Notes on Discrete Mathematics
CS 202: Fall 2013

James Aspnes

2013-12-03 11:27

Contents

Table of contents **i**

List of figures **xiv**

List of tables **xv**

List of algorithms **xvi**

Preface **xvii**

Syllabus **xviii**

Resources **xxii**

Lecture schedule **xxiv**

1 Introduction **1**

1.1 So why do I need to learn all this nasty mathematics? 1
1.2 But isn't math hard? 2
1.3 Thinking about math with your heart 3
1.4 What you should know about math 3
1.4.1 Foundations and logic 4
1.4.2 Basic mathematics on the real numbers 4
1.4.3 Fundamental mathematical objects 5
1.4.4 Modular arithmetic and polynomials 6
1.4.5 Linear algebra 6
1.4.6 Graphs 6
1.4.7 Counting 7
1.4.8 Probability 7
1.4.9 Tools 8

2 Mathematical logic 9
2.1 The basic picture . . . 9
2.1.1 Axioms, models, and inference rules . . . 9
2.1.2 Consistency . . . 10
2.1.3 What can go wrong . . . 10
2.1.4 The language of logic . . . 11
2.1.5 Standard axiom systems and models . . . 11
2.2 Propositional logic . . . 12
2.2.1 Operations on propositions . . . 13
2.2.1.1 Precedence . . . 15
2.2.2 Truth tables . . . 16
2.2.3 Tautologies and logical equivalence . . . 17
2.2.3.1 Inverses, converses, and contrapositives . . . 19
2.2.3.2 Equivalences involving true and false . . . 21
Example . . . 22
2.2.4 Normal forms . . . 23
2.3 Predicate logic . . . 24
2.3.1 Variables and predicates . . . 25
2.3.2 Quantifiers . . . 25
2.3.2.1 Universal quantifier . . . 26
2.3.2.2 Existential quantifier . . . 26
2.3.2.3 Negation and quantifiers . . . 27
2.3.2.4 Restricting the scope of a quantifier . . . 27
2.3.2.5 Nested quantifiers . . . 28
2.3.2.6 Examples . . . 30
2.3.3 Functions . . . 31
2.3.4 Equality . . . 31
2.3.4.1 Uniqueness . . . 32
2.3.5 Models . . . 32
2.3.5.1 Examples . . . 33
2.4 Proofs . . . 34
2.4.1 Inference Rules . . . 35
2.4.2 Proofs, implication, and natural deduction . . . 37
2.4.2.1 The Deduction Theorem . . . 37
2.5 Natural deduction . . . 38
2.5.1 Inference rules for equality . . . 39
2.5.2 Inference rules for quantified statements . . . 39
2.6 Proof techniques . . . 42

3 Set theory **47**
3.1 Naive set theory 47
3.2 Operations on sets 49
3.3 Proving things about sets 50
3.4 Axiomatic set theory 52
3.5 Cartesian products, relations, and functions 53
3.5.1 Examples of functions 55
3.5.2 Sequences 55
3.5.3 Functions of more (or less) than one argument 56
3.5.4 Composition of functions 56
3.5.5 Functions with special properties 56
3.5.5.1 Surjections 57
3.5.5.2 Injections 57
3.5.5.3 Bijections 57
3.5.5.4 Bijections and counting 57
3.6 Constructing the universe 58
3.7 Sizes and arithmetic 60
3.7.1 Infinite sets 60
3.7.2 Countable sets 62
3.7.3 Uncountable sets 62
3.8 Further reading 63

4 The real numbers **64**
4.1 Field axioms 65
4.1.1 Axioms for addition 65
4.1.2 Axioms for multiplication 66
4.1.3 Axioms relating multiplication and addition 68
4.1.4 Other algebras satisfying the field axioms 69
4.2 Order axioms 70
4.3 Least upper bounds 71
4.4 What's missing: algebraic closure 73
4.5 Arithmetic 73
4.6 Connection between the reals and other standard algebras . . 74
4.7 Extracting information from reals 76

5 Induction and recursion **77**
5.1 Simple induction 77
5.2 Alternative base cases 79
5.3 Recursive definitions work 80
5.4 Other ways to think about induction 80

5.5 Strong induction 81
5.5.1 Examples 82
5.6 Recursively-defined structures 83
5.6.1 Functions on recursive structures 84
5.6.2 Recursive definitions and induction 84
5.6.3 Structural induction 85

6 Summation notation 86
6.1 Summations 86
6.1.1 Formal definition 87
6.1.2 Scope 87
6.1.3 Summation identities 87
6.1.4 Choosing and replacing index variables 89
6.1.5 Sums over given index sets 89
6.1.6 Sums without explicit bounds 91
6.1.7 Infinite sums 91
6.1.8 Double sums 91
6.2 Products 92
6.3 Other big operators 92
6.4 Closed forms 93
6.4.1 Some standard sums 94
6.4.2 Guess but verify 95
6.4.3 Ansatzes 96
6.4.4 Strategies for asymptotic estimates 97
6.4.4.1 Pull out constant factors 97
6.4.4.2 Bound using a known sum 97
Geometric series 97
Constant series 98
Arithmetic series 98
Harmonic series 98
6.4.4.3 Bound part of the sum 98
6.4.4.4 Integrate 98
6.4.4.5 Grouping terms 98
6.4.4.6 Oddities 99
6.4.4.7 Final notes 99

7 Asymptotic notation 100
7.1 Definitions 100
7.2 Motivating the definitions 100
7.3 Proving asymptotic bounds 101

7.4 Asymptotic notation hints 102
7.4.1 Remember the difference between big-O, big-Ω, and big-Θ 102
7.4.2 Simplify your asymptotic terms as much as possible . 103
7.4.3 Remember the limit trick 103
7.5 Variations in notation 103
7.5.1 Absolute values 104
7.5.2 Abusing the equals sign 104

8 Number theory 106
8.1 Divisibility and division 106
8.2 Greatest common divisors 108
8.2.1 The Euclidean algorithm for computing $\gcd(m,n)$ . . 108
8.2.2 The extended Euclidean algorithm 109
8.2.2.1 Example 109
8.2.2.2 Applications 109
8.3 The Fundamental Theorem of Arithmetic 112
8.3.1 Applications 112
8.4 Modular arithmetic and residue classes 113
8.4.1 Arithmetic on residue classes 114
8.4.2 Division in $\mathbb{Z}_m$ 115
8.4.3 The Chinese Remainder Theorem 116
8.4.4 The size of $\mathbb{Z}_m^*$ and Euler's Theorem 119
8.5 RSA encryption 120

9 Relations 122
9.1 Representing relations 122
9.1.1 Directed graphs 122
9.1.2 Matrices 123
9.2 Operations on relations 124
9.2.1 Composition 124
9.2.2 Inverses 125
9.3 Classifying relations 125
9.4 Equivalence relations 125
9.4.1 Why we like equivalence relations 128
9.5 Partial orders 128
9.5.1 Drawing partial orders 130
9.5.2 Comparability 130
9.5.3 Lattices 131
9.5.4 Minimal and maximal elements 131

9.5.5 Total orders . . . 132
9.5.5.1 Topological sort . . . 132
9.5.6 Well orders . . . 135
9.6 Closures . . . 137
9.6.1 Examples . . . 139

10 Graphs 141
10.1 Types of graphs . . . 142
10.1.1 Directed graphs . . . 142
10.1.2 Undirected graphs . . . 142
10.1.3 Hypergraphs . . . 143
10.2 Examples of graphs . . . 144
10.3 Local structure of graphs . . . 145
10.4 Some standard graphs . . . 145
10.5 Subgraphs and minors . . . 148
10.6 Graph products . . . 150
10.6.1 Functions . . . 151
10.7 Paths and connectivity . . . 152
10.8 Cycles . . . 154
10.9 Proving things about graphs . . . 156
10.9.1 Paths and simple paths . . . 156
10.9.2 The Handshaking Lemma . . . 157
10.9.3 Characterizations of trees . . . 157
10.9.4 Spanning trees . . . 161
10.9.5 Eulerian cycles . . . 161

11 Counting 163
11.1 Basic counting techniques . . . 164
11.1.1 Equality: reducing to a previously-solved case . . . 164
11.1.2 Inequalities: showing $|A| \leq |B|$ and $|B| \leq |A|$ . . . 164
11.1.3 Addition: the sum rule . . . 165
11.1.3.1 For infinite sets . . . 166
11.1.3.2 The Pigeonhole Principle . . . 166
11.1.4 Subtraction . . . 166
11.1.4.1 Inclusion-exclusion for infinite sets . . . 167
11.1.4.2 Combinatorial proof . . . 167
11.1.5 Multiplication: the product rule . . . 168
11.1.5.1 Examples . . . 169
11.1.5.2 For infinite sets . . . 169
11.1.6 Exponentiation: the exponent rule . . . 169

11.1.6.1 Counting injections 170
11.1.7 Division: counting the same thing in two different ways 171
11.1.8 Applying the rules 173
11.1.9 An elaborate counting problem 174
11.1.10 Further reading 177
11.2 Binomial coefficients 177
11.2.1 Recursive definition 178
11.2.1.1 Pascal's identity: algebraic proof 179
11.2.2 Vandermonde's identity 180
11.2.2.1 Combinatorial proof 180
11.2.2.2 Algebraic proof 181
11.2.3 Sums of binomial coefficients 181
11.2.4 The general inclusion-exclusion formula 182
11.2.5 Negative binomial coefficients 183
11.2.6 Fractional binomial coefficients 184
11.2.7 Further reading 184
11.3 Generating functions 184
11.3.1 Basics 184
11.3.1.1 A simple example 184
11.3.1.2 Why this works 185
11.3.1.3 Formal definition 186
11.3.2 Some standard generating functions 189
11.3.3 More operations on formal power series and generating functions 189
11.3.4 Counting with generating functions 190
11.3.4.1 Disjoint union 190
11.3.4.2 Cartesian product 191
11.3.4.3 Repetition 191
Example: (0|11)∗ 191
Example: sequences of positive integers 191
11.3.4.4 Pointing 193
11.3.4.5 Substitution 193
Example: bit-strings with primes 194
Example: (0|11)* again 194
11.3.5 Generating functions and recurrences 194
11.3.5.1 Example: A Fibonacci-like recurrence 195
11.3.6 Recovering coefficients from generating functions . . . 195
11.3.6.1 Partial fraction expansion and Heaviside's cover-up method 197
Example: A simple recurrence 197

Example: Coughing cows 198
Example: A messy recurrence 199
11.3.6.2 Partial fraction expansion with repeated roots 201
Solving for the PFE directly 201
Solving for the PFE using the extended cover-up method 203
11.3.7 Asymptotic estimates 204
11.3.8 Recovering the sum of all coefficients 205
11.3.8.1 Example 205
11.3.9 A recursive generating function 206
11.3.10 Summary of operations on generating functions 209
11.3.11 Variants 210
11.3.12 Further reading 210

12 Probability theory **211**
12.1 Events and probabilities 212
12.1.1 Probability axioms 212
12.1.1.1 The Kolmogorov axioms 213
12.1.1.2 Examples of probability spaces 214
12.1.2 Probability as counting 214
12.1.2.1 Examples 215
12.1.3 Independence and the intersection of two events . . . 215
12.1.3.1 Examples 216
12.1.4 Union of events 217
12.1.4.1 Examples 217
12.1.5 Conditional probability 218
12.1.5.1 Conditional probabilities and intersections of non-independent events 218
12.1.5.2 The law of total probability 219
12.1.5.3 Bayes's formula 220
12.2 Random variables 220
12.2.1 Examples of random variables 220
12.2.2 The distribution of a random variable 221
12.2.2.1 Some standard distributions 222
12.2.2.2 Joint distributions 223
Examples 223
12.2.3 Independence of random variables 224
12.2.3.1 Examples 224
12.2.3.2 Independence of many random variables . . . 225
12.2.4 The expectation of a random variable 225

12.2.4.1 Variables without expectations 226
12.2.4.2 Expectation of a sum 227
Example 227
12.2.4.3 Expectation of a product 227
12.2.4.4 Conditional expectation 228
Examples 229
12.2.4.5 Conditioning on a random variable 231
12.2.5 Markov's inequality 232
12.2.5.1 Example 233
12.2.5.2 Conditional Markov's inequality 233
12.2.6 The variance of a random variable 233
12.2.6.1 Multiplication by constants 234
12.2.6.2 The variance of a sum 235
12.2.6.3 Chebyshev's inequality 236
Application: showing that a random variable is close to its expectation 236
Application: lower bounds on random variables 237
12.2.7 Probability generating functions 238
12.2.7.1 Sums 238
12.2.7.2 Expectation and variance 238
12.2.8 Summary: effects of operations on expectation and variance of random variables 240
12.2.9 The general case 240
12.2.9.1 Densities 241
12.2.9.2 Independence 242
12.2.9.3 Expectation 242

13 Linear algebra 243
13.1 Vectors and vector spaces 243
13.1.1 Relative positions and vector addition 244
13.1.2 Scaling 245
13.2 Abstract vector spaces 246
13.3 Matrices 247
13.3.1 Interpretation 248
13.3.2 Operations on matrices 249
13.3.2.1 Transpose of a matrix 249
13.3.2.2 Sum of two matrices 249
13.3.2.3 Product of two matrices 250
13.3.2.4 The inverse of a matrix 251
Example 252

13.3.2.5 Scalar multiplication 253
13.3.3 Matrix identities 253
13.4 Vectors as matrices 255
13.4.1 Length 255
13.4.2 Dot products and orthogonality 256
13.5 Linear combinations and subspaces 257
13.5.1 Bases 258
13.6 Linear transformations 259
13.6.1 Composition 260
13.6.2 Role of rows and columns of M in the product Mx . . 260
13.6.3 Geometric interpretation 261
13.6.4 Rank and inverses 263
13.6.5 Projections 264
13.7 Further reading 267

14 Finite fields 268
14.1 A magic trick 268
14.2 Fields and rings 269
14.3 Polynomials over a field 271
14.4 Algebraic field extensions 272
14.5 Applications 274
14.5.1 Linear-feedback shift registers 274
14.5.2 Checksums 275
14.5.3 Cryptography 276

A Assignments 277
A.1 Assignment 1: due Thursday, 2013-09-12, at 5:00 pm (Huamin)277
A.1.1 Tautologies 278
A.1.2 Positively equivalent 279
A.1.3 A theory of leadership 280
A.2 Assignment 2: due Thursday, 2013-09-19, at 5:00 pm (Xiao) . 281
A.2.1 Subsets 281
A.2.2 A distributive law 282
A.2.3 Exponents 282
A.3 Assignment 3: due Thursday, 2013-09-26, at 5:00 pm (Mike) . 283
A.3.1 Surjections 283
A.3.2 Proving an axiom the hard way 284
A.3.3 Squares and bigger squares 285
A.4 Assignment 4: due Thursday, 2013-10-03, at 5:00 pm (Huamin)285
A.4.1 A fast-growing function 285

A.4.2 A slow-growing set 286
A.4.3 Double factorials 287
A.5 Assignment 5: due Thursday, 2013-10-10, at 5:00 pm (Xiao) . 289
A.5.1 A bouncy function 289
A.5.2 Least common multiples of greatest common divisors . 290
A.5.3 Adding and subtracting 290
A.6 Assignment 6: due Thursday, 2013-10-31, at 5:00 pm (Mike) . 291
A.6.1 Factorials mod n 291
A.6.2 Indivisible and divisible 291
A.6.3 Equivalence relations 292
A.7 Assignment 7: due Thursday, 2013-11-07, at 5:00 pm (Mike) 293
A.7.1 Flipping lattices with a function 293
A.7.2 Splitting graphs with a mountain 293
A.7.3 Drawing stars with modular arithmetic 294
A.8 Assignment 8: due Thursday, 2013-11-14, at 5:00 pm (Huamin)
. . . . 295
A.8.1 Two-path graphs 295
A.8.2 Even teams 297
A.8.3 Inflected sequences 297
A.9 Assignment 9: due Thursday, 2013-11-21, at 5:00 pm (Xiao) 297
A.9.1 Guessing the median 297
A.9.2 Two flushes 298
A.9.3 Dice and more dice 300

B Exams 302
B.1 CS202 Exam 1, October 17th, 2013 302
B.1.1 A tautology (20 points) 302
B.1.2 A system of equations (20 points) 302
B.1.3 A sum of products (20 points) 303
B.1.4 A subset problem (20 points) 304

C Midterm exams from previous semesters 305
C.1 Midterm Exam, October 12th, 2005 305
C.1.1 A recurrence (20 points) 305
C.1.2 An induction proof (20 points) 306
C.1.3 Some binomial coefficients (20 points) 307
C.1.4 A probability problem (20 points) 307
C.2 Midterm Exam, October 24th, 2007 308
C.2.1 Dueling recurrences (20 points) 308
C.2.2 Seating arrangements (20 points) 308

C.2.3 Non-attacking rooks (20 points) 309
C.2.4 Subsets (20 points) 310
C.3 Midterm Exam, October 24th, 2008 310
C.3.1 Some sums (20 points) 310
C.3.2 Nested ranks (20 points) 310
C.3.3 Nested sets (20 points) 311
C.3.4 An efficient grading method (20 points) 312
C.4 Midterm exam, October 21st, 2010 312
C.4.1 A partial order (20 points) 313
C.4.2 Big exponents (20 points) 313
C.4.3 At the playground (20 points) 313
C.4.4 Gauss strikes back (20 points) 314

D Final exams from previous semesters 315
D.1 CS202 Final Exam, December 15th, 2004 315
D.1.1 A multiplicative game (20 points) 315
D.1.2 An equivalence in space (20 points) 317
D.1.3 A very big fraction (20 points) 317
D.1.4 A pair of odd vertices (20 points) 318
D.1.5 How many magmas? (20 points) 318
D.1.6 A powerful relationship (20 points) 318
D.1.7 A group of archaeologists (20 points) 319
D.2 CS202 Final Exam, December 16th, 2005 319
D.2.1 Order (20 points) 320
D.2.2 Count the subgroups (20 points) 320
D.2.3 Two exits (20 points) 320
D.2.4 Victory (20 points) 321
D.2.5 An aggressive aquarium (20 points) 321
D.2.6 A subspace of matrices (20 points) 322
D.3 CS202 Final Exam, December 20th, 2007 323
D.3.1 A coin-flipping problem (20 points) 323
D.3.2 An ordered group (20 points) 324
D.3.3 Weighty vectors (20 points) 324
D.3.4 A dialectical problem (20 points) 325
D.3.5 A predictable pseudorandom generator (20 points) . . 326
D.3.6 At the robot factory (20 points) 327
D.4 CS202 Final Exam, December 19th, 2008 328
D.4.1 Some logical sets (20 points) 328
D.4.2 Modularity (20 points) 328
D.4.3 Coin flipping (20 points) 329

D.4.4 A transitive graph (20 points) 329
D.4.5 A possible matrix identity (20 points) 329
D.5 CS202 Final Exam, December 14th, 2010 330
D.5.1 Backwards and forwards (20 points) 330
D.5.2 Linear transformations (20 points) 331
D.5.3 Flipping coins (20 points) 332
D.5.4 Subtracting dice (20 points) 333
D.5.5 Scanning an array (20 points) 334

E How to write mathematics 335
E.1 By hand 335
E.2 LaTeX 335
E.3 Microsoft Word equation editor 336
E.4 ASCII and/or Unicode art 336

F Tools from calculus 338
F.1 Limits 338
F.2 Derivatives 340
F.3 Integrals 341

G The natural numbers 344
G.1 The Peano axioms 344
G.2 A simple proof 346
G.3 Defining addition 347
G.3.1 Other useful properties of addition 349
G.4 A scary induction proof involving even numbers 350
G.5 Defining more operations 351

Bibliography 353

Index 356

List of Figures

8.1 Trace of extended Euclidean algorithm 110
8.2 Python code for extended Euclidean algorithm 111

9.1 A directed graph 123
9.2 Relation as a directed graph 123
9.3 Factors of 12 partially ordered by divisibility 130
9.4 Maximal and minimal elements 132
9.5 Topological sort 133
9.6 Reflexive, symmetric, and transitive closures 138
9.7 Strongly-connected components 139

10.1 A directed graph 142
10.2 A graph 143
10.3 Two representations of a hypergraph 144
10.4 Complete graphs K_1 through K_{10} 146
10.5 Cycle graphs C_3 through C_{11} 147
10.6 Path graphs P_0 through P_4 148
10.7 Complete bipartite graph $K_{3,4}$ 148
10.8 star graphs $K_{1,3}$ through $K_{1,8}$ 149
10.9 Two presentations of the cube graph Q_3 149
10.10Subgraphs and minors 150
10.11Cycles and closed walks 155

13.1 Geometric interpretation of vector addition 244

A.1 Examples of $S_{m,k}$ for Problem A.7.3 294
A.2 All 90 two-path graphs on five vertices 296

List of Tables

2.1 Compound propositions 15
2.2 Common logical equivalences 20
2.3 Absorption laws 22
2.4 Natural deduction: introduction and elimination rules 40
2.5 Proof techniques 46

4.1 Features of various standard algebras 76

8.1 Addition and multiplication tables for $\mathbb{Z}_3$ 115
8.2 Multiplication table for $\mathbb{Z}_9$ 116

F.1 Table of derivatives 341
F.2 Table of integrals 342

List of Algorithms

Preface

These are notes for the Fall 2013 semester version of the Yale course CPSC 202a, Mathematical Tools for Computer Science. This document also incorporates the lecture schedule and assignments, as well as some sample assignments from previous semesters. Because this is a work in progress, it will be updated frequently over the course of the semester.

Notes from some previous versions of the course can be found at `http://pine.cs.yale.edu/pinewiki/CS202/`.

Syllabus

Description

Introduction to formal methods for reasoning and to mathematical techniques basic to computer science. Topics include propositional logic, discrete mathematics, and linear algebra. Emphasis on applications to computer science: recurrences, sorting, graph traversal, Gaussian elimination.

Meeting times

Tuesday and Thursday 13:00–14:15 in LC 317.

On-line course information

The lecture schedule, course notes, and all assignments can be found in a single gigantic PDF file at `http://cs.yale.edu/homes/aspnes/classes/202/notes.pdf`. You should probably bookmark this file, as it will be updated frequently.

Office hours and other information can be found in the course calendar at Google Calendar.

Staff

If you can't make the open office hours listed in the course calendar, you may be able to schedule appointments at other times by email.

Instructor

James Aspnes. Office: AKW 401. Email: `james.aspnes@gmail.com`.

Teaching fellows

- Jerrod Ankenman. Office: AKW 114. Email:
- Huamin Li. Office: AKW 015. Email: `huamin.li@yale.edu`.
- Mike Marmar. Office: AKW 313. Email: `michael.marmar@yale.edu`.

Undergraduate course grader

- Xiao Shi. Email: `xiao.shi@yale.edu`.

Peer tutors

- Iulia Tamas. Email: `iulia.tamas@yale.edu`.
- Michael Tan. Email: `michael.tan@yale.edu`.
- Michael Zhao. Email: `michael.zhao@yale.edu`.

Textbook

The textbook for the class is Kevin Ferland, *Discrete Mathematics*, Cengage Learning, 2008. ISBN 0618415386.

Course requirements

Nine weekly homework assignments and two exams held at the regular lecture time. The exams will count for approximately three homework assignments each.

Use of outside help

Students are free to discuss homework problems and course material with each other, and to consult with the instructor or a TA. Solutions handed in, however, should be the student's own work. If a student benefits substantially from hints or solutions received from fellow students or from outside sources, then the student should hand in their solution but acknowledge the outside sources, and we will apportion credit accordingly. Using outside resources in solving a problem is acceptable but plagiarism is not.

Clarifications for homework assignments

From time to time, ambiguities and errors may creep into homework assignments. Questions about the interpretation of homework assignments should be sent to the instructor at `james.aspnes@gmail.com`. Clarifications will appear in an updated version of the assignment.

Late assignments

Late assignments will not be accepted without a Dean's Excuse.

Topics

The course will cover the minimal topics in mathematics that you will need to survive the Computer Science major. We assume that coming in to the course you will already have a thorough grounding in high school algebra, but may or may not have taken any other math classes. By the end of the course, you should:

- Understand definitions and proofs, including quantifiers and induction.
- Understand basic set theory and set-theoretic notation.
- Be comfortable with manipulating commonly-used functions like exponentials and logarithms.
- Know how to count (not as easy as you think).
- Understand asymptotic notation.
- Know how to solve recurrences.
- Understand basic probability.
- Have a passing familiarity with standard mathematical concepts that show up in computer science, including graphs, algebraic structures (e.g., groups, rings, and fields), linear algebra and matrices, and basic number theory.

Because CS202 is only a one-semester course, coverage of most topics will necessarily be rather sketchy. If you expect to do further work in the theoretical end of computer science or in math-intensive fields like graphics, vision, neural networks, robotics, or scientific computation, you should plan to take further courses in mathematics (a serious linear algebra course is particularly recommended). One of the goals of CS202 is to provide you with the mathematical maturity you will need to profit from such courses.

Resources

In addition to the textbook and these notes, you may find many other resources useful.

Reserved books at Bass Library

- Kevin Ferland. *Discrete Mathematics.* Cengage Learning, 2008. ISBN 978-0618415380. [Fer08]
- Kenneth H. Rosen. *Discrete Mathematics and Its Applications*, Seventh Edition, McGraw-Hill, 2012. ISBN 978-0072899054. QA39.3 R67 2012 (LC). [Ros12]
- Norman L. Biggs. *Discrete Mathematics*, second edition. Oxford University Press, 2002. ISBN 0198507178. QA76.9 M35 B54 2002. [Big02]
- Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, 1989. QA39.2 G733X 1989. [GKP94]
- George Polya. *How to Solve It: A New Aspect of Mathematical Method.* Princeton University Press, 2004. QA11 .P66 2004 (LC). [Pol04]
- Daniel Solow. *How to Read and Do Proofs: An Introduction to Mathematical Thought Processes.* Wiley, 2005. QA9.54 .S65X 2005 (LC) [Sol05]

Internet resources

PlanetMath `http://planetmath.org`

Wolfram MathWorld `http://mathworld.wolfram.com`

WikiPedia `http://en.wikipedia.org`

Google `http://www.google.com`

Lecture schedule

As always, the future is uncertain, so you should take parts of the schedule that haven't happened yet with a grain of salt. Readings refer to chapters or sections in the course notes, except for those specified as Ferland, which refer to the course textbook [Fer08].

See also the course calendar, which lists lecture times, office hours for the instructor and TA, and assignment due dates.

2013-08-29 Overview of the course. Start of mathematical logic: basic principles and propositional logic. Boolean operators, truth tables, tautologies, and logical equivalence. Readings: Chapter 1, §§2.1–2.2; Ferland §1.1.

2013-09-03 More proposition logic: simplifying expressions using standard equivalences. Predicate logic: constants, predicates, variables and quantifiers. Nested quantifiers at the playground and the subtle but important difference between $\exists x : \forall y : \text{likes}(y, x)$ and $\forall y : \exists x : \text{likes}(y, x)$. Readings: §2.3 through §2.3.2; Ferland §1.3.

2013-09-05 More predicate logic: Function symbols and equality. Models in predicate logic. Start of proofs and inference rules. Readings: §§2.3.3–2.5.2; Ferland §1.5.

2013-09-10 Start of sets: set notation, basic operations on sets, examples of set-theory axioms. Readings: §§3.1–3.4; Ferland §§1.2 and §1.4.

2013-09-12 More set theory: Cartesian products, relations, and functions. Sequences as functions. Bijections and counting. Readings: §§3.5–3.7.1. Ferland §5.1 through Example 5.5, §5.3 through Example 5.37, §5.4 through Example 5.47.

2013-09-17 More on writing proofs about relations between sets. Countable vs uncountable sets. Readings: §§3.3, 3.7.2, and 3.7.3; Ferland §§2.2–2.3.

2013-09-19 The real numbers and their properties. Readings: Chapter 4; Ferland Appendix B.

2013-09-24 Induction and recursion. Readings: §§5.1–5.5; Ferland §§4.1, 4.3, and 4.5.

2013-09-26 Summation notation. Readings: §§6.1–6.3; Ferland §§4.2 and 4.4.

2013-10-01 Closed-form solutions to summations. Asymptotic notation. Readings: §6.4, Chapter 7; Ferland §10.4.

2013-10-03 Number theory: divisibility and greatest common divisors. Readings: §§8.1–8.2.1; Ferland §§3.1–3.2.

2013-10-08 Modular arithmetic. The extended Euclidean algorithm and inverses mod m. Readings: §§8.2.2 and 8.4; Ferland §§3.3 and 3.6.

2013-10-10 Various number theorems: The Fundamental Theorem of Arithmetic, the Chinese Remainder Theorem, Euler's Theorem. RSA encryption. Readings: §§8.3 and 8.5.

2013-10-15 Start of relations: relations digraphs, and matrices; equivalence relations. Readings: Chapter 9 through §9.4; Ferland §5.1, second part of §5.2.

2013-10-17 **Exam 1**. The first exam was given at the usual class time in SCL 110. It was a closed-book exam covering all material discussed up to this point. Sample solutions are in Appendix B.1.

2013-10-22 Partial orders. Readings: §9.5 though §9.5.3; Ferland first part of §5.2.

2013-10-29 More on relations and digraphs. Minimal and maximal elements and topological sort. A brief digression on well-ordered sets and transfinite induction. Reflexive, symmetric, and transitive closures. Strongly-connected components and pre-orders. Readings: §§9.5.4–§§9.6.

2013-10-31 More graph theory. Readings: Chapter 10; Ferland §§8.1, 8.3, 9.1–9.3, and 10.1.

2013-11-05 Basic counting principles: Equality by direct bijection or $\leq$ plus $\geq$; the sum rule; subtraction and inclusion-exclusion for two sets; the product rule; exponentiation and variants (n^k, $(n)_k$); division and binomial coefficients $\binom{n}{k}$. Readings: §11.1; Ferland §6.1–6.3.

2013-11-07 Binomial coefficients and the binomial theorem: Pascal's triangle identity, symmetry, Vandermonde's identity. Basics of generating functions. Readings: §11.2 and §11.3 through §11.3.4.2; Ferland §§7.1 and 7.3.

2013-11-12 Discrete probability theory: probability spaces, events, and conditional probability. Readings: §12.1; Ferland §6.4.

2013-11-14 More probability theory: random variables and expectations. Readings: §12.2 through §12.2.5.

2013-11-19 Linear algebra: vector and matrix operations. Computing matrix inverses using Gauss-Jordan elimination. Readings: §§13.1–13.3.

2013-11-21 More linear algebra: Vectors as matrices. Dot products, linear combinations, and linear transformations. Readings: §§13.4–13.7.

2013-12-03 Finite fields. Readings: Chapter 14.

2013-12-05 **Exam 2**. The second exam will be given at the usual class time in **SPL 59**. It will be a closed-book exam covering all material discussed during the semester. See Appendix D for examples of exams from previous semesters.

Chapter 1

Introduction

This is a course on discrete mathematics as used in Computer Science. It's only a one-semester course, so there are a lot of topics that it doesn't cover or doesn't cover in much depth. But the hope is that this will give you a foundation of skills that you can build on as you need to, and particularly to give you a bit of **mathematical maturity**—the basic understanding of what mathematics is and how mathematical definitions and proofs work.

1.1 So why do I need to learn all this nasty mathematics?

Why you should know about mathematics, if you are interested in Computer Science: or, more specifically, why you should take CS202 or a comparable course:

- Computation is something that you can't see and can't touch, and yet (thanks to the efforts of generations of hardware engineers) it obeys strict, well-defined rules with astonishing accuracy over long periods of time.
- Computations are too big for you to comprehend all at once. Imagine printing out an execution trace that showed every operation a typical \$500 desktop computer executed in one (1) second. If you could read one operation per second, for eight hours every day, you would die of old age before you got halfway through. Now imagine letting the computer run overnight.

So in order to understand computations, we need a language that allows us to reason about things we can't see and can't touch, that are too big for us to understand, but that nonetheless follow strict, simple, well-defined rules. We'd like our reasoning to be consistent: any two people using the language should (barring errors) obtain the same conclusions from the same information. Computer scientists are good at inventing languages, so we could invent a new one for this particular purpose, but we don't have to: the exact same problem has been vexing philosophers, theologians, and mathematicians for much longer than computers have been around, and they've had a lot of time to think about how to make such a language work. Philosophers and theologians are still working on the consistency part, but mathematicians (mostly) got it in the early 20th-century. Because the first virtue of a computer scientist is laziness, we are going to *steal their code.*

1.2 But isn't math hard?

Yes and no. The human brain is not really designed to do formal mathematical reasoning, which is why most mathematics was invented in the last few centuries and why even apparently simple things like learning how to count or add require years of training, usually done at an early age so the pain will be forgotten later. But mathematical reasoning is very close to legal reasoning, which we do seem to be very good at.[1]

There is very little structural difference between the two sentences:

1. If x is in S, then $x + 1$ is in S.
2. If x is of royal blood, then x's child is of royal blood.

But because the first is about boring numbers and the second is about fascinating social relationships and rules, most people have a much easier time deducing that to show somebody is royal we need to start with some known royal and follow a chain of descendants than they have deducing that to show that some number is in the set S we need to start with some known element of S and show that repeatedly adding 1 gets us to the number we want. And yet to a logician these are the same processes of reasoning.

So why is statement (1) trickier to think about than statement (1)? Part of the difference is familiarity—we are all taught from an early age what it means to be somebody's child, to take on a particular social role, etc. For mathematical concepts, this familiarity comes with exposure and practice, just as with learning any other language. But part of the difference is that

[1] For a description of some classic experiments that demonstrate this, see `http://en.wikipedia.org/wiki/Wason_selection_task`.

we humans are wired to understand and appreciate social and legal rules: we are very good at figuring out the implications of a (hypothetical) rule that says that any contract to sell a good to a consumer for $100 or more can be canceled by the consumer within 72 hours of signing it provided the good has not yet been delivered, but we are not so good at figuring out the implications of a rule that says that a number is composite if and only if it is the product of two integer factors neither of which is 1. It's a lot easier to imagine having to cancel a contract to buy swampland in Florida that you signed last night while drunk than having to prove that 82 is composite. But again: there is nothing more natural about contracts than about numbers, and if anything the conditions for our contract to be breakable are more complicated than the conditions for a number to be composite.

1.3 Thinking about math with your heart

There are two things you need to be able to do to get good at mathematics (the creative kind that involves writing proofs, not the mechanical kind that involves grinding out answers according to formulas). One of them is to learn the language: to attain what mathematicians call **mathematical maturity**. You'll do that in CS202, if you pay attention. But the other is to learn how to activate the parts of your brain that are good at mathematical-style reasoning when you do math—the parts evolved to detect when the other primates in your band of hunter-gatherers are cheating.

To do this it helps to get a little angry, and imagine that finishing a proof or unraveling a definition is the only thing that will stop your worst enemy from taking some valuable prize that *you* deserve. (If you don't have a worst enemy, there is always the universal quantifier.) But whatever motivation you choose, you need to be fully engaged in what you are doing. Your brain is smart enough to know when you don't care about something, and if you don't believe that thinking about math is important, it will think about something else.

1.4 What you should know about math

We won't be able to cover all of this, but the list below might be a minimal set of topics it would be helpful to understand for computer science. Topics that we didn't do this semester are marked with (*).

1.4.1 Foundations and logic

Why: This is the assembly language of mathematics—the stuff at the bottom that everything else complies to.

- Propositional logic.
- Predicate logic.
- Axioms, theories, and models.
- Proofs.
- Induction and recursion.

1.4.2 Basic mathematics on the real numbers

Why: You need to be able to understand, write, and prove equations and inequalities involving real numbers.

- Standard functions and their properties: addition, multiplication, exponentiation, logarithms.
- More specialized functions that come up in algorithm analysis: floor, ceiling, max, min.
- Techniques for proving inequalities, including:
  - General inequality axioms (transitivity, anti-symmetry, etc.)
  - Inequality axioms for $\mathbb{R}$ (i.e., how $<$ interacts with addition, multiplication, etc.)
  - Techniques involving derivatives (assumes calculus) (*):
    - Finding local extrema of f by solving for $f'(x) = 0$. (*)
    - Using f'' to distinguish local minima from local maxima. (*)
    - Using $f'(x) \leq g'(x)$ in $[a, b]$ and $f(a) \leq g(a)$ or $f(b) \leq g(b)$ to show $f(x) \leq g(x)$ in $[a, b]$. (*)
- Special subsets of the real number: rationals, integers, natural numbers.

1.4.3 Fundamental mathematical objects

Why: These are the mathematical equivalent of data structures, the way that more complex objects are represented.

- Set theory.
  - Naive set theory.
  - Predicates vs sets.
  - Set operations.
  - Set comprehension.
  - Russell's paradox and axiomatic set theory.
- Functions.
  - Functions as sets.
  - Injections, surjections, and bijections.
  - Cardinality.
  - Finite vs infinite sets.
  - Sequences.
- Relations.
  - Equivalence relations, equivalence classes, and quotients.
  - Orders.
- The basic number tower.
  - Countable universes: $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$. (Can be represented in a computer.)
  - Uncountable universes: $\mathbb{R}, \mathbb{C}$. (Can only be approximated in a computer.)
- Other algebras.
  - The string monoid. (*)
  - $\mathbb{Z}_m$ and $\mathbb{Z}_p$.
  - Polynomials over various rings and fields.

1.4.4 Modular arithmetic and polynomials

Why: Basis of modern cryptography.

- Arithmetic in $\mathbb{Z}_m$.
- Primes and divisibility.
- Euclid's algorithm and inverses.
- The Chinese Remainder Theorem.
- Fermat's Little Theorem and Euler's Theorem.
- RSA encryption.
- Galois fields and applications.

1.4.5 Linear algebra

Why: Shows up everywhere.

- Vectors and matrices.
- Matrix operations and matrix algebra.
- Inverse matrices and Gaussian elimination.
- Geometric interpretations.

1.4.6 Graphs

Why: Good for modeling interactions. Basic tool for algorithm design.

- Definitions: graphs, digraphs, multigraphs, etc.
- Paths, connected components, and strongly-connected components.
- Special kinds of graphs: paths, cycles, trees, cliques, bipartite graphs.
- Subgraphs, induced subgraphs, minors.

1.4.7 Counting

Why: Basic tool for knowing how much resources your program is going to consume.

- Basic combinatorial counting: sums, products, exponents, differences, and quotients.
- Combinatorial functions.
  - Factorials.
  - Binomial coefficients.
  - The 12-fold way. (*)
- Advanced counting techniques.
  - Inclusion-exclusion.
  - Recurrences. (*)
  - Generating functions. (Limited coverage.)

1.4.8 Probability

Why: Can't understand randomized algorithms or average-case analysis without it. Handy if you go to Vegas.

- Discrete probability spaces.
- Events.
- Independence.
- Random variables.
- Expectation and variance.
- Probabilistic inequalities.
  - Markov's inequality.
  - Chebyshev's inequality. (*)
  - Chernoff bounds. (*)
- Stochastic processes. (*)
  - Markov chains. (*)
  - Martingales. (*)
  - Branching processes. (*)

1.4.9 Tools

Why: Basic computational stuff that comes up, but doesn't fit in any of the broad categories above. These topics will probably end up being mixed in with the topics above.

- Things you may have forgotten about exponents and logarithms. (*)
- Inequalities and approximations.
- $\sum$ and $\prod$ notation.
- How to differentiate and integrate simple functions. (*)
- Computing or approximating the value of a sum.
- Asymptotic notation.

Chapter 2

Mathematical logic

Mathematical logic is the discipline that mathematicians invented in the late nineteenth and early twentieth centuries so they could stop talking nonsense. It's the most powerful tool we have for reasoning about things that we can't really comprehend, which makes it a perfect tool for Computer Science.

2.1 The basic picture

Reality		Model		Theory
herds of sheep				
piles of rocks	$\rightarrow$	$\mathbb{N} = \{0, 1, 2, \ldots\}$	$\rightarrow$	$\forall x : \exists y : y = x + 1$
tally marks				

We want to model something we see in reality with something we can fit in our heads. Ideally we drop most of the features of the real thing that we don't care about and keep the parts that we do care about. But there is a second problem: if our model is very big (and the natural numbers are very *very* big), how do we know what we can say about them?

2.1.1 Axioms, models, and inference rules

One approach is to true to come up with a list of **axioms** that are true statements about the model and a list of **inference rules** that let us derive new true statements from the axioms. The axioms and inference rules together generate a **theory** that consists of all statements that can be constructed from the axioms by applying the inference rules. The rules of the game are that we can't claim that some statement is true unless it's a **theorem**: something we can derive as part of the theory.

Simple example: All fish are green (axiom). George Washington is a fish (axiom). From "all X are Y" and "Z is X", we can derive "Z is Y" (inference rule). Thus George Washington is green (theorem). Since we can't do anything else with our two axioms and one inference rule, these three statements together form our entire theory about George Washington, fish, greenness, etc.

Theories are attempts to describe **models**. A model is typically a collection of objects and relations between them. For a given theory, there may be many models that are consistent with it: for example, a model that includes both green fishy George Washington and MC 900-foot Abraham Lincoln is consistent with the theory above, because the theory doesn't say anything about Abraham Lincoln.

2.1.2 Consistency

A theory is **consistent** if it can't prove both P and not-P for any P. Consistency is incredibly important, since all the logics people actually use can prove anything starting from P and not-P.

2.1.3 What can go wrong

If we throw in too many axioms, you can get an inconsistency: "All fish are green; all sharks are not green; all sharks are fish; George Washington is a shark" gets us into trouble pretty fast.

If we don't throw in enough axioms, we underconstrain the model. For example, the Peano axioms for the natural numbers (see example below) say (among other things) that there is a number 0 and that any number x has a successor $S(x)$ (think of $S(x)$ as $x+1$). If we stop there, we might have a model that contains only 0, with $S(0) = 0$. If we add in $0 \neq S(x)$ for any x, then we can get stuck at $S(0) = 1 = S(1)$. If we add yet another axiom that says $S(x) = S(y)$ if and only if $x = y$, then we get all the ordinary natural numbers $0, S(0) = 1, S(1) = 2$, etc., but we could also get some extras: say $0', S(0') = 1', S(1') = 0'$. Characterizing the "correct" natural numbers historically took a lot of work to get right, even though we all know what we mean when we talk about them. The situation is of course worse when we are dealing with objects that we don't really understand; here the most we can hope for is to try out some axioms and see if anything strange happens.

Better yet is to use some canned axioms somebody else has already debugged for us. In this respect the core of mathematics acts like a system library—it's a collection of useful structures and objects that are known to work, and (if we are lucky) may even do exactly what we expect.

2.1.4 The language of logic

The basis of mathematical logic is **propositional logic**, which was essentially invented by Aristotle. Here the model is a collection of **statements** that are either true or false. There is no ability to refer to actual things; though we might include the statement "George Washington is a fish", from the point of view of propositional logic that is an indivisible atomic chunk of truth or falsehood that says nothing in particular about George Washington or fish. If we treat it as an axiom we can prove the truth of more complicated statements like "George Washington is a fish or 2+2=5" (true since the first part is true), but we can't really deduce much else. Still, this is a starting point.

If we want to talk about things and their properties, we must upgrade to **predicate logic**. Predicate logic adds both **constants** (stand-ins for objects in the model like "George Washington") and **predicates** (stand-ins for properties like "is a fish"). It also lets use **quantify** over variables and make universal statements like "For all x, if x is a fish then x is green." As a bonus, we usually get functions ("$f(x)$ = the number of books George Washington owns about x") and equality ("George Washington = 12" implies "George Washington + 5 = 17"). This is enough machinery to define and do pretty much all of modern mathematics.

We will discuss both of these logics in more detail below.

2.1.5 Standard axiom systems and models

Rather than define our own axiom systems and models from scratch, it helps to use ones that already have a track record of consistency and usefulness. Almost all mathematics fits in one of the following models:

- The natural numbers $\mathbb{N}$. These are defined using the Peano axioms, and if all you want to do is count, add, and multiply, you don't need much else. (If you want to subtract, things get messy.)
- The integers $\mathbb{Z}$. Like the naturals, only now we can subtract. Division is still a problem.
- The rational numbers $\mathbb{Q}$. Now we can divide. But what about $\sqrt{2}$?
- The real numbers $\mathbb{R}$. Now we have $\sqrt{2}$. But what about $\sqrt{(-1)}$?

- The complex numbers $\mathbb{C}$. Now we are pretty much done. But what if we want to talk about more than one complex number at a time?
- The universe of sets. These are defined using the axioms of set theory, and produce a rich collection of sets that include, among other things, structures equivalent to the natural numbers, the real numbers, collections of same, sets so big that we can't even begin to imagine what they look like, and even bigger sets so big that we can't use the usual accepted system of axioms to prove whether they exist or not. Fortunately, in computer science we can mostly stop with finite sets, which makes life less confusing.
- Various alternatives to set theory, like lambda calculus, category theory, or second-order arithmetic. We won't talk about these, since they generally don't let you do anything you can't do already with sets. However, lambda calculus and category theory are both important to know about if you are interested in programming language theory.

In practice, the usual way to do things is to start with sets and then define everything else in terms of sets: e.g., 0 is the empty set, 1 is a particular set with 1 element, 2 a set with 2 elements, etc., and from here we work our way up to the fancier numbers. The idea is that if we trust our axioms for sets to be consistent, then the things we construct on top of them should also be consistent, although if we are not careful in our definitions they may not be exactly the things we think they are.

2.2 Propositional logic

Propositional logic is the simplest form of logic. Here the only statements that are considered are **propositions**, which contain no variables. Because propositions contain no variables, they are either always true or always false.

Examples of propositions:

- $2 + 2 = 4$. (Always true).
- $2 + 2 = 5$. (Always false).

Examples of non-propositions:

- $x + 2 = 4$. (May be true, may not be true; it depends on the value of x.)

- $x \cdot 0 = 0$. (Always true, but it's still not a proposition because of the variable.)
- $x \cdot 0 = 1$. (Always false, but not a proposition because of the variable.)

As the last two examples show, it is not enough for a statement to be always true or always false—whether a statement is a proposition or not is a structural property. But if a statement doesn't contain any variables (or other undefined terms), it is a proposition, and as a side-effect of being a proposition it's always true or always false.

2.2.1 Operations on propositions

Propositions by themselves are pretty boring. So boring, in fact, that logicians quickly stop talking about specific propositions and instead haul out placeholder names like p, q, or r. But we can build slightly more interesting propositions by combining propositions together using various logical connectives, such as:

Negation The **negation** of p is written as $\neg p$, or sometimes $-p$ or $\overline{p}$. It has the property that it is false when p is true, and true when p is false.

Or The **or** of two propositions p and q is written as $p \vee q$, and is true as long as at least one, or possibly both, of p and q is true.[1] This is not always the same as what "or" means in English; in English, "or" often is used for *exclusive or* which is not true if both p and q are true. For example, if someone says "You will give me all your money or I will stab you with this table knife", you would be justifiably upset if you turn over all your money and still get stabbed. But a logician would not be at all surprised, because the standard "or" in propositional logic is an **inclusive or** that allows for both outcomes.

Exclusive or If you want to exclude the possibility that both p and q are true, you can use **exclusive or** instead. This is written as $p \oplus q$, and is true precisely when exactly one of p or q is true. Exclusive or is not used in classical logic much, but is important for many computing applications, since it corresponds to addition modulo 2 (see §8.4) and

[1]The symbol $\vee$ is a stylized V, intended to represent the Latin word *vel*, meaning "or." (Thanks to Noel McDermott for remembering this.) Much of this notation is actually pretty recent (early 20th century): see `http://jeff560.tripod.com/set.html` for a summary of earliest uses of each symbol.

has nice reversibility properties (e.g. $p \oplus (p \oplus q)$ always has the same truth-value as q).

And The **and** of p and q is written as $p \wedge q$, and is true only when both p and q are true.[2] This is pretty much the same as in English, where "I like to eat ice cream and I own a private Caribbean island" is not a true statement when made by most people even though most people like to eat ice cream. The only complication in translating English expressions into logical *and*s is that logicians can't tell the difference between "and" and "but": the statement "$2 + 2 = 4$ but $3 + 3 = 6$" becomes simply "$(2 + 2 = 4) \wedge (3 + 3 = 6)$."

Implication This is the most important connective for proofs. An **implication** represents an "if…then" claim. If p implies q, then we write $p \rightarrow q$ or $p \Rightarrow q$, depending on our typographic convention and the availability of arrow symbols in our favorite font. In English, $p \rightarrow q$ is usually rendered as "If p, then q," as in "If you step on your own head, it will hurt." The meaning of $p \rightarrow q$ is that q is true whenever p is true, and the proposition $p \rightarrow q$ is true provided (a) p is false (in which case all bets are off), or (b) q is true.

In fact, the only way for $p \rightarrow q$ to be *false* is for p to be true but q to be false. Because of this, $p \rightarrow q$ can be rewritten as $\neg p \vee q$. So, for example, the statements "If $2 + 2 = 5$, then I'm the Pope", "If I'm the Pope, then $2 + 2 = 4$", and "If $2 + 2 = 4$, then $3 + 3 = 6$", are all true, provided the if/then is interpreted as implication. Normal English usage does not always match this pattern; instead, if/then in normal speech is often interpreted as the much stronger *biconditional* (see below).

Biconditional Suppose that $p \rightarrow q$ and $q \rightarrow p$, so that either both p and q are true or both p and q are false. In this case, we write $p \leftrightarrow q$ or $p \Leftrightarrow q$, and say that p holds **if and only if** q holds. The truth of $p \leftrightarrow q$ is still just a function of the truth or falsehood of p and q; though there doesn't need to be any connection between the two sides of the statement, "$2 + 2 = 5$ if and only if I am the Pope" is a true statement (provided it is not uttered by the Pope). The only way for $p \leftrightarrow q$ to be false is for one side to be true and one side to be false.

The result of applying any of these operations is called a **compound proposition**.

[2]The symbol $\wedge$ is a stylized A, short for the latin word *atque*, meaning "and also."

NOT p	$\neg p$	$\overline{p}, {\sim}p$
p AND q	$p \wedge q$	
p XOR q	$p \oplus q$	
p OR q	$p \vee q$	
p implies q	$p \rightarrow q$	$p \Rightarrow q$, $p \supset q$
p if and only if q	$p \leftrightarrow q$	$p \Leftrightarrow q$

Table 2.1: Compound propositions. The rightmost column gives alternate forms. Precedence goes from strongest for $\neg$ to weakest for $\leftrightarrow$ (but see §2.2.1.1 for a discussion of variation in conventions for this).

Table 2.1 shows what all of this looks like when typeset nicely. Note that in some cases there is more than one way to write a compound expression. Which you choose is a matter of personal preference, but you should try to be consistent.

2.2.1.1 Precedence

The short version: for the purposes of this course, we will use the ordering in Table 2.1, which corresponds roughly to precedence in C-like programming languages. But see caveats below. Remember always that there is no shame in putting in a few extra parentheses if it makes a formula more clear.

Examples: $(\neg p \vee q \wedge r \rightarrow s \leftrightarrow t)$ is interpreted as $((((\neg p) \vee (q \wedge r)) \rightarrow s) \leftrightarrow t)$. Both OR and AND are associative, so $(p \vee q \vee r)$ is the same as $((p \vee q) \vee r)$ and as $(p \vee (q \vee r))$, and similarly $(p \wedge q \wedge r)$ is the same as $((p \wedge q) \wedge r)$ and as $(p \wedge (q \wedge r))$.

Note that this convention is not universal: many mathematicians give AND and OR equal precedence, so that the meaning of $p \wedge q \vee r$ is ambiguous without parentheses. There are good arguments for either convention. Making AND have higher precedence than OR is analogous to giving multiplication higher precedence than addition, and makes sense visually when AND is written multiplicatively (as in $pq \vee qr$ for $(p \wedge q) \vee (q \wedge r)$. Making them have the same precedence emphasizes the symmetry between the two operations, which we'll see more about later when we talk about De Morgan's laws in §2.2.3. But as with anything else in mathematics, either convention can be adopted, as long as you are clear about what you are doing and it doesn't cause annoyance to the particular community you are writing for.

There does not seem to be a standard convention for the precedence of XOR, since logicians don't use it much. There are plausible arguments for putting XOR in between AND and OR, but it's probably safest just to use parentheses.

Implication is not associative, although the convention is that it binds "to the right," so that $a \to b \to c$ is read as $a \to (b \to c)$; except for type theorists and Haskell programmers, few people ever remember this, so it is usually safest to put in the parentheses. I personally have no idea what $p \leftrightarrow q \leftrightarrow r$ means, so any expression like this should be written with parentheses as either $(p \leftrightarrow q) \leftrightarrow r$ or $p \leftrightarrow (q \leftrightarrow r)$.

2.2.2 Truth tables

To define logical operations formally, we give a **truth table**. This gives, for any combination of truth values (true or false, which as computer scientists we often write as 1 or 0) of the inputs, the truth value of the output. In this usage, truth tables are to logic what addition and multiplication tables are to arithmetic.

Here is a truth table for negation:

p	$\neg p$
0	1
1	0

And here is a truth table for the rest of the logical operators:

p	q	$p \vee q$	$p \oplus q$	$p \wedge q$	$p \to q$	$p \leftrightarrow q$
0	0	0	0	0	1	1
0	1	1	1	0	1	0
1	0	1	1	0	0	0
1	1	1	0	1	1	1

See also [Fer08, §1.1], [Ros12, §§1.1–1.2], or [Big02, §§3.1–3.3].

We can think of each row of a truth table as a model for propositional logic, since the only things we can describe in propositional logic are whether particular propositions are true or not. Constructing a truth table corresponds to generating all possible models.

This can be useful if we want to figure out when a particular proposition is true. Proving a proposition using a truth table is a simple version of **model checking**: we enumerate all possible **models** of a given collection of simple propositions, and see if what we want to prove holds in all models. This works for propositional logic because the list of models is just the list of possible combinations of truth values for all the simple propositions P,

Q, etc. We can check that each truth table we construct works by checking that the truth values each column (corresponding to some subexpression of the thing we are trying to prove) follow from the truth values in previous columns according to the rules established by the truth table defining the appropriate logical operation.

For predicate logic, model checking becomes more complicated, because a typical system of axioms is likely to have infinitely many models, many of which are likely to be infinitely large. There we will need to rely much more on proofs constructed by applying inference rules.

2.2.3 Tautologies and logical equivalence

A compound proposition that is true no matter what the truth-values of the propositions it contains is called a **tautology**. For example, $p \rightarrow p$, $p \vee \neg p$, and $\neg(p \wedge \neg p)$ are all tautologies, as can be verified by constructing truth tables. If a compound proposition is always false, it's a **contradiction**. The negation of a tautology is a contradiction and vice versa.

The most useful class of tautologies are **logical equivalences**. This is a tautology of the form $X \leftrightarrow Y$, where X and Y are compound propositions. In this case, X and Y are said to be **logically equivalent** and we can substitute one for the other in more complex propositions. We write $X \equiv Y$ if X and Y are logically equivalent.

The nice thing about logical equivalence is that is does the same thing for Boolean formulas that equality does for algebraic formulas: if we know (for example), that $p \vee \neg p$ is equivalent to 1, and $q \vee 1$ is equivalent to q, we can grind $q \vee p \vee \neg p \equiv q \vee 1 \equiv 1$ without having to do anything particularly clever. (We will need cleverness later when we prove things where the consequent isn't logically equivalent to the premise.)

To prove a logical equivalence, one either constructs a truth table to show that $X \leftrightarrow Y$ is a tautology, or transforms X to Y using previously-known logical equivalences.

Some examples:

- $p \wedge \neg p \equiv 0$: Construct a truth table

p	$\neg p$	$p \wedge \neg p$	0
0	1	0	0
1	0	0	0

  and observe that the last two columns are always equal.

- $p \vee p \equiv p$: Use the truth table

p	$p \vee p$
0	0
1	1

- $p \to q \equiv \neg p \vee q$: Again construct a truth table

p	q	$p \to q$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

- $\neg(p \vee q) \equiv \neg p \wedge \neg q$: (one of De Morgan's laws; the other is $\neg(p \wedge q) \equiv \neg p \vee \neg q$).

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	1	0	**0**
1	0	1	**0**	0	1	**0**
1	1	1	**0**	0	0	**0**

- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ (one of the distributive laws; the other is $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$).

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$
0	0	0	0	**0**	0	0	**0**
0	0	1	0	**0**	1	0	**0**
0	1	0	0	**0**	0	1	**0**
0	1	1	1	**1**	1	1	**1**
1	0	0	0	**1**	1	1	**1**
1	0	1	0	**1**	1	1	**1**
1	1	0	0	**1**	1	1	**1**
1	1	1	1	**1**	1	1	**1**

- $(p \to r) \vee (q \to r) \equiv (p \wedge q) \to r$. Now things are getting messy, so building a full truth table may take awhile. But we have take a shortcut by using logical equivalences that we've already proved (plus associativity of $\vee$):

$$\begin{aligned}
(p \rightarrow r) \vee (q \rightarrow r) &\equiv (\neg p \vee r) \vee (\neg q \vee r) && [\text{Using } p \rightarrow q \equiv \neg p \vee q \text{ twice}] \\
&\equiv \neg p \vee \neg q \vee r \vee r && [\text{Associativity and commutativity of } \vee] \\
&\equiv \neg p \vee \neg q \vee r && [p \equiv p \vee p] \\
&\equiv \neg(p \wedge q) \vee r && [\text{De Morgan's law}] \\
&\equiv (p \wedge q) \rightarrow r. && [p \rightarrow q \equiv \neg p \vee q]
\end{aligned}$$

This last equivalence is a little surprising. It shows, for example, that if somebody says "It is either the case that if you study you will graduate from Yale with distinction, or that if you join the right secret society you will graduate from Yale with distinction", then this statement (assuming we treat the or as $\vee$) is logically equivalent to "If you study and join the right secret society, then you will graduate from Yale with distinction." It is easy to get tangled up in trying to parse the first of these two propositions; translating to logical notation and simplifying using logical equivalence is a good way to simplify it.

Over the years, logicians have given names to many logical equivalences. Some of the more useful ones are summarized in Table 2.2. More complicated equivalences can often be derived from these. If that doesn't work (and you don't have too many variables to work with), you can always try writing out a truth table.

2.2.3.1 Inverses, converses, and contrapositives

The **contrapositive** of $p \rightarrow q$ is $\neg q \rightarrow \neg p$; it is logically equivalent to the original implication. For example, the contrapositive of "If I am Barack Obama then I am a Democrat" is "If I am not a Democrat then I am not Barack Obama". A **proof by contraposition** demonstrates that p implies q by assuming $\neg q$ and then proving $\neg p$; it is similar but not identical to an **indirect proof**, which assumes $\neg p$ and derives a contradiction.

The **inverse** of $p \rightarrow q$ is $\neg p \rightarrow \neg q$. So the inverse of "If you take CPSC 202, you will surely die" is "If you do not take CPSC 202, you will not surely die." There is often no connection between the truth of an implication and the truth of its inverse: "If I am Barack Obama then I am a Democrat" does not have the same truth-value as "If I am not Barack Obama then I am not a Democrat", at least according to current polling numbers.

The **converse** of $p \rightarrow q$ is $q \rightarrow p$. E.g. the converse of "If I am Barack Obama then I am a Democrat" is "If I am a Democrat then I am Barack

$\neg\neg p \equiv p$	Double negation
$\neg(p \wedge q) \equiv \neg p \vee \neg q$	De Morgan's law
$\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's law
$p \wedge q \equiv q \wedge p$	Commutativity of AND
$p \vee q \equiv q \vee p$	Commutativity of OR
$p \wedge (q \wedge r) \equiv p \wedge (q \wedge r)$	Associativity of AND
$p \vee (q \vee r) \equiv p \vee (q \vee r)$	Associativity of OR
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	AND distributes over OR
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	OR distributes over AND
$p \to q \equiv \neg p \vee q$	Equivalence of implication and OR
$p \to q \equiv \neg q \to \neg p$	Contraposition
$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$	Expansion of if and only if
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$	Inverse of if and only f
$p \leftrightarrow q \equiv q \leftrightarrow p$	Commutativity of if and only if

Table 2.2: Common logical equivalences (see also [Fer08, Theorem 1.1])

Obama." The converse of a statement is always logically equivalent to the inverse. Often in proving a biconditional (e.g., "I am Barack Obama if and only if I am a Democrat"), one proceeds by proving first the implication in one direction and then either the inverse or the converse (which are logically equivalent).

2.2.3.2 Equivalences involving true and false

Any tautology is equivalent to true; any contradiction is equivalent to false. Two important cases of this are the **law of the excluded middle**

$$P \vee \neg P \equiv 1$$

and its dual, the **law of non-contradiction**

$$P \wedge \neg P \equiv 0.$$

The law of the excluded middle is what allows us to do **case analysis**, where we prove that some proposition Q holds by showing first that P implies Q and then that $\neg P$ also implies Q.[3]

[3]Though we will use the law of the excluded middle, it has always been a little bit controversial, because it is **non-constructive**: it tells you that one of P or $\neg P$ is true, but it doesn't tell you which.

There is a variant of classical logic called **intuitionistic logic** where the law of the excluded middle does not hold. The reason for the controversy is that the law of the excluded middle is **non-constructive**: it tells you that one of P or $\neg P$ is true, but it doesn't tell you which.

Though intuititionistic logic might seem like a curiosity, there is a deep connection between computer programs and proofs in intuitionistic logic, known as the **Curry-Howard isomorphism**. The idea is that you get intuitionistic logic if you interpret

- P as an object of type P;
- $P \rightarrow Q$ as a function that takes a P as an argument and returns a Q;
- $P \wedge Q$ as an object that contains both a P and a Q (like a `struct` in C);
- $P \vee Q$ as an object that contains either a P or a Q (like a `union` in C); and
- $\neg P$ as $P \rightarrow \bot$, a function that given a P produces a special error value $\bot$ that can't otherwise be generated.

With this interpretation, many theorems of classical logic continue to hold. For example, *modus ponens* says

$$(P \wedge (P \rightarrow Q)) \rightarrow Q.$$

Seen through the Curry-Howard isomorphism, this means that there is a function that, given a P and a function that generates a Q from a P, generates a Q. For example, the following Scheme function:

$P \wedge 0 \equiv 0$	$P \vee 0 \equiv P$
$P \wedge 1 \equiv P$	$P \vee 1 \equiv 1$
$P \leftrightarrow 0 \equiv \neg P$	$P \oplus 0 \equiv P$
$P \leftrightarrow 1 \equiv P$	$P \oplus 1 \equiv \neg P$
$P \rightarrow 0 \equiv \neg P$	$0 \rightarrow P \equiv 1$
$P \rightarrow 1 \equiv 1$	$1 \rightarrow P \equiv P$

Table 2.3: Absorption laws. The first four are the most important. Note that $\wedge$, $\vee$, $\oplus$, and $\leftrightarrow$ are all commutative, so reversed variants also work.

One strategy for simplifying logical expressions is to try to apply known equivalences to generate sub-expressions that reduce to true or false via the law of the excluded middle or the law of non-contradiction. These can then be absorbed into nearby terms using various absorption laws, shown in Table 2.3.

Example Let's show that $(P \wedge (P \rightarrow Q)) \rightarrow Q$ is a tautology. (This justifies the inference rule *modus ponens*, defined below.) Working from the

```
(define (modus-ponens p p-implies q) (p-implies-q p))
```

Similarly, in a sufficiently sophisticated programming language we can show $P \rightarrow \neg\neg P$, since this expands to $P \rightarrow ((P \rightarrow \bot) \rightarrow \bot)$, and we can write a function that takes a P as its argument and returns a function that takes a $P \rightarrow \bot$ function and feeds the P to it:

```
(define (double-negation p) (lambda (p-implies-fail)
(p-implies-fail p)))
```

But we can't generally show $\neg\neg P \rightarrow P$, since there is no way to take a function of type $(P \rightarrow \bot) \rightarrow \bot$ and extract an actual example of a P from it. Nor can we expect to show $P \vee \neg P$, since this would require exhibiting either a P or a function that takes a P and produces an error, and for any particular type P we may not be able to do either.

For normal mathematical proofs, we won't bother with this, and will just assume $P \vee \neg P$ always holds.

inside out, we can compute

$$\begin{aligned}
(P \wedge (P \to Q)) \to Q &\equiv (P \wedge (\neg P \vee Q)) \to Q && \text{expand } \to \\
&\equiv ((P \wedge \neg P) \vee (P \wedge Q)) \to Q && \text{distribute } \vee \text{ over } \wedge \\
&\equiv (0 \vee (P \wedge Q)) \to Q && \text{non-contradiction} \\
&\equiv (P \wedge Q) \to Q && \text{absorption} \\
&\equiv \neg(P \wedge Q) \vee Q && \text{expand } \to \\
&\equiv (\neg P \vee \neg Q) \vee Q && \text{De Morgan's law} \\
&\equiv \neg P \vee (\neg Q \vee Q) && \text{associativity} \\
&\equiv \neg P \vee 1 && \text{excluded middle} \\
&\equiv 1 && \text{absorption}
\end{aligned}$$

In this derivation, we've labeled each step with the equivalence we used. Most of the time we would not be this verbose.

2.2.4 Normal forms

A compound proposition is in **conjuctive normal form** (**CNF** for short) if it is obtained by ANDing together ORs of one or more variables or their negations (an OR of one variable is just the variable itself). So for example P, $(P \vee Q) \wedge R$, $(P \vee Q) \wedge (Q \vee R) \wedge (\neg P)$, and $(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q \vee S \vee T \vee \neg U)$ are in CNF, but $(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \wedge Q)$, $(P \vee Q) \wedge (P \to R) \wedge (\neg P \vee Q)$, and $(P \vee (Q \wedge R)) \wedge (P \vee \neg R) \wedge (\neg P \vee Q)$ are not. Using the equivalence $P \to Q \equiv \neg P \vee Q$, De Morgan's laws, and the distributive law, it is possible to rewrite any compound proposition in CNF.

CNF formulas are particularly useful because they support **resolution** (see §2.4.1). Using the tautology $(P \vee Q) \wedge (\neg P \vee R) \to Q \vee R$, we can construct proofs from CNF formulas by looking for occurrences of some simple proposition and its negation and **resolving** them, which generates a new clause we can add to the list. For example, we can compute

$$\begin{aligned}
&(\mathbf{P} \vee Q) \wedge (P \vee \neg R) \wedge (\neg \mathbf{P} \vee Q) \wedge (\neg Q \vee R) \\
\vdash&(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q) \wedge (\neg \mathbf{Q} \vee R) \wedge \mathbf{Q} \\
\vdash&(P \vee Q) \wedge (P \vee \neg \mathbf{R}) \wedge (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge Q \wedge \mathbf{R} \\
\vdash&(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge Q \wedge R \wedge P \\
\vdash&P.
\end{aligned}$$

This style of proof is called a **resolution proof**. Because of its simplicity it is particularly well-suited for mechanical theorem provers. Such proofs can also encode traditional proofs based on *modus ponens*: the inference $P \wedge (P \rightarrow Q) \vdash Q$ can be rewritten as resolution by expanding $\rightarrow$ to get $P \wedge (\neg P \vee Q) \vdash Q$.

Similarly, a compound proposition is in **disjunctive normal form** (**DNF**) if it consists of an OR of ANDs, e.g. $(P \wedge Q) \vee (P \wedge \neg R) \vee (\neg P \wedge Q)$. Just as any compound proposition can be transformed into CNF, it can similarly be transformed into DNF.

Note that conjunctive normal forms are not unique; for example, $P \vee Q$ and $(P \wedge \neg Q) \vee (P \wedge Q) \vee (\neg P \wedge Q)$ are both in conjunctive normal form and are logically equivalent to each other. So while CNF can be handy as a way of reducing the hairiness of a formula (by eliminating nested parentheses or negation of non-variables, for example), it doesn't necessarily let us see immediately if two formulas are really the same.

2.3 Predicate logic

Using only propositional logic, we can express a simple version of a famous argument:

- Socrates is a man.
- If Socrates is a man, then Socrates is mortal.
- Therefore, Socrates is mortal.

This is an application of the inference rule called **modus ponens**, which says that from p and $p \rightarrow q$ you can deduce q. The first two statements are axioms (meaning we are given them as true without proof), and the last is the conclusion of the argument.

What if we encounter Socrates's infinitely more logical cousin Spocrates? We'd like to argue

- Spocrates is a man.
- If Spocrates is a man, then Spocrates is mortal.
- Therefore, Spocrates is mortal.

Unfortunately, the second step depends on knowing that humanity implies mortality for everybody, not just Socrates. If we are unlucky in our choice of axioms, we may not know this. What we would like is a general way to say that humanity implies mortality for everybody, but with just propositional logic, we can't write this fact down.

2.3.1 Variables and predicates

The solution is to extend our language to allow formulas that involve variables. So we might let x, y, z, etc. stand for any element of our **universe of discourse** or **domain**—essentially whatever things we happen to be talking about at the moment. We can now write statements like:

- "x is human."
- "x is the parent of y."
- "$x + 2 = x^2$."

These are not propositions because they have variables in them. Instead, they are **predicates**; statements whose truth-value depends on what concrete object takes the place of the variable. Predicates are often abbreviated by single capital letters followed by a list of **arguments**, the variables that appear in the predicate, e.g.:

- $H(x)$ = "x is human."
- $P(x, y)$ = "x is the parent of y."
- $Q(x)$ = "x + 2 = x^2."

We can also fill in specific values for the variables, e.g. $H(\text{Spocrates})$ = "Spocrates is human." If we fill in specific values for all the variables, we have a proposition again, and can talk about that proposition being true (e.g. $H(2)$ and $H(-1)$ are true) or false ($H(0)$ is false).

In **first-order logic**, which is what we will be using in this course, variables always refer to things and never to predicates: any predicate symbol is effectively a constant. There are higher-order logics that allow variables to refer to predicates, but most mathematics accomplishes the same thing by representing predicates with sets (see Chapter 3).

2.3.2 Quantifiers

What we really want is to be able to say when H or P or Q is true for many different values of their arguments. This means we have to be able to talk about the truth or falsehood of statements that include variables. To do this, we **bind** the variables using **quantifiers**, which state whether the claim we are making applies to all values of the variable (**universal quantification**), or whether it may only apply to some (**existential quantification**).

2.3.2.1 Universal quantifier

The **universal quantifier** $\forall$ (pronounced "for all") says that a statement must be true for all values of a variable within some *universe* of allowed values (which is often implicit). For example, "all humans are mortal" could be written $\forall x : \text{Human}(x) \rightarrow \text{Mortal}(x)$ and "if x is positive then $x + 1$ is positive" could be written $\forall x : x > 0 \rightarrow x + 1 > 0$.

If you want to make the universe explicit, use set membership notation, e.g. $\forall x \in Z : x > 0 \rightarrow x + 1 > 0$. This is logically equivalent to writing $\forall x : x \in Z \rightarrow (x > 0 \rightarrow x + 1 > 0)$ or to writing $\forall x : (x \in Z \wedge x > 0) \rightarrow x + 1 > 0$, but the short form makes it more clear that the intent of $x \in Z$ is to restrict the range of x.[4]

The statement $\forall x : P(x)$ is equivalent to a very large AND; for example, $\forall x \in \mathbb{N} : P(x)$ could be rewritten (if you had an infinite amount of paper) as $P(0) \wedge P(1) \wedge P(2) \wedge P(3) \wedge \dots$. Normal first-order logic doesn't allow infinite expressions like this, but it may help in visualizing what $\forall x : P(x)$ actually means. Another way of thinking about it is to imagine that x is supplied by some adversary and you are responsible for showing that $P(x)$ is true; in this sense, the universal quantifier chooses the *worst case* value of x.

2.3.2.2 Existential quantifier

The **existential quantifier** $\exists$ (pronounced "there exists") says that a statement must be true for at least one value of the variable. So "some human is mortal" becomes $\exists x : \text{Human}(x) \wedge \text{Mortal}(x)$. Note that we use AND rather than implication here; the statement $\exists x : \text{Human}(x) \rightarrow \text{Mortal}(x)$ makes the much weaker claim that "there is some thing x, such that if x is human, then x is mortal," which is true in any universe that contains an immortal purple penguin—since it isn't human, $\text{Human}(\text{penguin}) \rightarrow \text{Mortal}(\text{penguin})$ is true.

As with $\forall$, $\exists$ can be limited to an explicit universe with set membership notation, e.g., $\exists x \in Z : x = x^2$. This is equivalent to writing $\exists x : x \in Z \wedge x = x^2$.

[4]Programmers will recognize this as a form of *syntactic sugar*.

The formula $\exists x : P(x)$ is equivalent to a very large OR, so that $\exists x \in \mathbb{N} : P(x)$ could be rewritten as $P(0) \vee P(1) \vee P(2) \vee P(3) \vee \ldots$. Again, you can't generally write an expression like this if there are infinitely many terms, but it gets the idea across.

2.3.2.3 Negation and quantifiers

The following equivalences hold:

$$\neg \forall x : P(x) \equiv \exists x : \neg P(x). \neg \exists x : P(x) \qquad \equiv \forall x : \neg P(x).$$

These are essentially the quantifier version of De Morgan's laws: the first says that if you want to show that not all humans are mortal, it's equivalent to finding some human that is not mortal. The second says that to show that no human is mortal, you have to show that all humans are not mortal.

2.3.2.4 Restricting the scope of a quantifier

Sometimes we want to limit the universe over which we quantify to some restricted set, e.g., all positive integers or all baseball teams. We've previously seen how to do this using set-membership notation, but can also do this for more general predicates either explicitly using implication:

$$\forall x : x > 0 \to x - 1 \geq 0$$

or in abbreviated form by including the restriction in the quantifier expression itself:

$$\forall x > 0 : x - 1 \geq 0.$$

Similarly

$$\exists x : x > 0 \wedge x^2 = 81$$

can be written as

$$\exists x > 0 : x^2 = 81.$$

Note that constraints on $\exists$ get expressed using AND rather than implication.

The use of set membership notation to restrict a quantifier is a special case of this. Suppose we want to say that 79 is not a perfect square, by which we mean that there is no integer whose square is 79. If we are otherwise talking about real numbers (two of which happen to be square roots of 79), we can exclude the numbers we don't want by writing

$$\neg \exists x \in \mathbb{Z} : x^2 = 79$$

which is interpreted as

$$\neg\exists x : (x \in \mathbb{Z} \wedge x^2 = 79)$$

or, equivalently

$$\forall x : x \in \mathbb{Z} \rightarrow x^2 \neq 79.$$

Here $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ is the standard set of integers.

For more uses of $\in$, see Chapter 3.

2.3.2.5 Nested quantifiers

It is possible to nest quantifiers, meaning that the statement bound by a quantifier itself contains quantifiers. For example, the statement "there is no largest prime number" could be written as

$$\neg\exists x : (\text{Prime}(x) \wedge \forall y : y > x \rightarrow \neg\text{Prime}(y))$$

i.e., "there does not exist an x that is prime and any y greater than x is not prime." Or in a shorter (though not strictly equivalent) form:

$$\forall x \exists y : y > x \wedge \text{Prime}(y)$$

which we can read as "for any x there is a bigger y that is prime."

To read a statement like this, treat it as a game between the $\forall$ player and the $\exists$ player. Because the $\forall$ comes first in this statement, the for-all player gets to pick any x it likes. The exists player then picks a y to make the rest of the statement true. The statement as a whole is true if the $\exists$ player always wins the game. So if you are trying to make a statement true, you should think of the universal quantifier as the enemy (the **adversary** in algorithm analysis) who says "nya-nya: try to make *this* work, bucko!", while the existential quantifier is the friend who supplies the one working response.

As in many two-player games, it makes a difference who goes first. If we write likes(x, y) for the predicate that x likes y, the statements

$$\forall x \exists y : \text{likes}(x, y)$$

and

$$\exists y \forall x : \text{likes}(x, y)$$

mean very different things. The first says that for every person, there is somebody that that person likes: we live in a world with no complete misanthropes. The second says that there is some single person who is so immensely popular that everybody in the world likes them. The nesting of the quantifiers is what makes the difference: in $\forall x \exists y : \text{likes}(x, y)$, we are saying that no matter who we pick for x, $\exists y : \text{likes}(x, y)$ is a true statement; while in $\exists y \forall x : \text{likes}(x, y)$, we are saying that there is some y that makes $\forall x : \text{likes}(x, y)$ true.

Naturally, such games can go on for more than two steps, or allow the same player more than one move in a row. For example

$$\forall x \forall y \exists z : x^2 + y^2 = z^2$$

is a kind of two-person challenge version of the Pythagorean theorem where the universal player gets to pick x and y and the existential player has to respond with a winning z. (Whether the statement itself is true or false depends on the range of the quantifiers; it's false, for example, if x, y, and z are all natural numbers or rationals but true if they are all real or complex. Note that the universal player only needs to find one bad (x, y) pair to make it false.)

One thing to note about nested quantifiers is that we can switch the order of two universal quantifiers or two existential quantifiers, but we can't swap a universal quantifier for an existential quantifier or vice versa. So for example $\forall x \forall y : (x = y \rightarrow x + 1 = y + 1)$ is logically equivalent to $\forall y \forall x : (x = y \rightarrow y + 1 = x + 1)$, but $\forall x \exists y : y < x$ is not logically equivalent to $\exists y \forall x : y < x$. This is obvious if you think about it in terms of playing games: if I get to choose two things in a row, it doesn't really matter which order I choose them in, but if I choose something and then you respond it might make a big difference if we make you go first instead.

One measure of the complexity of a mathematical statement is how many layers of quantifiers it has, where a layer is a sequence of all-universal or all-existential quantifiers. Here's a standard mathematical definition that involves three layers of quantifiers, which is about the limit for most humans:

$$\left[\lim_{x \to \infty} f(x) = y\right] \equiv \left[\forall \epsilon > 0 : \exists N : \forall x > N : |f(x) - y| < \epsilon\right].$$

Now that we know how to read nested quantifiers, it's easy to see what the right-hand side means:

1. The adversary picks ϵ, which must be greater than 0.
2. We pick N.

3. The adversary picks x, which must be greater than N.
4. We win if $f(x)$ is within ϵ of y.

So, for example, a proof of

$$\lim_{x\to\infty} 1/x = 0$$

would follow exactly this game plan:

1. Choose some $\epsilon > 0$.
2. Let $N > 1/\epsilon$. (Note that we can make our choice depend on previous choices.)
3. Choose any $x > N$.
4. Then $x > N > 1/\epsilon > 0$, so $1/x < 1/N < \epsilon \to |1/x - 0| < \epsilon$. QED!

2.3.2.6 Examples

Here we give some more examples of translating English into statements in predicate logic.

All crows are black.	$\forall x : \text{Crow}(x) \to \text{Black}(x)$

The formula is logically equivalent to either of

$$\neg\exists x \text{Crow}(x) \wedge \neg\text{Black}(x)$$

or

$$\forall x : \neg\text{Black}(x) \to \neg\text{Crow}(x).$$

The latter is the core of a classic "paradox of induction" in philosophy: if seeing a black crow makes me think it's more likely that all crows are black, shouldn't seeing a logically equivalent non-black non-crow (e.g., a banana yellow AMC Gremlin) also make me think all non-black objects are non-crows, i.e., that all crows are black? The paradox suggests that logical equivalence works best for true/false and not so well for probabilities.

Some cows are brown.	$\exists x : \text{Cow}(x) \wedge \text{Brown}(x)$

No cows are blue.	$\neg\exists x : \text{Cow}(x) \wedge \text{Blue}(x)$

Some other equivalent versions:

$$\begin{aligned}
&\forall x : \neg(\text{Cow}(x) \wedge \text{Blue}(x) \\
&\forall x : (\neg\text{Cow}(x) \vee \neg\text{Blue}(x)) \\
&\forall x : \text{Cow}(x) \rightarrow \neg\text{Blue}(x) \\
&\forall x : \text{Blue}(x) \rightarrow \neg\text{Cow}(x).
\end{aligned}$$

All that glitters is not gold.	$\neg\forall x : \text{Glitters}(x) \rightarrow \text{Gold}(x)$

Or $\exists x : \text{Glitters}(x) \wedge \neg\text{Gold}(x)$. Note that the English syntax is a bit ambiguous: a literal translation might look like $\forall x : \text{Glitters}(x) \rightarrow \neg\text{Gold}(x)$, which is *not* logically equivalent. This is an example of how predicate logic is often more precise than natural language.

No shirt, no service.	$\forall x : \neg\text{Shir}t(x) \rightarrow \neg\text{Served}(x)$
Every event has a cause.	$\forall x \exists y : \text{Causes}(y, x)$

And a more complicated statment: Every even number greater than 2 can be expressed as the sum of two primes.

$$\forall x : (\text{Even}(x) \wedge x > 2) \rightarrow (\exists p \exists q : \text{Prime}(p) \wedge \text{Prime}(q) \wedge (x = p + q))$$

The last one is **Goldbach's conjecture**. The truth value of this statement is currently unknown.

2.3.3 Functions

A **function symbol** looks like a predicate but instead of computing a truth value it returns an object. So for example the successor function S in the Peano axioms for the natural numbers returns $x + 1$ when applied as $S(x)$. Sometimes when there is only a single argument we omit the parentheses, e.g., $Sx = S(x), SSSx = S(S(S(x)))$.

2.3.4 Equality

Often we include a special **equality** predicate $=$, written $x = y$. The interpretation of $x = y$ is that x and y are the same element of the domain. It satisfies the **reflexivity axiom** $\forall x : x = x$ and the **substitution axiom schema**:

$$\forall x \forall y : (x = y \rightarrow (Px \leftrightarrow Py))$$

where P is any predicate. This immediately gives a **substitution rule** that says $x = y, P(x) \vdash P(y)$. It's likely that almost every proof you ever wrote down in high school algebra consisted only of many applications of the substitution rule.

Example: We'll prove $\forall x \forall y : (x = y \rightarrow y = x)$ from the above axioms (this property is known as **symmetry**). Apply substitution to the predicate $Px \equiv y = x$ to get $\forall x \forall y : (x = y \rightarrow (y = x \leftrightarrow x = x))$. Use reflexivity to rewrite this as $\forall x \forall y : (x = y \rightarrow (y = x \leftrightarrow T))$ or $\forall x \forall y : (x = y \rightarrow y = x)$ as claimed.

Exercise: Prove $\forall x \forall y \forall z : (x = y \wedge y = z \rightarrow x = z)$. (This property is known as **transitivity**.)

2.3.4.1 Uniqueness

An occasionally useful abbreviation is $\exists! x P(x)$, which stands for "there exists a *unique* x such that $P(x)$." This is short for

$$(\exists x P(x)) \wedge (\forall x \forall y : P(x) \wedge P(y) \rightarrow x = y).$$

An example is $\exists! x x + 1 = 12$. To prove this we'd have to show not only that there is some x for which $x + 1 = 12$ (11 comes to mind), but that if we have any two values x and y such that $x + 1 = 12$ and $y + 1 = 12$, then $x = y$ (this is not hard to do). So the exclamation point encodes quite a bit of extra work, which is why we usually hope that $\exists x x + 1 = 12$ is good enough.

2.3.5 Models

In propositional logic, we can build truth tables that describe all possible settings of the truth-values of the literals. In predicate logic, the analogous concept to an assignment of truth-values is a **structure**. A structure consists of a set of objects or elements (built using set theory, as described in Chapter 3), together with a description of which elements fill in for the constant symbols, which predicates hold for which elements, and what the value of each function symbol is when applied to each possible list of arguments (note that this depends on knowing what constant, predicate, and function symbols are available—this information is called the **signature** of the structure). A structure is a **model** of a particular **theory** (set of statements), if each statement in the theory is true in the model.

In general we can't hope to find all possible models of a given theory. But models are useful for two purposes: if we can find some model of a particular theory, then the existence of this model demonstrates that the theory is consistent; and if we can find a model of the theory in which some additional statement S doesn't hold, then we can demonstrate that there is no way to prove S from the theory (i.e. it is not the case that $T \vdash S$, where T is the list of axioms that define the theory).

2.3.5.1 Examples

- Consider the axiom $\neg\exists x$. This axiom has exactly one model (it's empty).
- Now consider the axiom $\exists! x$. This axiom also has exactly one model (with one element).
- We can enforce exactly k elements with one rather long axiom, e.g. for $k = 3$ do $\exists x_1 \exists x_2 \exists x_3 \forall y : y = x_1 \vee y = x_2 \vee y = x_3$. In the absence of any special symbols, a structure of 3 undifferentiated elements is the unique model of this axiom.
- Suppose we add a predicate P and consider the axiom $\exists x Px$. Now we have many models: take any nonempty model you like, and let P be true of at least one of its elements. If we take a model with two elements a and b, with Pa and $\neg Pb$, we get that $\exists x Px$ is not enough to prove $\forall x Px$, since the later statement isn't true in this model.
- Now let's bring in a function symbol S and constant symbol 0. Consider a stripped-down version of the Peano axioms that consists of just the axiom $\forall x \forall y : Sx = Sy \to x = y$. Both the natural numbers $\mathbb{N}$ and the integers $\mathbb{Z}$ are a model for this axiom, as is the set $\mathbb{Z}_m$ of integers mod m for any m (see §8.4). In each case each element has a unique predecessor, which is what the axiom demands. If we throw in the first Peano axiom $\forall x : Sx \neq 0$, we eliminate $\mathbb{Z}$ and $\mathbb{Z}_m$ because in each of these models 0 is a successor of some element. But we don't eliminate a model that consists of two copies of $\mathbb{N}$ sitting next to each other (only one of which contains the "real" 0), or even a model that consists of one copy of $\mathbb{N}$ (to make 0 happy) plus any number of copies of $\mathbb{Z}$ and $\mathbb{Z}_m$.

- A practical example: The family tree of the kings of France is a model of the theory containing the two axioms $\forall x \forall y \forall z \text{Parent}(x, y) \wedge \text{Parent}(y, z) \rightarrow \text{GrandParent}(x, z)$ and $\forall x \forall y \text{Parent}(x, y) \rightarrow \neg\text{Parent}(y, x)$. But this set of axioms could use some work, since it still allows for the possibility that $\text{Parent}(x, y)$ and $\text{GrandParent}(y, x)$ are both true.

2.4 Proofs

A proof is a way to derive statements from other statements. It starts with **axioms** (statements that are assumed in the current context always to be true), **theorems** or **lemmas** (statements that were proved already; the difference between a theorem and a lemma is whether it is intended as a final result or an intermediate tool), and **premises** P (assumptions we are making for the purpose of seeing what consequences they have), and uses **inference rules** to derive Q. The axioms, theorems, and premises are in a sense the starting position of a game whose rules are given by the inference rules. The goal of the game is to apply the inference rules until Q pops out. We refer to anything that isn't proved in the proof itself (i.e., an axiom, theorem, lemma, or premise) as a **hypothesis**; the result Q is the **conclusion**.

When a proof exists of Q from some premises $P_1, P_2, \ldots$, we say that Q is **deducible** or **provable** from $P_1, P_2, \ldots$, which is written as

$$P_1, P_2, \ldots \vdash Q.$$

If we can prove Q directly from our inference rules without making any assumptions, we may write

$$\vdash Q$$

The **turnstile** symbol $\vdash$ has the specific meaning that we can derive the conclusion Q by applying inference rules to the premises. This is not quite the same thing as saying $P \rightarrow Q$. If our inference rules are particularly weak, it may be that $P \rightarrow Q$ is true but we can't prove Q starting with P. Conversely, if our inference rules are too strong (maybe they can prove anything, even things that aren't true) we might have $P \vdash Q$ but $P \rightarrow Q$ is false.

For propositions, most of the time we will use inference rules that are just right, meaning that $P \vdash Q$ implies $P \rightarrow Q$ (**soundness**) and $P \rightarrow Q$ implies $P \vdash Q$ (**completeness**). Here the distinction between $\vdash$ and $\rightarrow$ is then whether we want to talk about the existence of a proof (the first case) or about the logical relation between two statements (the second). Things get a little more complicated with statements involving predicates; in this case there are **incompleteness theorems** that say that sufficiently powerful sets of axioms have consequences that can't be proven unless the theory is inconsistent.

2.4.1 Inference Rules

Inference rules let us construct **valid** arguments, which have the useful property that if their premises are true, their conclusions are also true.

The main source of inference rules is tautologies of the form $P_1 \wedge P_2 \ldots \rightarrow Q$; given such a tautology, there is a corresponding inference rule that allows us to assert Q once we have $P_1, P_2, \ldots$ (either because each P_i is an axiom/theorem/premise or because we proved it already while doing the proof). The most important inference rule is **modus ponens**, based on the tautology $(p \wedge (p \rightarrow q)) \rightarrow q$; this lets us, for example, write the following famous argument:[5]

1. If it doesn't fit, you must acquit. [Axiom]
2. It doesn't fit. [Premise]
3. You must acquit. [Modus ponens applied to 1+2]

There are many named inference rules in classical propositional logic. We'll list some of them below. You don't need to remember the names of anything except modus ponens, and most of the rules are pretty much straightforward applications of modus ponens plus some convenient tautology that can be proved by truth tables or stock logical equivalences. (For example, the "addition" rule below is just the result of applying modus ponens to p and the tautology $p \rightarrow (p \vee q)$.)

Inference rules are often written by putting the premises above a horizontal line and the conclusion below. In text, the horizontal line is often replaced by the symbol $\vdash$, which means exactly the same thing. Premises are listed on the left-hand side separated by commas, and the conclusion is

[5]OK, maybe not as famous as it once was.

placed on the right. We can then write

$p \vdash p \vee q.$	Addition
$p \wedge q \vdash p.$	Simplification
$p, q \vdash p \wedge q.$	Conjunction
$p, p \rightarrow q \vdash q.$	Modus ponens
$\neg q, p \rightarrow q \vdash \neg p.$	Modus tollens
$p \rightarrow q, q \rightarrow r \vdash p \rightarrow r.$	Hypothetical syllogism
$p \vee q, \neg p \vdash q.$	Disjunctive syllogism
$p \vee q, \neg p \vee r \vdash q \vee r.$	Resolution

Of these rules, addition, simplification, and conjunction are mostly used to pack and unpack pieces of arguments. Modus ponens "the method of affirming" (and its reversed cousin modus tollens "the method of denying") let us apply implications. You don't need to remember modus tollens if you can remember the contraposition rule $(p \rightarrow q) \equiv (\neg q \rightarrow \neg p)$. Hypothetical syllogism just says that implication is transitive; it lets you paste together implications if the conclusion of one matches the premise of the other. Disjunctive syllogism is again a disguised version of modus ponens (via the logical equivalence $(p \vee q) \equiv (\neg p \rightarrow q)$); you don't need to remember it if you can remember this equivalence. Resolution is almost never used by humans but is very popular with computer theorem provers.

An argument is **valid** if the conclusion is true whenever the hypotheses are true. Any proof constructed using the inference rules is valid. It does not necessarily follow that the conclusion is true; it could be that one or more of the hypotheses is false:

1. If you give a mouse a cookie, he's going to ask for a glass of milk. [Axiom]
2. If he asks for a glass of milk, he will want a straw. [Axiom]
3. You gave a mouse a cookie. [Premise]
4. He asks for a glass of milk. [Modus ponens applied to 1 and 3.]
5. He will want a straw. [Modus ponens applied to 2 and 4.]

Will the mouse want a straw? No: Mice can't ask for glasses of milk, so Axiom 1 is false.

2.4.2 Proofs, implication, and natural deduction

Recall that $P \vdash Q$ means there is a proof of Q by applying inference rules to P, while $P \rightarrow Q$ says that Q holds whenever P does. These are not the same thing: provability ($\vdash$) is outside the theory (it's a statement about whether a proof exists or not) while implication ($\rightarrow$) is inside (it's a logical connective for making compound propositions). But most of the time they mean almost the same thing.

For example, suppose that $P \rightarrow Q$ is provable without any assumptions:

$$\vdash P \rightarrow Q.$$

Since we can always ignore extra premises, we get

$$P \vdash P \rightarrow Q$$

and thus

$$P \vdash P, P \rightarrow Q,$$

which gives

$$P \vdash Q$$

by applying modus ponens to the right-hand side.

So we can go from $\vdash P \rightarrow Q$ to $P \vdash Q$.

This means that provability is in a sense weaker than implication: it holds (assuming modus ponens) whenever implication does. But we usually don't use this fact much, since $P \rightarrow Q$ is a much more useful statement than $P \vdash Q$. Can we go the other way?

2.4.2.1 The Deduction Theorem

Yes, using the **Deduction Theorem**.

Often we want to package the result of a proof as a Theorem (a proven statement that is an end in itself) or Lemma (a proven statement that is intended mostly to be used in other proofs). Typically a proof shows that, given some base assumptions Γ, if certain premises $P_1, P_2, \ldots P_n$ hold, then some conclusion Q holds (with various axioms or previously-established theorems assumed to be true from context). To use this result later, it is useful to be able to package it as an implication $P_1 \wedge P_2 \wedge \dots P_n \rightarrow Q$. In other words, we want to go from

$$\Gamma, P_1, P_2, \dots, P_n \vdash Q$$

to

$$\Gamma \vdash (P_1 \wedge P_2 \wedge \dots \wedge P_n) \rightarrow Q.$$

The statement that we can do this, for a given collection of inference rules, is the Deduction Theorem:

Theorem 2.4.1 (Deduction Theorem)**.** *If there is a proof of Q from premises $\Gamma, P_1, P_2, \dots, P_n$, then there is a proof of $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$ from Γ alone.*

The actual proof of the theorem depends on the particular set of inference rules we start with, but the basic idea is that there exists a mechanical procedure for extracting a proof of the implication from the proof of Q assuming P_1 etc.

Caveat: In predicate logic, the deduction theorem only applies if none of the premises contain any free variables (which are variables that aren't bound by a containing quantifier). Usually you won't run into this, but there are some bad cases that arise without this restriction.

2.5 Natural deduction

In practice, we usually don't refer to the Deduction Theorem directly, and instead adopt a new inference rule:

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \rightarrow Q} \qquad (\rightarrow I)$$

This says that if we can prove Q using assumptions Γ and P, then we can prove $P \rightarrow Q$ using just Γ. Note that the horizontal line acts like a higher-order version of $\vdash$; it lets us combine one or more proofs into a new, bigger proof.

This style of inference rule, where we explicitly track what assumptions go into a particular result, is known as **natural deduction**. The natural deduction approach was invented by Gentzen [Gen35a, Gen35b] as a way to make inference rules more closely match actual mathematical proof-writing practice than the modus-ponens-only approach that modern logicians had been using up to that point.[6]

The particular rule $(\to I)$ is called introducing implication. There is a corresponding rule for eliminating implication that is essentially just modus ponens:

$$\frac{\Gamma \vdash P \to Q \quad \Gamma \vdash P}{\Gamma \vdash Q} \tag{$\to E$}$$

If we want to be really systematic about things, we can rewrite most of our standard inference rules as introduction and elimination rules for particular operators. See Table 2.4.

2.5.1 Inference rules for equality

The equality predicate is special, in that it allows for the **substitution rule**

$$x = y, P(x) \vdash P(y).$$

If we don't want to include the substitution rule as an inference rule, we could instead represent it as an axiom schema:

$$\forall x : \forall y : ((x = y \wedge P(x)) \to P(y)).$$

But this is messier.

We can also assert $x = x$ directly:

$$\vdash x = x$$

2.5.2 Inference rules for quantified statements

Universal generalization If y is a variable that does not appear in Γ, then

$$\frac{\Gamma \vdash P(y)}{\Gamma \vdash \forall x : P(x)}$$

This says that if we can prove that some property holds for a "generic" y, without using any particular properties of y, then in fact the property holds for all possible x.

In a written proof, this will usually be signaled by starting with something like "Let y be an arbitrary [member of some universe]". For

[6]See `http://plato.stanford.edu/entries/proof-theory-development/` for a more detailed history of the development of proof theory in general and [Pel99] for a discussion of how different versions of proof theory have been adopted in textbooks.

$$\frac{\Gamma \vdash P}{\Gamma \vdash \neg\neg P} \qquad (\neg I)$$

$$\frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P} \qquad (\neg E)$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q} \qquad (\wedge I)$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P} \qquad (\wedge E_1)$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q} \qquad (\wedge E_2)$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q} \qquad (\vee I_1)$$

$$\frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q} \qquad (\vee I_2)$$

$$\frac{\Gamma \vdash P \vee Q \quad \Gamma \vdash \neg Q}{\Gamma \vdash P} \qquad (\vee E_1)$$

$$\frac{\Gamma \vdash P \vee Q \quad \Gamma \vdash \neg P}{\Gamma \vdash Q} \qquad (\vee E_2)$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q} \qquad (\to I)$$

$$\frac{\Gamma \vdash P \to Q \quad \Gamma \vdash P}{\Gamma \vdash Q} \qquad (\to E_1)$$

$$\frac{\Gamma \vdash P \to Q \quad \Gamma \vdash \neg Q}{\Gamma \vdash \neg P} \qquad (\to E_2)$$

Table 2.4: Natural deduction: introduction and elimination rules

example: Suppose we want to show that there is no biggest natural number, i.e. that $\forall n \in \mathbb{N} : \exists n' \in \mathbb{N} : n' > n$. Proof: Let n be any element of $\mathbb{N}$. Let $n = n + 1$. Then $n' > n$. (Note: there is also an instance of existential generalization here.)

Universal instantiation In the other direction, we have

$$\forall x : Q(x) \vdash Q(c).$$

Here we go from a general statement about all possible values x to a statement about a particular value. Typical use: Given that all humans are mortal, it follows that Spocrates is mortal.

Existential generalization This is essentially the reverse of universal instantiation: it says that, if c is some particular object, we get

$$Q(c) \vdash \exists x : Q(x).$$

The idea is that to show that $Q(x)$ holds for at least one x, we can point to c as a specific example of an object for which Q holds. The corresponding style of proof is called a **proof by construction** or **proof by example**.

For example: We are asked to prove that there exists an even prime number. Look at 2: it's an even prime number. QED.

Not all proofs of existential statements are **constructive**, in the sense of identifying a single object that makes the existential statement true. An example is a well-known **non-constructive** proof that there are irrational numbers a and b for which a^b is rational. The non-constructive proof is to consider $\sqrt{2}^{\sqrt{2}}$. If this number is rational, it's an example of the claim; if not, $\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^2 = 2$ works.[7] Non-constructive proofs are generally not as useful as constructive proofs, because the example used in a constructive proof may have additional useful properties in other contexts.

Existential instantiation $\exists x : Q(x) \vdash Q(c)$ for some c, where c is a new name that hasn't previously been used (this is similar to the requirement for universal generalization, except now the new name is on the right-hand side).

[7]For this particular claim, there is also a constructive proof: $\sqrt{2}^{\log_2 9} = 3$ [Sch01].

The idea here is that we are going to give a name to some c that satisfies $Q(c)$, and we know that we can get away this because $\exists x : Q(x)$ says that some such thing exists.[8]

In a proof, this is usually signaled by "let x be… " or "call it x." For example: Suppose we know that there exists a prime number greater than 7. Let p be some such prime number greater than 7.

2.6 Proof techniques

A proof technique is a template for how to go about proving particular classes of statements: this template guides you in the choice of inference rules (or other proof techniques) to write the actual proof. This doesn't substitute entirely for creativity (there is no efficient mechanical procedure for generating even short proofs unless $\mathbf{P} = \mathbf{NP}$), but it can give you some hints for how to get started.

Table 2.5 gives techniques for trying to prove $A \to B$ for particular statements A and B. The techniques are mostly classified by the structure of B. Before applying each technique, it may help to expand any definitions that appear in A or B.[9]

If you want to prove $A \leftrightarrow B$, the usual approach is to prove $A \to B$ and $A \leftarrow B$ separately. Proving $A \to B$ and $\neg A \to \neg B$ also works (because of contraposition).

Strategy	When	Assume	Conclude	What to do/why it works

[8]This is actually a fairly painful idea to formalize. One version in pure first-order logic is the axiom

$$((\forall x : (Q(x) \to P)) \wedge \exists y : Q(y)) \to P$$

. Nobody but a logician would worry about this.

[9]These strategies are largely drawn from [Sol05], particularly the summary table in the appendix, which is the source of the order and organization of the table and the names of most of the techniques. Ferland [Fer08] has an entire chapter on proof techniques of various sorts. Rosen [Ros12] describes proof strategies in §§1.5–1.7 and Biggs [Big02] describes various proof techniques in Chapters 1, 3, and 4; both descriptions are a bit less systematic than the ones in Solow or Ferland, but also include a variety of specific techniques that are worth looking at.

Direct proof	Try it first	A	B	Apply inference rules to work forward from A and backward from B; when you meet in the middle, pretend that you were working forward from A all along.
Contraposition	$B = \neg Q$	$\neg B$	$\neg A$	Apply any other technique to show $\neg B \rightarrow \neg A$ and then apply the contraposition rule. Sometimes called an *indirect proof* although the term *indirect proof* is often used instead for proofs by contradiction (see below).

Contradiction	When $B = \neg Q$, or when you are stuck trying the other techniques.	$A \wedge \neg B$	False	Apply previous methods to prove both P and $\neg P$ for some P. Note: this can be a little dangerous, because you are assuming something that is (probably) not true, and it can be hard to detect as you prove further false statements whether the reason they are false is that they follow from your false assumption, or because you made a mistake. Direct or contraposition proofs are preferred because they don't have this problem.
Construction	$B = \exists x P(x)$	A	$P(c)$ for some specific object c.	Pick a likely-looking c and prove that $P(c)$ holds.
Counterexample	$B = \neg\forall x P(x)$	A	$\neg P(c)$ for some specific object c.	Pick a likely-looking c and show that $\neg P(c)$ holds. This is identical to a proof by construction, except that we are proving $\exists x \neg P(x)$, which is equivalent to $\neg\forall x P(x)$.

Choose	$B = \forall x(P(x) \to Q(x))$	A, $P(c)$, where c is chosen arbitrarily.	$Q(c)$	Choose some c and assume A and $P(c)$. Prove $Q(c)$. Note: c is a placeholder here. If $P(c)$ is "c is even" you can write "Let c be even" but you can't write "Let $c = 12$", since in the latter case you are assuming extra facts about c.
Instantiation	$A = \forall x P(x)$	A	B	Pick some particular c and prove that $P(c)$ $\to$B. Here you *can* get away with saying "Let $c = 12$." (If $c = 12$ makes B true).
Elimination	$B = C \vee D$	$A \wedge \neg C$	D	The reason this works is that $A \wedge \neg C \to D$ is equivalent to $\neg(A \wedge \neg C) \to D \equiv \neg A \vee C \vee D \equiv A \to (C \vee D)$. Of course, it works equally well if you start with $A \wedge \neg D$ and prove C.
Case analysis	$A = C \vee D$	C, D	B	Here you write two separate proofs: one that assumes C and proves B, and one that assumes D and proves B. A special case is when $D = \neg C$. You can also consider more cases, as long as A implies at least one of the cases holds.

Induction	$B = \forall x \in \mathbb{N} P(x)$	A	$P(0)$ and $\forall x \in \mathbb{N} : (P(x) \rightarrow P(x+1))$.	If $P(0)$ holds, and $P(x)$ implies $P(x+1)$ for all x, then for any specific natural number n we can consider constructing a sequence of proofs $P(0) \rightarrow P(1) \rightarrow P(2) \rightarrow \ldots \rightarrow P(n)$. (This is actually a defining property of the natural numbers; see the example below.)

Table 2.5: Proof techniques (adapted from [Sol05]

Some others that are mentioned in Solow [Sol05]: Direct Uniqueness, Indirect Uniqueness, various max/min arguments. We will cover induction proofs later.

Chapter 3

Set theory

Set theory is the dominant foundation for mathematics. The idea is that everything else in mathematics—numbers, functions, etc.—can be written in terms of sets, so that if you have a consistent description of how sets behave, then you have a consistent description of how everything built on top of them behaves. If predicate logic is the machine code of mathematics, set theory would be assembly language.

3.1 Naive set theory

Naive set theory is the informal version of set theory that corresponds to our intuitions about sets as unordered collections of objects (called **elements**) with no duplicates. A set can be written explicitly by listing its elements using curly braces:

- $\{\}$ = the **empty set** $\emptyset$, which has no elements.
- $\{\text{Moe}, \text{Curly}, \text{Larry}\}$ = the **Three Stooges**.
- $\{0, 1, 2, \ldots\} = \mathbb{N}$, the **natural numbers**. Note that we are relying on the reader guessing correctly how to continue the sequence here.
- $\{\{\}, \{0\}, \{1\}, \{0, 1\}, \{0, 1, 2\}, 7\}$ = a set of sets of natural numbers, plus a stray natural number that is directly an element of the outer set.

Membership in a set is written using the $\in$ symbol (pronounced "is an element of," "is a member of," or just "is in"). So we can write Moe $\in$ the

Three Stooges or $4 \in \mathbb{N}$. We can also write $\notin$ for "is not an element of", as in Moe $\notin \mathbb{N}$.

A fundamental axiom in set theory (the **Axiom of Extensionality**; see §3.4) is that the only distinguishing property of a set is its list of members: if two sets have the same members, they are the same set.

For nested sets like $\{\{1\}\}$, $\in$ represents only direct membership: the set $\{\{1\}\}$ only has one element, $\{1\}$, so $1 \notin \{\{1\}\}$. This can be confusing if you think of $\in$ as representing the English "is in," because if I put my lunch in my lunchbox and put my lunchbox in my backpack, then my lunch is in my backpack. But my lunch is *not* an element of $\{\{\text{my lunch}\}, \text{my textbook}, \text{my slingshot}\}$. In general, $\in$ is not transitive (see §9.3)—it doesn't behave like $\leq$ unless there is something very unusual about the set you are applying it to—and there is no particular notation for being a deeply-buried element of an element of an element (etc.) of some set.

In addition to listing the elements of a set explicitly, we can also define a set by **set comprehension**, where we give a rule for how to generate all of its elements. This is pretty much the only way to define an infinite set without relying on guessing, but can be used for sets of any size. Set comprehension is usually written using **set-builder notation**, as in the following examples:

- $\{x \mid x \in \mathbb{N} \wedge x > 1 \wedge (\forall y \in \mathbb{N} : \forall z \in \mathbb{N} : yz = x \rightarrow y = 1 \vee z = 1)\} =$ the prime numbers.
- $\{2x \mid x \in \mathbb{N}\}$ = the even numbers.
- $\{x \mid x \in \mathbb{N} \wedge x < 12\} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$.

Sometimes the original set that an element has to be drawn from is put on the left-hand side of the vertical bar:

- $\{n \in \mathbb{N} \mid \exists x, y, z \in \mathbb{N} \setminus \{0\} : x^n + y^n = z^n\}$. This is a fancy name for $\{1, 2\}$, but this fact is not obvious [Wil95].

Using set comprehension, we can see that every set in naive set theory is equivalent to some predicate. Given a set S, the corresponding predicate is $x \in S$, and given a predicate P, the corresponding set is $\{x \mid Px\}$. But watch out for **Russell's paradox**: what is $\{S \mid S \notin S\}$?

3.2 Operations on sets

If we think of sets as representing predicates, each logical connective gives rise to a corresponding operation on sets:

- $A \cup B = \{x \mid x \in A \vee x \in B\}$. The **union** of A and B.
- $A \cap B = \{x \mid x \in A \wedge x \in B\}$. The **intersection** of A and B.
- $A \setminus B = \{x \mid x \in A \wedge x \notin B\}$. The **set difference** of A and B.
- $A \triangle B = \{x \mid x \in A \oplus x \in B\}$. The **symmetric difference** of A and B.

(Of these, union and intersection are the most important in practice.)

Corresponding to implication is the notion of a **subset**:

- $A \subseteq B$ ("A is a subset of B") if and only if $\forall x : x \in A \rightarrow x \in B$.

Sometimes one says A is **contained in** B if $A \subseteq B$. This is one of two senses in which A can be "in" B—it is also possible that A is in fact an element of B ($A \in B$). For example, the set $A = \{12\}$ is an element of the set $B = \{\text{Moe}, \text{Larry}, \text{Curly}, \{12\}\}$, but A is not a subset of B, because A's element 12 is not an element of B. Usually we will try to reserve "is in" for $\in$ and "is contained in" for $\subseteq$, but it's safest to use the symbols (or "is an element/subset of") to avoid any possibility of ambiguity.

Finally we have the set-theoretic equivalent of negation:

- $\bar{A} = \{x \mid x \notin A\}$. The set $\bar{A}$ is known as the **complement** of A.

If we allow complements, we are necessarily working inside some fixed **universe**, since the complement $U = \bar{\emptyset}$ of the empty set contains all possible objects. This raises the issue of where the universe comes from. One approach is to assume that we've already fixed some universe that we understand (e.g. $\mathbb{N}$), but then we run into trouble if we want to work with different classes of objects at the same time. The set theory used in most of mathematics is defined by a collection of axioms that allow us to construct, essentially from scratch, a universe big enough to hold all of mathematics without apparent contradictions while avoiding the paradoxes that may arise in naive set theory. However, one consequence of this construction is that the universe (a) much bigger than anything we might ever use, and (b) *not* a set, making complements not very useful. The usual solution to this is to replace complements with explicit set differences: $U \setminus A$ for some specific universe U instead of $\bar{A}$.

3.3 Proving things about sets

We have three predicates so far in set theory, so there are essentially three positive things we could try to prove about sets:

1. Given x and S, show $x \in S$. This requires looking at the definition of S to see if x satisfies its requirements, and the exact structure of the proof will depend on what the definition of S is.

2. Given S and T, show $S \subseteq T$. Expanding the definition of subset, this means we have to show that every x in S is also in T. So a typical proof will pick an arbitrary x in S and show that it must also be an element of T. This will involve unpacking the definition of S and using its properties to show that x satisfies the definition of T.

3. Given S and T, show $S = T$. Typically we do this by showing $S \subseteq T$ and $T \subseteq S$ separately. The first shows that $\forall x : x \in S \rightarrow x \in T$; the second shows that $\forall x : x \in T \rightarrow x \in S$. Together, $x \in S \rightarrow x \in T$ and $x \in T \rightarrow x \in S$ gives $x \in S \leftrightarrow x \in T$, which is what we need for equality.

There are also the corresponding negative statements:

1. For $x \notin S$, use the definition of S as before.

2. For $S \not\subseteq T$, we only need a counterexample: pick any one element of S and show that it's not an element of T.

3. For $S \neq T$, prove one of $S \not\subseteq T$ or $T \not\subseteq S$.

Note that because $S \not\subseteq T$ and $S \neq T$ are existential statements rather than universal ones, they tend to have simpler proofs.

Here are some examples, which we'll package up as a lemma:

Lemma 3.3.1. *The following statements hold for all sets S and T, and all predicates P:*

$$S \cap T \subseteq S \tag{3.3.1}$$

$$S \cup T \supseteq S \tag{3.3.2}$$

$$\{x \in S \mid P(x)\} \subseteq S \tag{3.3.3}$$

$$S = (S \cap T) \cup (S \setminus T) \tag{3.3.4}$$

Proof.

- (3.3.1) Let x be in $S \cap T$. Then $x \in S$ and $x \in T$, from the definition of $S \cap T$. It follows that $x \in S$. Since x was arbitrary, we have that for all x in $S \cap T$, x is also in T; in other words, $S \cap T \subseteq T$.

- (3.3.2). Let x be in S.[1] Then $x \in S \vee x \in T$ is true, giving $x \in S \cup T$.

- (3.3.3) Let x be in $\{x \in S \mid P(x)\}$. Then, by the definition of set comprehension, $x \in S$ and $P(x)$. We don't care about $P(x)$, so we drop it to just get $x \in S$.

- (3.3.4). This is a little messy, but we can solve it by breaking it down into smaller problems.

  First, we show that $S \subseteq (S \setminus T) \cup (S \cap T)$. Let x be an element of S. There are two cases:

  1. If $x \in T$, then $x \in (S \cap T)$.
  2. If $x \notin T$, then $x \in (S \setminus T)$.

  In either case, we have shown that x is in $(S \cap T) \cup (S \setminus T)$. This gives $S \subseteq (S \cap T) \cup (S \setminus T)$.

  Conversely, we show that $(S \setminus T) \cup (S \cap T) \subseteq S$. Suppose that $x \in (S \setminus T) \cup (S \cap T)$. Again we have two cases:

  1. If $x \in (S \setminus T)$, then $x \in S$ and $x \notin T$.
  2. If $x \in (S \cap T)$, then $x \in S$ and $x \in T$.

  In either case, $x \in S$.

  Since we've shown that both the left-hand and right-hand sides of (3.3.4) are subsets of each other, they must be equal.

$\square$

Using similar arguments, we can show that properties of $\wedge$ and $\vee$ that don't involve negation carry over to $\cap$ and $\cup$ in the obvious way. For example, both operations are commutative and associative, and each distributes over the other.

[1]Note that we are starting with S here because we are really showing the equivalent statement $S \subseteq S \cup T$.

3.4 Axiomatic set theory

The problem with naive set theory is that unrestricted set comprehension is too strong, leading to contradictions. **Axiomatic set theory** fixes this problem by being more restrictive about what sets one can form. The axioms most commonly used are known as **Zermelo-Fraenkel set theory with choice** or **ZFC**. We'll describe the axioms of ZFC below, but in practice you mostly just need to know what constructions you can get away with.

The short version is that you can construct sets by (a) listing their members, (b) taking the union of other sets, (c) taking the set of all subsets of a set, or (d) using some predicate to pick out elements or subsets of some set.[2] The starting points for this process are the empty set $\emptyset$ and the set $\mathbb{N}$ of all natural numbers (suitably encoded as sets). If you can't construct a set in this way (like the Russell's Paradox set), odds are that it isn't a set.

These properties follow from the more useful axioms of ZFC:

Extensionality Any two sets with the same elements are equal.[3]

Existence The empty set $\emptyset$ is a set.[4]

Pairing Given sets x and y, $\{x, y\}$ is a set.[5]

Union For any set of sets $S = \{x, y, z, \dots\}$, the set $\bigcup S = x \cup y \cup z \cup \dots$ exists.[6]

Power set For any set S, the power set $\mathcal{P}(S) = \{A \mid A \subseteq S\}$ exists.[7]

Specification For any set S and any predicate P, the set $\{x \in S \mid P(x)\}$ exists.[8] This is called **restricted comprehension**, and is an **axiom schema** instead of an axiom, since it generates an infinite list of axioms, one for each possible P. Limiting ourselves to constructing subsets of existing sets avoids Russell's Paradox, because we can't construct $S = \{x \mid x \notin x\}$. Instead, we can try to construct $S = \{x \in T \mid x \notin x\}$, but we'll find that S isn't an element of T, so it doesn't contain itself but also doesn't create a contradiction.

[2]Technically this only gives us Z, a weaker set theory than ZFC that omits Replacement (Fraenkel's contribution) and Choice.

[3]$\forall x : \forall y : (x = y) \leftrightarrow (\forall z : z \in x \leftrightarrow z \in y)$.

[4]$\exists x : \forall y : y \notin x$.

[5]$\forall x : \forall y : \exists z : \forall q : q \in z \leftrightarrow q = x \vee q = y$.

[6]$\forall x : \exists y : \forall z : z \in y \leftrightarrow (\exists q : z \in q \wedge q \in x)$.

[7]$\forall x : \exists y : \forall z : z \in y \leftrightarrow z \subseteq x$.

[8]$\forall x : \exists y : \forall z : z \in y \leftrightarrow z \in x \wedge P(z)$.

Infinity There is a set that has $\emptyset$ as a member and also has $x \cup \{x\}$ whenever it has x. This gives an encoding of $\mathbb{N}$.[9] Here $\emptyset$ represents 0 and $x \cup \{x\}$ represents $x + 1$. This effectively defines each number as the set of all smaller numbers, e.g. $3 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$. Without this axiom, we only get finite sets. (Technical note: the set whose existence is given by the Axiom of Infinity may also contain some extra elements, but we can strip them out—with some effort—using Specification.)

There are three other axioms that don't come up much in computer science:

Foundation Every nonempty set A contains a set B with $A \cap B = \emptyset$.[10] This rather technical axiom prevents various weird sets, such as sets that contain themselves or infinite descending chains $A_0 \ni A_1 \ni A_2 \ni \dots$. Without it, we can't do induction arguments once we get beyond $\mathbb{N}$.

Replacement If S is a set, and $R(x, y)$ is a predicate with the property that $\forall x : \exists! y : R(x, y)$, then $\{y \mid \exists x \in SR(x, y)\}$ is a set.[11] Like comprehension, replacement is an axiom schema. Mostly used to construct astonishingly huge infinite sets.

Choice For any set of nonempty sets S there is a function f that assigns to each x in S some $f(x) \in x$. This axiom is unpopular in some circles because it is **non-constructive**: it tells you that f exists, but it doesn't give an actual definition of f. But it's too useful to throw out.

Like everything else in mathematics, the particular system of axioms we ended up with is a function of the history, and there are other axioms that could have been included but weren't. Some of the practical reasons for including some axioms but not others are described in a pair of classic papers by Maddy [Mad88a, Mad88b].

3.5 Cartesian products, relations, and functions

Sets are unordered: the set $\{a, b\}$ is the same as the set $\{b, a\}$. Sometimes it is useful to consider **ordered pairs** (a, b), where we can tell which element comes first and which comes second. These can be encoded as sets using the

[9] $\exists x : \emptyset \in x \wedge \forall y \in x : y \cup \{y\} \in x$.

[10] $\forall x \neq \emptyset : \exists y \in x : x \cap y = \emptyset$.

[11] $(\forall x : \exists! y : R(x, y)) \rightarrow \forall z : \exists q : \forall r : r \in q \leftrightarrow (\exists s \in z : R(s, r))$.

rule $(a,b) = \{\{a\},\{a,b\}\}$, which was first proposed by Kuratowski [Kur21, Definition V].[12]

Given sets A and B, their **Cartesian product** $A \times B$ is the set $\{(x,y) \mid x \in A \wedge y \in B\}$, or in other words the set of all ordered pairs that can be constructed by taking the first element from A and the second from B. If A has n elements and B has m, then $A \times B$ has nm elements.[13] For example, $\{1,2\} \times \{3,4\} = \{(1,3),(1,4),(2,3),(2,4)\}$.

Because of the ordering, Cartesian product is not commutative in general. We usually have $A \times B \neq B \times A$ (Exercise: when are they equal?).

The existence of the Cartesian product of any two sets can be proved using the axioms we already have: if (x,y) is defined as $\{\{x\},\{x,y\}\}$, then $\mathcal{P}(A \cup B)$ contains all the necessary sets $\{x\}$ and $\{x,y\}$, and $\mathcal{P}(\mathcal{P}(A \cup B))$ contains all the pairs $\{\{x\},\{x,y\}\}$. It also contains a lot of other sets we don't want, but we can get rid of them using Specification.

A special class of relations are **functions**. A function from a **domain** A to a **codomain**[14] B is a relation on A and B (i.e., a subset of $A \times B$ such that every element of A appears on the left-hand side of exactly one ordered pair. We write $f : A \to B$ as a short way of saying that f is a function from A to B, and for each $x \in A$ write $f(x)$ for the unique $y \in B$ with $(x,y) \in f$.[15]

The set of *all* functions from A to B is written as B^A: note that the order of A and B is backwards here from $A \to B$. Since this is just the subset of $\mathcal{P}(A \times B)$ consisting of functions as opposed to more general relations, it exists by the Power Set and Specification axioms.

When the domain of a function is is finite, we can always write down a list of all its values. For infinite domains (e.g. $\mathbb{N}$), almost all functions are impossible to write down, either as an explicit table (which would need to be infinitely long) or as a formula (there aren't enough formulas). Most

[12]This was not the only possible choice. Kuratowski cites a previous encoding suggested by Hausdorff [Hau14] of (a,b) as $\{\{a,1\},\{b,2\}\}$, where 1 and 2 are tags not equal to a or b. He argues that this definition "seems less convenient to me" than $\{\{a\},\{a,b\}\}$, because it requires tinkering with the definition if a or b do turn out to be equal to 1 or 2. This is a nice example of how even though mathematical definitions arise through convention, some definitions are easier to use than others.

[13]In fact, this is the most direct way to *define* multiplication on $\mathbb{N}$, and pretty much the only sensible way to define multiplication for infinite cardinalities; see §11.1.5.

[14]The codomain is sometimes called the *range*, but most mathematicians will use **range** for $\{f(x) \mid x \in A\}$, which may or may not be equal to the codomain B, depending on whether f is or is not surjective.

[15]Technically, knowing f alone does not tell you what the codomain is, since some elements of B may not show up at all. This can be fixed by representing a function as a pair (f,B), but it's generally not something most people worry about.

of the time we will be interested in functions that have enough structure that we can describe them succinctly, for obvious practical reasons. But in a sense these other, ineffable functions still exist, so we use a definition of a function that encompasses them.

Often, a function is specified not by writing out some huge set of ordered pairs, but by giving a rule for computing $f(x)$. An example: $f(x) = x^2$. Particular trivial functions can be defined in this way anonymously; another way to write $f(x) = x^2$ is as the anonymous function $x \mapsto x^2$.

3.5.1 Examples of functions

- $f(x) = x^2$. Note: this single rule gives several different functions, e.g. $f : \mathbb{R} \to \mathbb{R}$, $f : \mathbb{Z} \to \mathbb{Z}$, $f : \mathbb{N} \to \mathbb{N}$, $f : \mathbb{Z} \to \mathbb{N}$. Changing the domain or codomain changes the function.
- $f(x) = x + 1$.
- Floor and ceiling functions: when x is a real number, the **floor** of x (usually written $\lfloor x \rfloor$) is the largest integer less than or equal to x and the **ceiling** of x (usually written $\lceil x \rceil$) is the smallest integer greater than or equal to x. E.g., $\lfloor 2 \rfloor = \lceil 2 \rceil = 2$, $\lfloor 2.337 \rfloor = 2$, $\lceil 2.337 \rceil = 3$.
- The function from $\{0, 1, 2, 3, 4\}$ to $\{a, b, c\}$ given by the following table:

0	a
1	c
2	b
3	a
4	b

3.5.2 Sequences

Functions let us define sequences of arbitrary length: for example, the infinite sequence $x_0, x_1, x_2, \ldots$ of elements of some set A is represented by a function $x : \mathbb{N} \to A$, while a shorter sequence (a_0, a_1, a_2) would be represented by a function $a : \{0, 1, 2\} \to A$. In both cases the **subscript** takes the place of a function argument: we treat x_n as **syntactic sugar** for $x(n)$. Finite sequences are often called **tuples**, and we think of the result of taking the Cartesian product of a finite number of sets $A \times B \times C$ as a set of tuples (a, b, c), even though the actual structure may be $((a, b), c)$ or $(a, (b, c))$ depending on which product operation we do first.

We can think of the Cartesian product of k sets (where k need not be 2) as a set of sequences indexed by the set $\{1 \ldots k\}$ (or sometimes $\{0 \ldots k - 1\}$).

Technically this means that $A \times B \times C$ (a set of functions) is not the same as $(A \times B) \times C$ (the set of all ordered pairs whose first element is an ordered pair in $A \times B$ and whose second element is in C) or $A \times (B \times C)$ (the set of ordered pairs whose first element is in A and whose second element is in $B \times C$). This distinction has no practical effect and so we typically ignore it; the technical justification for this is that the three different representations are all **isomorphic** in the sense that a translation exists between each pair of them that preserves their structure.

A special case is the Cartesian product of no sets. This is just the set containing a single element, the empty sequence.

Cartesian products over indexed collections of sets can be written using product notation (see §6.2), as in

$$\prod_{i=1}^{n} A_n$$

or even

$$\prod_{x \in \mathbb{R}} A_x.$$

3.5.3 Functions of more (or less) than one argument

If $f : A \times B \to C$, then we write $f(a, b)$ for $f((a, b))$. In general we can have a function with any number of arguments (including 0); a function of k arguments is just a function from a domain of the form $A_1 \times A_2 \times \dots A_k$ to some codomain B.

3.5.4 Composition of functions

Two functions $f : A \to B$ and $g : B \to C$ can be **composed** to give a **composition** $g \circ f$. This is a function from A to C defined by $(g \circ f)(x) = g(f(x))$. Composition is often implicit in definitions of functions: the function $x \mapsto x^2 + 1$ is the composition of two functions $x \mapsto x + 1$ and $x \mapsto x^2$.

3.5.5 Functions with special properties

We can classify functions $f : A \to B$ based on how many elements x of the domain A get mapped to each element y of the codomain B. If every y is the image of at least one x, f is **surjective**. If every y is the image of at most one x, f is **injective**. If every y is the image of exactly one x, f is **bijective**. These concepts are formalized below.

3.5.5.1 Surjections

A function $f : A \to B$ that covers every element of B is called **onto**, **surjective**, or a **surjection**. This means that for any y in B, there exists some x in A such that $y = f(x)$. An equivalent way to show that a function is surjective is to show that its **range** $\{f(x) \mid x \in A\}$ is equal to its codomain.

For example, the function $f(x) = x^2$ from $\mathbb{N}$ to $\mathbb{N}$ is not surjective, because its range includes only perfect squares. The function $f(x) = x + 1$ from $\mathbb{N}$ to $\mathbb{N}$ is not surjective because its range doesn't include 0. However, the function $f(x) = x + 1$ from $\mathbb{Z}$ to $\mathbb{Z}$ *is* surjective, because for every y in $\mathbb{Z}$ there is some x in $\mathbb{Z}$ such that $y = x + 1$.

3.5.5.2 Injections

If $f : A \to B$ maps distinct elements of A to distinct elements of B (i.e., if $x \neq y$ implies $f(x) \neq f(y)$), it is called **one-to-one**, **injective**, or an **injection**. By contraposition, an equivalent definition is that $f(x) = f(y)$ implies $x = y$ for all x and y in the domain. For example, the function $f(x) = x^2$ from $\mathbb{N}$ to $\mathbb{N}$ is injective. The function $f(x) = x^2$ from $\mathbb{Z}$ to $\mathbb{Z}$ is *not* injective (for example, $f(-1) = f(1) = 1$). The function $f(x) = x + 1$ from $\mathbb{N}$ to $\mathbb{N}$ is injective.

3.5.5.3 Bijections

A function that is both surjective and injective is called a **one-to-one correspondence**, **bijective**, or a **bijection**.[16] Any bijection f has an **inverse** f^{-1}; this is the function $\{(y, x) \mid (x, y) \in f\}$.

Of the functions we have been using as examples, only $f(x) = x+1$ from $\mathbb{Z}$ to $\mathbb{Z}$ is bijective.

3.5.5.4 Bijections and counting

Bijections let us define the size of arbitrary sets without having some special means to count elements. We say two sets A and B have the same **size** or **cardinality** if there exists a bijection $f : A \leftrightarrow B$.

Often it is convenient to have standard representatives of sets of a given cardinality. A common trick is to use the **Von Neumann ordinals**, which

[16]The terms *onto*, *one-to-one*, and *bijection* are probably the most common, although *injective* and *surjective* are often used as well, as injective in particular avoids any possible confusion between *one-to-one* and *one-to-one correspondence*. The terms *injective*, *surjective*, and *bijective* are generally attributed to the pseudonymous collective intelligence Bourbaki [Bou70].

are sets that are constructed recursively so that each contains all the smaller ordinals as elements.[17] The empty set $\emptyset$ represents 0, the set $\{0\}$ represents 1, $\{0, 1\}$ represents 2, and so on. The first infinite ordinal is $\omega = \{0, 1, 2, \ldots\}$, which is followed by $\omega + 1 = \{0, 1, 2, \ldots; \omega\}$, $\omega + 2 = \{0, 1, 2, \ldots; \omega, \omega + 1\}$, and so forth; there are also much bigger ordinals like ω^2 (which looks like ω many copies of ω stuck together), ω^ω (which is harder to describe, but can be visualized as the set of infinite sequences of natural numbers with an appropriate ordering), and so on. Given any collection of ordinals, it has a smallest element, equal to the intersection of all elements: this means that Von Neumann ordinals are **well-ordered** (see §9.5.6). So we can define the cardinality $|A|$ of a set A formally as the unique smallest ordinal B such that there exists a bijection $f : A \leftrightarrow B$.

This is exactly what we do when we do counting: to know that there are 3 stooges, we count them off $0 \to \text{Moe}, 1 \to \text{Larry}, 2 \to \text{Curly}$, giving a bijection between the set of stooges and $3 = \{0, 1, 2\}$.

Because different infinite ordinals may have the same cardinality, infinite cardinalities are generally not named for the smallest ordinal of that cardinality, but get their own names. So the cardinality $|\mathbb{N}|$ of the naturals is written as $\aleph_0$, the next largest possible cardinality as $\aleph_1$, etc. See §3.7.1 for more details.

3.6 Constructing the universe

With power set, Cartesian product, the notion of a sequence, etc., we can construct all of the standard objects of mathematics. For example:

Integers The integers are the set $\mathbb{Z} = \{\ldots, -2, -1, 0, -1, 2, \ldots\}$. We represent each integer z as an ordered pair (x, y), where $x = 0 \vee y = 0$; formally, $\mathbb{Z} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x = 0 \vee y = 0\}$. The interpretation of (x, y) is $x - y$; so positive integers z are represented as $(z, 0)$ while negative integers are represented as $(0, -z)$. It's not hard to define addition, subtraction, multiplication, etc. using this representation.

Rationals The rational numbers $\mathbb{Q}$ are all fractions of the form p/q where p is an integer, q is a natural number not equal to 0, and p and q have

[17]The formal definition is that S is an ordinal if (a) every element of S is also a subset of S; and (b) every subset T of S contains an element x with the property that $x = y$ or $x \in y$ for all $y \in T$. The second property says that S is **well-ordered** if we treat $\in$ as meaning $<$ (see §9.5.6. The fact that every subset of S has a minimal element means that we can do induction on S, since if there is some property that does not hold for all x in S, there must be some minimal x for which it doesn't hold.

no common factors. Each such fraction can be represented as a set using an ordered pair (p, q). Operations on rationals are defined as you may remember from grade school.

Reals The real numbers $\mathbb{R}$ can be defined in a number of ways, all of which turn out to be equivalent. The simplest to describe is that a real number x is represented by pair of sets $\{y \in \mathbb{Q} \mid y < x\}$ and $\{y \in Q \mid y \geq x\}$; this is known as a **Dedekind cut**.[Ded01] Formally, a Dedekind cut is any pair of subsets (S, T) of $\mathbb{Q}$ with the properties that (a) S and T **partition** $\mathbb{Q}$, meaning that $S \cap T = \emptyset$ and $S \cup T = \mathbb{Q}$; (b) every element of S is less than every element of T $(\forall s \in S\, \forall t \in T : s < t)$; and (c) S contains no largest element $(\forall x \in S\, \exists y \in S : x < y)$. Note that real numbers in this representation may be hard to write down.

A simpler but equivalent representation is to drop T, since it is just $\mathbb{Q} \setminus S$: this gives use a real number for any proper subset S of $\mathbb{Q}$ that is **downward closed**, meaning that $x < y \in S$ implies $x \in S$. Real numbers this representation may still be hard to write down.

More conventionally, a real number can be written as an infinite decimal expansion like

$$\pi \approx 3.14159265358979323846264338327950288419716939937510582\ldots,$$

which is a special case of a **Cauchy sequence** that gives increasingly good approximations to the actual real number the further along you go.

We can also represent standard objects of computer science:

Deterministic finite state machines A **deterministic finite state machine** is a tuple $(\Sigma, Q, q_0, \delta, Q_{\text{accept}})$ where Σ is an **alphabet** (some finite set), Q is a **state space** (another finite set), $q_0 \in Q$ is an **initial state**, $\delta : Q \times \Sigma \to Q$ is a **transition function** specifying which state to move to when processing some symbol in Σ, and $Q_{\text{accept}} \subseteq Q$ is the set of **accepting states**. If we represent symbols and states as natural numbers, the set of all deterministic finite state machines is then just a subset of $\mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N}) \times \mathbb{N} \times \left(\mathbb{N}^{\mathbb{N} \times \mathbb{N}}\right) \times \mathcal{P}(\mathbb{N})$ satisfying some consistency constraints.

3.7 Sizes and arithmetic

We can compute the size of a set by explicitly counting its elements; for example, $|\emptyset| = 0$, $|\{\text{Larry}, \text{Moe}, \text{Curly}\}| = 3$, and $|\{x \in \mathbb{N} \mid x < 100 \wedge x \text{ is prime}\}| = |\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97\}| = 25$. But sometimes it is easier to compute sizes by doing arithmetic. We can do this because many operations on sets correspond in a natural way to arithmetic operations on their sizes. (For much more on this, see Chapter 11.)

Two sets A and B that have no elements in common are said to be **disjoint**; in set-theoretic notation, this means $A \cap B = \emptyset$. In this case we have $|A \cup B| = |A| + |B|$. The operation of **disjoint union** acts like addition for sets. For example, the disjoint union of 2-element set $\{0, 1\}$ and the 3-element set $\{\text{Wakko}, \text{Jakko}, \text{Dot}\}$ is the 5-element set $\{0, 1, \text{Wakko}, \text{Jakko}, \text{Dot}\}$.

The size of a Cartesian product is obtained by multiplication: $|A \times B| = |A| \cdot |B|$. An example would be the product of the 2-element set $\{a, b\}$ with the 3-element set $\{0, 1, 2\}$: this gives the 6-element set $\{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)\}$. Even though Cartesian product is not generally commutative, since ordinary natural number multiplication is, we always have $|A \times B| = |B \times A|$.

For power set, it is not hard to show that $|\mathcal{P}(S)| = 2^{|S|}$. This is a special case of the size of A^B, the set of all functions from B to A, which is $|A|^{|B|}$; for the power set we can encode $\mathcal{P}(S)$ using 2^S, where 2 is the special set $\{0, 1\}$, and a subset T of S is encoded by the function that maps each $x \in S$ to 0 if $x \notin T$ and 1 if $x \in T$.

3.7.1 Infinite sets

For infinite sets, we take the above properties as *definitions* of addition, multiplication, and exponentiation of their sizes. The resulting system is known as **cardinal arithmetic**, and the sizes that sets (finite or infinite) might have are known as **cardinal numbers**.

The finite cardinal numbers are just the natural numbers: $0, 1, 2, 3, \ldots$. The first infinite cardinal number is the size of the set of natural numbers, and is written as $\aleph_0$ (**aleph-zero**, **aleph-null**, or **aleph-nought**). The next infinite cardinal number is $\aleph_1$ (**aleph-one**): it might or might not be the size of the set of real numbers, depending on whether you include the **Generalized Continuum Hypothesis** in your axiom system or not.[18]

[18]The generalized continuum hypothesis says (essentially) that there aren't any more cardinalities out there in between the ones that are absolutely necessary; a consequence of this is that there are no cardinalities between $|\mathbb{N}|$ and $|\mathbb{R}|$. An alternative notation exists if

Infinite cardinals can behave very strangely. For example:

- $\aleph_0 + \aleph_0 = \aleph_0$. In other words, it is possible to have two sets A and B that both have the same size as $\mathbb{N}$, take their disjoint union, and get another set $A + B$ that has the same size as $\mathbb{N}$. To give a specific example, let $A = \{2x \mid x \in \mathbb{N}\}$ (the concepteven numbers) and $B = \{2x + 1 \mid x \in \mathbb{N}\}$ (the **odd numbers**). These have $|A| = |B| = |\mathbb{N}|$ because there is a bijection between each of them and $\mathbb{N}$ built directly into their definitions. It's also not hard to see that A and B are disjoint, and that $A \cup B = \mathbb{N}$. So $|A| = |B| = |A| + |B|$ in this case.

  The general rule for cardinal addition is that $\kappa + \lambda = \max(\kappa, \lambda)$ if at least one of κ and λ is infinite. Sums of finite cardinals behave exactly the way you expect.

- $\aleph_0 \cdot \aleph_0 = \aleph_0$. Example: A bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$ using the **Cantor pairing function** $\langle x, y\rangle = (x + y + 1)(x + y)/2 + y$. The first few values of this are $\langle 0, 0\rangle = 0, \langle 1, 0\rangle = 2 \cdot 1/2 + 0 = 1, \langle 0, 1\rangle = 2 \cdot 1/2 + 1 = 1, \langle 2, 0\rangle = 3 \cdot 2/2 + 0 = 3, \langle 1, 1\rangle = 3 \cdot 2/2 + 1 = 4, \langle 0, 2\rangle = 3 \cdot 2/2 + 2 = 5$, etc. The basic idea is to order all the pairs by increasing $x + y$, and then order pairs with the same value of $x + y$ by increasing y; eventually every pair is reached.

  The general rule for cardinal multiplication is that $\kappa \cdot \lambda = \max(\kappa, \lambda)$ if at least one of κ or λ is infinite. So $\kappa \cdot \lambda = \kappa + \lambda$ if either is infinite (or both are zero).

- $\mathbb{N}^* = \{\text{all finite sequences of elements of } \mathbb{N}\}$ has size $\aleph_0$. One way to do this to define a function recursively by setting $f([]) = 0$ and $f([\text{first}, \text{rest}]) = 1 + \langle \text{first}, f(\text{rest})\rangle$, where first is the first element of

you don't want to take a position on GCH: this writes $\beth_0$ ("beth-0") for $|\mathbb{N}|$, $\beth_1$ ("beth-1") for $|\mathbb{R}| = |\mathcal{P}(\mathbb{N})|$, with the general rule $\beth_{i+1} = 2^{\beth_i}$. This avoids the issue of whether there exist sets with size between $\mathbb{N}$ and $\mathbb{R}$, for example. In my limited experience, only hard-core set theorists ever use $\beth$ instead of $\aleph$: in the rare cases where the distinction matters, most normal mathematicians will just assume GCH, which makes $\beth_i = \aleph_i$ for all i.

the sequence and rest is all the other elements. For example,

$$\begin{aligned} f(0,1,2) &= 1 + \langle 0, f(1,2)\rangle \\ &= 1 + \langle 0, 1 + \langle 1, f(2)\rangle\rangle \\ &= 1 + \langle 0, 1 + \langle 1, 1 + \langle 2, 0\rangle\rangle\rangle \\ &= 1 + \langle 0, 1 + \langle 1, 1 + 3\rangle\rangle = 1 + \langle 0, 1 + \langle 1, 4\rangle\rangle \\ &= 1 + \langle 0, 1 + 19\rangle \\ &= 1 + \langle 0, 20\rangle \\ &= 1 + 230 \\ &= 231. \end{aligned}$$

This assigns a unique element of $\mathbb{N}$ to each finite sequence, which is enough to show $|\mathbb{N}^*| \le |\mathbb{N}|$. With some additional effort one can show that f is in fact a bijection, giving $|\mathbb{N}^*| = |\mathbb{N}|$.

3.7.2 Countable sets

All of these sets have the property of being **countable**, which means that they can be put into a bijection with $\mathbb{N}$ or one of its subsets. The general principle is that any sum or product of infinite cardinal numbers turns into taking the maximum of its arguments. The last case implies that anything you can write down using finitely many symbols (even if they are drawn from an infinite but countable alphabet) is countable. This has a lot of applications in computer science: one of them is that the set of all computer programs in any particular programming language is countable.

3.7.3 Uncountable sets

Exponentiation is different. We can easily show that $2^{\aleph_0} \ne \aleph_0$, or equivalently that there is no bijection between $\mathcal{P}(\mathbb{N})$ and $\mathbb{N}$. This is done using **Cantor's diagonalization argument**, which appears in the proof of the following theorem.

Theorem 3.7.1. *Let S be any set. Then there is no surjection $f : S \to \mathcal{P}(S)$.*

Proof. Let $f : S \to \mathcal{P}(S)$ be some function from S to subsets of S. We'll construct a subset of S that f misses, thereby showing that f is not a surjection. Let $A = \{x \in S \mid x \notin f(x)\}$. Suppose $A = f(y)$. Then $y \in A \leftrightarrow y \notin A$, a contradiction.[19] $\square$

[19]Exercise: Why does A exist even though the Russell's Paradox set doesn't?

Since any bijection is also a surjection, this means that there's no bijection between S and $\mathcal{P}(S)$ either, implying, for example, that $|\mathbb{N}|$ is strictly less than $|\mathcal{P}(\mathbb{N})|$.

(On the other hand, it is the case that $\left|\mathbb{N}^{\mathbb{N}}\right| = \left|2^{\mathbb{N}}\right|$, so things are still weird up here.)

Sets that are larger than $\mathbb{N}$ are called **uncountable**. A quick way to show that there is no surjection from A to B is to show that A is countable but B is uncountable. For example:

Corollary 3.7.2. *There are functions $f : \mathbb{N} \to \{0, 1\}$ that are not computed by any computer program.*

Proof. Let P be the set of all computer programs that take a natural number as input and always produce 0 or 1 as output (assume some fixed language), and for each program $p \in P$, let f_p be the function that p computes. We've already argued that P is countable (each program is a finite sequence drawn from a countable alphabet), and since the set of all functions $f : \mathbb{N} \to \{0, 1\} = 2^{\mathbb{N}}$ has the same size as $\mathcal{P}(\mathbb{N})$, it's uncountable. So some f gets missed: there is at least one function from $\mathbb{N} to \{0, 1\}$ that is not equal to f_p for any program p. ∎

The fact that there are more functions from $\mathbb{N}$ to $\mathbb{N}$ than there are elements of $\mathbb{N}$ is one of the reasons why set theory (slogan: "everything is a set") beat out **lambda calculus** (slogan: "everything is a function from functions to functions") in the battle over the foundations of mathematics. And this is why we do set theory in CS202 and lambda calculus (disguised as Scheme) in CS201.

3.8 Further reading

See [Ros12, §§2.1–2.2], [Big02, Chapter 2], or [Fer08, §1.3, §1.5].

Chapter 4

The real numbers

The **real numbers** $\mathbb{R}$ are the subject of high-school algebra and most practical mathematics. Some important restricted classes of real numbers are the **naturals** $\mathbb{N} = 0, 1, 2, \dots$, the integers $\mathbb{Z} = \dots, -2, -1, 0, 1, 2, \dots$, and the **rationals** $\mathbb{Q}$, which consist of all real numbers that can be written as ratios of integers p/q, otherwise known as **fractions**.

The rationals include $1, 3/2, 22/7, -355/113$, an so on, but not some common mathematical constants like $e \approx 2.718281828\dots$ or $\pi \approx 3.141592\dots$. Real numbers that are not rational are called **irrational**. There is now single-letter abbreviation for the irrationals.

The typeface used for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ is called **blackboard bold** and originates from the practice of emphasizing a letter on a blackboard by writing it twice. Some writers just use ordinary boldface: $\mathbf{N}$, etc., but this does not scream out "this is a set of numbers" as loudly as blackboard bold. You may also see blackboard bold used for more exotic sets of numbers like the **complex numbers** $\mathbb{C}$, which are popular in physics and engineering, and for some more exotic number systems like the $\mathbb{H}$,[1] which are sometimes used in graphics, or the **octonions** $\mathbb{O}$, which exist mostly to see how far complex numbers can be generalized.

Like any mathematical structure, the real numbers are characterized by a list of **axioms**, which are the basic facts from which we derive everything we know about the reals. There are many equivalent ways of axiomatizing the real numbers; we will give one here. Many of these properties can also be found in [Fer08, Appendix B]. These should mostly be familiar to you from high-school algebra, but we include them here because we need to know

[1]Why $\mathbb{H}$? The rationals already took $\mathbb{Q}$ (for "quotient"), so the quaternions are abbreviated by the initial of their discoverer, William Rowan Hamilton.

what we can assume when we want to prove something about reals.

4.1 Field axioms

The real numbers are a **field**, which means that they support the operations of addition $+$, multiplication $\cdot$, and their inverse operations subtraction $-$ and division $/$. The behavior of these operations is characterized by the **field axioms**.

4.1.1 Axioms for addition

Addition in a field satisfies the axioms of a **commutative group** (often called an **abelian group**, after Niels Henrik Abel, an early nineteenth-century mathematician). These characterize the behavior of the addition operation $+$ and sums of the form $a + b$ ("a **plus** b").

Axiom 4.1.1 (Commutativity of addition)**.** *For all numbers,*

$$a + b = b + a. \tag{4.1.1}$$

Any operation that satisfies Axiom 4.1.1 is called **commutative**. Commutativity lets us ignore the order of arguments to an operation. Later, we will see that multiplication is also commutative.

Axiom 4.1.2 (Associativity of addition)**.** *For all numbers,*

$$a + (b + c) = (a + b) + c. \tag{4.1.2}$$

An operation that satisfies Axiom 4.1.2 is called **associative**. Associativity means we don't have to care about how a sequence of the same associative operation is parenthesized, letting us write just $a + b + c$ for $a + (b + c) = (a + b) + c$.[2]

[2] A curious but important practical fact is that addition is often *not* associative in computer arithmetic. This is because computers (and calculators) approximate real numbers by **floating-point numbers**, which only represent the some limited number of digits of an actual real number in order to make it fit in limited memory. This means that low-order digits on very large numbers can be lost to **round-off error**. So a computer might report $(1000000000000 + -1000000000000) + 0.00001 = 0.00001$ but $1000000000000 + (-1000000000000 + 0.00001) = 0.0$. Since we don't have to write any programs in this class, we will just work with actual real numbers, and not worry about such petty numerical issues.

Axiom 4.1.3 (Additive identity)**.** *There exists a number* 0 *such that, for all numbers* a*,*

$$a + 0 = 0 + a = a. \tag{4.1.3}$$

An object satisfies the condition $a + 0 = 0 + a = 1$ for some operation is called an **identity** for that operation. Later we will see that 1 is an identity for multiplication.

It's not hard to show that identities are unique:

Lemma 4.1.4. *Let* $0' + a = a + 0' = a$ *for all* a*. Then* $0' = 0$*.*

Proof. Compute $0' = 0' + 0 = 0$. (The first equality holds by the fact that $a = a + 0$ for all a and the second from the assumption that $0' + a = a$ for all a.) ☐

Axiom 4.1.5 (Additive inverses)**.** *For each* a*, there exists a number* $-a$*, such that*

$$a + (-a) = (-a) + a = 0. \tag{4.1.4}$$

For convenience, we will often write $a + (-b)$ as $a - b$ ("a **minus** b"). This gives us the operation of **subtraction**. The operation that returns $-a$ given a is called **negation** and $-a$ can be read as "**negative** a," "**minus** a",[3] or the "negation of a."

Like identities, inverses are also unique:

Lemma 4.1.6. *If* $a' + a = a + a' = 0$*, then* $a' = -a$*.*

Proof. Starting with $0 = a' + a$, add $-a$ on the right to both sides to get $-a = a' + a + -a = a'$. ☐

4.1.2 Axioms for multiplication

Multiplication in a field satisfies the axioms of a commutative group, if the additive identity 0 is excluded.

For convenience[4], the multiplication operation $\cdot$ is often omitted, allowing us to write ab for $a \cdot b$. We will use this convention when it will not cause confusion.

[3]Warning: Some people will get annoyed with you over "minus a" and insist on reserving "minus" for the operation in $a - b$. In extreme cases, you may see $-a$ typeset differently: ^{-}a. Pay no attention to these people. Though not making the distinction makes life more difficult for calculator designers and compiler writers, as a working mathematician you are entitled to **abuse notation** by using the same symbol for multiple purposes when it will not lead to confusion.

[4]Also called "laziness."

Axiom 4.1.7 (Commutativity of multiplication)**.** *For all numbers,*

$$ab = ba. \tag{4.1.5}$$

Axiom 4.1.8 (Associativity of multiplication)**.** *For all numbers,*

$$a(bc) = (ab)c. \tag{4.1.6}$$

Axiom 4.1.9 (Multiplicative identity)**.** *There exists a number* $1 \neq 0$ *such that, for all numbers* a*,*

$$a \cdot 1 = 1 \cdot a = a. \tag{4.1.7}$$

We insist that $1 \neq 0$ because we want Axiom 4.1.9 to hold in $\mathbb{R} \setminus \{0\}$. This also has the beneficial effect of preventing us from having $\mathbb{R} = \{0\}$, which would otherwise satisfy all of our axioms.

Since the only difference between the multiplicative identity and the additive identity is notation, Lemma 4.1.4 applies here as well: if there is any $1'$ such that $a \cdot 1' = 1' \cdot a = a$ for all a, then $1' = 1$.

Axiom 4.1.10 (Multiplicative inverses)**.** *For every* a *except* 0*, there exists a number* a^{-1}*, such that*

$$a \cdot a^{-1} = a^{-1} \cdot a = 1. \tag{4.1.8}$$

Lemma 4.1.6 applies here to show that a^{-1} is also unique for each a.

For convenience, we will often write $a \cdot b^{-1}$ as a/b or the vertical version $\frac{a}{b}$. This gives us the operation of **division**. The expression a/b or $\frac{a}{b}$ is pronounced "a **over** b" or (especially in grade school, whose occupants are generally not as lazy as full-grown mathematicians) "a **divided by** b." Some other notations (mostly used in elementary school) for this operation are $a \div b$ and $a : b$.[5]

Note that because 0 is not guaranteed to have an inverse,[6] the meaning of $a/0$ is not defined.

The number a^{-1}, when it does exist, is often just called the **inverse** of a or sometimes "inverse a." (The ambiguity that might otherwise arise with the additive inverse $-a$ is avoided by using negation for $-a$.) The multiplicative inverse a^{-1} can also be written using the division operation as $1/a$.

[5]Using a colon for division is particularly popular in German-speaking countries, where the "My Dear Aunt Sally" rule for remembering that multiplication and division bind tighter than addition and subtraction becomes the more direct *Punktrechnung vor Strichrechnung*—"point reckoning before stroke reckoning."

[6]In fact, once we get a few more axioms, terrible things will happen if we try to make 0 have an inverse.

4.1.3 Axioms relating multiplication and addition

Axiom 4.1.11 (Distributive law)**.** *For all a, b, and c,*

$$a \cdot (b + c) = ab + ac \tag{4.1.9}$$

$$(a + b) \cdot c = ac + bc \tag{4.1.10}$$

Since multiplication is commutative, we technically only need one of (4.1.9) and (4.1.10), but there are other structures we will see called **rings** that satisfy the distributive law without having a commutative multiplication operation, so it's safest to include both.

The additive identity 0 also has a special role in multiplication, which is a consequence of the distributive law: it's an **annihilator**:

Lemma 4.1.12. *For all a,*

$$a \cdot 0 = 0 \cdot a = 0. \tag{4.1.11}$$

Proof. Because $0 = 0 + 0$, we have $a \cdot 0 = a \cdot (0 + 0) = a \cdot 0 + a \cdot 0$. But then adding $-(a \cdot 0)$ to both sides gives $0 = a \cdot 0$. ∎

Annihilation is why we don't want to define 0^{-1}, and thus won't allow division by zero. If there were a real number that was 0^{-1}, then for any a and b we would have:

$$\begin{aligned} a \cdot 0 &= b \cdot 0 = 0 \\ (a \cdot 0) \cdot 0^{-1} &= (b \cdot 0) \cdot 0^{-1} \\ a \cdot (0 \cdot 0^{-1}) &= b \cdot (0 \cdot 0^{-1}) \\ a \cdot 1 &= b \cdot 1 \\ a &= b. \end{aligned}$$

(Exercise: which axiom is used at each step in this proof?)

In particular, we would get $1 = 0$, contradicting Axiom 4.1.9.

A similar argument shows that

Lemma 4.1.13. *If $a \cdot b = 0$, then $a = 0$ or $b = 0$.*

Proof. Suppose $a \cdot b = 0$ but $a \neq 0$.[7] Then a has an inverse a^{-1}. So we can compute

$$a \cdot b = 0 \tag{4.1.12}$$

$$a^{-1} \cdot a \cdot b = a^{-1} \cdot 0 \tag{4.1.13}$$

$$b = 0. \tag{4.1.14}$$

[7]This is an example of the proof strategy where we show $P \vee Q$ by assuming $\neg P$ and proving Q.

□

Another consequence of the distributive law is that we can determine how multiplication interacts with negation. You may recall being taught at an impressionable age that

$$a \cdot (-b) = -(ab), \tag{4.1.15}$$
$$(-a) \cdot b = -(ab), \tag{4.1.16}$$

and

$$(-a) \cdot (-b) = ab. \tag{4.1.17}$$

Like annihilation, these are not axioms—or at least, we don't have to include them as axioms if we don't want to. Instead, we can prove them directly from axioms and theorems we've already got. For example, here is a proof of (4.1.15):

$$\begin{aligned}
a \cdot 0 &= 0 \\
a \cdot (b + (-b)) &= 0 \\
ab + a \cdot (-b) &= 0 \\
-(ab) + (ab + a \cdot (-b)) &= -(ab) \\
(-(ab) + ab) + a \cdot (-b) &= -(ab) \\
0 + a \cdot (-b) &= -(ab) \\
a \cdot (-b) &= -(ab).
\end{aligned}$$

Similar proofs can be given for (4.1.16) and (4.1.17).

A special case of this is that multiplying by -1 is equivalent to negation:

Corollary 4.1.14. *For all a,*

$$(-1) \cdot a = -a. \tag{4.1.18}$$

Proof. Using (4.1.17), $(-1) \cdot a = -(1 \cdot a) = -a$. □

4.1.4 Other algebras satisfying the field axioms

The field axioms so far do not determine the real numbers: they also hold for any number of other fields, including the rationals $\mathbb{Q}$, the complex numbers $\mathbb{C}$, and various finite fields such as the integers modulo a prime p (written as $\mathbb{Z}_p$; we'll see more about these in Chapter 14).

They do not hold for the integers $\mathbb{Z}$ (which don't have multiplicative inverses) or the natural numbers $\mathbb{N}$ (which don't have additive inverses either). This means that $\mathbb{Z}$ and $\mathbb{N}$ are not fields, although they are examples of weaker algebraic structures (a **ring** in the case of $\mathbb{Z}$ and a **semiring** in the case of $\mathbb{N}$).

4.2 Order axioms

Unlike $\mathbb{C}$ and $\mathbb{Z}_p$ (but like $\mathbb{Q}$), the real numbers are an **ordered field**, meaning that in addition to satisfying the field axioms, there is a relation $\leq$ that satisfies the axioms:

Axiom 4.2.1 (Comparability)**.** *$a \leq b$ or $b \leq a$.*

Axiom 4.2.2 (Antisymmetry)**.** *If $a \leq b$ and $b \leq a$, then $a = b$.*

Axiom 4.2.3 (Transitivity)**.** *If $a \leq b$ and $b \leq c$, then $a \leq c$.*

Axiom 4.2.4 (Translation invariance)**.** *If $a \leq b$, then $a + c \leq b + c$.*

Axiom 4.2.5 (Scaling invariance)**.** *If $a \leq b$ and $0 \leq c$, then $a \cdot c \leq b \cdot c$.*

The first three of these mean that $\leq$ is a **total order** (see §9.5.5). The other axioms describe how $\leq$ interacts with addition and multiplication.

For convenience, we define $a < b$ as shorthand for $a \leq b$ and $a \neq b$, and define reverse operations $a \geq b$ (meaning $b \leq a$) and $a > b$ (meaning $b > a$). If $a > 0$, we say that a is **positive**. If $a < 0$, it is **negative**. If $a \geq 0$, it is **non-negative**. **Non-positive** can be used to say $a \leq 0$, but this doesn't seem to come up as much as non-negative.

Other properties of $\leq$ can be derived from these axioms.

Lemma 4.2.6 (Reflexivity)**.** *For all x, $x \leq x$.*

Proof. Apply comparability with $y = x$. ☐

Lemma 4.2.7 (Trichotomy)**.** *Exactly one of $x < y$, $x = y$, or $x > y$ holds.*

Proof. First, let's show that at least one holds. If $x = y$, we are done. Otherwise, suppose $x \neq y$. From comparability, we have $x \leq y$ or $y \leq x$. Since $x \neq y$, this gives either $x < y$ or $x > y$.

Next, observe that $x = y$ implies $x \not< y$ and $x \not> y$, since $x < y$ and $x > y$ are both defined to hold only when $x \neq y$. This leaves the possibility that $x < y$ and $x > y$. But then $x \leq y$ and $y \leq$, so by anti-symmetry, $x = y$, contradicting our assumption. So at most one holds. ☐

Trichotomy lets us treat, for example, $x \not< y$ and $x \geq y$ as equivalent.

Lemma 4.2.8. *If $a \geq 0$, then $-a \leq 0$.*

Proof. Take $a \geq 0$ and add $-a$ to both sides (using Axiom 4.2.4) to get $0 \geq -a$. ☐

Lemma 4.2.9. *For all a and b, $a \geq b$ if and only if $a - b \geq 0$.*

Proof. Given $a \geq b$, add $-b$ to both sides to get $a - b \geq 0$. Given $a - b \geq 0$, do the reverse by adding b to both sides. ☐

Lemma 4.2.10. *If $a \geq 0$ and $b \geq 0$, then $a + b \geq 0$.*

Proof. From Lemma 4.2.8, $a \geq 0$ implies $0 \geq -a$. So $b \geq 0 \geq -a$ by transitivity. Add a to both sides to get $a + b \geq 0$. ☐

Theorem 4.2.11. *If $a \geq b$ and $c \geq d$, then $a + c \geq b + d$.*

Proof. From Lemma 4.2.9, $a - b \geq 0$ and $c - d \geq 0$. From Lemma 4.2.10, we get $(a-b)+(c-d) \geq 0$. Now add $b+d$ to both sides to get $a+c \geq b+d$. ☐

Lemma 4.2.12. *If $a \leq b$, then $-b \leq -a$.*

Proof. Subtract $a + b$ from both sides. ☐

Theorem 4.2.13. *If $a \leq b$ and $c \leq 0$, then $a \cdot c \geq b \cdot c$.*

Proof. From Lemma 4.2.8, $-c \geq 0$, so from Axiom 4.2.5, $-c \cdot a \leq -c \cdot b$. Now apply Lemma 4.2.12 to get $c \cdot a \geq c \cdot b$. ☐

4.3 Least upper bounds

One more axiom is needed to characterize the reals. A subset S of $\mathbb{R}$ has an **upper bound** if there is some $x \in \mathbb{R}$ such that $y \leq x$ for all y in S. It has a **least upper bound** if there is a smallest z with this property: some z such that (a) z is an upper bound on S and (b) whenever q is an upper bound on S, $z \leq q$.

Axiom 4.3.1 (Least upper bound property)**.** *Every nonempty subset of $\mathbb{R}$ that has an upper bound has a least upper bound.*

More formally, if for some $S \subseteq \mathbb{R}$, $S \neq \emptyset$ and there exists some x such that $y \leq x$ for all y in S, then there exists $z \in \mathbb{R}$ such that $y \leq z$ for all y in S and whenever $y \leq q$ for all y in S, $z \leq q$.

The least upper bound of a set S, if there is one, is called the **supremum** of S and written as $\sup S$. A consequence of the least upper bound property is that every nonempty set S that has a **lower bound** has a **greatest lower bound**, or **infimum**: $\inf S = -\sup\{-x \mid x \in S\}$. Neither the supremum nor the infimum is defined for empty or unbounded sets.[8]

It may be that $\sup S$ or $\inf S$ is not actually an element of S. For example, $\sup\{x \in \mathbb{R} \mid x < 1\} = 1$, but $1 \notin \{x \in \mathbb{R} \mid x < 1\}$.

Having least upper bounds distinguishes the reals from the rationals: The bounded nonempty set $\{x \in \mathbb{Q} | x \cdot x \leq 2\}$ has no least upper bound in $\mathbb{Q}$ (because $\sqrt{2}$ is not rational), but it does in $\mathbb{R}$. (This is another example of a set that doesn't include its least upper bound.)

A consequence of having least upper bounds is that reals do not get too big or too small:

Theorem 4.3.2 (Archimedean property)**.** *For any two real numbers $0 < x < y$, there exists some $n \in \mathbb{N}$ such that $n \cdot x > y$.*

Proof. The proof is by contradiction.

Suppose that this is not true, that is, that there exist $0 < x < y$ such that $n \cdot x \leq y$ for all $n \in \mathbb{N}$. Dividing both sides by x gives $n \leq x/y$ for all $n \in \mathbb{N}$, meaning that x/y is an upper bound on $\mathbb{N}$. From the least upper bound property, there exists a least upper bound z on $\mathbb{N}$.

Now consider $z - 1$. This is less than z, so it's not an upper bound on $\mathbb{N}$. If we negate the statement $\forall n \in \mathbb{N}, n \leq z - 1$, we get $\exists n \in \mathbb{N}, n > z - 1$. But then $n + 1 > z$, contradicting the claim that z is an upper bound. $\square$

This excludes the possibility of **infinitesimals**, nonzero values that are nonetheless smaller than every positive rational number. You can blame Bishop Berkeley [Ber34] for the absence of these otherwise very useful objects from our standard mathematical armory. However, in return for losing infinitesimals we do get that the rationals are **dense** in the reals, meaning that between any two reals is a rational, as well as many other useful properties.

[8] It's sometimes convenient to extend $\mathbb{R}$ by adding two extra elements $-\infty$ and $+\infty$, where $-\infty$ is smaller than all reals and $+\infty$ is bigger. In the resulting **extended real line**, we can define $\inf S = -\infty$ when S has no lower bound, $\sup S = +\infty$ when S has no upper bound, and $\inf \emptyset = +\infty$ and $\sup \emptyset = -\infty$. These last two conventions are chosen because they preserve the rules $\inf(S \cup T) = \min(\inf S, \inf T)$ and $\sup(S \cup T) = \max(\sup S, \sup T)$.

4.4 What's missing: algebraic closure

One way to think about the development of number systems is that each system $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ adds the ability to solve equations that have no solutions in the previous system. Some specific examples are

$x + 1 = 0$	Solvable in $\mathbb{Z}$ but not $\mathbb{N}$
$2x = 1$	Solvable in $\mathbb{Q}$ but not $\mathbb{Z}$
$x \cdot x = 2$	Solvable in $\mathbb{R}$ but not $\mathbb{Q}$
$x \cdot x + 1 = 0$	Solvable in $\mathbb{C}$ but not $\mathbb{R}$

This process stops with the complex numbers $\mathbb{C}$, which consist of pairs of the form $a + bi$ where $i^2 = -1$. The reason is that the complex numbers are **algebraically closed**: if you write an equation using only complex numbers, $+$, and $\cdot$, and it has some solution x in any field bigger than $\mathbb{C}$, then x is in $\mathbb{C}$ as well. The down side in comparison to the reals is that we lose order: there is no ordering of complex numbers that satisfies the translation and scaling invariance axioms. As in many other areas of mathematics and computer science, we are forced to make trade-offs based on what is important to us at the time.

4.5 Arithmetic

In principle, it is possible to show that the standard grade-school algorithms for arithmetic all work in $\mathbb{R}$ as defined by the axioms in the preceding sections. This is sometimes trickier than it looks: for example, just showing that 1 is positive requires a sneaky application of Axiom 4.2.5.[9]

To avoid going nuts, we will adopt the following rule:

Rule 4.5.1. *Any grade-school fact about arithmetic that does not involve any variables will be assumed to be true in* $\mathbb{R}$.

So for example, you don't need to write out a proof using the definition of multiplicative inverses and the distributive law to conclude that $\frac{1}{2} + \frac{3}{5} = \frac{11}{10}$; just remembering how to add fractions (or getting a smart enough computer to do it for you) is enough.

Caveat: Dumb computers will insist on returning useless decimals like 1.1. As mathematicians, we don't like decimal notation, because it can't

[9]Suppose $1 \leq 0$. Then $1 \cdot 1 \geq 0 \cdot 1$ (Theorem 4.2.13, which simplifies to $1 \geq 0$. Since $1 \neq 0$, this contradicts our assumption, showing that $1 > 0$.

represent exactly even trivial values like $\frac{1}{3}$. Similarly, mixed fractions like $1\frac{1}{10}$, while useful for carpenters, are not popular in mathematics.

4.6 Connection between the reals and other standard algebras

The reals are an example of an **algebra**, which is a set with various operations attached to it: the set is $\mathbb{R}$ itself with the operations being 0, 1, $+$, and $\cdot$. A **sub-algebra** is a subset that is **closed** under the operations, meaning that the results of any operation applied to elements of the subsets (no elements in the case of 0 or 1) yields an element of the subset.

All sub-algebras of $\mathbb{R}$ inherit any properties that don't depend on the existence of particular elements other than 0 and 1; so addition and multiplication are still commutative and associative, multiplication still distributes over addition, and 0 and 1 are still identities. But other axioms may fail.

Some interesting sub-algebras of $\mathbb{R}$ are:

- The **natural numbers** $\mathbb{N}$. This is the smallest sub-algebra of $\mathbb{R}$, because once you have 0, 1, and addition, you can construct the rest of the naturals as $1 + 1$, $1 + 1 + 1$, etc. They do not have additive or multiplicative inverses, but they do satisfy the order axioms, as well as the extra axiom that $0 \leq x$ for all $x \in \mathbb{N}$.

- The **integers** $\mathbb{Z}$. These are what you get if you throw in additive inverses: now in addition to 0, 1, $1+1$, etc., you also get -1, $-(1+1)$, etc. The order axioms are still satisfied. No multiplicative inverses, though.

- The **dyadics** $\mathbb{D}$. These are numbers of the form $m2^{-n}$ where $m \in \mathbb{Z}$ and $n \in \mathbb{N}$. These are of some importance in computing because almost all numbers represented inside a computer are really dyadics, although in mathematics they are not used much. Like the integers, they still don't have multiplicative inverses: there is no way to write $1/3$ (for example) as $m2^{-n}$.

- The rationals $\mathbb{Q}$. Now we ask for multiplicative inverses, and get them. Any rational can be written as p/q where p and q are integers. Unless extra restrictions are put on p and q, these representations are not unique: $22/7 = 44/14 = 66/21 = (-110)/(-35)$. You probably first saw these in grade school as **fractions**, and one way to describe $\mathbb{Q}$ is as the **field of fractions** of $\mathbb{Z}$.

> The rationals satisfy all the field axioms, and are the smallest subfield of $\mathbb{R}$. They also satisfy all the ordered field axioms and the Archimedean property. But they are not complete. Adding completeness gives the real numbers.

An issue that arises here is that, strictly speaking, the natural numbers $\mathbb{N}$ we defined back in §3.4 are *not* elements of $\mathbb{R}$ as defined in terms of, say, Dedekind cuts. The former are finite ordinals while the latter are downward-closed sets of rationals, themselves represented as elements of $\mathbb{N} \times \mathbb{N}$. Similarly, the integer elements of $\mathbb{Q}$ will be pairs of the form $(n, 1)$ where $n \in \mathbb{N}$ rather than elements of $\mathbb{N}$ itself. We also have a definition (§G.1) that builds natural numbers out of 0 and a successor operation S. So what does it mean to say $\mathbb{N} \subseteq \mathbb{Q} \subseteq \mathbb{R}$?

One way to think about it is that the sets

$$\begin{aligned}
&\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \dots\}, \\
&\{(0,1), (1,1), (2,1), (3,1), \dots\}, \\
&\{\{(p,q) \mid p < 0\}, \{(p,q) \mid p < q\}, \{(p,q) \mid p < 2q\}, \{(p,q) \mid p < 3q\}, \dots\},
\end{aligned}$$

and

$$\{0, S0, SS0, SSS0, \dots\}$$

are all **isomorphic**: there are bijections between them that preserve the behavior of 0, 1, $+$, and $\cdot$. So we think of $\mathbb{N}$ as representing some Platonic ideal of natural-numberness that is only defined up to isomorphism.[10] So in the context of $\mathbb{R}$, when we write $\mathbb{N}$, we mean the version of $\mathbb{N}$ that is a subset of $\mathbb{R}$, and in other contexts, we might mean a different set that happens to behave in exactly the same way.

In the other direction, the complex numbers are a super-algebra of the reals: we can think of any real number x as the complex number $x + 0i$, and this complex number will behave exactly the same as the original real number x when interacting with other real numbers carried over into $\mathbb{C}$ in the same way.

The various features of these algebras are summarized in Table 4.1.

[10] In programming terms, $\mathbb{N}$ is an interface that may have multiple equivalent implementations.

Symbol	$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{C}$
Name	Naturals	Integers	Rationals	Reals	Complex numbers
Typical element	12	-12	$-\frac{12}{7}$	$\sqrt{12}$	$\sqrt{12}+\frac{22}{7}i$
Associative	Yes	Yes	Yes	Yes	Yes
0 and 1	Yes	Yes	Yes	Yes	Yes
Inverses	No	+ only	Yes	Yes	Yes
Associative	Yes	Yes	Yes	Yes	Yes
Ordered	Yes	Yes	Yes	Yes	No
Least upper bounds	Yes	Yes	No	Yes	No
Algebraically closed	No	No	No	No	Yes

Table 4.1: Features of various standard algebras

4.7 Extracting information from reals

The **floor** function $\lfloor x \rfloor$ and **ceiling** function $\lceil x \rceil$ can be used to convert an arbitrary real to an integer: the floor of x is the largest integer less than or equal to x, while the ceiling of x is the smallest integer greater than or equal to x. More formally, they are defined by $\lfloor x \rfloor = \sup \{y \in \mathbb{Z} \mid y \leq x\}$ and $\lceil x \rceil = \inf \{y \in \mathbb{Z} \mid y \geq x\}$. The floor and ceiling will always be integers, with $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$. If x is already an integer, $\lfloor x \rfloor = x = \lceil x \rceil$. Some examples: $\lfloor \pi \rfloor = 3$, $\lceil \pi \rceil = 4$, $\lfloor -1/2 \rfloor = -1$, $\lceil -1/2 \rceil = 0$, $\lfloor 12 \rfloor = \lceil 12 \rceil = 12$.

If you want the **fractional part** of a real number x, you can compute it as $x - \lfloor x \rfloor$.

The **absolute value** $|x|$ of x is defined by

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

The absolute value function erases the sign of x: $|-12| = |12| = 12$.

The **signum** function $\operatorname{sgn}(x)$ returns the sign of its argument, encode as -1 for negative, 0 for zero, and $+1$ for positive:

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ +1 & \text{if } x > 0. \end{cases}$$

So $\operatorname{sgn}(-12) = -1$, $\operatorname{sgn}(0) = 0$, and $\operatorname{sgn}(12) = 1$. This allows for an alternative definition of $|x|$ as $\operatorname{sgn}(x) \cdot x$.

Chapter 5

Induction and recursion

Induction is a technique for proving universal statements about some class of objects built from smaller objects: the idea is to show that each if each object has a property provided all the smaller objects do, then every object in the class has the property. **Recursion** is the same technique applied to definitions instead of proofs: an object is defined in terms of smaller objects of the same type.

5.1 Simple induction

The simplest form of induction goes by the name of **simple induction**, and it's what we use to show that something is true for all natural numbers.

We have several equivalent definitions of the natural numbers $\mathbb{N}$, but what they have in common is the following basic pattern, which goes back to Peano [Pea89]:

- 0 is a natural number.
- If x is a natural number, so is $x + 1$.

This is an example of a **recursive definition**: it gives us a base object to start with (0) and defines new natural numbers ($x+1$) by applying some operation ($+1$) to natural numbers we already have x.

Because these are the only ways to generate natural numbers, we can prove that a particular natural number has some property P by showing that you can't construct a natural number without having P be true. This means showing that $P(0)$ is true, and that $P(x)$ implies $P(x + 1)$. If both of these statements hold, then P is baked into each natural number as part of its construction.

We can express this formally as the **induction schema**:

$$(P(0) \land \forall x \in \mathbb{N} : (P(x) \to P(x+1))) \to \forall x \in \mathbb{N} : P(x). \tag{5.1.1}$$

Any proof that uses the induction schema will consist of two parts, the **base case** showing that $P(0)$ holds, and the **induction step** showing that $P(x) \to P(x+1)$. The assumption $P(x)$ used in the induction step is called the **induction hypothesis**.

For example, let's suppose we want to show that for all $n \in \mathbb{N}$, either $n = 0$ or there exists n' such that $n = n' + 1$. Proof: We are trying to show that $P(n)$ holds for all n, where $P(n)$ says $x = 0 \lor (\exists x' : x = x' + 1)$. The base case is when $n = 0$, and here the induction hypothesis holds by the addition rule. For the induction step, we are given that $P(x)$ holds, and want to show that $P(x+1)$ holds. In this case, we can do this easily by observing that $P(x+1)$ expands to $(x+1) = 0 \lor (\exists x' : x + 1 = x' + 1)$. So let $x' = x$ and we are done.[1]

Here's a less trivial example. So far we have not defined exponentiation. Let's solve this by declaring

$$x^0 = 1 \tag{5.1.2}$$

$$x^{n+1} = x \cdot x^n \tag{5.1.3}$$

where n ranges over all elements of $\mathbb{N}$.

This is a **recursive definition**: to compute, say, 2^4, we expand it out using (5.1.3) until we bottom out at (5.1.2). This gives $2^4 = 2 \cdot 2^3 = 2 \cdot 2 \cdot 2^2 = 2 \cdot 2 \cdot 2 \cdot 2^0 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 1 = 16$.

If we want to *prove* something about our newly-defined operation, we are likely to end up using induction.

Theorem 5.1.1. *If $a > 1$, then $a^n > 1$ for all $n > 0$*

Proof. Let $a > 1$.

Since we are looking at a universal statement about almost all naturals, we're going to prove it by induction. This requires choosing an induction hypothesis. We can rewrite the claim slightly as for all n, $n > 0$ implies $a^n > 1$.

Base case: If $n = 0$, then $n \not> 0$, so the induction hypothesis holds vacuously.

[1] This is admittedly not a very interesting use of induction, since we don't actually use $P(x)$ in proving $P(x+1)$.

Induction step: Suppose the induction hypothesis holds for n, i.e., that $n > 0 \to a^n > 1$. We want to show that it also holds for $n+1$. Annoyingly, there are two cases we have to consider:

1. $n = 0$. Then we can compute $a^1 = a \cdot a^0 = a \cdot 1 = a > 1$.
2. $n > 0$. The induction hypothesis now gives $a^n > 1$ (since in this case the premise $n > 0$ holds), so $a^{n+1} = a \cdot a^n > a \cdot 1 > 1$.

$\square$

5.2 Alternative base cases

One of the things that is apparent from the proof of Theorem 5.1.1 is that being forced to start at 0 may require painful circumlocutions if 0 is not the first natural for which we the predicate we care about holds. So in practice it is common to use a different base case. This gives a generalized version of the induction schema that works for any integer base:

$$(P(z_0) \wedge \forall z \in \mathbb{Z}, z \geq z_0 : (P(z) \to P(z+1))) \to \forall z \in \mathbb{Z}, z \geq z_0 : P(z) \tag{5.2.1}$$

Intuitively, this works for the same reason (5.1.1) works: if P is true for z_0, then any larger integer can be reached by applying $+1$ enough times, and each $+1$ operation preserves P. If we want to prove it formally, observe that (5.2.1) turns into (5.1.1) if we do a change of variables and define $Q(n) = P(z - z_0)$.

Here's an example of starting at a non-zero base case:

Theorem 5.2.1. *Let $n \in \mathbb{N}$. If $n \geq 4$, then $2^n \geq n^2$.*

Proof. Base case: Let $n = 4$, then $2^n = 16 = n^2$.

For the induction step, assume $2^n \geq n^2$. We need to show that $2^{n+1} \geq (n+1)^2 = n^2 + 2n + 1$. Using the assumption and the fact that $n \geq 4$, we can compute

$$\begin{aligned}
2^{n+1} &= 2 \cdot 2^n \\
&\geq 2n^2 \\
&= n^2 + n^2 \\
&\geq n^2 + 4n \\
&= n^2 + 2n + 2n \\
&\geq n^2 + 2n + 1 \\
&= (n+1)^2.
\end{aligned}$$

□

5.3 Recursive definitions work

In §5.1, we defined x^n recursively, by giving rules for computing x^0 and computing x^{n+1} given x^n. We can show using induction that such definitions actually work.

Lemma 5.3.1. *Let S be some codomain, and let $f(n) : \mathbb{N} \to S$ satisfy*

$$\begin{aligned} f(0) &= x_0 \\ f(n+1) &= g(f(n)) \end{aligned}$$

Then there is a unique function f with this property.

Proof. Suppose that there is some f' such that $f'(0) = x_0$ and $f'(n+1) = g(f'(n))$. We can show by induction on n that $f'(n) = f(n)$ for all n. The base case is $f'(0) = x_0 = f(0)$. For the induction step, if $f'(n) = f(n)$, then $f'(n+1) = g(f'(n)) = g(f(n)) = f(n+1)$. □

5.4 Other ways to think about induction

In set-theoretic terms, the principle of induction says that if S is a subset of $\mathbb{N}$, and both

1. $0 \in S$ and
2. $x \in S$ implies $x + 1 \in S$,

then $S = \mathbb{N}$.

This is logically equivalent to the fact that the naturals are **well-ordered**. This means that any non-empty subset S of $\mathbb{N}$ has a smallest element. More formally: for any $S \subseteq \mathbb{N}$, if $S \neq \emptyset$, then there exists $x \in S$ such that for all $y \in S$, $x \leq y$.

It's easy to see that well-ordering implies induction. Let S be a subset of $\mathbb{N}$, and consider its complement $\mathbb{N} \setminus S$. Then either $N \setminus S$ is empty, meaning $S = \mathbb{N}$, or $\mathbb{N} \setminus S$ has a least element y. But in the second case either $y = 0$ and $0 \notin S$ or $y = x + 1$ for some x and $x \in S$ but $x + 1 \notin S$. So $S \neq \mathbb{N}$ implies $0 \notin S$ or there exists x such that $x \in S$ but $x + 1 \notin S$. Taking the contraposition of this statement gives induction.

The converse is a little trickier, since we need to figure out how to use induction to prove things about subsets of $\mathbb{N}$, but induction only talks about elements of $\mathbb{N}$. The trick is consider only the part of S that is smaller than some variable n, and show that any S that contains an element smaller than n has a smallest element.

Lemma 5.4.1. *For all $n \in \mathbb{N}$, if S is a subset of $\mathbb{N}$ that contains an element less than or equal to n, then S has a smallest element.*

Proof. By induction on n.

The base case is$n = 0$. Here $0 \in S$ and $0 \leq x$ for any $x \in \mathbb{N}$, so in particular $0 \leq x$ for any $x \in S$, making 0 the smallest element in S.

For the induction step, suppose that the claim in the lemma holds for n. To show that it holds for $n+1$, suppose that $n+1 \in S$. Then either (a) S contains an element less than or equal to n, so S has a smallest element by the induction hypothesis, or (b) S does not contain an element less than or equal to n. But in this second case, S must contain $n+1$, and since there are no elements less than $n+1$ in S, $n+1$ is the smallest element. $\square$

To show the full result, let n be some element of S. Then S contains an element less than or equal to n, and so S contains a smallest element.

5.5 Strong induction

Sometimes when proving that the induction hypothesis holds for $n+1$, it helps to use the fact that it holds for all $n' < n+1$, not just for n. This sort of argument is called **strong induction**. Formally, it's equivalent to simple induction: the only difference is that instead of proving $\forall k : P(k) \rightarrow P(k+1)$, we prove $\forall k : (\forall m \leq k : Q(m)) \rightarrow Q(k+1)$. But this is exactly the same thing if we let $P(k) \equiv \forall m \leq k : Q(m)$, since if $\forall m \leq k : Q(m)$ implies $Q(k+1)$, it also implies $\forall m \leq k+1 : Q(m)$, giving us the original induction formula $\forall k P(k) \rightarrow P(k+1)$.

As with simple induction, it can be helpful to think of this approach backwards, by taking the contraposition. This gives the **method of infinite descent**, due to Fermat. The idea is to give a method for taking some n_0 for which $P(n_0)$ doesn't hold, and use it to show that there is some $n_1 < n_0$ for which $P(n_1)$ also doesn't hold. Repeating this process forever gives an infinite descending sequence $n_0 > n_1 > n_2 > \ldots$, which would give a subset of $\mathbb{N}$ with no smallest element. As with any recursive definition, the "repeat" step is secretly using an induction argument.

An alternative formulation of the method of infinite descent is that since the naturals are well-ordered, if there is some n for which $P(n)$ doesn't hold, there is a smallest n for which it doesn't hold. But if we can take this n can find a smaller n', then we get a contradiction.

Historical note: Fermat may have used this technique to construct a plausible but invalid proof of his famous "Last Theorem" that $a^n + b^n = c^n$ has no non-trivial integer solutions for $n > 2$.

5.5.1 Examples

- Every $n > 1$ can be factored into a product of one or more prime numbers.[2] Proof: By induction on n. The base case is $n = 2$, which factors as $2 = 2$ (one prime factor). For $n > 2$, either (a) n is prime itself, in which case $n = n$ is a prime factorization; or (b) n is not prime, in which case $n = ab$ for some a and b, both greater than 1. Since a and b are both less than n, by the induction hypothesis we have $a = p_1 p_2 \dots p_k$ for some sequence of one or more primes and similarly $b = p'_1 p'_2 \dots p'_{k'}$. Then $n = p_1 p_2 \dots p_k p'_1 p'_2 \dots p'_{k'}$ is a prime factorization of n.

- Every deterministic bounded two-player perfect-information game that can't end in a draw has a winning strategy for one of the players. A perfect-information game is one in which both players know the entire state of the game at each decision point (like Chess or Go, but unlike Poker or Bridge); it is deterministic if there is no randomness that affects the outcome (this excludes Backgammon and Monopoly, some variants of Poker, and multiple hands of Bridge), and it's bounded if the game is guaranteed to end in at most a fixed number of moves starting from any reachable position (this also excludes Backgammon and Monopoly). Proof: For each position x, let $b(x)$ be the bound on the number of moves made starting from x. Then if y is some position reached from x in one move, we have $b(y) < b(x)$ (because we just used up a move). Let $f(x) = 1$ if the first player wins starting from position x and $f(x) = 0$ otherwise. We claim that f is well-defined. Proof: If $b(x) = 0$, the game is over, and so $f(x)$ is either 0 or 1, depending on who just won. If $b(x) > 0$, then $f(x) = \max \{f(y) \mid y \text{ is a successor to } x\}$ if it's the first player's turn to move and $f(x) = \min \{f(y) \mid y \text{ is a successor to } x\}$ if it's the second player's turn to move. In either case each $f(y)$ is well-defined (by the induction hypothesis) and so $f(x)$ is also well-defined.

[2] A number is **prime** if it can't be written as $a \cdot b$ where a and b are both greater than 1.

- The **division algorithm**: For each $n, m \in \mathbb{N}$ with $m \neq 0$, there is a unique pair $q, r \in \mathbb{N}$ such that $n = qm + r$ and $0 \leq r < m$. Proof: Fix m then proceed by induction on n. If $n < m$, then if $q > 0$ we have $n = qm + r \geq 1 \cdot m \geq m$, a contradiction. So in this case $q = 0$ is the only solution, and since $n = qm + r = r$ we have a unique choice of $r = n$. If $n \geq m$, by the induction hypothesis there is a unique q' and r' such that $n - m = q'm + r'$ where $0 \leq r' < m$. But then $q = q' + 1$ and $r = r'$ satisfies $qm + r = (q' - 1 + 1)m + r = (q'm + r') + m = (n - m) + m = n$. To show that this solution is unique, if there is some other q'' and r'' such that $q''m + r'' = n$, then $(q'' - 1)m + r'' = n - m = q'm + r'$, and by the uniqueness of q' and r' (ind. hyp. again), we have $q'' - 1 = q' = q - 1$ and $r'' = r' = r$, giving that $q'' = q$ and $r'' = r$. So q and r are unique.

5.6 Recursively-defined structures

A definition with the structure of an inductive proof (give a base case and a rule for building bigger structures from smaller ones) Structures defined in this way are **recursively-defined**.

Examples of recursively-defined structures:

Finite Von Neumann ordinals A finite von Neumann ordinal is either (a) the empty set $\emptyset$, or (b) $x \cup \{x\}$, where x is a finite von Neumann ordinal.

Complete binary trees A complete binary tree consists of either (a) a *leaf node*, or (b) an *internal node* (the *root*) with two complete binary trees as children (or *subtrees*).

Boolean formulas A boolean formula consists of either (a) a variable, (b) the negation operator applied to a Boolean formula, (c) the AND of two Boolean formulas, or (d) the OR of two Boolean formulas. A monotone Boolean formula is defined similarly, except that negations are forbidden.

Finite sequences, recursive version Before we defined a finite sequence as a function from some natural number (in its set form: $n = \{0, 1, 2, ..., n - 1\}$) to some set S. We could also define a finite sequence over S recursively, by the rule: $\langle\rangle$ (the empty sequence) is a finite sequence, and if a is a finite sequence and $x \in S$, then (x, a) is a finite sequence. (Fans of LISP will recognize this method immediately.)

The key point is that in each case the definition of an object is **recursive**—the object itself may appear as part of a larger object. Usually we assume that this recursion eventually bottoms out: there are some base cases (e.g. leaves of complete binary trees or variables in Boolean formulas) that do not lead to further recursion. If a definition doesn't bottom out in this way, the class of structures it describes might not be well-defined (i.e., we can't tell if some structure is an element of the class or not).

5.6.1 Functions on recursive structures

We can also define functions on recursive structures recursively:

The depth of a binary tree For a leaf, 0. For a tree consisting of a root with two subtrees, $1 + \max(d_1, d_2)$, where d_1 and d_2 are the depths of the two subtrees.

The value of a Boolean formula given a particular variable assignment For a variable, the value (true or false) assigned to that variable. For a negation, the negation of the value of its argument. For an AND or OR, the AND or OR of the values of its arguments. (This definition is not quite as trivial as it looks, but it's still pretty trivial.)

Or we can define ordinary functions recursively:

The Fibonacci series Let $F(0) = F(1) = 1$. For $n > 1$, let $F(n) = F(n-1) + F(n-2)$.

Factorial Let $0! = 1$. For $n > 0$, let $n! = n \cdot ((n-1)!)$.

5.6.2 Recursive definitions and induction

Recursive definitions have the same form as an induction proof. There are one or more base cases, and one or more recursion steps that correspond to the induction step in an induction proof. The connection is not surprising if you think of a definition of some class of objects as a predicate that identifies members of the class: a recursive definition is just a formula for writing induction proofs that say that certain objects are members.

Recursively-defined objects and functions also lend themselves easily to induction proofs about their properties; on general structures, such induction arguments go by the name of *structural induction*.

5.6.3 Structural induction

For finite structures, we can do induction over the structure. Formally we can think of this as doing induction on the size of the structure or part of the structure we are looking at.

Examples:

Every complete binary tree with n leaves has $n-1$ internal nodes Base case is a tree consisting of just a leaf; here $n = 1$ and there are $n - 1 = 0$ internal nodes. The induction step considers a tree consisting of a root and two subtrees. Let n_1 and n_2 be the number of leaves in the two subtrees; we have $n_1 + n_2 = n$; and the number of internal nodes, counting the nodes in the two subtrees plus one more for the root, is $(n_1 - 1) + (n_2 - 1) + 1 = n_1 + n_2 - 1 = n - 1$.

Monotone Boolean formulas generate monotone functions What this means is that changing a variable from false to true can never change the value of the formula from true to false. Proof is by induction on the structure of the formula: for a naked variable, it's immediate. For an AND or OR, observe that changing a variable from false to true can only leave the values of the arguments unchanged, or change one or both from false to true (induction hypothesis); the rest follows by staring carefully at the truth table for AND or OR.

Bounding the size of a binary tree with depth d We'll show that it has at most $2^{d+1} - 1$ nodes. Base case: the tree consists of one leaf, $d = 0$, and there are $2^{0+1} - 1 = 2 - 1 = 1$ nodes. Induction step: Given a tree of depth $d > 1$, it consists of a root (1 node), plus two subtrees of depth at most $d - 1$. The two subtrees each have at most $2^{d-1+1} - 1 = 2^d - 1$ nodes (induction hypothesis), so the total number of nodes is at most $2(2^d - 1) + 1 = 2^{d+1} + 2 - 1 = 2^{d+1} - 1$.

Chapter 6

Summation notation

6.1 Summations

Summations are the discrete versions of integrals; given a sequence $x_a, x_{a+1}, \dots, x_b$, its sum $x_a + x_{a+1} + \cdots + x_b$ is written as $\sum_{i=a}^{b} x_i$.

The large jagged symbol is a stretched-out version of a capital Greek letter sigma. The variable i is called the **index of summation**, a is the **lower bound** or **lower limit**, and b is the **upper bound** or **upper limit**. Mathematicians invented this notation centuries ago because they didn't have `for` loops; the intent is that you loop through all values of i from a to b (including both endpoints), summing up the body of the summation for each i.

If $b < a$, then the sum is zero. For example,

$$\sum_{i=0}^{-5} \frac{2^i \sin i}{i^3} = 0.$$

This rule mostly shows up as an extreme case of a more general formula, e.g.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2},$$

which still works even when $n = 0$ or $n = -1$ (but not for $n \leq -2$).

Summation notation is used both for laziness (it's more compact to write $\sum_{i=0}^{n}(2i+1)$ than $1+3+5+7+\cdots+(2n+1)$) and precision (it's also more clear exactly what you mean).

6.1.1 Formal definition

For finite sums, we can formally define the value by either of two recurrences:

$$\sum_{i=a}^{b} f(i) = \begin{cases} 0 & \text{if } b < a \\ f(a) + \sum_{i=a+1}^{b} f(i) & \text{otherwise.} \end{cases} \tag{6.1.1}$$

$$\sum_{i=a}^{b} f(i) = \begin{cases} 0 & \text{if } b < a \\ f(b) + \sum_{i=a}^{b-1} f(i) & \text{otherwise.} \end{cases} \tag{6.1.2}$$

In English, we can compute a sum recursively by computing either the sum of the last $n-1$ values or the first $n-1$ values, and then adding in the value we left out. (For infinite sums we need a different definition; see below.)

6.1.2 Scope

The *scope* of a summation extends to the first addition or subtraction symbol that is not enclosed in parentheses or part of some larger term (e.g., in the numerator of a fraction). So

$$\sum_{i=1}^{n} i^2 + 1 = \left(\sum_{i=1}^{n} i^2\right) + 1 = 1 + \sum_{i=1}^{n} i^2 \neq \sum_{i=1}^{n} (i^2+1).$$

Since this can be confusing, it is generally safest to wrap the sum in parentheses (as in the second form) or move any trailing terms to the beginning. An exception is when adding together two sums, as in

$$\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n^2} i = \left(\sum_{i=1}^{n} i^2\right) + \left(\sum_{i=1}^{n^2} i\right).$$

Here the looming bulk of the second sigma warns the reader that the first sum is ending; it is much harder to miss than the relatively tiny plus symbol in the first example.

6.1.3 Summation identities

The summation operator is **linear**. This means that constant factors can be pulled out of sums:

$$\sum_{i=n}^{m} a x_i = a \sum_{i=n} x_i \tag{6.1.3}$$

and sums inside sums can be split:

$$\sum_{i=n}^{m}(x_i + y_i) = \sum_{i=n}^{m} x_i + \sum_{i \in S y_i} . \tag{6.1.4}$$

With multiple sums, the order of summation is not important, provided the bounds on the inner sum don't depend on the index of the outer sum:

$$\sum_{i=n}^{m} \sum_{j=n'}^{m'} x_{ij} = \sum_{j=n'}^{m'} \sum_{i=n}^{m} x_{ij}.$$

Products of sums can be turned into double sums of products and vice versa:

$$\left(\sum_{i=n}^{m} x_i\right)\left(\sum_{j=n'}^{m'} y_j\right) = \sum_{i=n}^{m} \sum_{j=n'}^{m'} x_i y_j.$$

These identities can often be used to transform a sum you can't solve into something simpler.

To prove these identities, use induction and (6.1.2). For example, the following lemma demonstrates a generalization of (6.1.3) and (6.1.4):

Lemma 6.1.1.

$$\sum_{i=n}^{m}(ax_i + by_i) = a\sum_{i=n}^{m} x_i + b\sum_{i=n}^{m} y_i.$$

Proof. If $m < n$, then both sides of the equation are zero. This proves that (6.1.1) holds for small m and gives us a base case for our induction at $m = n - 1$ that we can use to show it holds for larger m.

For the induction step, we want to show that (6.1.1) holds for $m + 1$ if it holds for m. This is a straightforward computation using (6.1.2) first to unpack the combined sum then to repack the split sums:

$$\begin{aligned}
\sum_{i=n}^{m+1}(ax_i + by_i) &= \sum_{i=n}^{m}(ax_i + by_i) + (ax_m + by_m) \\
&= a\sum_{i=n}^{m} x_i + b\sum_{i=n}^{m} y_i + ax_m + by_m \\
&= a\left(\sum_{i=n}^{m} x_i + x_m\right) + b\left(\sum_{i=n}^{m} y_i + y_m\right) \\
&= a\sum_{i=n}^{m+1} + b\sum_{i=n}^{m+1} y_i.
\end{aligned}$$

□

6.1.4 Choosing and replacing index variables

When writing a summation, you can generally pick any index variable you like, although i, j, k, etc., are popular choices. Usually it's a good idea to pick an index that isn't used outside the sum. Though

$$\sum_{n=0}^{n} n = \sum_{i=0}^{n} i$$

has a well-defined meaning, the version on the right-hand side is a lot less confusing.

In addition to renaming indices, you can also shift them, provided you shift the bounds to match. For example, rewriting

$$\sum_{i=1}^{n} (i-1)$$

by substituting j for $i - 1$ gives

$$\sum_{j=0}^{n-1} j,$$

which is easier to work with.

6.1.5 Sums over given index sets

Sometimes we'd like to sum an expression over values that aren't consecutive integers, or may not even be integers at all. This can be done using a sum over all indices that are members of a given index set, or in the most general form satisfy some given predicate (with the usual set-theoretic caveat that the objects that satisfy the predicate must form a set). Such a sum is written by replacing the lower and upper limits with a single subscript that gives the predicate that the indices must obey.

For example, we could sum i^2 for i in the set $\{3, 5, 7\}$:

$$\sum_{i \in \{3,5,7\}} i^2 = 3^2 + 5^2 + 7^2 = 83.$$

Or we could sum the sizes of all subsets of a given set S:

$$\sum_{A \subseteq S} |A| .$$

Or we could sum the inverses of all prime numbers less than 1000:

$$\sum_{p < 1000,\ p \text{ is prime}} 1/p.$$

Sometimes when writing a sum in this form it can be confusing exactly which variable or variables are the indices. The usual convention is that a variable is always an index if it doesn't have any meaning outside the sum, and if possible the index variable is put first in the expression under the sigma if possible. If it is not obvious what a complicated sum means, it is generally best to try to rewrite it to make it more clear; still, you may see sums that look like

$$\sum_{1 \le i < j \le n} \frac{i}{j}$$

or

$$\sum_{x \in A \subseteq S} |A|$$

where the first sum sums over all *pairs* of values (i, j) such that $1 \le i$, $i \le j$, and $j \le n$, with each pair appearing exactly once; and the second sums over all sets A that are subsets of S and contain x (assuming x and S are defined outside the summation). Hopefully, you will not run into too many sums that look like this, but it's worth being able to decode them if you do.

Sums over a given set are guaranteed to be well-defined only if the set is finite. In this case we can use the fact that there is a bijection between any finite set S and the ordinal $|S|$ to rewrite the sum as a sum over indices in $|S|$. For example, if $|S| = n$, then there exists a bijection $f : \{0 \dots n-1\} \leftrightarrow S$, so we can define

$$\sum_{i \in S} x_i = \sum_{i=0}^{n-1} x_{f(i)}. \tag{6.1.5}$$

This allows us to apply (6.1.2) to decompose the sum further:

$$\sum_{i \in S} x_i = \begin{cases} 0 & \text{if } S = \emptyset, \\ \left(\sum_{i \in S \setminus z} x_i\right) + x_z & \text{if } z \in S. \end{cases} \tag{6.1.6}$$

The idea is that for any particular $z \in S$, we can always choose a bijection that makes $z = f(|S| - 1)$.

If S is infinite, computing the sum is trickier. For countable S, where there is a bijection $f : \mathbb{N} \leftrightarrow S$, we can sometimes rewrite

$$\sum_{i \in S} x_i = \sum_{i=0}^{\infty} x_{f(i)}.$$

and use the definition of an infinite sum (given below). Note that if the x_i have different signs the result we get may depend on which bijection we choose. For this reason such infinite sums are probably best avoided unless you can explicitly use $\mathbb{N}$ or a subset of $\mathbb{N}$ as the index set.

6.1.6 Sums without explicit bounds

When the index set is understood from context, it is often dropped, leaving only the index, as in $\sum_i i^2$. This will generally happen only if the index spans all possible values in some obvious range, and can be a mark of sloppiness in formal mathematical writing. Theoretical physicists adopt a still more lazy approach, and leave out the $\sum_i$ part entirely in certain special types of sums: this is known as the **Einstein summation convention** after the notoriously lazy physicist who proposed it.

6.1.7 Infinite sums

Sometimes you may see an expression where the upper limit is infinite, as in

$$\sum_{i=0}^{\infty} \frac{1}{i^2}.$$

The meaning of this expression is the **limit** of the series s obtained by taking the sum of the first term, the sum of the first two terms, the sum of the first three terms, etc. The limit **converges** to a particular value x if for any $\epsilon > 0$, there exists an N such that for all $n > N$, the value of s_n is within ϵ of x (formally, $|s_n - x| < \epsilon$). We will see some examples of infinite sums when we look at generating functions in §11.3.

6.1.8 Double sums

Nothing says that the expression inside a summation can't be another summation. This gives double sums, such as in this rather painful definition of multiplication for non-negative integers:

$$a \times b \overset{\text{def}}{=} \sum_{i=1}^{a} \sum_{j=1}^{b} 1.$$

If you think of a sum as a *for* loop, a double sum is two nested *for* loops. The effect is to sum the innermost expression over all pairs of values of the two indices.

Here's a more complicated double sum where the limits on the inner sum depend on the index of the outer sum:

$$\sum_{i=0}^{n} \sum_{j=0}^{i} (i+1)(j+1).$$

When $n = 1$, this will compute $(0{+}1)(0{+}1)+(1{+}1)(0{+}1)+(1{+}1)(1{+}1) = 7$. For larger n the number of terms grows quickly.

There are also triple sums, quadruple sums, etc.

6.2 Products

What if you want to multiple a series of values instead of add them? The notation is the same as for a sum, except that you replace the sigma with a pi, as in this definition of the factorial function for non-negative n:

$$n! \overset{\text{def}}{=} \prod_{i=1}^{n} i = 1 \cdot 2 \cdot \dots \cdot n.$$

The other difference is that while an empty sum is defined to have the value 0, an empty product is defined to have the value 1. The reason for this rule (in both cases) is that an empty sum or product should return the **identity element** for the corresponding operation—the value that when added to or multiplied by some other value x doesn't change x. This allows writing general rules like:

$$\sum_{i \in A} f(i) + \sum_{i \in B} f(i) = \sum_{i \in A \cup B} f(i)$$

$$\left(\prod_{i \in A} f(i)\right) \cdot \left(\prod_{i \in B} f(i)\right) = \prod_{i \in A \cup B} f(i)$$

which holds as long as $A \cap B = \emptyset$. Without the rule that the sum of an empty set was 0 and the product 1, we'd have to put in a special case for when one or both of A and B were empty.

Note that a consequence of this definition is that $0! = 1$.

6.3 Other big operators

Some more obscure operators also allow you to compute some aggregate over a series, with the same rules for indices, lower and upper limits, etc., as $\sum$ and $\prod$. These include:

- Big AND:

$$\bigwedge_{x \in S} P(x) \equiv P(x_1) \wedge P(x_2) \wedge \ldots \equiv \forall x \in S : P(x).$$

- Big OR:

$$\bigvee_{x \in S} P(x) \equiv P(x_1) \vee P(x_2) \vee \ldots \equiv \exists x \in S : P(x).$$

- Big Intersection:

$$\bigcap_{i=1}^{n} A_i = A_1 \cap A_2 \cap \ldots \cap A_n.$$

- Big Union:

$$\bigcup_{i=1}^{n} A_i = A_1 \cup A_2 \cup \ldots \cup A_n.$$

These all behave pretty much the way one would expect. One issue that is not obvious from the definition is what happens with an empty index set. Here the rule as with sums and products is to return the identity element for the operation. This will be True for AND, False for OR, and the empty set for union; for intersection, there is no identity element in general, so the intersection over an empty collection of sets is undefined.

6.4 Closed forms

When confronted with some nasty sum, it is nice to be able to convert into a simpler expression that doesn't contain any summation signs or other operators that iterate over some bound variable. Such an expression is known as a **closed form**.

It is not always possible to do this, and the problem of finding a closed form for a summation is very similar to the problem of computing an integral (see §F.3): in both cases the techniques available are mostly limited to massaging the summation until it turns into something whose simpler expression you remember. To do this, it helps to both (a) have a big toolbox of summations with known values, and (b) have some rules for manipulating summations to get them into a more convenient form. We'll start with the toolbox.

6.4.1 Some standard sums

Here are the three formulas you should either memorize or remember how to derive:

$$\sum_{i=1}^{n} 1 = n$$
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$
$$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$$

Rigorous proofs of these can be obtained by induction on n.

For not so rigorous proofs, the second identity can be shown using a trick alleged to have been invented by the legendary 18th-century mathematician Carl Friedrich Gauss, at a frighteningly early age, by adding up two copies of the sequence running in opposite directions, one term at a time:

$$\begin{array}{ccccccccccc}
S & = & 1 & + & 2 & + & \dots & + & n & & \\
S & = & n & + & n-1 & + & \dots & + & 1 & & \\
\hline
2S & = & (n+1) & + & (n+1) & + & \dots & + & (n+1) & = & n(n+1),
\end{array}$$

and from $2S = n(n+1)$ we get $S = n(n+1)/2$.

For the last identity, start with

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r},$$

which holds when $|r| < 1$. The proof is that if

$$S = \sum_{i=0}^{\infty} r^i$$

then

$$rS = \sum_{i=0}^{\infty} r^{i+1} = \sum_{i=1}^{\infty} r^i$$

and so

$$S - rS = r^0 = 1.$$

Solving for S gives $S = 1/(1-r)$.

We can now get the sum up to n by subtracting off the extra terms starting with $rn+1$:

$$\sum_{i=0}^{n} r^i = \sum_{i=0}^{\infty} r^i - r^{n+1}\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} - \frac{r^{n+1}}{1-r} = \frac{1-r^{n+1}}{1-r}.$$

Amazingly enough, this formula works even when r is greater than 1. If r is equal to 1, then the formula doesn't work (it requires dividing zero by zero), but there is an easier way to get the solution.

These standard summations can be combined with linearity to solve more complicated problems. For example, we can directly compute

$$\begin{aligned}\sum_{i=0}^{n}(3\cdot 2^n + 5) &= 3\sum_{i=0}^{n} 2^n + 5\sum_{i=0}^{n} 1 \\ &= 3\cdot\left(2^{n+1}-1\right) + 5(n+1) \\ &= 3\cdot 2^{n+1} + 5n + 2.\end{aligned}$$

Other useful summations can be found in various places. Rosen [Ros12] and and Graham *et al.* [GKP94] both provide tables of sums in their chapters on generating functions. But it is usually better to be able to reconstruct the solution of a sum rather than trying to memorize such tables.

6.4.2 Guess but verify

If nothing else works, you can try using the "guess but verify" method, which also works more generally for identifying sequences defined recursively. Here we write out the values of the summation for the first few values of the upper limit (for example), and hope that we recognize the sequence. If we do, we can then try to prove that a formula for the sequence of sums is correct by induction.

Example: Suppose we want to compute

$$S(n) = \sum_{k=1}^{n}(2k-1)$$

but that it doesn't occur to us to split it up and use the $\sum_{k=1}^{n} k$ and $\sum_{k=1}^{n} 1$ formulas. Instead, we can write down a table of values:

n	S(n)
0	0
1	1
2	$1 + 3 = 4$
3	$1 + 3 + 5 = 9$
4	$1 + 3 + 5 + 7 = 16$
5	$1 + 3 + 5 + 7 + 9 = 25$

At this point we might guess that $S(n) = n^2$. To verify this, observe that it holds for $n = 0$, and for larger n we have $S(n) = S(n-1)+(2n-1) = (n-1)^2 + 2n - 1 = n^2 - 2n + 1 - 2n - 1 = n^2$. So we can conclude that our guess was correct.

6.4.3 Ansatzes

A slightly more sophisticated approach to guess but verify involves guessing the form of the solution, but leaving a few parameters unfixed so that we can adjust them to match the actual data. This parameterized guess is called an **ansatz**, from the German word for "starting point," because guesswork sound much less half-baked if you can refer to it in German.

To make this works, it helps to have some idea of what the solution to a sum might look like. One useful rule of thumb is that a sum over a degree-d polynomial is usually a degree-$(d + 1)$ polynomial.

For example, let's guess that

$$\sum_{i=0}^{n} i^2 = c_3 n^3 + c_2 n^2 + c_1 n + c_0, \tag{6.4.1}$$

when $n \geq 0$.

Under the assumption that (6.4.1) holds, we can plug in $n = 0$ to get $\sum_{i=0}^{0} i^2 = 0 = c_0$. This means that we only need to figure out c_3, c_2, and c_1.

Plugging in some small values for n gives

$$\begin{aligned} 0 + 1 = 1 &= c_3 + c_2 + c_1 \\ 0 + 1 + 4 = 5 &= 8c_3 + 4c_2 + 2c_1 \\ 0 + 1 + 4 + 9 = 14 &= 27c_3 + 8c_2 + 3c_1 \end{aligned}$$

With some effort, this system of equations can be solved to obtain $c_3 = 1/3, c_2 = 1/2, c_1 = 1/6$, giving the formula

$$\sum_{i=0}^{n} i^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n. \tag{6.4.2}$$

We still don't know that (6.4.2) actually works: in principle, we could have fit coefficients $c_0 \dots c_3$ to any sequence of four values. To show that it does work, we do an induction argument.

The base case is $n = 0$, which we know works. For the induction step, compute

$$\begin{aligned}\frac{1}{3}(n+1)^3 + \frac{1}{2}(n+1)^2 + \frac{1}{6}(n+1) &= \left(\frac{1}{3}n^3 + n^2 + n + \frac{1}{3}\right) + \left(\frac{1}{2}n^2 + n + \frac{1}{2}\right) + \left(\frac{1}{6}n + \frac{1}{6}\right) \\ &= \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n + n^2 + 2n + 1 \\ &= \sum_{i=0}^{n} i^2 + (n+1)^2 \\ &= \sum_{i=0}^{n+1} i^2.\end{aligned}$$

6.4.4 Strategies for asymptotic estimates

Mostly in algorithm analysis, we do not need to compute sums exactly, because we are just going to wrap the result up in some asymptotic expression anyway (see Chapter 7). This makes our life much easier, because we only need an approximate solution.

Here's my general strategy for computing sums:

6.4.4.1 Pull out constant factors

Pull as many constant factors out as you can (where constant in this case means anything that does not involve the summation index). Example: $\sum_{i=1}^{n} \frac{n}{i} = n \sum_{i=1}^{n} \frac{1}{i} = nH_n = \Theta(n \log n)$. (See harmonic series below.)

6.4.4.2 Bound using a known sum

See if it's bounded above or below by some other sum whose solution you already know. Good sums to try (you should memorize all of these):

Geometric series $\sum_{i=0}^{n} x^i = \frac{1-x^{n+1}}{1-x}$ and $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$.

The way to recognize a geometric series is that the ratio between adjacent terms is constant. If you memorize the second formula, you can rederive the first one. If you're Gauss, you can skip memorizing the second formula.

A useful trick to remember for geometric series is that if x is a constant that is not exactly 1, the sum is always big-Theta of its largest term. So for example $\sum_{i=1}^{n} 2^i = \Theta(2^n)$ (the exact value is $2^{n+1} - 2$), and $\sum_{i=1}^{n} 2^{-i} = \Theta(1)$ (the exact value is $1 - 2^{-n}$).

If the ratio between terms equals 1, the formula doesn't work; instead, we have a constant series (see below).

Constant series $\sum_{i=1}^{n} 1 = n$.

Arithmetic series The simplest arithmetic series is $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. The way to remember this formula is that it's just n times the average value $(n+1)/2$. The way to recognize an arithmetic series is that the difference between adjacent terms is constant. The general arithmetic series is of the form $\sum_{i=1}^{n}(ai + b) = \sum_{i=1}^{n} ai + \sum_{i=1}^{n} b = an(n+1)/2 + bn$. Because the general series expands so easily to the simplest series, it's usually not worth memorizing the general formula.

Harmonic series $\sum_{i=1}^{n} 1/i = H_n = \Theta(n \log n)$.

Can be rederived using the integral technique given below or by summing the last half of the series, so this is mostly useful to remember in case you run across H„n„ (the ”n-th harmonic number”).

6.4.4.3 Bound part of the sum

See if there's some part of the sum that you can bound. For example, $\sum_{i=1}^{n} i^3$ has a (painful) exact solution, or can be approximated by the integral trick described below, but it can very quickly be solved to within a constant factor by observing that $\sum_{i=1}^{n} i^3 \leq \sum_{i=1}^{n} n^3 = O(n^4)$ and $\sum_{i=1}^{n} i^3 \geq \sum_{i=n/2}^{n} i^3 \geq \sum_{i=n/2}^{n} (n/2)^3 = \Omega(n^4)$.

6.4.4.4 Integrate

Integrate. If $f(n)$ is non-decreasing and you know how to integrate it, then $\int_{a-1}^{b} f(x)dx \leq \sum_{i=a}^{b} f(i) \leq \int_{a}^{b+1} f(x)dx$, which is enough to get a big-Theta bound for almost all functions you are likely to encounter in algorithm analysis. If you don't know how to integrate it, see §F.3.

6.4.4.5 Grouping terms

Try grouping terms together. For example, the standard trick for showing that the harmonic series is unbounded in the limit is to argue that $1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \ldots \geq 1 + 1/2 + (1/4 + 1/4) + (1/8 +$

$1/8 + 1/8 + 1/8) + \ldots \geq 1 + 1/2 + 1/2 + 1/2 + \ldots$. I usually try everything else first, but sometimes this works if you get stuck.

6.4.4.6 Oddities

One oddball sum that shows up occasionally but is hard to solve using any of the above techniques is $\sum_{i=1}^{n} a^i i$. If $a < 1$, this is $\Theta(1)$ (the exact formula for $\sum_{i=1}^{\infty} a^i i$ when $a < 1$ is $a/(1-a)^2$, which gives a constant upper bound for the sum stopping at n); if $a = 1$, it's just an arithmetic series; if $a > 1$, the largest term dominates and the sum is $\Theta(a^n n)$ (there is an exact formula, but it's ugly—if you just want to show it's $O(a^n n)$, the simplest approach is to bound the series $\sum_{i=0}^{n-1} a^{n-i}(n-i)$ by the geometric series $\sum_{i=0}^{n-1} a^{n-i} n \leq a^n n/(1-a^{-1}) = O(a^n n)$. I wouldn't bother memorizing this one provided you remember how to find it in these notes.

6.4.4.7 Final notes

In practice, almost every sum you are likely to encounter in algorithm analysis will be of the form $\sum_{i=1}^{n} f(n)$ where $f(n)$ is exponential (so that it's bounded by a geometric series and the largest term dominates) or polynomial (so that $f(n/2) = \Theta(f(n))$) and the sum is $\Theta(nf(n))$ using the $\sum_{i=n/2}^{n} f(n) = \Omega(nf(n))$ lower bound).

Graham *et al.* [GKP94] spend a lot of time on computing sums exactly. The most generally useful technique for doing this is to use generating functions (see §11.3).

Chapter 7

Asymptotic notation

Asymptotic notation is a tool for describing the behavior of functions on large values, which is used extensively in the analysis of algorithms.

7.1 Definitions

$O(f(n))$ A function $g(n)$ is in $O(f(n))$ ("**big O** of $f(n)$") if there exist constants $c > 0$ and N such that $|g(n)| \leq c\,|f(n)|$ for all $n > N$.

$\Omega(f(n))$ A function $g(n)$ is in $\Omega(f(n))$ ("**big Omega** of $f(n)$") if there exist constants $c > 0$ and N such that $|g(n)| \geq c\,|f(n)|$ for all $n > N$.

$\Theta(f(n))$ A function $g(n)$ is in $\Theta(f(n))$ ("**big Theta** of $f(n)$") if there exist constants $c_1 > 0$, $c_2 > 0$, and N such that $c_1\,|f(n)| \leq |g(n)| \leq c_2\,|f(n)|$ for all $n > N$. This is equivalent to saying that $g(n)$ is in both $O(f(n))$ and $\Omega(f(n))$.

$o(f(n))$ A function $g(n)$ is in $o(f(n))$ ("**little o** of $f(n)$") if for *every* $c > 0$ there exists an N such that $|g(n)| \leq c\,|f(n)|$ for all $n > N$. This is equivalent to saying that $\lim_{n\to\infty} g(n)/f(n) = 0$.

$\omega(f(n))$ A function $g(n)$ is in $\omega(f(n)$ ("**little omega** of $f(n)$") if for *every* $c > 0$ there exists an N such that $|g(n)| \geq c\,|f(n)|$ for all $n > N$. This is equivalent to saying that $\lim_{n\to\infty} |g(n)|\,/\,|f(n)|$ diverges to infinity.

7.2 Motivating the definitions

Why would we use this notation?

- Constant factors vary from one machine to another. The c factor hides this. If we can show that an algorithm runs in $O(n^2)$ time, we can be confident that it will continue to run in $O(n^2)$ time no matter how fast (or how slow) our computers get in the future.
- For the N threshold, there are several excuses:
  - Any problem can theoretically be made to run in $O(1)$ time for any finite subset of the possible inputs (e.g. all inputs expressible in 50 MiB or less), by prefacing the main part of the algorithm with a very large table lookup. So it's meaningless to talk about the relative performance of different algorithms for bounded inputs.
  - If $f(n) > 0$ for all n, then we can get rid of N (or set it to zero) by making c large enough. But some functions $f(n)$ take on zero—or undefined—values for interesting n (e.g., $f(n) = n^2$ is zero when n is zero, and $f(n) = \log n$ is undefined for $n = 0$ and zero for $n = 1$). Allowing the minimum N lets us write $O(n^2)$ or $O(\log n)$ for classes of functions that we would otherwise have to write more awkwardly as something like $O(n^2 + 1)$ or $O(\log(n + 2))$.
  - Putting the $n > N$ rule in has a natural connection with the definition of a limit, where the limit as n goes to infinity of $g(n)$ is defined to be x if for each $\epsilon > 0$ there is an N such that $|g(n) - x| < \epsilon$ for all $n > N$. Among other things, this permits the limit test that says $g(n) = O(f(n))$ if the $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ exists and is finite.

7.3 Proving asymptotic bounds

Most of the time when we use asymptotic notation, we compute bounds using stock theorems like $O(f(n))+O(g(n)) = O(\max(f(n), g(n))$ or $O(cf(n)) = O(f(n))$. But sometimes we need to unravel the definitions to see whether a given function fits in a given class, or to prove these utility theorems to begin with. So let's do some examples of how this works.

Theorem 7.3.1. *The function n is in $O(n^3)$.*

Proof. We must find c, N such that for all $n > N$, $|n| < c\,|n^3|$. Since n^3 is much bigger than n for most values of n, we'll pick c to be something convenient to work with, like 1. So now we need to choose N so that for all

$n > N$, $|n| < |n^3|$. It is not the case that $|n| < |n^3|$ for all n (try plotting n vs n^3 for $n < 1$) but if we let $N = 1$, then we have $n > 1$, and we just need to massage this into $n^3 > n$. There are a couple of ways to do this, but the quickest is probably to observe that squaring and multiplying by n (a positive quantity) are both increasing functions, which means that from $n > 1$ we can derive $n^2 > 1^2 = 1$ and then $n^2 \cdot n = n^3 > 1 \cdot n = n$. ☐

Theorem 7.3.2. *The function n^3 is not in $O(n)$.*

Proof. Here we need to negate the definition of $O(n)$, a process that turns all existential quantifiers into universal quantifiers and vice versa. So what we need to show is that for all $c > 0$ and N, there exists some $n > N$ for which $|n^3|$ is not less than $c\,|n|$. So fix some such $c > 0$ and N. We must find an $n > N$ for which $n^3 > cn$. Solving for n in this inequality gives $n > c^{1/2}$; so setting $n > \max(N, c^{1/2})$ finishes the proof. ☐

Theorem 7.3.3. *If $f_1(n)$ is in $O(g(n))$ and $f_2(n)$ is in $O(g(n))$, then $f_1(n) + f_2(n)$ is in $O(g(n))$.*

Proof. Since $f_1(n)$ is in $O(g(n))$, there exist constants c_1, N_1 such that for all $n > N_1$, $|f_1(n)| < c\,|g(n)|$. Similarly there exist c_2, N_2 such that for all $n > N_2$, $|f_2(n)| < c\,|g(n)|$.

To show $f_1(n) + f_2(n)$ in $O(g(n))$, we must find constants c and N such that for all $n > N$, $|f_1(n) + f_2(n)| < c\,|g(n)|$. Let's let $c = c_1 + c_2$. Then if n is greater than $\max(N_1, N_2)$, it is greater than both N_1 and N_2, so we can add together $|f_1| < c_1\,|g|$ and $|f_2| < c_2\,|g|$ to get $|f_1 + f_2| \leq |f_1| + |f_2| < (c_1 + c_2)\,|g| = c\,|g|$. ☐

7.4 Asymptotic notation hints

7.4.1 Remember the difference between big-O, big-Ω, and big-Θ

- Use big-O when you have an upper bound on a function, e.g. the zoo never got more than $O(1)$ new gorillas per year, so there were at most $O(t)$ gorillas at the zoo in year t.

- Use big-Ω when you have a lower bound on a function, e.g. every year the zoo got at least one new gorilla, so there were at least $\Omega(t)$ gorillas at the zoo in year t.

- Use big-Θ when you know the function exactly to within a constant-factor error, e.g. every year the zoo got exactly five new gorillas, so there were $\Theta(t)$ gorillas at the zoo in year t.

For the others, use little-o and ω when one function becomes vanishingly small relative to the other, e.g. new gorillas arrived rarely and with declining frequency, so there were $o(t)$ gorillas at the zoo in year t. These are not used as much as big-O, big-Ω, and big-Θ in the algorithms literature.

7.4.2 Simplify your asymptotic terms as much as possible

- $O(f(n)) + O(g(n)) = O(f(n))$ when $g(n) = O(f(n))$. If you have an expression of the form $O(f(n) + g(n))$, you can almost always rewrite it as $O(f(n))$ or $O(g(n))$ depending on which is bigger. The same goes for Ω and Θ.
- $O(cf(n)) = O(f(n))$ if c is a constant. You should never have a constant inside a big O. This includes bases for logarithms: since $\log_a x = \log_b x / \log_b a$, you can always rewrite $O(lgn)$, $O(lnn)$, or $O(log_{1.4467712}n)$ as just $O(logn)$.
- But watch out for exponents and products: $O(3^n n^{3.1178} \log^{1/3} n)$ is already as simple as it can be.

7.4.3 Remember the limit trick

If you are confused whether e.g. $\log n$ is $O(f(n))$, try computing the limit as n goes to infinity of $\frac{\log n}{n}$, and see if it's a constant (zero is OK).

You may need to use L'Hôpital's Rule to evaluate such limits if they aren't obvious. This says that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

when $f(n)$ and $g(n)$ both diverge to infinity or both converge to zero. Here f' and g' are the derivatives of f and g with respect to n; see §F.2.

7.5 Variations in notation

As with many tools in mathematics, you may see some differences in how asymptotic notation is defined and used.

7.5.1 Absolute values

Some authors leave out the absolute values. For example, Biggs [Big02] defines $f(n)$ as being in $O(g(n))$ if $f(n) \leq cg(n)$ for sufficiently large n. If $f(n)$ and $g(n)$ are non-negative, this is not an unreasonable definition. But it produces odd results if either can be negative: for example, by this definition, $-n^{1000}$ is in $O(n^2)$. Some authors define O, Ω, and Θ *only* for non-negative functions, avoiding this problem.

The most common definition (which we will use) says that $f(n)$ is in $O(g(n))$ if $|f(n)| \leq c\,|g(n)|$ for sufficiently large n; by this definition $-n^{1000}$ is *not* in $O(n^2)$, though it is in $O(n^{1000})$. This definition was designed for error terms in asymptotic expansions of functions, where the error term might represent a positive or negative error.

7.5.2 Abusing the equals sign

Formally, we can think of $O(g(n))$ as a predicate on functions, which is true of all functions $f(n)$ that satisfy $f(n) \leq cg(n)$ for some c and sufficiently large n. This requires writing that n^2 is $O(n^2)$ where most computer scientists or mathematicians would just write $n^2 = O(n^2)$. Making sense of the latter statement involves a standard convention that is mildly painful to define formally but that greatly simplifies asymptotic analyses.

Let's take a statement like the following:

$$O(n^2) + O(n^3) + 1 = O(n^3).$$

What we want this to mean is that the left-hand side can be replaced by the right-hand side without causing trouble. To make this work formally, we define the statement as meaning that for any f in $O(n^2)$ and any g in $O(n^3)$, there exists an h in $O(n^3)$ such that $f(n) + g(n) + 1 = h(n)$.

In general, any appearance of O, Ω, or Θ on the left-hand side gets a universal quantifier (for all) and any appearance of O, Ω, or Θ on the right-hand side gets an existential quantifier (there exists). So

$$f(n) + o(f(n)) = \Theta(f(n))$$

means that for any g in $o(f(n))$, there exists an h in $\Theta(f(n))$ such that $f(n) + g(n) = h(n)$, and

$$O(f(n)) + O(g(n)) + 1 = O(\max(f(n), g(n))) + 1$$

means that for any r in $O(f(n))$ and s in $O(g(n))$, there exists t in $O(\max(f(n), g(n))$ such that $r(n) + s(n) + 1 = t(n) + 1$.

The nice thing about this definition is that as long as you are careful about the direction the equals sign goes in, you can treat these complicated pseudo-equations like ordinary equations. For example, since $O(n^2) + O(n^3) = O(n^3)$, we can write

$$\begin{aligned}\frac{n^2}{2} + \frac{n(n+1)(n+2)}{6} &= O(n^2) + O(n^3)\\ &= O(n^3),\end{aligned}$$

which is much simpler than what it would look like if we had to talk about particular functions being elements of particular sets of functions.

This is an example of **abuse of notation**, the practice of redefining some standard bit of notation (in this case, equations) to make calculation easier. It's generally a safe practice as long as everybody understands what is happening. But beware of applying facts about unabused equations to the abused ones. Just because $O(n^2) = O(n^3)$ doesn't mean $O(n^3) = O(n^2)$—the big-O equations are not reversible the way ordinary equations are.

More discussion of this can be found in [Fer08, §10.4] and [GKP94, Chapter 9].

Chapter 8

Number theory

Number theory is the study of the natural numbers, particularly their divisibility properties. Throughout this chapter, when we say *number*, we mean an element of $\mathbb{N}$.

8.1 Divisibility and division

A number m **divides** n, written $m|n$, if there is some number k such that $km = n$. In this case m is said to be a **factor** or **divisor** of n. A number greater than 1 whose only factors are 1 and itself is called **prime**. Non-primes that are greater than 1 are called **composite**. The remaining numbers 0 and 1 are by convention neither prime nor composite; this allows us to avoid writing "except 0 or 1" in a lot of places later.

We can use the same definition of divisibility for integers, by letting m divide n if there is an integer k such that $km = n$. This gives $m|n$ if and only $|m|$ divides $|n|$. This does have some odd consequences, like -7 being prime. The integer -1 gets the same special exemption as 1—both are units, numbers that, because they divide the identity, are considered neither prime nor composite.

Some useful facts about divisibility:

- If $d|m$ and $d|n$, then $d|(m + n)$. Proof: Let $m = ad$ and $n = bd$, then $(m + n) = (a + b)d$.
- If $d|n$ and $n \neq 0$, then $d \leq n$. Proof: $n = kd \neq 0$ implies $k \geq 1$ implies $n = kd \geq d$.
- For all d, $d|0$. Proof: $0 \cdot d = 0$.

- If $d|m$ or $d|n$, then $d|mn$. Proof: Suppose $m = kd$, then $mn = (nk)d$. Alternatively, if $n = kd$, then $mn = (mk)d$.
- If p is prime, then $p|ab$ if and only if $p|a$ or $p|b$. Proof: follows from the extended Euclidean algorithm (see below).

If n is not divisible by m, then any attempt to divide n things into m piles will leave some things left over.

The **division algorithm**, due to Euclid, yields for any pair of integers (which might or might not be natural numbers) n and $m \neq 0$ unique integers q and r such that $n = qm + r$ and $0 \leq r < |m|$. The number q is called the **quotient** of n divided by m and $r = n \bmod m$ is called the **remainder** of n divided by m. The number m is called the **divisor** or (when we are mostly interested in remainders) the **modulus**.

For positive m, the quotient is often written as $\lfloor n/m \rfloor$, the **floor** of n/m, to make it clear that we want the integer version of the quotient and not some nasty fraction; for negative m, we'll get $\lceil n/m \rceil$ instead. The remainder is often written as $(n \bmod m)$, pronounced "the remainder of n modulo m" when paid by the word but usually just "n mod m." Saying that $n \bmod m = 0$ is that same as saying that m divides n.

For non-negative n and m, the "algorithm" for the division algorithm is that if n is already less than m, we can set $q = 0$ and $r = n$, while for larger n, we can compute $n - m = q'm + r$ recursively and then set $q = q + 1$. Showing that this algorithm works is an application of strong induction.

Theorem 8.1.1 (Division algorithm)**.** *Let n, m be integers with $m \neq 0$. Then there exist unique integers q and r such that $0 \leq r < |m|$ and $n = qm + r$.*

Proof. First we show that q and r exist for $n \geq 0$ and $m > 0$. This is done by induction on n. If $n < m$, then $q = 0$ and $r = n$ satisfies $n = qm + r$ and $0 \leq r < m$. If $n \geq m$, then $n - m \geq 0$ and $n - m < n$, so from the induction hypothesis there exist some q', r such that $n - m = q'm + r$ and $0 \leq r < m$. Then if $q = q + 1$, we have $n = (n - m) + m = q'm + r + m = (q' + 1)m + r = qm + r$.

Next we extend to the cases where n might be negative. If $n < 0$ and $m > 0$, then there exist q', r' with $0 \leq r < m$ such that $-n = q'm + r$. If $r' = 0$, let $q = -q'$ and $r = 0$, giving $n = -(-n) = -(q'm + r') = qm + r$. If $r' \neq 0$, let $q = -q' - 1$ and $r = m - r$; now $n = -(-n) = -(q'm + r') = -(-(q+1)m+(m-r)) = -(-qm-r) = qm+r$. So in either case appropriate q and r exist.

Finally, we consider the case where m is negative. Let $n = q'(-m) + r$, where $0 \leq r < -m$. Let $q = -q'$. Then $n = q'(-m) + r = (-q') \cdot m + r = qm + r$.

So far we have only shown that q and r exist; we haven't shown that they are unique. For uniqueness, suppose that $n = qm+r = q'm+r'$, where $0 \leq r \leq r' < |m|$. Then $(q'm + r') - (qm + r) = 0$, which we can rearrange to get $r' - r = (q - q')m$. In particular, $m|(r' - r)$, so there exists some k such that $r' - r = k \cdot |m|$. If $k = 0$, then $r' = r$, from which we can conclude $q' = q$. If $k \neq 0$, $k \geq 1$, so $r' \geq r' - r \geq |m|$, contradicting the requirement that $r' < |m|$. ☐

Note that quotients of negative numbers always round down. For example, $\lfloor(-3)/17\rfloor = -1$ even though -3 is much closer to 0 than it is to -17. This is so that the remainder is always non-negative (14 in this case). This may or may not be consistent with the behavior of the remainder operator in your favorite programming language.

8.2 Greatest common divisors

Let m and n be numbers, where at least one of m and n is nonzero, and let k be the largest number for which $k|m$ and $k|n$. Then k is called the **greatest common divisor** or **gcd** of m and n, written $\gcd(m, n)$ or sometimes just (m, n). A similar concept is the **least common multiple** (**lcm**), written $\operatorname{lcm}(m, n)$, which is the smallest k such that $m|k$ and $n|k$.

Formally, $g = \gcd(m, n)$ if $g|m$, $g|n$, and for any g' that divides both m and n, $g'|g$. Similarly, $\ell = \operatorname{lcm}(m, n)$ if $m|\ell$, $n|\ell$, and for any ℓ' with $m|\ell'$ and $n|\ell'$, $\ell|\ell'$.

Two numbers m and n whose gcd is 1 are said to be **relatively prime** or **coprime**, or simply to have no common factors.

If divisibility is considered as a partial order, the naturals form a **lattice** (see §9.5.3), which is a partial order in which every pair of elements x and y has both a unique greatest element that is less than or equal to both (the **meet** $x \wedge y$, equal to $\gcd(x, y)$ in this case) and a unique smallest element that is greater than or equal to both (the **join** $x \vee y$, equal to $\operatorname{lcm}(x, y)$ in this case).

8.2.1 The Euclidean algorithm for computing $\gcd(m, n)$

Euclid described in Book VII of his *Elements* what is now known as the **Euclidean algorithm** for computing the gcd of two numbers (his original version was for finding the largest square you could use to tile a given rectangle, but the idea is the same). Euclid's algorithm is based on the recurrence

$$\gcd(m, n) = \begin{cases} n & \text{if } m = 0, \\ \gcd(n \bmod m, m) & \text{if } m > 0. \end{cases}$$

The first case holds because $n|0$ for all n. The second holds because if k divides both n and m, then k divides $n \bmod m = n - \lfloor n/m \rfloor m$; and conversely if k divides m and $n \bmod m$, then k divides $n = (n \bmod m) + m \lfloor n/m \rfloor$. So (m, n) and $(n \bmod m, m)$ have the same set of common factors, and the greatest of these is the same.

So the algorithm simply takes the remainder of the larger number by the smaller recursively until it gets a zero, and returns whatever number is left.

8.2.2 The extended Euclidean algorithm

The **extended Euclidean algorithm** not only computes $\gcd(m, n)$, but also computes integer coefficients m' and n' such that

$$m'm + n'n = \gcd(m, n).$$

It has the same structure as the Euclidean algorithm, but keeps track of more information in the recurrence. Specifically:

- For $m = 0$, $\gcd(m, n) = n$ with $n' = 1$ and $m' = 0$.
- For $m > 0$, let $n = qm + r$ where $0 \le r < m$, and use the algorithm recursively to compute a and b such that $ar + bm = \gcd(r, m) = \gcd(m, n)$. Substituting $r = n-qm$ gives $\gcd(m, n) = a(n-qm)+bm = (b-aq)m+an$. This gives both the gcd and the coefficients $m' = b-aq$ and $n' = a$.

8.2.2.1 Example

Figure 8.1 gives a computation of the gcd of 176 and 402, together with the extra coefficients. The code used to generate this figure is given in Figure 8.2.

8.2.2.2 Applications

- If $\gcd(n, m) = 1$, then there is a number n' such that $nn' + mm' = 1$. This number n' is called the **multiplicative inverse** of n mod m and acts much like $1/n$ in modular arithmetic (see §8.4.2).

```
Finding gcd(176,402)
q = 2  r = 50
 Finding gcd(50,176)
 q = 3  r = 26
  Finding gcd(26,50)
  q = 1  r = 24
   Finding gcd(24,26)
   q = 1  r = 2
    Finding gcd(2,24)
    q = 12  r = 0
     Finding gcd(0,2)
     base case
     Returning 0*0 + 1*2 = 2
    a = b1 - a1*q = 1 - 0*12 = 1
    Returning 1*2 + 0*24 = 2
   a = b1 - a1*q = 0 - 1*1 = -1
   Returning -1*24 + 1*26 = 2
  a = b1 - a1*q = 1 - -1*1 = 2
  Returning 2*26 + -1*50 = 2
 a = b1 - a1*q = -1 - 2*3 = -7
 Returning -7*50 + 2*176 = 2
a = b1 - a1*q = 2 - -7*2 = 16
Returning 16*176 + -7*402 = 2
```

Figure 8.1: Trace of extended Euclidean algorithm

```
#!/usr/bin/python3

def euclid(m, n, trace = False, depth = 0):
    """Implementation of extended Euclidean algorithm.

    Returns triple (a, b, g) where am + bn = g and g = gcd(m, n).

    Optional argument trace, if true, shows progress."""

    def output(s):
        if trace:
            print("{}{}".format(' ' * depth, s))

    output("Finding gcd({},{})".format(m, n))

    if m == 0:
        output("base case")
        a, b, g = 0, 1, n
    else:
        q = n//m
        r = n % m
        output("q = {}  r = {}".format(q, r))
        a1, b1, g = euclid(r, m, trace, depth + 1)
        a = b1 - a1*q
        b = a1
        output("a = b1 - a1*q = {} - {}*{} = {}".format(b1, a1, q, a))

    output("Returning {}*{} + {}*{} = {}".format(a, m, b, n, g))
    return a, b, g

if __name__ == '__main__':
    import sys

    euclid(int(sys.argv[1]), int(sys.argv[2]), True)
```

Figure 8.2: Python code for extended Euclidean algorithm

- If p is prime and $p|ab$, then either $p|a$ or $p|b$. Proof: suppose $p \not| a$; since p is prime we have $\gcd(p, a) = 1$. So there exist r and s such that $rp + sa = 1$. Multiply both sides by b to get $rpb + sab = b$. Then $p|rpb$ and $p|sab$ (the latter because $p|ab$), so p divides their sum and thus $p|b$.

8.3 The Fundamental Theorem of Arithmetic

Let n be a number greater than 0. Then there is a unique sequence of primes $p_1 \le p_2 \le \ldots \le p_k$ such that $n = p_1 p_2 \ldots p_k$. This fact is known as the **Fundamental Theorem of Arithmetic**, and the sequence $p_1 \ldots p_k$ is called the **prime factorization** of n.

Showing that there is *at least* one such sequence is an easy induction argument. If $n = 1$, take the empty sequence; by convention, the product of the empty sequence is the multiplicative identity, 1. If n is prime, take $p_1 = n$; otherwise, let $n = ab$ where a and b are both greater than 1. Then $n = p_1 \ldots p_k q_1 \ldots q_m$ where the p_i are the prime factors of a and the q_i are the prime factors of b. Unfortunately, this simple argument does not guarantee *uniqueness* of the sequence: it may be that there is some n with two or more distinct prime factorizations.

We can show that the prime factorization is unique by an induction argument that uses the fact that $p|ab$ implies $p|a$ or $p|b$ (which we proved using the extended Euclidean algorithm). If $n = 1$, then any non-empty sequence of primes has a product greater than 1; it follows that the empty sequence is the unique factorization of 1. If n is prime, any factorization other than n alone would show that it isn't; this provides a base case of $n = 2$ and $n = 3$ as well as covering larger values of n that are prime. So suppose that n is composite, and that $n = p_1 \ldots p_k = q_1 \ldots q_m$, where $\{p_i\}$ and $\{q_i\}$ are nondecreasing sequences of primes. Suppose also (by the induction hypothesis) that any $n' < n$ has a unique prime factorization.

If $p_1 = q_1$, then $p_2 \ldots p_k = q_2 \ldots q_m$, and so the two sequences are identical by the induction hypothesis. Alternatively, suppose that $p_1 < q_1$; note that this also implies $p_1 < q_i$ for all i, so that p_1 doesn't appear anywhere in the second factorization of n. But then p doesn't divide $q_1 \ldots q_m = n$, a contradiction.

8.3.1 Applications

Some consequences of unique factorization:

- We can compute $\gcd(a, b)$ by factoring both a and b and retaining all the common factors, which is the algorithm favored in grade school when dealing with small numbers. Without unique factorization, this wouldn't work: we might get unlucky and factor a or b the wrong way so that the common factors didn't line up. For very large numbers, computing prime factorizations becomes impractical, so Euclid's algorithm is a better choice.

- Similarly, for every a and b, we can compute the least common multiple $\operatorname{lcm}(a, b)$ by taking the maximum of the exponents on each prime that appears in the factorization of a or b. It can also be found by computing $\operatorname{lcm}(a, b) = ab/\gcd(a, b)$, which is more efficient for large a and b because we don't have to factor.

- The natural numbers without zero, partially ordered by divisibility, form a lattice that is isomorphic to the lattice obtained by taking all infinite sequences of natural numbers whose sum is nonzero and finite, and letting $x \leq y$ if $x_i \leq y_i$ for all i. The isomorphism maps n to x where x_i is the exponent in the prime factorization of n on the i-th smallest of all primes (e.g. $24 = 2^3 \times 3^1 \mapsto 3, 1, 0, 0, 0, \dots$). If we throw in 0, it becomes a new element at the top of the lattice.

  The divisors of any particular number n also form a lattice, which is a **sublattice** of $(\mathbb{N}, |)$. See Figure 9.3 for an example.

8.4 Modular arithmetic and residue classes

From the division algorithm, we have that for each pair of integers n and $m \neq 0$, there is a unique remainder r with $0 \leq r < |m|$ and $n = qm + r$ for some q; this unique r is written as $(n \bmod m)$. Define $n \equiv_m n'$ (read "n is **congruent to** n' **mod** m") if $(n \bmod m) = (n' \bmod m)$, or equivalently if there is some $q \in \mathbb{Z}$ such that $n = n' + qm$.

The set of integers congruent to n mod m is called the **residue class** of n (*residue* is an old word for *remainder*), and is written as $[n]_m$. The sets $[0]_m, [1]_m, \dots [m-1]_m$ between them partition the integers, and the set $\{[0]_m, [1]_m, \dots [m-1]_m\}$ defines the **integers mod** m, written $\mathbb{Z}_m$. We will see that $\mathbb{Z}_m$ acts very much like $\mathbb{Z}$, with well-defined operations for addition, subtraction, and multiplication. In the case where the modulus is prime, we even get divison: $\mathbb{Z}_p$ is a **finite field** for any prime p.

The most well-known instance of $\mathbb{Z}_m$ is $\mathbb{Z}_2$, the integers mod 2. The class $[0]_2$ is the **even** numbers and the class $[1]_2$ is the **odd** numbers.

8.4.1 Arithmetic on residue classes

We can define arithmetic operations on residue classes in $\mathbb{Z}_m$ just as we defined arithmetic operations on integers (defined as equivalence classes of pairs of naturals). Given residue classes $[x]_m$ and $[y]_m$, define $[x]_m + [y]_m = [x + y]_m$, where the addition in the right-hand side is the usual integer addition in $\mathbb{Z}$. Here x and y act as **representatives** for their respective residue classes.

For example, in $\mathbb{Z}_2$ we have $0+0 = 0$, $0+1 = 1$, $1+0 = 1$, and $1+1 = 0$ (since $1+1 = 2 \in [0]_m$). Note that as with the integers, we must verify that this definition of addition is well-defined: in particular, it shouldn't matter which representatives x and y we pick.

To prove this, let's start with an alternative characterization of when $x \equiv_m y$:

Lemma 8.4.1. *Let $x, y \in \mathbb{Z}$ and let $m \in \mathbb{N}^+$. Then $x \equiv_m y$ if and only if $m|(x - y)$.*

Proof. Write $x = qm + r$, $y = sm + t$, where $0 \le r, s < m$. Then $x - y = (q - s)m + (r - t)$. If $m|(x - y)$, then $m|(r - t)$; since $-m < r - t < m$ this implies $r - t = 0$ and thus $x \bmod m = r = t = y \bmod m$. Conversely, suppose $x \bmod m = y \bmod m$. Then $r - t = 0$ giving $x - y = (b - s)m$, so $m|(x - y)$. ☐

Theorem 8.4.2. *If $x \equiv_m x'$ and $y \equiv_m y'$, then $x + y \equiv_m x' + y'$.*

Proof. From Lemma 8.4.1, $m|(x-x')$ and $m|(y-y')$. So $m|((x-x')+(y-y'))$, which we can rearrange as $m|((x + y) - (x' + y'))$. Apply Lemma 8.4.1 in the other direction to get $x + y \equiv_m x' + y'$. ☐

Similarly, we can define $-[x]_m = [-x]_m$ and $[x]_m \cdot [y]_m = [x \cdot y]_m$. A similar argument to the proof of Theorem 8.4.2 shows that these definitions also give well-defined operations on residue classes.[1]

All of the usual properties of addition, subtraction, and multiplication are inherited from $\mathbb{Z}$: addition and multiplication are commutative and associative, the distributive law applies, etc.

For example, Table 8.1 gives addition and multiplication tables for $\mathbb{Z}_3$.

[1] For $[-x]_m$: Suppose $x \equiv_m x'$; then $m|(x - x')$, which implies $m|(-(x - x'))$ or $m|((-x) - (-x'))$, giving $-x \equiv_m -x'$.

For $[x]_m \cdot [y]_m$: Suppose $x \equiv_m x'$. Then $m|(x - x')$ implies $m|((x - x')y)$ implies $m|(xy - x'x)$. So $xy \equiv_m x'y$. Applying the same argument shows that if $y \equiv_m y'$, $xy \equiv_m xy'$. Transitivity can then be used to show $xy \equiv_m x'y \equiv_m x'y'$.

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Table 8.1: Addition and multiplication tables for $\mathbb{Z}_3$

It may help to think of 2 as -1; so it's not surprising that $2 \cdot 2 = 4 \equiv_3 1 = (-1) \cdot (-1)$.

Using these tables, we can do arbitrarily horrible calculations in $\mathbb{Z}_3$ using the same rules as in $\mathbb{Z}$, e.g., $2 \cdot (1+2) - 2 = 2 \cdot (0) - 2 = -2 = 1 \pmod 3$. Note we tack on the " (mod 3)" at the end so that the reader won't think we've gone nuts.

The fact that $[x]_m + [y]_m = [x+y]_m$ and $[x]_m \times [y]_m = [xy]_m$ for all x and y means that the remainder operation $x \mapsto x \bmod m$ is a **homomorphism** from $\mathbb{Z}$ to $\mathbb{Z}_m$: it preserves the operations $+$ and $\times$ on $\mathbb{Z}$.

The formal definition of a homomorphism that preserves an operation (say $+$) is a function f such that $f(a+b) = f(a) + f(b)$. The function x mod m preserves not only the operations $+$ and $\times$ but the constants 0 and 1. This means that it doesn't matter when you perform the mod operation when converting a complicated expression in $\mathbb{Z}$ to the corresponding expression in $\mathbb{Z}_m$.

8.4.2 Division in $\mathbb{Z}_m$

One thing we don't get general in $\mathbb{Z}_m$ is the ability to divide. This is not terribly surprising, since we don't get to divide (without remainders) in $\mathbb{Z}$ either. But for some values of x and m we can in fact do division: for these x and m there exists a **multiplicative inverse** $x^{-1} \pmod m$ such that $xx^{-1} = 1 \pmod m$. We can see the winning x's for $\mathbb{Z}_9$ by looking for ones in the multiplication table for $\mathbb{Z}_9$, given in Table 8.2.

Here we see that $1^{-1} = 1$, as we'd expect, but that we also have $2^{-1} = 5$, $4^{-1} = 7$, $5^{-1} = 2$, $7^{-1} = 4$, and $8^{-1} = 8$. There are no inverses for 0, 3, or 6.

What 1, 2, 4, 5, 7, and 8 have in common is that they are all relatively prime to 9. This is not an accident: when $\gcd(x, m) = 1$, we can use the extended Euclidean algorithm (§8.2.2) to find $x^{-1} \pmod m$. Observe that what we want is some x' such that $xx' \equiv_m 1$, or equivalently such that $x'x + qm = 1$ for some q. But the extended Euclidean algorithm finds such an x' (and q) whenever $\gcd(x, m) = 1$.

$\times$	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6	7	8
2	0	2	4	6	8	**1**	3	5	7
3	0	3	6	0	3	6	0	3	6
4	0	4	8	3	7	2	6	**1**	5
5	0	5	**1**	6	2	7	3	8	4
6	0	6	3	0	6	3	0	6	3
7	0	7	5	3	**1**	8	6	4	2
8	0	8	7	6	5	4	3	2	**1**

Table 8.2: Multiplication table for $\mathbb{Z}_9$

If $\gcd(x, m) \neq 1$, then x has no multiplicative inverse in $\mathbb{Z}_m$. The reason is that if some $d > 1$ divides both x and m, it continues to divide xx' and m for any $x' \neq 0$. So in particular xx' can't be congruent to 1 mod m since $qm + 1$ and m don't share any common factors for any value of q.

The set of of residue classes $[x]_m$ where $\gcd(x, m) = 1$ is written as $\mathbb{Z}_m^*$. For a prime p, $\mathbb{Z}_p^*$ includes all non-zero elements of $\mathbb{Z}_p$, since $\gcd(x, p) = 1$ for any x that is not 0 or a multiple of p. This means that $\mathbb{Z}_p$ satisfies the same field axioms (§4.1) as $\mathbb{Q}$ or $\mathbb{R}$: in $\mathbb{Z}_p$, we can add, subtract, multiply, and divide by any number that's not 0, and all of these operations behave the way we expect. However, $\mathbb{Z}_p$ is not an ordered field, since the fact that numbers wrap around means that we can't define a $\leq$ relation that is invariant with respect to translation or scaling.

One thing that is sometimes confusing about division in $\mathbb{Z}_p$ is that it doesn't project down from a corresponding operation in $\mathbb{Z}$, the way that addition, subtraction, and multiplication do. So while we can write $\frac{3}{4} = 2 \pmod 5$, if we compute $\frac{3}{4}$ first (in $\mathbb{Q}$, say), there is no natural mapping from $\mathbb{Q}$ to $\mathbb{Z}_5$ that sends $\frac{3}{4}$ to 2.[2] Division in $\mathbb{Z}_p$ requires finding inverses, either by guessing them correctly or using the extended Euclidean algorithm.

8.4.3 The Chinese Remainder Theorem

Here is the **Chinese Remainder Theorem**, in the form typically used today:[3]

[2] While we could define a mapping $f(p/q) = (p \bmod 5)(q \bmod 5)^{-1}$ that would work for many rationals, the problem is what to do with fractions like $\frac{3}{5}$.

[3] The earliest known written version of the theorem appeared in *The Mathematical Classic of Sunzi*, a Chinese text from the Song dynasty. The first convincing proof of the

Theorem 8.4.3 (Chinese Remainder Theorem)**.** *Let m_1 and m_2 be relatively prime.*[4] *Then for each pair of equations*

$$
\begin{aligned}
n \bmod m_1 &= n_1, \\
n \bmod m_2 &= n_2,
\end{aligned}
$$

there is a unique solution n with $0 \leq n < m_1 m_2$.

Example: let $m_1 = 3$ and $m_2 = 4$. Then the integers n from 0 to 11 can be represented as pairs (n_1, n_2) with no repetitions as follows:

n	n_1	n_2
0	0	0
1	1	1
2	2	2
3	0	3
4	1	0
5	2	1
6	0	2
7	1	3
8	2	0
9	0	1
10	1	2
11	2	3

Proof. We'll show an explicit algorithm for constructing the solution. The first trick is to observe that if $a|b$, then $(x \bmod b) \bmod a = x \bmod a$. The proof is that $x \bmod b = x - qb$ for some q, so $(x \bmod b) \bmod a = (x \bmod a) - (qb \bmod a) = x \bmod a$ since any multiple of b is also a multiple of a, giving $qb \bmod a = 0$.

Since m_1 and m_2 are relatively prime, the extended Euclidean algorithm gives m_1' and m_2' such that $m_1' m_1 = 1 \pmod{m_2}$ and $m_2' m_2 = 1 \pmod{m_1}$.

result is due to the fifth-century Indian mathematician Aryabhata. The name "Chinese Remainder Theorem" appears to be much more recent. See `http://mathoverflow.net/questions/11951/what-is-the-history-of-the-name-chinese-remainder-theorem` for a discussion of the history of the name.

[4]This means that $\gcd(m_1, m_2) = 1$.

Let $n = (n_1 m'_2 m_2 + n_2 m'_1 m_1) \bmod m_1 m_2$. Then

$$\begin{aligned}
n \bmod m_1 &= ((n_1 m'_2 m_2 + n_2 m'_1 m_1) \bmod m_1 m_2) \bmod m_1 \\
&= (n_1 m'_2 m_2 + n_2 m'_1 m_1) \bmod m_1 \\
&= (n_1 \cdot 1 + n_2 m'_1 \cdot 0) \bmod m_1 \\
&= n_1 \bmod m_1 \\
&= n_1.
\end{aligned}$$

A nearly identical calculation shows $n \bmod m_2 = n_2$ as well.

The intuition is that $m'_2 m_2$ acts like 1 mod m_1 and 0 mod m_2, and vice versa for $m'_1 m_1$. Having found these basic solutions for $(1, 0)$ and $(0, 1)$, solutions for arbitrary (n_1, n_2) are just a matter of adding up enough of each.

That the solution for each pair is unique can be shown by counting: there are $m_1 m_2$ possible choices for both pairs (n_1, n_2) and solutions n, so if some pair has more than one solution, some other pair must have none. But we have just given an algorithm for generating a solution for any pair. ☐

The general version allows for any number of equations, as long as the moduli are all relatively prime.

Theorem 8.4.4 (Chinese Remainder Theorem (general version)). *Let $m_1, \dots, m_k$ satsify $\gcd(m_i, m_j) = 1$ for all pairs i, j with $i \neq j$. Then any system of equations*

$$n = n_i \pmod{m_i}$$

has a unique solution n with $0 \leq n \leq \prod_i m_i$.

Proof. The solution can be computed using the formula

$$n = \left(\sum_i n_i \prod_{j \neq i} (m_j^{-1} \pmod{m_i}) m_j \right) \bmod \prod_i m_i.$$

As in the two-modulus case, the factor $(m_j^{-1} \pmod{m_i}) m_j$, where m_j^{-1} $\pmod{m_i}$ is the multiplicative inverse of m_j mod m_i, acts like 1 mod m_i

and 0 mod m_j. So for any fixed k,

$$\begin{aligned}
n \bmod m_k &= \left(\left(\sum_i n_i \prod_{j\neq i} (m_j^{-1} \pmod{m_i}) m_j\right) \bmod \prod_i m_i\right) \bmod m_k \\
&= \left(\sum_i n_i \prod_{j\neq i} (m_j^{-1} \pmod{m_i}) m_j\right) \bmod m_k \\
&= \left(n_k \cdot 1 + \sum_{i\neq k} (n_i \cdot 0)\right) \bmod m_k \\
&= n_k.
\end{aligned}$$

∎

8.4.4 The size of $\mathbb{Z}_m^*$ and Euler's Theorem

The size of $\mathbb{Z}_m^*$, or equivalently the number of numbers less than m whose gcd with m is 1, is written $\phi(m)$ and is called **Euler's totient function** or just the **totient** of m. When p is prime, $\gcd(n, p) = 1$ for all n with $0 < n < p$, so $\phi(p) = \left|\mathbb{Z}_p^*\right| = p - 1$. For a prime power p^k, we similarly have $\gcd(n, p^k) = 1$ unless $p|n$. There are exactly p^{k-1} numbers less than p^k that are divisible by p (they are $0, p, 2p, \dots (p^k - 1)p$), so $\phi(p^k) = p^k - p^{k-1} = p^{k-1}(p - 1)$.[5] For composite numbers m that are not prime powers, finding the value of $\phi(m)$ is more complicated; but we can show using the Chinese Remainder Theorem (Theorem 8.4.3) that in general

$$\phi\left(\prod_{i=1}^k p_i^{e_i}\right) = \prod_{i=1}^k p_i^{e_i-1}(p_i - 1).$$

One reason $\phi(m)$ is important is that it plays a central role in **Euler's Theorem**:

Theorem 8.4.5. *Let* $\gcd(a, m) = 1$. *Then*

$$a^{\phi(m)} = 1 \pmod{m}.$$

Proof. We will prove this using an argument adapted from the proof of [Big02, Theorem 13.3.2]. Let $z_1, z_2, \dots, z_{\phi(m)}$ be the elements of $\mathbb{Z}_m^*$. For any $y \in \mathbb{Z}_m^*$, define $y\mathbb{Z}_m^* = \left\{yz_1, yz_2, \dots, yz_{\phi(m)}\right\}$. Since y has a multiplicative

[5]Note that $\phi(p) = \phi(p^1) = p^{1-1}(p - 1) = p - 1$ is actually a special case of this.

inverse mod m, the mapping $z \mapsto yz \pmod{m}$ is a bijection, and so $y\mathbb{Z}_m^* = \mathbb{Z}_m^* \pmod{m}$. It follows that $\prod_i z_i = \prod_i yz_i = y^{\phi(m)} \prod_i z_i \pmod{m}$. But now multiply both sides by $\left(\prod_i z_i\right)^{-1} = \prod_i z_i^{-1}$ to get $1 = y^{\phi(m)} \pmod{m}$ as claimed. □

For the special case that m is a prime, Euler's Theorem is known as **Fermat's Little Theorem**, and says that $a^{p-1} = 1 \pmod{p}$ for all primes p and all a such that $p \not| a$. Fermat proved this result before Euler generalized it to composite m, which is why we have two names.

8.5 RSA encryption

Euler's Theorem is useful in cryptography; for example, the **RSA encryption** system is based on the fact that $(x^e)^d = x \pmod{m}$ when $de = 1 \pmod{\phi(m)}$. So x can be encrypted by raising it to the e-th power mod m, and decrypted by raising the result to the d-th power. It is widely believed that publishing e and m reveals no useful information about d provided e and m are chosen carefully.

Specifically, the person who wants to receive secret messages chooses large primes p and q, and finds d and e such that $de = 1 \pmod{(p-1)(q-1)}$. They then publish $m = pq$ (the product, not the individual factors and e.

Encrypting a message x involves computing $x^e \bmod m$. If x and e are both large, computing x^e and then taking the remainder is an expensive operation; but it is possible to get the same value by computing x^e in stages by repeatedly squaring x and taking the product of the appropriate powers. To decrypt x^e, compute $(x^e)^d \bmod m$.

For example, let $p = 7$, $q = 13$, so $m = 91$. The totient $\phi(m)$ of m is $(p-1)(q-1) = 6 \cdot 12 = 72$. Next pick some e relatively prime to $\phi(m)$: $e = 5$. Since $5 \cdot 29 = 72 \cdot 2 + 1$ we can make $d = 29$. Note that to compute d in this way, we needed to know how to factor m so that we could compute $(p-1)(q-1)$; it's not known how to find d otherwise.

Now let's encrypt a message. Say we want to encrypt 11. Using $e = 5$ and $m = 91$, we can compute:

$$\begin{aligned}
11^1 &= 11 \\
11^2 &= 121 = 30 \\
11^4 &= 30^2 = 900 = 81 \\
11^5 &= 11^4 \cdot 11^1 = 81 \cdot 11 = 891 = 72.
\end{aligned}$$

When the recipient (who knows d) receives the encrypted message 72, they can recover the original by computing $72^{29} \bmod 91$:

$$\begin{aligned}
72^1 &= 72 \\
72^2 &= 5184 = 88 \\
72^4 &= 88^2 = (-3)^2 = 9 \\
72^8 &= 9^2 = 81 \\
72^{16} &= 81^2 = (-10)^2 = 100 = 9 \\
72^{29} &= 72^{16} \cdot 72^8 \cdot 72^4 \cdot 72^1 = 9 \cdot 81 \cdot 9 \cdot 72 = 81^2 \cdot 72 = 9 \cdot 72 = 648 = 11.
\end{aligned}$$

Note that we are working in $\mathbb{Z}_{91}$ throughout. This is what saves us from computing the actual value of 72^{29} in $\mathbb{Z}$,[6] and only at the end taking the remainder.

For actual security, we need m to be large enough that it's hard to recover p and q using presently conceivable factoring algorithms. Typical applications choose m in the range of 2048 to 4096 bits (so each of p and q will be a random prime somewhere between 10^{308} and 10^{617}. This is too big to show a hand-worked example, or even to fit into the much smaller integer data types shipped by default in many programming languages, but it's not too large to be able to do the computations efficiently with good large integer arithmetic library.

[6]If you're curious, it's 728857113063526668247098229876984590549890725463457792.

Chapter 9

Relations

A **binary relation** from a set A to a set B is a subset of $A \times B$. In general, an n**-ary relation** on sets $A_1, A_2, \ldots, A_n$ is a subset of $A_1 \times A_2 \times \ldots \times A_n$. We will mostly be interested in binary relations, although n-ary relations are important in databases. Unless otherwise specified, a **relation** will be a binary relation. A relation from A to A is called a relation **on** A; many of the interesting classes of relations we will consider are of this form. Some simple examples are the relations $=$, $<$, $\leq$, and $\mid$ (divides) on the integers.

You may recall that functions are a special case of relations, but most of the relations we will consider now will not be functions.

Binary relations are often written in **infix notation**: instead of writing $(x, y) \in R$, we write xRy. This should be pretty familiar for standard relations like $<$ but might look a little odd at first for relations named with capital letters.

9.1 Representing relations

In addition to representing a relation by giving an explicit table ($\{(0,1),(0,2),(1,2)\}$) or rule (xRy if $x < y$, where $x, y \in \{0,1,2\}$), we can also visualize relations in terms of other structures built on pairs of objects.

9.1.1 Directed graphs

A **directed graph** consists of a set of **vertices** V and a set of **edges** E, where each edge E has an **initial vertex** or **source** $\text{init}(e)$ and a **terminal vertex** or **sink** $\text{term}(E)$. A **simple** directed graph has no **parallel edges**: there are no edges e_1 and e_2 with $\text{init}(e_1) = \text{init}(e_2)$ and

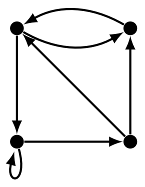


Figure 9.1: A directed graph

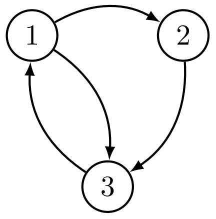


Figure 9.2: Relation $\{(1,2),(1,3),(2,3),(3,1)\}$ represented as a directed graph

$\text{term}(e_1) = \text{term}(e_2)$.

If we don't care about the labels of the edges, a simple directed graph can be described by giving E as a subset of $V \times V$; this gives a one-to-one correspondence between relations on a set V and (simple) directed graphs. For relations from A to B, we get a bipartite directed graph, where all edges go from vertices in A to vertices in B.

Directed graphs are drawn using a dot or circle for each vertex and an arrow for each edge, as in Figure 9.1.

This also gives a way to draw relations. For example, the relation on $\{1,2,3\}$ given by $\{(1,2),(1,3),(2,3),(3,1)\}$ can be depicted as show in Figure 9.2.

A directed graph that contains no sequence of edges leading back to their starting point is called a **directed acyclic graph** or **DAG**. DAGs are important for representing partially-ordered sets (see §9.5).

9.1.2 Matrices

A **matrix** is a two-dimensional analog of a sequence: in full generality, it is a function $A : S \times T \to U$, where S and T are the index sets of the matrix (typically $\{1 \ldots n\}$ and $\{1 \ldots m\}$ for some n and m). As with sequences, we write A_{ij} for $A(i,j)$. Matrices are typically drawn inside square brackets like this:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix}$$

The first index of an entry gives the row it appears in and the second one the column, so in this example $A_{1,2} = 2$ and $A_{3,4} = -1$. The **dimensions** of a matrix are the numbers of rows and columns; in the example, A is a 3×4 (pronounced “3 by 4”) matrix.

Note that rows come before columns in both indexing (A_{ij}: i is row, j is column) and giving dimensions ($n \times m$: n is rows, m is columns). Like the convention of driving on the right (in many countries), this choice is arbitrary, but failing to observe it may cause trouble.

Matrices are used heavily in linear algebra (Chapter 13), but for the moment we will use them to represent relations from $\{1 \dots n\}$ to $\{1 \dots m\}$, by setting $A_{ij} = 0$ if (i, j) is not in the relation and $A_{ij} = 1$ if (i, j) is. So for example, the relation on $\{1 \dots 3\}$ given by $\{(i, j) \mid i < j\}$ would appear in matrix form as

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

When used to represent the edges in a directed graph, a matrix of this form is called an **adjacency matrix**.

9.2 Operations on relations

9.2.1 Composition

Just like functions, relations can be composed: given relations $R \subseteq A \times B$ and $S \subseteq B \times C$ we define $(S \circ R) \subseteq A \times C$ by the rule $(x, z) \in (S \circ R)$ if and only if there exists some $y \in B$ such that $(x, y) \in R$ and $(y, z) \in S$. (In infix notation: $x(S \circ R)z \leftrightarrow \exists y : xRy \wedge ySz$.) It’s not hard to see that ordinary function composition ($(f \circ g)(x) = f(g(x))$) is just a special case of relation composition.

In matrix terms, composition acts like matrix multiplication, where we replace scalar multiplication with AND and scalar addition with OR: $(S \circ R)_{ij} = \bigvee_k (R_{ik} \wedge S_{kj})$. Note that if we use the convention that $R_{ij} = 1$ if iRj the order of the product is reversed from the order of composition.

For relations on a single set, we can iterate composition: R^n is defined by $R^0 = (=)$ and $R^{n+1} = R \circ R^n$. (This also works for functions, bearing in mind that the equality relation is also the constant function.) In directed graph terms, xR^ny if and only if there is a path of exactly n edges from x to y (possibly using the same edge more than once).

9.2.2 Inverses

Relations also have **inverses**: $xR^{-1}y \leftrightarrow yRx$. Unlike functions, every relation has an inverse.

9.3 Classifying relations

Certain properties of relations on a set are important enough to be given names that you should remember.

Reflexive A relation R on a set A is **reflexive** if (a, a) is in R for all a in A. The relations $=$ and $\leq$ are both reflexive; $<$ is not. The equality relation is in a sense particularly reflexive: a relation R is reflexive if and only if it is a superset of $=$.

Symmetric A relation R is **symmetric** if (a, b) is in R whenever (b, a) is. Equality is symmetric, but $\leq$ is not. Another way to state symmetry is that $R = R^{-1}$.

Antisymmetric A relation R is **antisymmetric** if the only way that both (a, b) and (b, a) can be in R is if $a = b$. (More formally: $aRb \wedge bRa \rightarrow a = b$.) The "less than" relation $<$ is antisymmetric: if a is less than b, b is *not* less than a, so the premise of the definition is never satisfied. The "less than or equal to" relation $\leq$ is also antisymmetric; here it is possible for $a \leq b$ and $b \leq a$ to both hold, but only if $a = b$. The set-theoretic statement is R is symmetric if and only if $R \cap R^{-1} \subseteq (=)$. This is probably not as useful as the simple definition.

Transitive A relation R is **transitive** if (a, b) in R and (b, c) in R implies (a, c) in R. The relations $=$, $<$, and $\leq$ are all transitive. The relation $\{(x, x + 1) \mid x \in \mathbb{N}\}$ is not. The set-theoretic form is that R is transitive if $R^2 \subseteq R$, or in general if $R^n \subseteq R$ for all $n > 0$.

9.4 Equivalence relations

An **equivalence relation** is a relation that is reflexive, symmetric, and transitive. Equality is the model of equivalence relations, but some other examples are:

- Equality mod m: The relation $x = y \pmod{m}$ that holds when x and y have the same remainder when divided by m is an equivalence relation. This is often written as $x \equiv_m y$.

- Equality after applying a function: Let $f : A \to B$ be any function, and define $x \sim_f y$ if $f(x) = f(y)$. Then $\sim_f$ is an equivalence relation. Note that $\equiv_m$ is a special case of this.

- Membership in the same block of a partition: Let A be the union of a collection of sets A_i where the A_i are all disjoint. The set $\{A_i\}$ is called a **partition** of A, and each individual set A_i is called a **block** of the partition. Let $x \sim y$ if x and y appear in the same block A_i for some i. Then $\sim$ is an equivalence relation.

- Directed graph isomorphism: Suppose that $G = (V, E)$ and $G' = (V', E')$ are directed graphs, and there exists a bijection $f : V \to V'$ such that (u, v) is in E if and only if $(f(u), f(v))$ is in E'. Then G and G' are said to be **isomorphic** (from Greek "same shape"). The relation $G \cong G'$ that holds when G and G' are isomorphic is easily seen to be reflexive (let f be the identity function), symmetric (replace f by f^{-1}), transitive (compose $f : G \to G'$ and $g : G' \to G''$); thus it is an equivalence relation.

- Partitioning a plane: draw a curve in a plane (i.e., pick a continuous function $f : [0, 1] \to R^2$). Let $x \sim y$ if there is a curve from x to y (i.e., a curve g with $g(0) = x$ and $g(1) = y$) that doesn't intersect the first curve. Then $x \sim y$ is an equivalence relation on points in the plane excluding the curve itself. Proof: To show $x \sim x$, let g be the constant function $g(t) = x$. To show $x \sim y \leftrightarrow y \sim x$, consider some function g demonstrating $x \sim y$ with $g(0) = x$ and $g(1) = y$ and let $g'(t) = g(1 - t)$. To show $x \sim y$ and $y \sim z$ implies $x \sim z$, let g be a curve from x to y and g' a curve from y to z, and define a new curve $(g + g')$ by $(g + g')(t) = g(2t)$ when $t \leq 1/2$ and $(g + g')(t) = g'(2t - 1)$ when $t \geq 1/2$.

Any equivalence relation $\sim$ on a set A gives rise to a set of **equivalence classes**, where the equivalence class of an element a is the set of all b such that $a \sim b$. Because of transitivity, the equivalence classes form a partition of the set A, usually written $A/\sim$ (pronounced "the **quotient set** of A by $\sim$,

"A slash $\sim$," or sometimes "A modulo $\sim$"). A member of a particular equivalence class is said to be a **representative** of that class. For example, the equivalence classes of equality mod m are the sets $[i]_m = \{i + km \mid k \in \mathbb{N}\}$, with one collection of representatives being $\{0, 1, 2, 3, \dots, m-1\}$. A more complicated case are the equivalence classes of the plane partitioning example; here the equivalence classes are essentially the pieces we get after cutting out the curve f, and any point on a piece can act as a representative for that piece.

This gives us several equally good ways of showing that a particular relation $\sim$ is an equivalence relation:

Theorem 9.4.1. *Let $\sim$ be a relation on A. Then each of the following conditions implies the others:*

1. *$\sim$ is reflexive, symmetric, and transitive.*
2. *There is a partition of A into disjoint **equivalence classes** A_i such that $x \sim y$ if and only if $x \in A_i$ and $y \in A_i$ for some i.*
3. *There is a set B and a function $f : A \to B$ such that $x \sim y$ if and only if $f(x) = f(y)$.*

Proof. We do this in three steps:

- $(1 \to 2)$. For each $x \in A$, let $A_x = [x]_\sim = \{y \in A \mid y \sim x\}$, and let the partition be $\{A_x \mid x \in A\}$. (Note that this may produce duplicate indexes for some sets.) By reflexivity, $x \in A_x$ for each x, so $A = \bigcup_x A_x$.

  To show that distinct equivalence classes are disjoint, suppose that $A_x \cap A_y \neq \emptyset$. Then there is some z that is in both A_x and A_y, which means that $z \sim x$ and $z \sim y$; symmetry reverses these to get $x \sim z$ and $y \sim z$. If $q \in A_x$, then $q \sim x \sim z \sim y$, giving $q \in A_y$; conversely, if $q \in A_y$, then $q \sim y \sim z \sum x$, giving $q \in A_x$. It follows that $A_x = A_y$.

- $(2 \to 3)$. Let $B = A/\sim = \{A_x\}$, where each A_x is defined as above. Let $f(x) = A_x$. Then $x \sim y$ implies $x \in A_y$ implies $A_x \cap A_y \neq \emptyset$. We've shown above that if this is the case, $A_x = A_y$, giving $f(x) = f(y)$. Conversely, if $f(x) \neq f(y)$, then $A_x \neq A_y$, giving $A_x \cap A_y = \emptyset$. In particular, $x \in A_x$ means $x \notin A_y$, so $x \not\sim y$.

- $(3 \to 1)$. Suppose $x \sim y$ if and only if $f(x) = f(y)$ for some f. Then $f(x) = f(x)$, so $x \sim x$: $(\sim)$ is reflexive. If $x \sim y$, then $f(x) = f(y)$, giving $f(y) = f(x)$ and thus $y \sim x$: $(\sim)$ is symmetric. If $x \sim y \sim z$, then $f(x) = f(y) = f(z)$, and $f(x) = f(z)$, giving $x \sim z$: $(\sim)$ is transitive.

□

9.4.1 Why we like equivalence relations

Equivalence relations are the way that mathematicians say "I don't care." If you don't care about which integer you've got except for its remainder when divided by m, then you define two integers that don't differ in any way that you care about to be equivalent and work in $\mathbb{Z}/\equiv_m$. This turns out to be incredibly useful for defining new kinds of things: for example, we can define **multisets** (sets where elements can appear more than once) by starting with sequences, declaring $x \sim y$ if there is a permutation of x that reorders it into y, and then defining a multiset as an equivalence class with respect to this relation.

This can also be used informally: "I've always thought that brocolli, spinach, and kale are in the same equivalence class."[1]

9.5 Partial orders

A **partial order** is a relation $\leq$ that is reflexive, transitive, and antisymmetric. The last means that if $x \leq y$ and $y \leq x$, then $x = y$. A set S together with a partial order $\leq$ is called a **partially ordered set** or **poset**. A **strict partial order** is a relation $<$ that is irreflexive and transitive (which implies antisymmetry as well). Any partial order $\leq$ can be converted into a strict partial order and vice versa by deleting/including the pairs (x, x) for all x.

A **total order** is a partial order $\leq$ in which any two elements are **comparable**. This means that, given x and y, either $x \leq y$ or $y \leq x$. A poset $(S, \leq)$ where $\leq$ is a total order is called **totally ordered**. Not all partial orders are total orders; for an extreme example, the poset $(S, =)$ for any set S with two or more elements is partially ordered by not totally ordered.

Examples:

- $(\mathbb{N}, \leq)$ is a poset. It is also totally ordered.
- $(\mathbb{N}, \geq)$ is also a both partially ordered and totally ordered. In general, if R is a partial order, then R^{-1} is also a partial order; similarly for total orders.
- The **divisibility relation** $a|b$ on natural numbers, where $a|b$ if and only if there is some k in $\mathbb{N}$ such that $b = ak$, is reflexive (let $k = 1$),

[1]Curious fact: two of these unpopular vegetables are in fact cultivars of the same species *Brassica oleracea* of cabbage.

antisymmetric (if $a|b$, then $a \leq b$, so if $a|b$ and $b|a$ then $a \leq b$ and $b \leq a$ implying $a = b$) and transitive (if $b = ak$ and $c = bk'$ then $c = akk'$). Thus it is a partial order.

- Let $(A, \leq_A)$ and $(B, \leq_B)$ be posets. Then the relation $\leq$ on $A \times B$ defined by $(a, b) \leq (a', b')$ ff and only if $a \leq a'$ and $b \leq b'$ is a partial order. The poset $(A \times B, \leq)$ defined in this way is called the **product poset** of A and B.

- Again let $(A, \leq_A)$ and $(B, \leq_B)$ be posets. The relation $\leq$ on $A \times B$ defined by $(a, b) \leq (a', b')$ if either (1) $a < a'$ or (2) $a = a'$ and $b \leq b'$ is called **lexicographic order** on $A \times B$ and is a partial order. The useful property of lexicographic order (**lex order** for short) is that if the original partial orders are total, so is the lex order: this is why dictionary-makers use it. This also gives a source of very difficult-to-visualize total orders, like lex order on $\mathbb{R} \times \mathbb{R}$, which looks like the classic real number line where every point is replaced by an entire copy of the reals.

- Let Σ be some alphabet and consider the set $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots$ of all finite words drawn from Σ. Given two words x and y, let $x \leq y$ if x is a **prefix** of y, i.e. if there is some word z such that $xz = y$. Then $(\Sigma^*, \leq)$ is a poset.

- Using the same set Σ^*, let $x \sqsubseteq y$ if x is a subsequence of y, i.e., if there is a sequence of increasing positions $i_1 < i_2 < \dots < i_k$ such that $x_j = y_{i_j}$. (For example, $bd \sqsubseteq a\mathbf{b}c\mathbf{d}e$.) Then $(\Sigma^*, \sqsubseteq)$ is a poset.

There are also some common relations that are not partial orders or strict partial orders but come close. For example, the element-of relation ($\in$) is irreflexive and antisymmetric (this ultimately follows from the Axiom of Foundation) but not transitive; if $x \in y$ and $y \in z$ we do not generally expect $x \in z$. The "is at least as rich as" relation is reflexive and transitive but not antisymmetric: if you and I have a net worth of 0, we are each as rich as the other, and yet we are not the same person. Relations that are reflexive and transitive (but not necessarily antisymmetric) are called **quasiorders** or **preorders** and can be turned into partial orders by defining an equivalence relation $x \sim y$ if $x \leq y$ and $y \leq x$ and replacing each equivalence class with respect to $\sim$ by a single element.

As far as I know, there is no standard term for relations that are irreflexive and antisymmetric but not necessarily transitive.

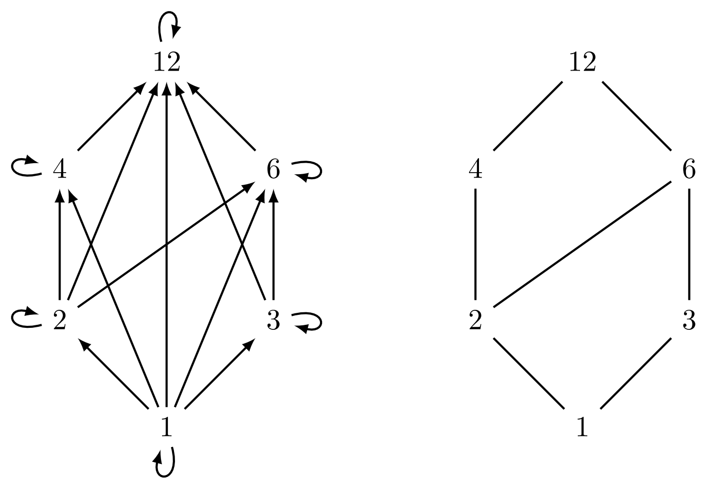


Figure 9.3: Factors of 12 partially ordered by divisibility. On the left is the full directed graph. On the right is a Hasse diagram, which uses relative height instead of arrowheads to indicate direction and omits edges implied by reflexivity and transitivity.

9.5.1 Drawing partial orders

Since partial orders are relations, we can draw them as directed graphs. But for many partial orders, this produces a graph with a lot of edges whose existence is implied by transitivity, and it can be easier to see what is going on if we leave the extra edges out. If we go further and line the elements up so that x is lower than y when $x < y$, we get a **Hasse diagram**: a representation of a partially ordered set as a graph where there is an edge from x to y if $x < y$ and there is no z such that $x < z < y$.[2]

Figure 9.3 gives an example of the divisors of 12 partially ordered by divisibility, represented both as a digraph and as a Hasse diagram. Even in this small example, the Hasse diagram is much easier to read.

9.5.2 Comparability

In a partial order, two elements x and y are **comparable** if $x \leq y$ or $y \leq x$. Elements that are not comparable are called **incomparable**. In a Hasse

[2]There is special terminology for this situation: such an x is called a **predecessor** or sometimes **immediate predecessor** of y; y in turn is a **successor** or sometimes **immediate successor** of x.

diagram, comparable elements are connected by a path that only goes up. For example, in the partial order whose Hasse diagram was given earlier, all elements are comparable to each other except the two elements at the top.

9.5.3 Lattices

A **lattice** is a partial order in which (a) each pair of elements x and y has a unique **greatest lower bound** or **meet**, written $x \wedge y$, with the property that $(x \wedge y) \leq x$, $(x \wedge y) \leq y$, and $z \leq (x \wedge y)$ for any z with $z \leq x$ and $z \leq y$; and (b) each pair of elements x and y has a unique **least upper bound** or **join**, written $x \vee y$, with the property that $(x \vee y) \geq x$, $(x \vee y) \geq y$, and $z \geq (x \vee y)$ for any z with $z \geq x$ and $z \geq y$.

Examples of lattices are any total order ($x \wedge y$ is $\min(x, y)$, $x \vee y$ is $\max(x, y)$), the subsets of a fixed set ordered by inclusion ($x \wedge y$ is $x \cap y$, $x \vee y$ is $x \cup y$), and the divisibility relation on the positive integers ($x \wedge y$ is the greatest common divisor, $x \vee y$ is the least common multiple—see Chapter 8). Products of lattices with the product order are also lattices: $(x_1, x_2) \wedge (y_1, y_2) = (x_1 \wedge_1 y_1, x_2 \wedge y_2)$ and $(x_1, x_2) \vee (y_1, y_2) = (x_1 \vee_1 y_1, x_2 \vee y_2)$.
[3]

9.5.4 Minimal and maximal elements

If for some x, $y \leq x$ only if $y = x$, then x is **minimal**. A partial order may have any number of minimal elements: the integers have no minimal element, the naturals have one minimal element, and a set with k elements none of which are comparable to each other has k minimal elements. If an element x satisfies $x \leq y$ for all y, then x is a **minimum**. A partial order may have at most one minimum (for example, 0 in the naturals), but may have no minimum, either because it contains an infinite descending chain (like the negative integers) or because it has more than one minimal element. Any minimum is also minimal.

The corresponding terms for elements that are not less than any other element or that are greater than all other elements are **maximal** and **maximum**, respectively.

Here is an example of the difference between a maximal and a maximum element: consider the family of all subsets of $\mathbb{N}$ with at most three elements,

[3]The product of two lattices with *lexicographic* order is not always a lattice. For example, consider the lex-ordered product of $(\{0, 1\}, \subseteq)$ with $(\mathbb{N}, \leq)$. For the elements $x = (\{0\}, 0)$ and $y = (\{1\}, 0)$, we have that z is a lower bound on x and y if and only if z is of the form $(\emptyset, k)$ for some $k \in \mathbb{N}$. But there is no greatest lower bound for x and y because given any particular k we can always choose a bigger k'.

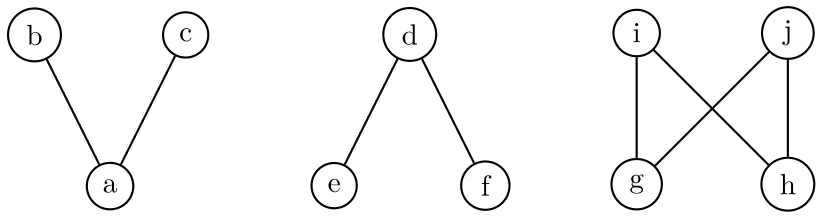


Figure 9.4: Maximal and minimal elements. In the first poset, a is minimal and a minimum, while b and c are both maximal but not maximums. In the second poset, d is maximal and a maximum, while e and f are both minimal but not minimums. In the third poset, g and h are both minimal, i and j are both maximal, but there are not minimums or maximums.

ordered by $\subseteq$. Then $\{0, 1, 2\}$ is a maximal element of this family (it's not a subset of any larger set), but it's not a maximum because it's not a superset of $\{3\}$. (The same thing works for any other three-element set.)

See Figure 9.4 for some more examples.

9.5.5 Total orders

If any two elements of a partial order are comparable (that is, if at least one of $x \leq y$ or $y \leq x$ holds for all x and y), then the partial order is a **total order**. Total orders include many of the familiar orders on the naturals, the reals, etc.

Any partial order $(S, \leq)$ can be **t**o a total order (generally more than one, if the partial order is not total itself). This means that we construct a new relation $\leq'$ on S that is a superset of $\leq$ and also totally ordered. There is a straightforward way to do this when S is finite, called a **topological sort**, and a less straightforward way to do this when S is infinite.

9.5.5.1 Topological sort

A **topological sort** is an algorithm for sorting objects that are partially ordered, in a way that preserves the partial order. (An example is given in Figure 9.5.) It can be used to construct a schedule for executing a sequence of operations that depend on each other, and efficient algorithms for topological sort exist. We won't bother with efficiency, and will just use the basic idea to show that a total extension of any finite partial order exists.

The simplest version of this algorithm is to find a minimal element, put it first, and then sort the rest of the elements; this is similar to **selection sort**,

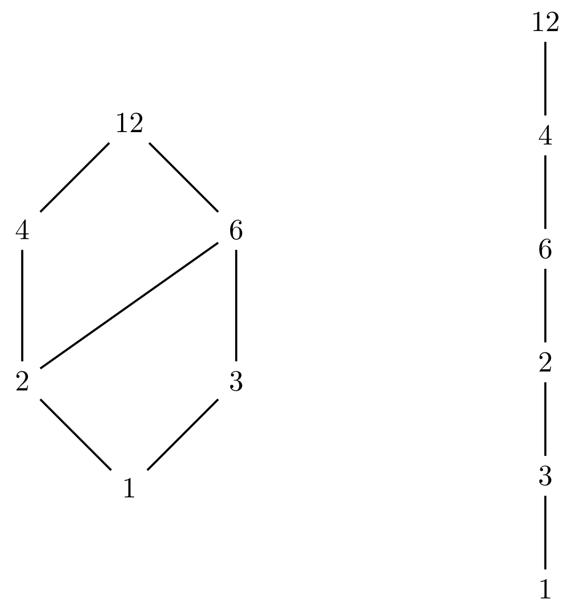


Figure 9.5: Topological sort. On the right is a total order extending the partial order on the left.

an algorithm for doing ordinary sorting, where we find the smallest element of a set, put it first, and then find the rest. In order for the selection-based version of topological sort to work, we have to know that there is, in fact, a minimal element.[4]

Lemma 9.5.1. *Every nonempty finite partially-ordered set has a minimal element.*

Proof. Let $(S, \leq)$ be a nonempty finite partially-ordered set. We will prove that S contains a minimal element by induction on $|S|$.

If $|S| = 1$, then $S = \{x\}$ for some x; x is the minimal element.

Now consider some S with $|S| > 2$. Pick some element $x \in S$, and let $T = S \setminus \{x\}$. Then by the induction hypothesis, T has a minimal element y, and since $|T| \geq 2$, T has at least one other element $z \not\leq y$.

If y is also a minimal element of S, we are done.

Otherwise, there is an element of $S \setminus T$ that is less than y. Since x is the only element of $S \setminus T$, $x < y$. If x is not a minimal element of S, there is some $z \in S$ such that $z < x$. But then $z \in T$ and $z < x < y$. If $z = y$, then $x \leq y$ and $y \leq x$, violating anti-symmetry. If $z \neq y$, then $z < y$ and $z \in T$, violating the assumption that y is minimal. Since we get a contradiction in either case, x is minimal. □

Now we can apply the selection-sort strategy previously described.

Theorem 9.5.2. *Every partial order on finite set has a total extension.*

Proof. Let $(S, \leq_S)$ be a finite partially-ordered set.

If S is empty, it contains no pair of incomparable elements, so it is already totally ordered.

For nonempty sets S, we will prove the result by induction on $|S|$.

If S is nonempty, then by Lemma 9.5.1, it has a minimal element x. Let $T = S \setminus \{x\}$ and let $\leq_T$ be the restriction of $\leq_S$ to T. Then $(T, \leq_T)$ has a total extension $\leq'_T$. Define $\leq'_S$ by $y \leq'_S z$ if $y = x$ or $y \leq'_T z$.

First, let us show that $\leq'_S$ extends $\leq_S$. Suppose $a \leq_S b$. There are three cases:

1. $a, b \in T$. Then $a \leq_S b \to a \leq_T b \to a \leq'_T b \to a \leq'_S b$.

2. $a = x$. Then $x \leq'_S b$ always.

3. $b = x$. Then $a \leq_S x \to a = x \to a \leq'_S x$.

[4]There may be more than one, but one is enough.

Next, let us show that $\le'_S$ is a partial order. This requires verifying that adding x to T doesn't break reflexivity, antisymmetry, or transitivity. For reflexivity, $x \le x$ from the first case of the definition. For antisymmetry, if $y \le'_S x$ then $y = x$, since $y \not\le'_T x$ for any y. For transitivity, if $x \le'_S y \le'_S z$ then $x \le'_S z$ (since $x \le'_S z$ for all z in S), and if $y \le'_S x \le'_S z$ then $y = x \le'_S z$ and if $y \le'_S z \le'_S x$ then $y = z = x$.

Finally, let's make sure that we actually get a total order. This means showing that any y and z in S are comparable. If $y \not\le'_S z$, then $y \ne x$, and either $z = x$ or $z \in T$ and $y \not\le'_T z$ implies $z \le'_T y$. In either case $z \le'_S y$. The case $y \not\le'_S z$ is symmetric. $\square$

For infinite partial orders the situation is more complicated, because an infinite partial order might not have any minimal elements (consider $(\mathbb{Z}, \le)$). The intuition is that we can always pick some pair of incomparable elements and declare one less than the other, fill in any other relations implied by transitivity, and repeat. Unfortunately this process may take infinitely long, so we have to argue that it converges in the limit to a genuine total order using a tool called **Zorn's lemma**, which itself is a theorem about partial orders.[5]

9.5.6 Well orders

A **well order** is a particularly restricted kind of total order. A partial order is a well order if it is a total order and every nonempty subset S has a

[5] You don't really need to know about Zorn's Lemma, but if you are curious, Zorn's Lemma says that if $(S, \le)$ is any poset, and every totally-ordered subset S' of S has an upper bound x, which is an element of S that is greater than or equal to any y in S', then S has a maximal element. Applying this to partial orders, let R be some partial order on a set A, and let S be the set of all partial orders R' on A that are supersets of R, ordered by the subset relation. Now given any chain of partial orders $R_1 \subseteq R_2, \dots$ in S, their union is also a partial order (this requires a proof) and any R_i is a subset of the union. So S has a maximal partial order $\hat{R}$.

If $\hat{R}$ is not a total order, then there is some pair of elements x and y that are incomparable. Let

$$T = \hat{R} \cup \{(x,y)\} \cup \left\{(x,z) \;\middle|\; (y,z) \in \hat{R}\right\} \cup \left\{(w,y) \;\middle|\; (w,x) \in \hat{R}\right\}$$

Then T is reflexive (because it contains all the pairs (x,x) that are in R) and transitive (by a tedious case analysis), and antisymmetric (by another tedious case analysis), meaning that it is a partial order that extends R—and thus an element of S—while also being a proper superset of $\hat{R}$. But this contradicts the assumption that $\hat{R}$ is maximal. So $\hat{R}$ is in fact the total order we are looking for.

minimum element x. An example of a well order is the usual order on $\mathbb{N}$.[6]

An equivalent definition is that a total order is a well order if it contains no **infinite descending chain**, which is an infinite sequence $x_1 > x_2 > x_3 > \dots$ To show that this is implied by every set having a least element, suppose that a given total order has the least-element property. Then given a would-be infinite descending chain $x_1, x_2, \dots$, let x_i be its least element. But then x_i is not greater than x_{i+1}. For the converse, suppose that some set S does not have a least element. Then we can construct an infinite descending chain by choosing any $x_1 in S$, then for each x_{i+1} choose some element less than the smallest of $x_1 \dots x_i$. Zorn's Lemma can be used to show that this process converges to an infinite descending chain in the limit.

The useful property of well-orders is that we can do induction on them. If it is the case that (a) $P(m)$ holds, where m is the smallest element in some set S, and (b) $P(x')$ for all $x < x'$ implies$P(x)$, then $P(x)$ holds for all x in S. The proof is that if $P(x)$ doesn't hold, there is a least element y in S for which it doesn't hold (this is the least element in the set $\{y \in S \mid \neg P(y)\}$, which exists because S is well ordered). But this contradicts (a) if $y = m$ and (b) otherwise.

For sets that aren't well-ordered, this argument generally doesn't work. For example, we can't do induction on the integers because there is no number negative enough to be the base case, and even if we add a new minimum element $-\infty$, when we do the induction step we can't find the minimum y in the set of integers excluding $-\infty$.

It is possible in an infinite set to have a well-ordering in which some elements do not have predecessors. For example, consider the order on $S = \mathbb{N} \cup \{\omega\}$ defined by $x \leq y$ if either (a) x and y are both in $\mathbb{N}$ and $x \leq y$ by the usual ordering on $\mathbb{N}$ or (b) $y = \omega$. This is a total order that is also a well order, but ω has no immediate predecessor. In this case we can still do induction proofs, but since ω is not $n + 1$ for any n, we need a special case in the proof to handle it. For a more complicated example, the set $\omega + \omega = \{0, 1, 2, \dots; \omega, \omega + 1, \omega + 2, \dots\}$ is also well-ordered, so we can do induction on it if we can show $P(0)$, $P(x) \to P(x + 1)$ (including for cases

[6]Proof: We can prove that any nonempty $S \subseteq \mathbb{N}$ has a minimum in a slightly round-about way by induction. The induction hypothesis (for x) is that if S contains some element y less than or equal to x, then S has a minimum element. The base case is when $x = 0$; here x is the minimum. Suppose now that the claim holds for x. Suppose also that S contains some element $y \leq x + 1$; if not, the induction hypothesis holds vacuously. If there is some $y \leq x$, then S has a minimum by the induction hypothesis. The alternative is that there is no y in S such that $y \leq x$, but there is a y in S with $y \leq x + 1$. This y must be equal to $x + 1$, and so y is the minimum.

like $x = \omega + 5$ and $x + 1 = \omega + 6$), and $P(\omega)$ (possibly using $P(x)$ for all $x < \omega$ in the last case).

9.6 Closures

In general, the **closure** of some mathematical object with respect to a given property is the smallest larger object that has the property. Usually "smaller" and "larger" are taken to mean subset or superset, so we are really looking at the intersection of all larger objects with the property. Such a closure always exists if the property is preserved by intersection (formally, if $(\forall i : P(S_i)) \rightarrow P(\cap S_i))$ and every object has at least one larger object with the property.

This rather abstract definition can be made more explicit for certain kinds of closures of relations. The **reflexive closure** of a relation R (whose domain and codomain are equal) is the smallest super-relation of R that is reflexive; it is obtained by adding (x, x) to R for all x in R's domain. The **symmetric closure** is the smallest symmetric super-relation of R; it is obtained by adding (y, x) to R whenever (x, y) is in R, or equivalently by taking $R \cup R^{-1}$. The **transitive closure** is obtained by adding (x, z) to R whenever (x, y) and (y, z) are both in R for some y—and continuing to do so until no new pairs of this form remain.[7] The transitive closure can also be computed as $R^+ = R^1 \cup R^2 \cup R^3 \ldots$; for reflexive R, this is equal to $R^* = R^0 \cup R^1 \cup R^2 \ldots$. Even if R is not already reflexive, R^* gives the **reflexive transitive closure** of R.

In digraph terms, the reflexive closure adds self-loops to all nodes, the symmetric closure adds a reverse edge for each edge, and the transitive closure adds an edge for each directed path through the graph (see Figure 9.6. One can also take the closure with respect to multiple properties, such as the **reflexive symmetric transitive closure** of R which will be the smallest equivalence relation in which any elements that are related by R are equivalent.

Closures provide a way of turning things that aren't already equivalence relations or partial orders into equivalence relations and partial orders. For equivalence relations this is easy: take the reflexive symmetric transitive closure, and you get a reflexive symmetric transitive relation. For partial orders it's trickier: antisymmetry isn't a closure property (even though it's preserved by intersection, a non-antisymmetric R can't be made antisymmetric by adding more pairs). Given a relation R on some set S, the

[7]This may not actually terminate if R's domain is not finite.

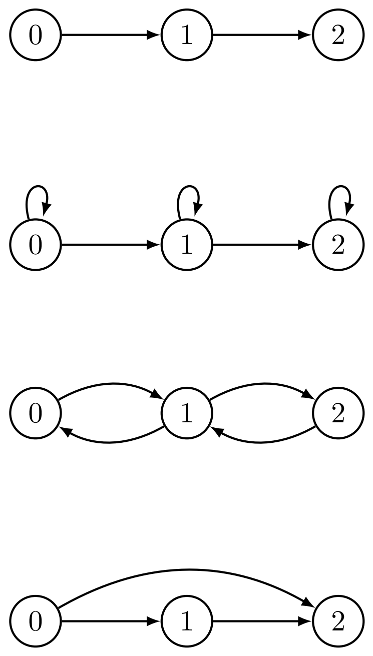


Figure 9.6: Reflexive, symmetric, and transitive closures of a relation represented as a directed graph. The original relation $\{(0,1),(1,2)\}$ is on top; the reflexive, symmetric, and transitive closures are depicted below it.

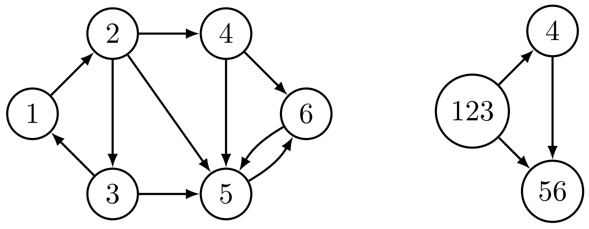


Figure 9.7: Reducing a graph to its strongly-connected components. On the left is the original graph. On the right, each strongly-connected component has been contracted to a single vertex. The contracted graph is acyclic.

best we can do is take the reflexive transitive closure R^* and hope that it's antisymmetric. If it is, we are done. If it isn't, we can observe that the relation $\sim$ defined by $x \sim y$ if xR^*y and yR^*x is an equivalence relation (Proof: $x \sim x$ because R^* is reflexive, $x \sim y \rightarrow y \sim x$ from the symmetry of the definition, and $x \sim y \wedge y \sim z \rightarrow x \sim z$ because transitivity of R^* gives $xR^*y \wedge yR^*z \rightarrow xR^*z$ and $yR^{*x} \wedge zR^*y \rightarrow zR^*x$). So we can take the quotient $S/\sim$, which smashes all the equivalence classes of $\sim$ into single points, define a quotient relation $R^*/\sim$ in the obvious way, and this quotient relation will be a partial order. This is the relational equivalent of the standard graph-theoretic algorithm that computes **strongly-connected components** (the equivalence classes of $\sim$) and constructs a **directed acyclic graph** from the original by contracting each strongly-connected component to a single vertex. See Figure 9.7 for an example.

9.6.1 Examples

- Let R be the relation on subsets of $\mathbb{N}$ given by xRy if there exists some $n \notin x$ such that $y = x \cup \{n\}$. The transitive closure of R is the proper subset relation $\subset$, where $x \subset y$ if $x \subseteq y$ but $x \neq y$. The reflexive transitive closure R^* of R *is* just the ordinary subset relation $\subseteq$. The reflexive symmetric transitive closure of R is the complete relation; given any two sets x and y, we can get from x to $\emptyset$ via $(R^*)^{-1}$ and then to y via R^*. So in this case the reflexive symmetric transitive closure is not very interesting.

- Let R be the relation on $\mathbb{N}$ given by xRy if $x = 2y$. Then the reflexive transitive closure R^* is the relation given by xR^*y if $x = 2^n y$ for some $n \in \mathbb{N}$, and the reflexive symmetric transitive closure is the relation given by $x \sim y$ if $x = 2^n y$ or $y = 2^n x$ for some $n \in \mathbb{N}$. For this R, not all elements of the underlying set are equivalent in the reflexive symmetric transitive closure; instead, we get a separate equivalence class $\{k, 2k, 4k, 8k, \dots\}$ for each odd number k.

Chapter 10

Graphs

A **graph** is a structure in which pairs of **vertices** are connected by **edges**. Each edge may act like an ordered pair (in a **directed graph**) or an unordered pair (in an **undirected graph**). We've already seen directed graphs as a representation for relations. Most work in graph theory concentrates instead on undirected graphs.

Because graph theory has been studied for many centuries in many languages, it has accumulated a bewildering variety of terminology, with multiple terms for the same concept (*e*.g. **node** for vertex or **arc** for edge) and ambiguous definitions of certain terms (*e*.g., a "graph" without qualification might be either a directed or undirected graph, depending on who is using the term: graph theorists tend to mean undirected graphs, but you can't always tell without looking at the context). We will try to stick with consistent terminology to the extent that we can. In particular, unless otherwise specified, a *graph* will refer to a **finite simple undirected graph**: an undirected graph with a finite number of vertices, where each edge connects two distinct vertices (thus no **self-loops**) and there is at most one edge between each pair of vertices (no **parallel edges**).

A reasonably complete glossary of graph theory can be found at at `http://en.wikipedia.org/wiki/Glossary_of_graph_theory`. See also Ferland [Fer08], Chapters 8 and 9; Rosen [Ros12] Chapter 10; or Biggs [Big02] Chapter 15 (for undirected graphs) and 18 (for directed graphs).

If you want to get a fuller sense of the scope of graph theory, Reinhard Diestel's (graduate) textbook *Graph Theory*[Die10] can be downloaded from `http://diestel-graph-theory.com`.

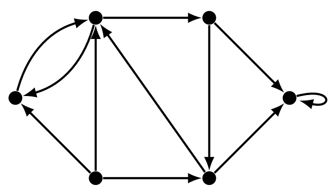


Figure 10.1: A directed graph

10.1 Types of graphs

Graphs are represented as ordered pairs $G = (V, E)$, where V is a set of vertices and E a set of edges. The differences between different types of graphs depends on what can go in E. When not otherwise specified, we usually think of a *graph* as an *undirected graph* (see below), but there are other variants.

10.1.1 Directed graphs

In a **directed graph** or *digraph*, each element of E is an ordered pair, and we think of edges as arrows from a **source**, **head**, or **initial vertex** to a **sink**, **tail**, or **terminal vertex**; each of these two vertices is called an **endpoint** of the edge. A directed graph is **simple** if there is at most one edge from one vertex to another. A directed graph that has multiple edges from some vertex u to some other vertex v is called a **directed multigraph**.

For simple directed graphs, we can save a lot of ink by adopting the convention of writing an edge (u, v) from u to v as just uv.

Directed graphs are drawn as in Figure 10.1.

As we saw in the notes on relations, there is a one-to-one correspondence between simple directed graphs with vertex set V and relations on V.

10.1.2 Undirected graphs

In an **undirected graph**, each edge is an undirected pair, which we can represent as subset of V with one or two elements. A **simple undirected graph** contains no duplicate edges and no **loops** (an edge from some vertex u back to itself); this means we can represent all edges as two-element subsets of V. Most of the time, when we say *graph*, we mean a simple undirected graph. Though it is possible to consider infinite graphs, for convenience we will limit ourselves to finite graphs, where $n = |V|$ and $m = |E|$ are both natural numbers.

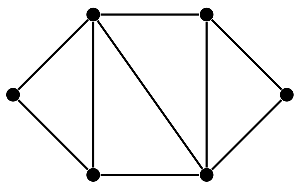


Figure 10.2: A graph

As with directed graphs, instead of writing an edge as $\{u, v\}$, we will write an edge between u and v as just uv. Note that in an undirected graph, uv and vu are the same edge.

Graphs are drawn just like undirected graphs, except that the edges don't have arrowheads on them. See Figure 10.2 for an example.

If we have loops or parallel edges, we have a more complicated structure called a **multigraph**. This requires a different representation where elements of E are abstract edges and we have a function mapping each element of E to its endpoints. Some authors make a distinction between *pseudographs* (with loops) and *multigraphs* (without loops), but we'll use multigraph for both.

Simple undirected graphs also correspond to relations, with the restriction that the relation must be irreflexive (no loops) and symmetric (undirected edges). This also gives a representation of undirected graphs as directed graphs, where the edges of the directed graph always appear in pairs going in opposite directions.

10.1.3 Hypergraphs

In a **hypergraph**, the edges (called **hyperedges**) are arbitrary nonempty sets of vertices. A k-**hypergraph** is one in which all such hyperedges connected exactly k vertices; an ordinary graph is thus a 2-hypergraph.

Hypergraphs can be drawn by representing each hyperedge as a closed curve containing its members, as in the left-hand side of Figure 10.3.

Hypergraphs aren't used very much, because it is always possible (though not always convenient) to represent a hypergraph by a **bipartite graph**. In a bipartite graph, the vertex set can be partitioned into two subsets S and T, such that every edge connects a vertex in S with a vertex in T. To represent a hypergraph H as a bipartite graph, we simply represent the vertices of H as vertices in S and the hyperedges of H as vertices in T, and put in an edge (s, t) whenever s is a member of the hyperedge t in H. The right-hand side of Figure 10.3 gives an example.

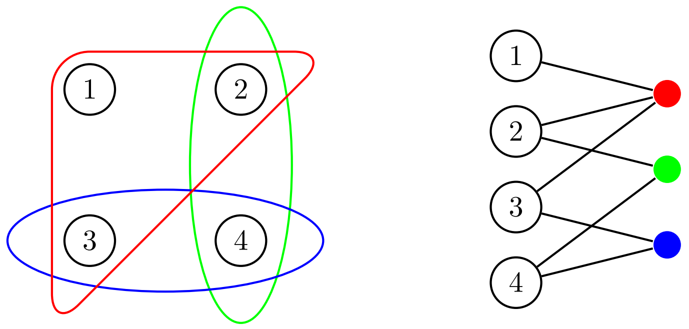
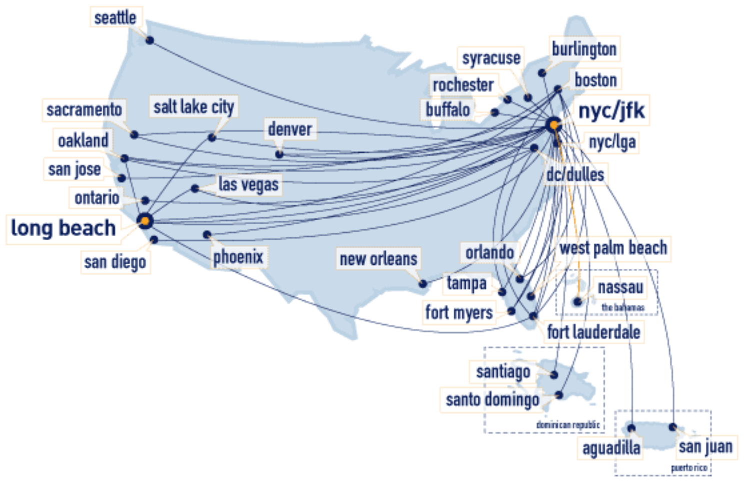


Figure 10.3: Two representations of a hypergraph. On the left, four vertices are connected by three hyperedges. On the right, the same four vertices are connected by ordinary edges to new vertices representing the hyperedges.

10.2 Examples of graphs

Any relation produces a graph, which is directed for an arbitrary relation and undirected for a symmetric relation. Examples are graphs of parenthood (directed), siblinghood (undirected), handshakes (undirected), etc.

Graphs often arise in transportation and communication networks. Here's a (now very out-of-date) route map for Jet Blue airlines, originally taken from `http://www.jetblue.com/travelinfo/routemap.html`:



Such graphs are often **labeled** with edge lengths, prices, etc. In computer networking, the design of network graphs that permit efficient routing of data without congestion, roundabout paths, or excessively large routing tables is a central problem.

The **web graph** is a directed multigraph with web pages for vertices and hyperlinks for edges. Though it changes constantly, its properties have been fanatically studied both by academic graph theorists and employees of search engine companies, many of which are still in business. Companies like Google base their search rankings largely on structural properties of the web graph.

Peer-to-peer systems for data sharing often have a graph structure, where each peer is a node and connections between peers are edges. The problem of designing efficient peer-to-peer systems is similar in many ways to the problem of designing efficient networks; in both cases, the structure (or lack thereof) of the underlying graph strongly affects efficiency.

10.3 Local structure of graphs

There are some useful standard terms for describing the immediate connections of vertices and edges:

- Incidence: a vertex is **incident** to any edge of which it is an endpoint (and vice versa).
- Adjacency, neighborhood: two vertices are **adjacent** if they are the endpoints of some edge. The **neighborhood** of a vertex v is the set of all vertices that are adjacent to v.
- Degree, in-degree, out-degree: the **degree** of v counts the number edges incident to v. In a directed graph, **in-degree** counts only incoming edges and **out-degree** counts only outgoing edges (so that the degree is always the in-degree plus the out-degree). The degree of a vertex v is often abbreviated as $d(v)$; in-degree and out-degree are similarly abbreviated as $d^{-}(v)$ and $d^{+}(v)$, respectively.

10.4 Some standard graphs

Most graphs have no particular structure, but there are some families of graphs for which it is convenient to have standard names. Some examples are:

- **Complete graph** K_n. This has n vertices, and every pair of vertices has an edge between them. See Figure 10.4.

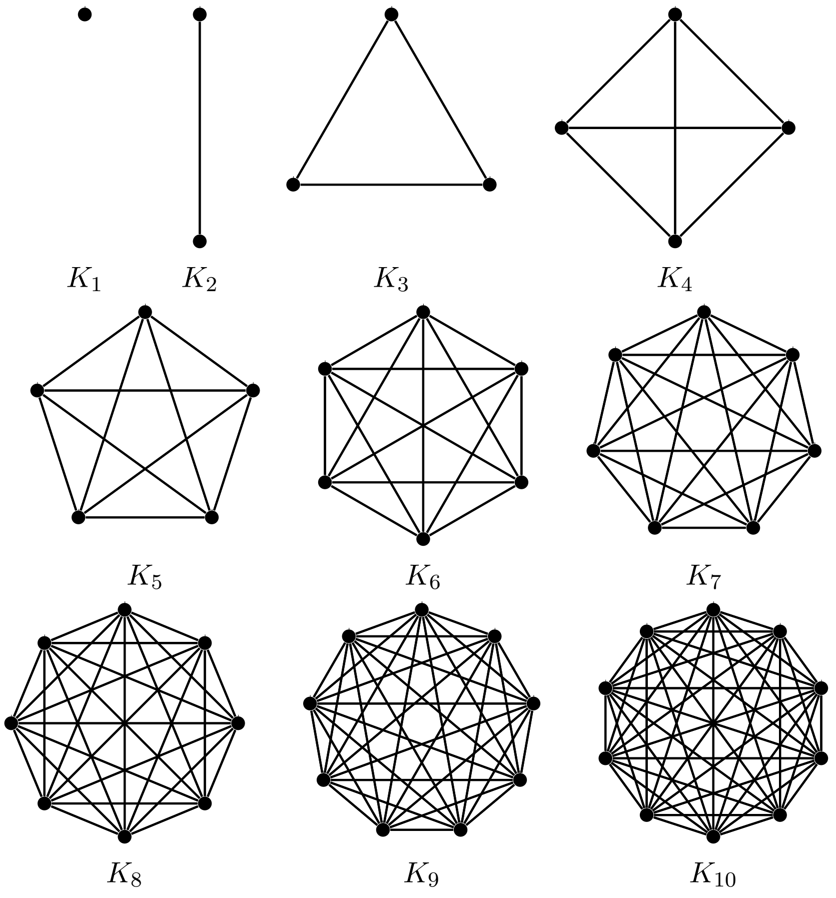


Figure 10.4: Complete graphs K_1 through K_{10}

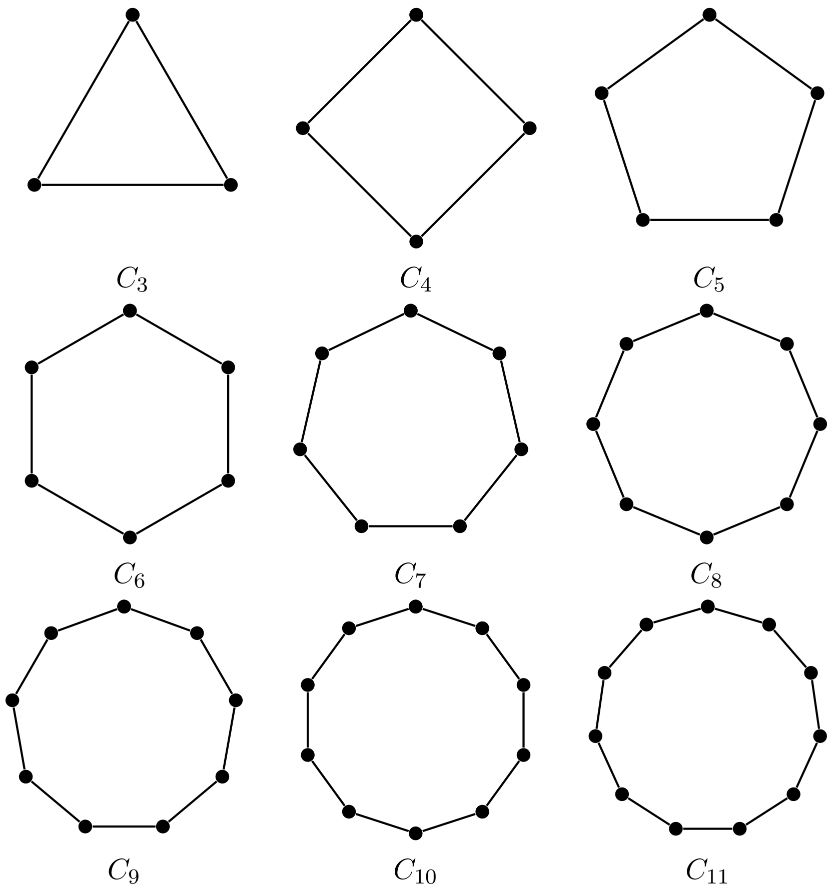


Figure 10.5: Cycle graphs C_3 through C_{11}

- **Cycle** graph C_n. This has vertices $\{0, 1, \dots n-1\}$ and an edge from i to $i+1$ for each i, plus an edge from $n-1$ to 0. For any cycle, n must be at least 3. See Figure 10.5.

- **Path** P_n. This has vertices $\{0, 1, 2, \dots n\}$ and an edge from i to $i+1$ for each i. Note that, despite the usual convention, n counts the number of *edges* rather than the number of vertices; we call the number of edges the **length** of the path. See Figure 10.6.

- **Complete bipartite graph** $K_{m,n}$. This has a set A of m vertices and a set B of n vertices, with an edge between every vertex in A and every vertex in B, but no edges within A or B. See Figure 10.7.

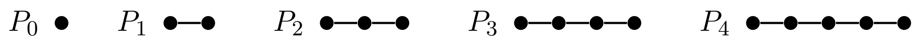


Figure 10.6: Path graphs P_0 through P_4

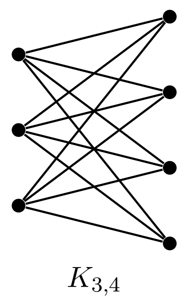


Figure 10.7: Complete bipartite graph $K_{3,4}$

- **Star graphs**. These have a single central vertex that is connected to n outer vertices, and are the same as $K_{1,n}$. See Figure 10.8.
- The **cube** Q_n. This is defined by letting the vertex set consist of all n-bit strings, and putting an edge between u and u' if u and u' differ in exactly one place. It can also be defined by taking the n-fold square product of an edge with itself (see §10.6).

Graphs may not always be drawn in a way that makes their structure obvious. For example, Figure 10.9 shows two different presentations of Q_3, neither of which looks much like the other.

10.5 Subgraphs and minors

A graph G is a **subgraph** of of a graph H, written $G \subseteq H$, if $V_G \subseteq V_H$ and $E_G \subseteq E_H$. We will also sometimes say that G is a subgraph of H if it is isomorphic to a subgraph of H, which is equivalent to having an injective homomorphism from G to H.

One can get a subgraph by deleting edges or vertices or both. Note that deleting a vertex also requires deleting any edges incident to the vertex (since we can't have an edge with a missing endpoint). If we delete as few edges as possible, we get an **induced subgraph**. Formally, the subgraph of a graph H whose vertex set is S and that contains every edge in H with endpoints in S is called the subgraph of H induced by S.

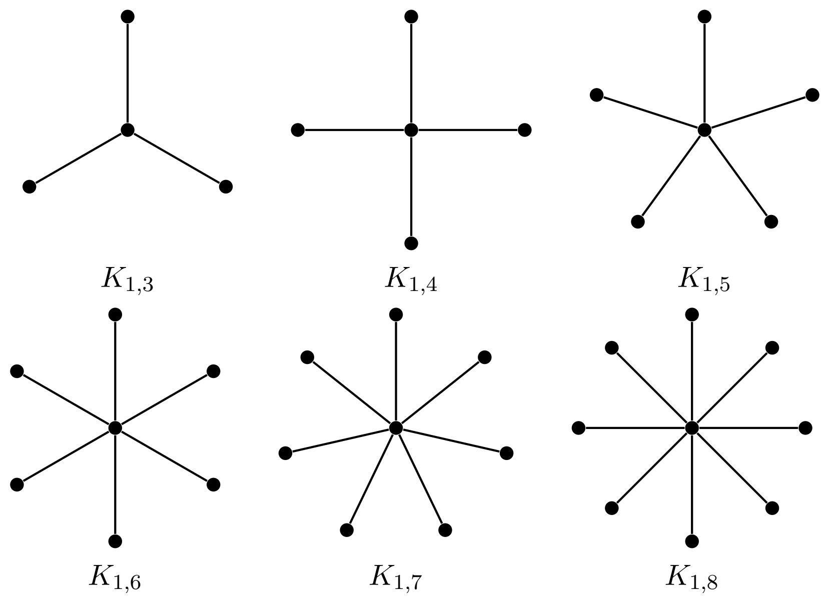


Figure 10.8: star graphs $K_{1,3}$ through $K_{1,8}$

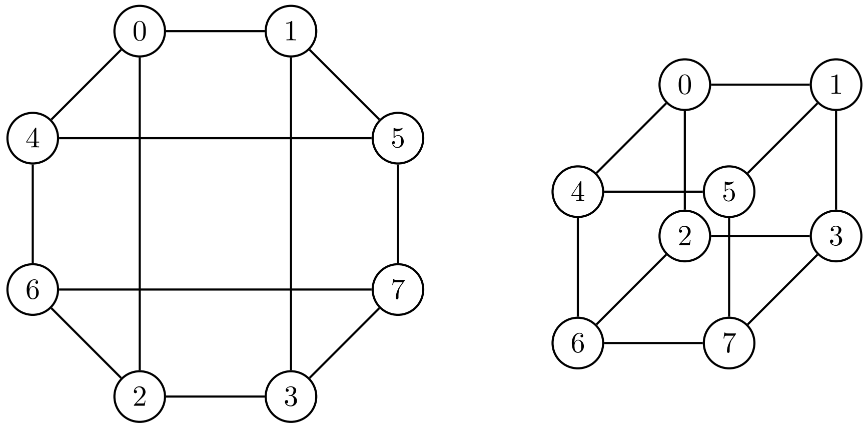


Figure 10.9: Two presentations of the cube graph Q_3

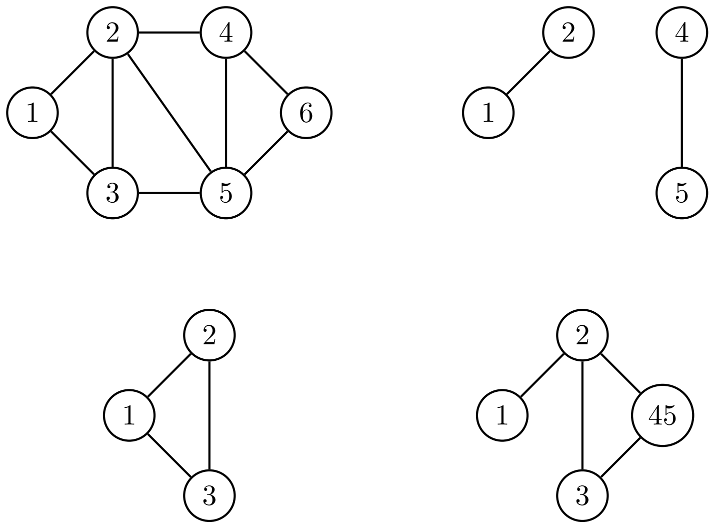


Figure 10.10: Examples of subgraphs and minors. Top left is the original graph. Top right is a subgraph that is not an induced subgraph. Bottom left is an induced subgraph. Bottom right is a minor.

A **minor** of a graph H is a graph obtained from H by deleting edges and/or vertices (as in a subgraph) and **contracting** edges, where two adjacent vertices u and v are merged together into a single vertex that is adjacent to all of the previous neighbors of both vertices. Minors are useful for recognizing certain classes of graphs. For example, a graph can be drawn in the plane without any crossing edges if and only if it doesn't contain K_5 or $K_{3,3}$ as a minor (this is known as **Wagner's theorem**).

Figure 10.10 shows some subgraphs and minors of the graph from Figure 10.2.

10.6 Graph products

There are at least five different definitions of the product of two graphs used by serious graph theorists. In each case the vertex set of the product is the Cartesian product of the vertex sets, but the different definitions throw in different sets of edges. Two of them are used most often:

- The **square product** or **graph Cartesian product** $G \mathbin{\square} H$. An edge $(u, u')(v, v')$ is in $G \mathbin{\square} H$ if and only if (a) $u = v$ and $u'v'$ is an edge in H, or (b) uv is an edge in G and $v = v'$. It's called the square product because the product of two (undirected) edges looks like a square. The intuition is that each vertex in G is replaced by a copy of H, and then corresponding vertices in the different copies of H are linked whenever the original vertices in G are adjacent. For algebraists, square products are popular because they behave correctly for Cayley graphs: if C_1 and C_2 are the Cayley graphs of G_1 and G_2 (for particular choices of generators), then $C_1 \square C_2$ is the Cayley graph of $G_1 \times G_2$.

  - The cube Q_n can be defined recursively by $Q_1 = P_1$ and $Q_n = Q_{n-1} \square Q_1$. It is also the case that $Q_n = Q_k \square Q_{n-k}$.
  - An n-by-m **mesh** is given by $P_{n-1} \square P_{m-1}$.

- The **cross product** or **categorical graph product** $G \times H$. Now $(u, u')(v, v')$ is in $G \times H$ if and only if uv is in G and $u'v'$ is in H. In the cross product, the product of two (again undirected) edges is a cross: an edge from (u, u') to (v, v') and one from (u, v') to (v, u'). The cross product is not as useful as the square product for defining nice-looking graphs, but it can arise in some other situations. An example is when G and H describe the positions (vertices) and moves (directed edges) of two solitaire games; then the cross product $G \times H$ describes the combined game in which at each step the player must make a move in both games. (In contrast, the square product $G \square H$ describes a game where the player can choose at each step to make a move in either game.)

10.6.1 Functions

A function from a graph G to another graph H typically maps V_G to V_H, with the edges coming along for the ride. Functions between graphs can be classified based on what they do to the edges:

- Isomorphisms: If $f : V_G \to V_H$ is a bijection and $uv \in E_G$ if and only if $f(u)f(v) \in E_G$, then G and H are said to be **isomorphic** and f is an **isomorphism**. Isomorphism is an equivalence relation—using it, we can divide graphs into equivalence classes and effectively forget the identities of the vertices. We don't currently know how to test whether two graphs are isomorphic (this is the **Graph Isomorphism Problem**); at the same time, we also don't know that testing isomorphism is hard, even assuming $P \neq NP$. Graph isomorphism is a rare example of a natural problem for which we have neither a good algorithm nor a (conditional) hardness proof.

- Automorphisms: An isomorphism from G to G is called an **automorphism** of G. Automorphisms correspond to internal symmetries of a graph. For example, the cycle C_n has $2n$ different automorphisms (to count them, observe there are n places we can send vertex 0 to, and having picked a place to send vertex 0 to, there are only 2 places to send vertex 1; so we have essentially n rotations times 2 for flipping or not flipping the graph). A path P_n (when $n > 1$) has 2 automorphisms (reverse the direction or not). Many graphs have no automorphisms except the identity map.

- Homomorphisms: A **homomorphism** f from a graph $G = (V_G, E_G)$ to a graph $H = (V_H, E_H)$ is a function $f : V_G \to V_H$ such that, for all uv in E_G, $f(u)f(v)$ is in E_H. (This definition assumes no parallel edges, but otherwise works for both directed and undirected graphs.) These are not generally as useful as isomorphisms or automorphisms, but can sometimes be used to characterize particular classes of graphs indirectly.

  - Intuition: a homomorphism is a function from vertices of G to vertices of H that also maps edges to edges.
  - Example: There is a unique homomorphism from the empty graph $(\emptyset, \emptyset)$ to any graph.
  - Example: Let $G = (V, E)$ be an undirected graph. Then there are exactly 2 homomorphisms from P_1 to G for each edge in G.
  - Example: There is a homomorphism from G to P_1 if and only if G is bipartite (see §A.7.2 for a proof). In general, there is a homomorphism from G to K_n if and only if G is n-partite (recall $P_1 = K_2$).
  - Comment: For multigraphs, one must define a homomorphism as a pair of functions $f_V : V_G \to V_H$ and $f_E : E_G \to E_H$ with appropriate consistency conditions.

10.7 Paths and connectivity

A fundamental property of graphs is connectivity: whether the graph can be divided into two or more pieces with no edges between them. Often it makes sense to talk about this in terms of reachability, or whether you can get from one vertex to another along some **path**.

- A **path** of **length** n in a graph is the image of a homomorphism from P_n.
  - In ordinary speech, it's a sequence of $n+1$ vertices $v_0, v_1, \dots, v_n$ such that $v_i v_{i+1}$ is an edge in the graph for each i.
  - A path is **simple** if the same vertex never appears twice (i.e. if the homomorphism is injective). If there is a path from u to v, there is a simple path from u to v obtained by removing cycles (Lemma 10.9.1).
- If there is a path from u to v, then v is **reachable** from u: $u \xrightarrow{*} v$. We also say that u is **connected to** v.
  - It's easy to see that connectivity is reflexive (take a path of length 0) and transitive (paste a path from u to v together with a path from v to w to get a path from u to w). But it's not necessarily symmetric if we have a directed graph.
- Connected components
  - In an undirected graph, connectivity *is* symmetric, so it's an equivalence relation.
    - Equivalence classes of $\xrightarrow{*}$ are called the **connected components** of G.
    - G itself is **connected** if and only if it has a single connected component, i.e., if every vertex is reachable from every other vertex.
    - Note that isolated vertices count as (separate) connected components.
  - In a directed graph, we can make connectivity symmetric in one of two different ways:
    - Define u to be **strongly connected** to v if $u \xrightarrow{*} v$ and $v \xrightarrow{*} u$. I.e., u and v are strongly connected if you can go from u to v and back again (not necessarily through the same vertices).
      - It's easy to see that strong connectivity is an equivalence relation.

        · The equivalence classes are called **strongly-connected components**.
        · A graph G is **strongly connected** if it has one strongly-connected component, i.e., if every vertex is reachable from every other vertex.
    * Define u to be **weakly connected** to v if $u \xrightarrow{*} v$ in the undirected graph obtained by ignoring edge orientation.
        · Intuition is that u is weakly connected to v if there is a path from u to v if you are allowed to cross edges backwards.
        · Weakly-connected components are defined by equivalence classes; a graph is weakly-connected if it has one component.
        · Weak connectivity is a "weaker" property that strong connectivity in the sense that if u is strongly connected to v, then u is weakly connected to v; but the converse does not necessarily hold.

- **Power** of a directed graph: The k-th power G^k has the same vertices as G, but uv is an edge in G^k if and only if there is a path of length k from u to v in G.

- **Transitive closure** of a directed graph: $G^* = \bigcup_{k=0}^{\infty} G^k$. I.e., there is an edge uv in G^* if and only if there is a path (of any length, including zero) from u to v in G, or in other words if $u \xrightarrow{*} v$. This is equivalent to taking the transitive closure of the adjacency relation.

10.8 Cycles

The standard cycle graph C_n has vertices $\{0, 1, \dots, n-1\}$ with an edge from i to $i+1$ for each i and from $n-1$ to 0. To avoid degeneracies, n must be at least 3. A **simple cycle** of length n in a graph G is an embedding of C_n in G: this means a sequence of distinct vertices $v_0 v_1 v_2 \dots v_{n-1}$, where each pair $v_i v_{i+1}$ is an edge in G, as well as $v_{n-1} v_0$. If we omit the requirement that the vertices are distinct, but insist on distinct edges instead, we have a **cycle**. If we omit both requirements, we get a **closed walk**; this includes very non-cyclic-looking walks like the short excursion uvu. We will mostly worry about cycles.[1] See Figure 10.11

[1]Some authors, including Ferland [Fer08], reserve *cycle* for what we are calling a *simple cycle*, and use *circuit* for *cycle*.

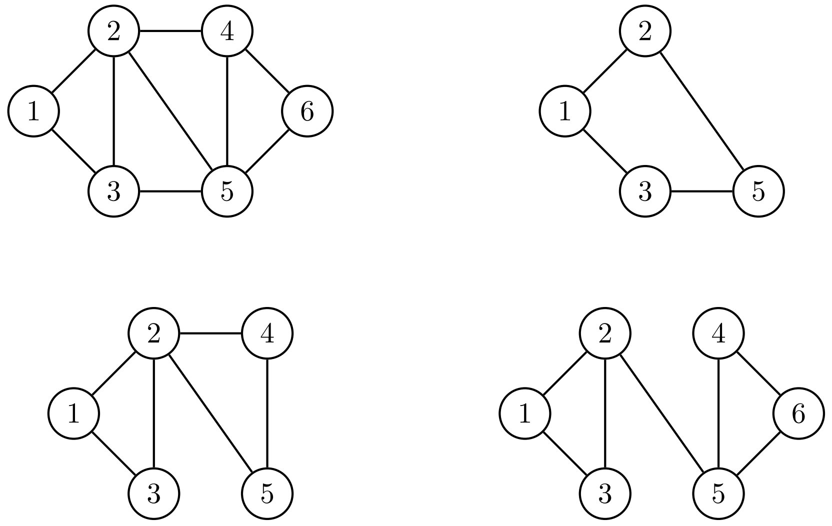


Figure 10.11: Examples of cycles and closed walks. Top left is a graph. Top right shows the simple cycle 1253 found in this graph. Bottom left shows the cycle 124523, which is not simple. Bottom right shows the closed walk 12546523.

Unlike paths, which have endpoints, no vertex in a cycle has a special role.

A graph with no cycles is **acyclic**. **Directed acyclic graphs** or **DAGs** have the property that their reachability relation $\overset{*}{\to}$ is a partial order; this is easily proven by showing that if $\overset{*}{\to}$ is not anti-symmetric, then there is a cycle consisting of the paths between two non-anti-symmetric vertices $u \overset{*}{\to} v$ and $v \overset{*}{\to} u$. Directed acyclic graphs may also be **topologically sorted**: their vertices ordered as $v_0, v_1, \dots, v_{n-1}$, so that if there is an edge from v_i to v_j, then $i < j$. The proof is by induction on $|V|$, with the induction step setting v_{n-1} to equal some vertex with out-degree 0 and ordering the remaining vertices recursively. (See §9.5.5.1.)

Connected acyclic undirected graphs are called **trees**. A connected graph $G = (V, E)$ is a tree if and only if $|E| = |V| - 1$; we'll prove this and other characterizations of tree in §10.9.3.

A cycle that includes every edge exactly once is called an **Eulerian cycle** or **Eulerian tour**, after Leonhard Euler, whose study of the *Seven bridges of Königsberg* problem led to the development of graph theory. A cycle that includes ever vertex exactly once is called a **Hamiltonian cycle** or **Hamiltonian tour**, after William Rowan Hamilton, another historical graph-theory heavyweight (although he is more famous for inventing quaternions and the Hamiltonian). Graphs with Eulerian cycles have a simple characterization: a graph has an Eulerian cycle if and only if every vertex has even degree. Graphs with Hamiltonian cycles are harder to recognize.

10.9 Proving things about graphs

Suppose we want to show that all graphs or perhaps all graphs satisfying certain criteria have some property. How do we do this? In the ideal case, we can decompose the graph into pieces somehow and use induction on the number of vertices or the number of edges. If this doesn't work, we may have to look for some properties of the graph we can exploit to construct an explicit proof of what we want.

10.9.1 Paths and simple paths

If all we care about is connectivity, we can avoid making distinctions between paths and simple paths.

Lemma 10.9.1. *If there is a path from s to t in G, there is a simple path from s to t in G.*

Proof. By induction on the length of the path. Specifically, we will show that if there is a path from s to t of length k, there is a simple path from s to t.

The base case is when $k = 1$; then the path consists of exactly one edge and is simple.

For larger k, let $s = v_0 \dots v_k = t$ be a path in G. If this path is simple, we are done. Otherwise, there exist positions $i < j$ such that $v_i = v_j$. Construct a new path $v_1 \dots v_i v_{j+1} \dots v_k$; this is an s–t path of length less than k, so by the induction hypothesis a simple s–t path exists. ☐

The converse of this lemma is trivial: any simple path is also a path.

Essentially the same argument works for cycles:

Lemma 10.9.2. *If there is a cycle in G, there is a simple cycle in G.*

Proof. As in the previous lemma, we prove that there exists a simple cycle if there is a cycle of length k for any k, by induction of k. The base case is $k = 3$: all 3-cycles are simple. For larger k, if $v_0 v_1 \dots v_{k-1}$ is a k-cycle that is not simple, there exist $i < j$ with $v_i = v_j$; patch the edges between them out to get a smaller cycle $v_0 \dots v_i v_{j+1} \dots v_{k-1}$. The induction hypothesis does the rest of the work. ☐

10.9.2 The Handshaking Lemma

This lemma relates the total degree of a graph to the number of edges. The intuition is that each edge adds one to the degree of both of its endpoints, so the total degree of all vertices is twice the number of edges.

Lemma 10.9.3. *For any graph $G = (V, E)$,*

$$\sum_{v \in V} d(v) = 2\left|E\right|.$$

Proof. By induction on $m = |E|$. If $m = 0$, G has no edges, and $\sum_{v \in V} d(v) = \sum_{v \in V} 0 = 0 = 2m$. If $m > 0$, choose some edge st and let $G' = G - st$ be the subgraph of G obtained by removing st. Applying the induction hypothesis to G',

$$\begin{aligned}
2(m-1) &= \sum_{v \in V} d_{G'}(v) \\
&= \sum_{v \in V \setminus \{s,t\}} d_{G'}(v) + d_{G'}(s) + d_{G'}(t) \\
&= \sum_{v \in V \setminus \{s,t\}} d_G(v) + (d_G(s) - 1) + (d_G(t) - 1) \\
&= \sum_{v \in V} d_G(v) - 2.
\end{aligned}$$

So $\sum_{v \in V} d_G(v) - 2 = 2m - 2$, giving $\sum_{v \in V} d_G(v) = 2m$. ☐

One application of the lemma is that the number of odd-degree vertices in a graph is always even (take both sides mod 2). Another, that we'll use below, is that if a graph has very few edges, then it must have some low-degree vertices.

10.9.3 Characterizations of trees

A **tree** is defined to be an acyclic connected graph. There are several equivalent characterizations.

Theorem 10.9.4. *A graph is a tree if and only if there is exactly one simple path between any two distinct vertices.*

Proof. A graph G is defined to be connected if and only if there is at least one simple path between any two distinct vertices. So we'll show that it is acyclic if and only if there is at most one simple path between any two distinct vertices.

First, suppose that G has two distinct simple paths $u = v_1 v_2 \dots v_k = v$ and $u = v'_1 v'_2 \dots v'_\ell = v$. Let i be the largest index for which $v_i = v'_i$; under the assumption that the paths are distinct and simple , we have $i < \min(k, \ell)$. Let $j > i$ be the smallest index for which $v_j = v'_m$ for some $m > i$; we know that some such j exists because, if nothing else, $v_k = v_\ell$. Let m is the smallest such m.

Now construct a cycle $v_i v_{i+1} \dots v_j v'_{m-1} v'_{m-2} \dots v'_i = v_i$. This is in fact a simple cycle, since the v_r are all distinct, the v'_s are all distinct, and if any v_r with $i < r < j$ equals v'_s with $i < s < m$, then j or m is not minimal. It follows that if G has two distinct simple paths between the same vertices, it contains a simple cycle, and is not acylic.

Conversely, suppose that G is not acyclic, and let $v_1 v_2 \dots v_k = v_1$ be a simple cycle in G. Then $v_1 v_2$ and $v_2 \dots v_k$ are both simple paths between v_1 and v_2, one of which contains v_3 and one of which doesn't. So if G is not acylic, it contains more than one simple path between some pair of vertices. ☐

An alternative characterization counts the number of edges: we will show that any graph with less than $|V| - 1$ edges is disconnected, and any graph with less than $|V| - 1$ edges is cyclic. With exactly $|V| - 1$ edges, we will show that a graph is connected if and only if it is acyclic.

The main trick involves reducing a $|V|$ by removing a degree-1 vertex. The following lemma shows that this does not change whether or not the graph is connected or acyclic:

Lemma 10.9.5. *Let G be a nonempty graph, and let v be a vertex of G with $d(v) = 1$. Let $G - v$ be the induced subgraph of G obtained by deleting v and its unique incident edge. Then*

1. *G is connected if and only if $G - v$ is connected.*
2. *G is acyclic if and only if $G - v$ is acyclic.*

Proof. Let w be v's unique neighbor.

If G is connected, for any two vertices s and t, there is a simple s–t path. If neither s nor t is v, this path can't include v, because w would appear both before and after v in the path, violating simplicity. So for any s, t in $G - v$, there is an s–t path in $G - v$, and $G - v$ is connected.

Conversely, if $G - v$ is connected, then any s and t not equal to v remain connected after adding vw, and if $s = v$, for any t there is a path $w = v_1 \dots v_k = t$, from which we can construct a path $vv_1 \dots v_k = t$ from v to t. The case $t = v$ is symmetric.

If G contains a cycle, then it contains a simple cycle; this cycle can't include v, so $G - v$ also contains the cycle.

Conversely, if $G - v$ contains a cycle, this cycle is also in G. ☐

Because a graph with two vertices and fewer than one edges is not connected, Lemma 10.9.5 implies that any graph with fewer than $|V| - 1$ edges is not connected.

Corollary 10.9.6. *Let $G = (V, E)$. If $|E| < |V| - 1$, G is not connected.*

Proof. By induction on $n = |V|$.

For the base case, if $n = 0$, then $|E| = 0 \not< n - 1$.

For larger n, suppose that $n \geq 1$ and $|E| < n-1$. From Lemma 10.9.3 we have $\sum_v d(v) < 2n-2$, from which it follows that there must be at least one vertex v with $d(v) < 2$. If $d(v) = 0$, then G is not connected. If $d(v) = 1$, then G is connected if and only if $G - v$ is connected. But $G - v$ has $n - 1$ vertices and $|E| - 1 < n - 2$ edges, so by the induction hypothesis, $G - v$ is not connected. So in either case, $|E| < n-1$ implies G is not connected. ☐

In the other direction, combining the lemma with the fact that the unique graph K_3 with three vertices and at least three edges is cyclic tells us that any graph with at least as many edges as vertices is cyclic.

Corollary 10.9.7. *Let $G = (V, E)$. If $|E| > |V - 1|$, G contains a cycle.*

Proof. By induction on $n = |V|$.

For $n \leq 2$, $|E| \not> |V - 1|$, so the claim holds vacuously.[2]

For larger n, there are two cases:

1. Some vertex v has degree $d(v) \leq 1$. Let $G' = (V', E') = G - v$. Then $|E'| \geq |E| - 1 > |V| - 2 = |V'| - 1$, and by the induction hypothesis G' contains a cycle. This cycle is also in G.

2. Every vertex v in G has $d(v) \geq 2$. We'll go for a walk: starting at some vertex v_0, choose at each step a vertex v_{i+1} adjacent to v_i that does not already appear in the walk. This process finishes when we reach a node v_k all of whose neighbors appear in the walk in a previous

[2]In fact, no graph with $|V| \leq 2$ contains a cycle, but we don't need to use this.

position. One of these neighbors may be v_{k-1}; but since $d(v_k) \geq 2$, there is another neighbor $v_j \neq v_{k-1}$. So $v_j \dots v_k v_j$ forms a cycle.

□

Now we can prove the full result:

Theorem 10.9.8. *Let $G = (V, E)$ be a nonempty graph. Then any two of the following statements implies the third:*

1. *G is connected.*
2. *G is acyclic.*
3. *$|E| = |V| - 1$.*

Proof. We will use induction on n for some parts of the proof. The base case is when $n = 1$; then all three statements hold always. For larger n, we show:

- (1) and (2) imply (3): Use Corollary 10.9.6 and Corollary 10.9.7.
- (1) and (3) imply (2). From Lemma 10.9.3, $\sum_{v \in V} d(v) = 2(n-1) < 2n$. It follows that there is at least one v with $d(v) \leq 1$. Because G is connected, we must have $d(v) = 1$. So $G' = G - v$ is a graph with $n - 2$ edges and $n - 1$ vertices. It is connected by Lemma 10.9.5, and thus it is acylic by the induction hypothesis. Applying the other case of Lemma 10.9.5 in the other direction shows G is also acyclic.
- (2) and (3) imply (1). As in the previous case, G contains a vertex v with $d(v) \leq 1$. If $d(v) = 1$, then $G - v$ is a nonempty graph with $n - 2$ edges and $n - 1$ vertices that is acyclic by Lemma 10.9.5. It is thus connected by the induction hypothesis, so G is also connected by Lemma 10.9.5. If $d(v) = 0$, then $G - v$ has $n - 1$ edges and $n - 1$ vertices. From Corollary 10.9.7, $G - v$ contains a cycle, contradicting (2).

□

For an alternative proof based on removing edges, see [Big02, Theorem 15.5]. This also gives the useful fact that removing one edge from a tree gives exactly two components.

10.9.4 Spanning trees

Here's another induction proof on graphs. A **spanning tree** of a nonempty connected graph G is a subgraph of G that includes all vertices and is a tree (i.e., is connected and acyclic).

Theorem 10.9.9. *Every nonempty connected graph has a spanning tree.*

Proof. Let $G = (V, E)$ be a nonempty connected graph. We'll show by induction on $|E|$ that G has a spanning tree. The base case is $|E| = |V| - 1$ (the least value for which G can be connected); then G itself is a tree (by the theorem above). For larger $|E|$, the same theorem gives that G contains a cycle. Let uv be any edge on the cycle, and consider the graph $G - uv$; this graph is connected (since we can route any path that used to go through uv around the other edges of the cycle) and has fewer edges than G, so by the induction hypothesis there is some spanning tree T of $G - uv$. But then T also spans G, so we are done. □

10.9.5 Eulerian cycles

Let's prove the vertex degree characterization of graphs with Eulerian cycles. As in the previous proofs, we'll take the approach of looking for something to pull out of the graph to get a smaller case.

Theorem 10.9.10. *Let G be a connected graph. Then G has an Eulerian cycle if and only if all nodes have even degree.*

Proof.

- (Only if part). Fix some cycle, and orient the edges by the direction that the cycle traverses them. Then in the resulting directed graph we must have $d^-(u) = d^+(u)$ for all u, since every time we enter a vertex we have to leave it again. But then $d(u) = 2d^+(u)$ is even.

- (If part). Suppose now that $d(u)$ is even for all u. We will construct an Eulerian cycle on all nodes by induction on $|E|$. The base case is when $|E| = 2\,|V|$ and $G = C_{|V|}$. For a larger graph, choose some starting node u_1, and construct a path $u_1 u_2 \ldots$ by choosing an arbitrary unused edge leaving each u_i; this is always possible for $u_i \neq u_1$ since whenever we reach u_i we have always consumed an even number of edges on previous visits plus one to get to it this time, leaving at least one remaining edge to leave on. Since there are only finitely many edges and we can only use each one once, eventually we must get stuck, and this must occur with $u_k = u_1$ for some k. Now delete all the edges in $u_1 \dots u_k$ from G, and consider the connected components of $G - (u_1 \dots u_k)$. Removing the cycle reduces $d(v)$ by an even number, so within each such connected component the degree of all vertices is even. It follows from the induction hypothesis that each connected component has an Eulerian cycle. We'll now string these per-component cycles together using our original cycle: while traversing $u_1 \dots, u_k$ when we encounter some component for the first time, we take a detour around the component's cycle. The resulting merged cycle gives an Eulerian cycle for the entire graph.

□

Why doesn't this work for Hamiltonian cycles? The problem is that in a Hamiltonian cycle we have too many choices: out of the $d(u)$ edges incident to u, we will only use two of them. If we pick the wrong two early on, this may prevent us from ever fitting u into a Hamiltonian cycle. So we would need some stronger property of our graph to get Hamiltonicity.

Chapter 11

Counting

Counting is the process of creating a bijection between a set we want to count and some set whose size we already know. Typically this second set will be a finite ordinal $[n] = \{0, 1, \ldots, n-1\}$.[1]

Counting a set A using a bijection $f : A \to [n]$ gives its size $|A| = n$; this size is called the **cardinality** of n. As a side effect, it also gives a well-ordering of A, since $[n]$ is well-ordered as we can define $x \leq y$ for x, y in A by $x \leq y$ if and only if $f(x) \leq f(y)$. Often the quickest way to find f is to line up all the elements of A in a well-ordering and then count them off: the smallest element of A gets mapped to 0, the next smallest to 1, and so on. Stripped of the mathematical jargon, this is exactly what you were taught to do as a small child.

Usually we will not provide an explicit bijection to compute the size of a set, but instead will rely on standard counting principles based on how we constructed the set. The branch of mathematics that studies sets constructed by combining other sets is called **combinatorics**, and the sub-branch that counts these sets is called **enumerative combinatorics**. In this chapter, we're going to give an introduction to enumerative combinatorics, but this basically just means counting.

For infinite sets, cardinality is a little more complicated. The basic idea is that we define $|A| = |B|$ if there is a bijection between them. This gives an equivalence relation on sets[2], and we define $|A|$ to be the equivalence class of this equivalence relation that contains A. For the finite case we represent

[1]Starting from 0 is traditional in computer science, because it makes indexing easier. Normal people count to n using $\{1, 2, \ldots, n\}$.

[2]Reflexivity: the identity function is a bijection from A to A. Symmetry: if $f : A \to B$ is a bijection, so is $f^{-1} : B \to A$. Transitivity: if $f : A \to B$ and $g : B \to C$ are bijections, so is $(g \circ f) : A \to C$.

the equivalence classes by taking representative elements $[n]$.

For the most part we will concentrate on counting finite sets, but will mention where the rules for finite sets break down with infinite sets.

11.1 Basic counting techniques

Our goal here is to compute the size of some set of objects, e.g., the number of subsets of a set of size n, the number of ways to put k cats into n boxes so that no box gets more than one cat, etc.

In rare cases we can use the definition of the size of a set directly, by constructing a bijection between the set we care about and some canonical set $[n]$. For example, the set $S_n = \left\{x \in \mathbb{N} \mid x < n^2 \wedge \exists y : x = y^2\right\}$ has exactly n members, because we can generate it by applying the one-to-one correspondence $f(y) = y^2$ to the set $\{0, 1, 2, 3, \ldots, n-1\} = [n]$. But most of the time constructing an explicit one-to-one correspondence is too time-consuming or too hard, so instead we will show how to map set-theoretic operations to arithmetic operations, so that from a set-theoretic construction of a set we can often directly read off an arithmetic computation that gives the size of the set.

11.1.1 Equality: reducing to a previously-solved case

If we can produce a bijection between a set A whose size we don't know and a set B whose size we do, then we get $|A| = |B|$. Pretty much all of our proofs of cardinality will end up looking like this.

11.1.2 Inequalities: showing $|A| \leq |B|$ and $|B| \leq |A|$

We write $|A| \leq |B|$ if there is an injection $f : A \to B$, and similarly $|B| \leq |A|$ if there is an injection $g : B \to A$. If both conditions hold, then there is a bijection between A and B, showing $|A| = |B|$. This fact is trivial for finite sets, but for infinite sets—even though it is still true—the actual construction of the bijection is a little trickier.[3]

[3]The claim for general sets is known as the Cantor-Bernstein-Schroeder theorem. One way to prove this is to assume that A and B are disjoint and construct a (not necessarily finite) graph whose vertex set is $A \cup B$ and that has edges for all pairs $(a, f(a))$ and $(b, g(b))$. It can then be shown that the connected components of this graph consist of (1) finite cycles, (2) doubly-infinite paths (i.e., paths with no endpoint in either direction), (3) infinite paths with an initial vertex in A, and (4) infinite paths with an initial vertex in B. For vertexes in all but the last class of components, define $h(x)$ to be $f(x)$ if x is in A and $f^{-1}(x)$ if x is in B. (Note that we are abusing notation slightly here by defining $f^{-1}(x)$

Similarly, if we write $|A| \geq |B|$ to indicate that there is a surjection from A to B, then $|A| \geq |B|$ and $|B| \geq |A|$ implies $|A| = |B|$. The easiest way to show this is to observe that if there is a surjection $f : A \to B$, then we can get an injection $f' : B \to A$ by letting $f'(y)$ be any element of $\{x \mid f(x) = y\}$, thus reducing to the previous case (this requires the Axiom of Choice, but pretty much everybody assumes the Axiom of Choice). Showing an injection $f : A \to B$ and a surjection $g : A \to B$ also works.

For example, $|\mathbb{Q}| = |\mathbb{N}|$. Proof: $|\mathbb{N}| \leq |\mathbb{Q}|$ because we can map any n in $\mathbb{N}$ to the same value in $\mathbb{Q}$; this is clearly an injection. To show $|\mathbb{Q}| \leq |\mathbb{N}|$, observe that we can encode any element $\pm p/q$ of $\mathbb{Q}$, where p and q are both natural numbers, as a triple (s, p, q) where $(s \in \{0, 1\}$ indicates $+$ (0) or $-$ (1); this encoding is clearly injective. Then use the bijection from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ twice to crunch this triple down to a single natural number, getting an injection from $\mathbb{Q}$ to $\mathbb{N}$.

11.1.3 Addition: the sum rule

The **sum rule** computes the size of $A \cup B$ when A and B are disjoint.

Theorem 11.1.1. *If A and B are finite sets with $A \cap B = \emptyset$, then*

$$|A \cup B| = |A| + |B| .$$

Proof. Let $f : A \to [|A|]$ and $g : B \to [|B|]$ be bijections. Define $h : A \cup B \to [|A| + |B|]$ by the rule $h(x) = f(x)$ for $x \in A$, $h(x) = |A| + g(x)$ for $x \in B$.

To show that this is a bijection, define $h^{-1}(y)$ for y in $[|A| + |b|]$ to be $f^{-1}(y)$ if $y < |A|$ and $g^{-1}(y - |A|)$ otherwise. Then for any y in $[|A| + |B|]$, either

1. $0y < |A|$, y is in the codomain of f (so $h^{-1}(y) = f^{-1}(y) \in A$ is well-defined), and $h(h^{-1}(y)) = f(f^{-1}(y)) = y$.

2. $|A| \leq y < |A| + |B|$. In this case $0 \leq y - |A| < |B|$, putting $y - |A|$ in the codomain of g and giving $h(h^{-1}(y)) = g(g^{-1}(y - |A|)) + |A| = y$.

So h^{-1} is in fact an inverse of h, meaning that h is a bijection. ☐

to be the unique y that maps to x when it exists.) For the last class of components, the initial B vertex is not the image of any x under f; so for these we define $h(x)$ to be $g(x)$ if x is in B and $g^{-1}(x)$ if x is in A. This gives the desired bijection h between A and B.

In the case where A and B are not disjoint, we can make them disjoint by replacing them with $A' = \{0\} \times A$ and $B' = \{1\} \times B$. (This is a pretty common trick for enforcing disjoint unions.)

One way to think about this proof is that we are constructing a total order on $A \cup B$ by putting all the A elements before all the B elements. This gives a straightforward bijection with $[|A| + |B|]$ by the usual preschool trick of counting things off in order.

Generalizations: If $A_1, A_2, A_3 \dots A_k$ are **pairwise disjoint** (i.e., $A_i \cap A_j = \emptyset$ for all $i \neq j$), then

$$\left| \bigcup_{i=1}^{k} A_i \right| = \sum_{i=1}^{k} |A_i| .$$

The proof is by induction on k.

Example: As I was going to Saint Ives, I met a man with 7 wives, 28 children, 56 grandchildren, and 122 great-grandchildren. Assuming these sets do not overlap, how many people did I meet? Answer: 1+7+28+56+122=214.

11.1.3.1 For infinite sets

The sum rule works for infinite sets, too; technically, the sum rule is used to *define* $|A| + |B|$ as $|A \cup B|$ when A and B are disjoint. This makes cardinal arithmetic a bit wonky: if at least one of A and B is infinite, then $|A| + |B| = \max(|A|, |B|)$, since we can space out the elements of the larger of A and B and shove the elements of the other into the gaps.

11.1.3.2 The Pigeonhole Principle

A consequence of the sum rule is that if A and B are both finite and $|A| > |B|$, you can't have an injection from A to B. The proof is by contraposition. Suppose $f : A \to B$ is an injection. Write A as the union of $f^{-1}(x)$ for each $x \in B$, where $f^{-1}(x)$ is the set of y in A that map to x. Because each $f^{-1}(x)$ is disjoint, the sum rule applies; but because f is an injection there is at most one element in each $f^{-1}(x)$. It follows that $|A| = \sum_{x \in B} |f^{-1}(x)| \leq \sum_{x \in B} 1 = |B|$. (Question: Why doesn't this work for infinite sets?)

The Pigeonhole Principle generalizes in an obvious way to functions with larger domains; if $f : A \to B$, then there is some x in B such that $|f^{-1}(x)| \geq |A| / |B|$.

11.1.4 Subtraction

For any sets A and B, A is the disjoint union of $A \cap B$ and $A \setminus B$. So $|A| = |A \cap B| + |A \setminus B|$ (for finite sets) by the sum rule. Rearranging gives

$$|A \setminus B| = |A| - |A \cap B| . \tag{11.1.1}$$

What makes (11.1.1) particularly useful is that we can use it to compute the size of $A \cup B$ even if A and B overlap. The intuition is that if we just add $|A|$ and $|B|$, then we count every element of $A \cap B$ twice; by subtracting off $|A \cap B|$ we eliminate the overcount. Formally, we have

Theorem 11.1.2. *For any sets A and B,*

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Proof. Compute

$$\begin{aligned} |A \cup B| &= |A \cap B| + |A \setminus B| + |B \setminus A| \\ &= |A \cap B| + (|A| - |A \cap B|) + (|B| - |A \cap B|) \\ &= |A| + |B| - |A \cap B|. \end{aligned}$$

□

This is a special case of the **inclusion-exclusion formula**, which can be used to compute the size of the union of many sets using the size of pairwise, triple-wise, etc. intersections of the sets. See §11.2.4 for the general rule.

11.1.4.1 Inclusion-exclusion for infinite sets

Subtraction doesn't work very well for infinite quantities (while $\aleph_0+\aleph_0 = \aleph_0$, that doesn't mean $\aleph_0 = 0$). So the closest we can get to the inclusion-exclusion formula is that $|A| + |B| = |A \cup B| + |A \cap B|$. If at least one of A or B is infinite, then $|A \cup B|$ is also infinite, and since $|A \cap B| \leq |A \cup B|$ we have $|A \cup B| + |A \cap B| = |A \cup B|$ by the usual but bizarre rules of cardinal arithmetic. So for infinite sets we have the rather odd result that $|A \cup B| = |A| + |B| = \max(|A|, |B|)$ whether the sets overlap or not.

11.1.4.2 Combinatorial proof

We can prove $|A|+|B| = |A \cup B|+|A \cap B|$ combinatorially, by turning both sides of the equation into disjoint unions (so the sum rule works) and then providing an explicit bijection between the resulting sets. The trick is that we can always force a union to be disjoint by tagging the elements with extra information; so on the left-hand side we construct $L = \{0\} \times A \cup \{1\} \times B$, and on the right-hand side we construct $R = \{0\} \times (A \cup B) \cup \{1\} \times (A \cap B)$. It is easy to see that both unions are disjoint, because we are always taking the union of a set of ordered pairs that start with 0 with a set of ordered pairs that start with 1, and no ordered pair can start with both tags; it follows that $|L| = |A| + |B|$ and $|R| = |A \cup B| + |A \cap B|$. Now define the function $f : L \to R$ by the rule

$$\begin{aligned} f((0, x)) &= (0, x). \\ f((1, x)) &= (1, x) \text{if } x \in B \cap A. \\ f((1, x)) &= (0, x) \text{if } x \in B \setminus A. \end{aligned}$$

Observe that f is surjective, because for any $(0, x)$ in $\{0\} \times (A \cup B)$, either x is in A and $(0, x) = f((0, x))$ where $(0, x) \in L$, or x is in $B \setminus A$ and $(0, x) = f((1, x))$ where $(1, x) \in L$. It is also true that f is injective; the only way for it not to be is if $f((0, x)) = f((1, x)) = (0, x)$ for some x. Suppose this occurs. Then $x \in A$ (because of the 0 tag) and $x \in B \setminus A$ (because $(1, x)$ is only mapped to $(0, x)$ if $x \in B \setminus A$). But x can't be in both A and $B \setminus A$, so we get a contradiction.

11.1.5 Multiplication: the product rule

The **product rule** says that Cartesian product maps to arithmetic product. Intuitively, we line the elements (a, b) of $A \times B$ in lexicographic order and count them off. This looks very much like packing a two-dimensional array in a one-dimensional array by mapping each pair of indices (i, j) to $i \cdot |B| + j$.

Theorem 11.1.3. *For any finite sets A and B,*

$$|A \times B| = |A| \cdot |B| .$$

Proof. The trick is to order $A \times B$ lexicographically and then count off the elements. Given bijections $f : A \to [|A|]$ and $g : B \to [|B|]$, define $h : (A \times B) \to [|A| \cdot |B|]$ by the rule $h((a, b)) = a \cdot |B| + b$. The division algorithm recovers a and b from $h(a, b)$ by recovering the unique natural numbers q and r such that $h(a, b) = q \cdot |B| + r$ and $0 \leq b < |B|$ and letting $a = f^{-1}(q)$ and $b = f^{-1}(r)$. $\square$

The general form is

$$\left| \prod_{i=1}^{k} A_i \right| = \prod_{i=1}^{k} |A_i| ,$$

where the product on the left is a Cartesian product and the product on the right is an ordinary integer product.

11.1.5.1 Examples

- As I was going to Saint Ives, I met a man with seven sacks, and every sack had seven cats. How many cats total? Answer: Label the sacks $0, 1, 2, \dots, 6$, and label the cats in each sack $0, 1, 2, \dots, 6$. Then each cat can be specified uniquely by giving a pair $(\text{sack number}, \text{cat number})$, giving a bijection between the set of cats and the set 7×7. Since $|7 \times 7| = 7 \cdot 7 = 49$, we have 49 cats.

- Dr. Frankenstein's trusty assistant Igor has brought him 6 torsos, 4 brains, 8 pairs of matching arms, and 4 pairs of legs. How many different monsters can Dr Frankenstein build? Answer: there is a one-to-one correspondence between possible monsters and 4-tuples of the form $(\text{torso}, \text{brain}, \text{pair of arms}, \text{pair of legs})$; the set of such 4-tuples has $6 \cdot 4 \cdot 8 \cdot 4 = 728$ members.

- How many different ways can you order n items? Call this quantity $n!$ (pronounced "n **factorial**"). With 0 or 1 items, there is only one way; so we have $0! = 1! = 1$. For $n > 1$, there are n choices for the first item, leaving $n-1$ items to be ordered. From the product rule we thus have $n! = n \cdot (n-1)!$, which we could also expand out as $\prod_{i=1}^{n} i$.

11.1.5.2 For infinite sets

The product rule also works for infinite sets, because we again use it as a definition: for any A and B, $|A| \cdot |B|$ is defined to be $|A \times B|$. One oddity for infinite sets is that this definition gives $|A| \cdot |B| = |A| + |B| = \max(|A|, |B|)$, because if at least one of A and B is infinite, it is possible to construct a bijection between $A \times B$ and the larger of A and B. Infinite sets are strange.

11.1.6 Exponentiation: the exponent rule

Given sets A and B, let A^B be the set of functions $f : B \to A$. Then $\left|A^B\right| = |A|^{|B|}$.

If $|B|$ is finite, this is just a $|B|$-fold application of the product rule: we can write any function $f : B \to A$ as a sequence of length $|B|$ that gives the value in A for each input in B. Since each element of the sequence contributes $|A|$ possible choices, we get $|A|^{|B|}$ choices total.

For infinite sets, the exponent rule is a definition of $|A|^{|B|}$. Some simple facts are that $n^\alpha = 2^\alpha$ whenever n is finite and α is infinite (this comes down to the fact that we can represent any element of $[n]$ as a finite sequence of bits) and $\alpha^n = \alpha$ under the same conditions (follows by induction on n from $\alpha \cdot \alpha = \alpha$). When α and β are both infinite, many strange things can happen.

To give a flavor of how exponentiation works for arbitrary sets, here's a combinatorial proof of the usual arithmetic fact that $x^a x^b = x^{a+b}$, for any cardinal numbers x, a, and b. Let $x = |X|$ and let $a = |A|$ and $b = |B|$ where A and B are disjoint (we can always use the tagging trick that we used for inclusion-exclusion to make A and B be disjoint). Then $x^a x^b = \left|X^A \times X^B\right|$ and $x^{a+b} = \left|X^{A \cup B}\right|$. We will now construct an explicit bijection $f : X^{A \cup B} \to X^A \times X^B$. The input to f is a function $g : A \cup B \to X$; the output is a pair of functions $(g_A : A \to X, g_B : B \to X)$. We define g_A by $g_A(x) = g(x)$ for all x in A (this makes g_A the **restriction** of g to A, usually written as $g \upharpoonright A$ or $g|A$); similarly $g_B = g \upharpoonright B$. This is easily seen to be a bijection; if $g = h$, then $f(g) = (g \upharpoonright A, g \upharpoonright B) = f(h) = (h \upharpoonright A, h \upharpoonright B)$, and if $g \neq h$ there is some x for which $g(x) \neq h(x)$, implying $g \upharpoonright A \neq h \upharpoonright A$ (if x is in A) or $g \upharpoonright B \neq h \upharpoonright B$ (if x is in B).

11.1.6.1 Counting injections

Counting injections from a k-element set to an n-element set corresponds to counting the number of ways $P(n,k)$ we can pick an ordered subset of k of n items without replacement, also known as picking a k-**permutation**. (The k elements of the domain correspond to the k positions in the order.)

There are n ways to pick the first item, $n-1$ to pick the second, and so forth, giving a total of

$$P(n,k) = \prod_{i=n-k+1}^{n} i = \frac{n!}{(n-k)!}$$

such k-permutations by the product rule.

Among combinatorialists, the notation $(n)_k$ (pronounced "n **lower-factorial** k") is more common than $P(n,k)$ for $n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1)$. As an extreme case we have $(n)_n = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-n+1) = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1 = n!$, so $n!$ counts the number of **permutations** of n.

This gives us three tools for counting functions between sets: n^k counts the number of functions from a k-element set to an n-element set, $(n)_k$ counts the number of injections from a k-element set to an n-element set, and $n!$ counts the number of bijections between two n-element sets (or from an n-element set to itself).

Counting surjections is messier. If you really need to do this, you will need to use **Stirling numbers**; see [GKP94, Chapter 6] or [Sta97, p. 33].

11.1.7 Division: counting the same thing in two different ways

An old farm joke:

Q: How do you count a herd of cattle?

A: Count the legs and divide by four.

Sometimes we can compute the size of a set S by using it (as an unknown variable) to compute the size of another set T (as a function of $|S|$), and then using some other way to count T to find its size, finally solving for $|S|$. This is known as **counting two ways** and is surprisingly useful when it works. We will assume that all the sets we are dealing with are finite, so we can expect things like subtraction and division to work properly.

Example: Let S be an n-element set, and consider the set $S_k = \{A \subseteq S \mid |A| = k\}$. What is $|S_k|$? Answer: First we'll count the number m of sequences of k elements of S with no repetitions. We can get such a sequence in two ways:

1. By picking a size-k subset A and then choosing one of $k!$ ways to order the elements. This gives $m = |S_k| \cdot k!$.
2. By choosing the first element in one of n ways, the second in one of $n-1$, the third in one of $n-2$ ways, and so on until the k-th element, which can be chosen in one of $n-k+1$ ways. This gives $m = (n)_k = n \cdot (n-1) \cdot (n-2) \cdot \ldots (n-k+1)$, which can be written as $n!/(n-k)!$. (Here we are using the factors in $(n-k)!$ to cancel out the factors in $n!$ that we don't want.)

So we have $m = |S_k| \cdot k! = n!/(n-k)!$, from which we get

$$|S_k| = \frac{n!}{k! \cdot (n-k)!}.$$

This quantity turns out to be so useful that it has a special notation:

$$\binom{n}{k} \overset{\text{def}}{=} \frac{n!}{k! \cdot (n-k)!}.$$

where the left-hand side is known as a **binomial coefficient** and is pronounced "n choose k." We discuss binomial coefficients at length in §11.2.

The secret of why it's called a binomial coefficient will be revealed when we talk about generating functions in §11.3.

Example: Here's a generalization of binomial coefficients: let the **multinomial coefficient**

$$\binom{n}{n_1\ n_2\ \dots\ n_k}$$

be the number of different ways to distribute n items among k bins where the i-th bin gets exactly n_i of the items and we don't care what order the items appear in each bin. (Obviously this only makes sense if $n_1+n_2+\dots+n_k = n$.) Can we find a simple formula for the multinomial coefficient?

Here are two ways to count the number of permutations of the n-element set:

1. Pick the first element, then the second, etc. to get $n!$ permuations.
2. Generate a permutation in three steps:
   (a) Pick a partition of the n elements into groups of size $n_1, n_2, \dots n_k$.
   (b) Order the elements of each group.
   (c) Paste the groups together into a single ordered list.

There are

$$\binom{n}{n_1\ n_2\ \dots\ n_k}$$

ways to pick the partition and

$$n_1! \cdot n_2! \cdots n_k!$$

ways to order the elements of all the groups, so we have

$$n! = \binom{n}{n_1\ n_2\ \dots\ n_k} \cdot n_1! \cdot n_2! \cdots n_k!,$$

which we can solve to get

$$\binom{n}{n_1\ n_2\ \dots\ n_k} = \frac{n!}{n_1! \cdot n_2! \cdots n_k!}.$$

This also gives another way to derive the formula for a binomial coefficient, since

$$\binom{n}{k} = \binom{n}{k\ (n-k)} = \frac{n!}{k! \cdot (n-k)!}.$$

11.1.8 Applying the rules

If you're given some strange set to count, look at the structure of its description:

- If it's given by a rule of the form x is in S if either $P(x)$ or $Q(x)$ is true, use the sum rule (if P and Q are mutually exclusive) or inclusion-exclusion. This includes sets given by recursive definitions, e.g. x is a tree of depth at most k if it is either (a) a single leaf node (provided $k > 0$) or (b) a root node with two subtrees of depth at most $k-1$. The two classes are disjoint so we have $T(k) = 1 + T(k-1)^2$ with $T(0) = 0$.[4]
- For objects made out of many small components or resulting from many small decisions, try to reduce the description of the object to something previously known, e.g. (a) a word of length k of letters from an alphabet of size n allowing repetition (there are n^k of them, by the product rule); (b) a word of length k not allowing repetition (there are $(n)_k$ of them—or $n!$ if $n = k$); (c) a subset of k distinct things from a set of size n, where we don't care about the order (there are $\binom{n}{k}$ of them); any subset of a set of n things (there are 2^n of them—this is a special case of (a), where the alphabet encodes non-membership as 0 and membership as 1, and the position in the word specifies the element). Some examples:
  - The number of games of Tic-Tac-Toe assuming both players keep playing until the board is filled is obtained by observing that each such game can be specified by listing which of the 9 squares are filled in order, giving $9! = 362880$ distinct games. Note that we don't have to worry about which of the 9 moves are made by X and which by O, since the rules of the game enforce it. (If we only consider games that end when one player wins, this doesn't work: probably the easiest way to count such games is to send a computer off to generate all of them. This gives 255168 possible games and 958 distinct final positions.)
  - The number of completely-filled-in Tic-Tac-Toe boards can be obtained by observing that any such board has 5 X's and 4 O's. So there are $\binom{9}{5} = 126$ such positions. (Question: Why would this be smaller than the actual number of final positions?)

[4]Of course, just setting up a recurrence doesn't mean it's going to be easy to actually solve it.

Sometimes reducing to a previous case requires creativity. For example, suppose you win n identical cars on a game show and want to divide them among your k greedy relatives. Assuming that you don't care about fairness, how many ways are there to do this?

- If it's OK if some people don't get a car at all, then you can imagine putting n cars and $k-1$ dividers in a line, where relative 1 gets all the cars up to the first divider, relative 2 gets all the cars between the first and second dividers, and so forth up to relative k who gets all the cars after the $(k-1)$-th divider. Assume that each car—and each divider—takes one parking space. Then you have $n+k-1$ parking spaces with $k-1$ dividers in them (and cars in the rest). There are exactly $\binom{n+k-1}{k-1}$ ways to do this.
- Alternatively, suppose each relative demands at least 1 car. Then you can just hand out one car to each relative to start with, leaving $n-k$ cars to divide as in the previous case. There are $\binom{(n-k)+k-1}{k-1} = \binom{n-1}{k-1}$ ways to do this. [[[**direct encoding for** $\binom{n-1}{k-1}$]]]

Finding such correspondences is a central part of enumerative combinatorics, the branch of mathematics that deals with counting things.

11.1.9 An elaborate counting problem

Suppose you have the numbers $\{1, 2, \ldots, 2n\}$, and you want to count how many sequences of k of these numbers you can have that are (a) increasing ($a[i] < a[i+1]$ for all i), (b) decreasing ($a[i] \geq a[i+1]$ for all i), or (c) made up only of even numbers.

This is the union of three sets A, B, and C, corresponding to the three cases. The first step is to count each set individually; then we can start thinking about applying inclusion-exclusion to get the size of the union.

For A, any increasing sequence can be specified by choosing its elements (the order is determined by the assumption it is increasing). So we have $|A| = \binom{2n}{k}$.

For B, by symmetry we have $|B| = |A| = \binom{2n}{k}$.

For C, we are just looking at n^k possible sequences, since there are n even numbers we can put in each position.

Inclusion-exclusion says that $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cup B \cup C|$. It's not hard to see that $A \cap B = \emptyset$ when

k is at least 2,[5] so we can reduce this to $|A| + |B| + |C| - |A \cap C| - |B \cap C|$. To count $A \cap C$, observe that we are now looking at increasing sequences chosen from the n possible even numbers; so there are exactly $\binom{n}{k}$ of them, and similarly for $B \cap C$. Summing up gives a total of

$$\binom{2n}{k} + \binom{2n}{k} + n^k - \binom{n}{k} - \binom{n}{k} = 2\left(\binom{2n}{k} - \binom{n}{k}\right) + n^k$$

sequences satisfying at least one of the criteria.

Note that we had to assume $k = 2$ to get $A \cap B = \emptyset$, so this formula might require some adjustment for $k < 2$. In fact we can observe immediately that the unique empty sequence for $k = 1$ fits in all of A, B, and C, so in this case we get 1 winning sequence (which happens to be equal to the value in the formula, because here $A \cap B = \emptyset$ for other reasons), and for $k = 1$ we get $2n$ winning sequences (which is less than the value $3n$ given by the formula).

To test that the formula works for at least some larger values, let $n = 3$ and $k = 2$. Then the formula predicts $2\left(\binom{6}{2} - \binom{3}{2}\right) + 3^2 = 2(15-3)+9 = 33$

[5]It's even easier to assume that $A \cap B = \emptyset$ always, but for $k = 1$ any sequence is both increasing and nonincreasing, since there are no pairs of adjacent elements in a 1-element sequence to violate the property.

total sequences.[6] And here they are:

$$\begin{array}{c}(1,2)\\(1,3)\\(1,4)\\(1,5)\\(1,6)\\(2,1)\\(2,2)\\(2,3)\\(2,4)\\(2,5)\\(2,6)\\(3,1)\\(3,2)\\(3,4)\\(3,5)\\(3,6)\\(4,1)\\(4,2)\\(4,3)\\(4,4)\\(4,5)\\(4,6)\\(5,1)\\(5,2)\\(5,3)\\(5,4)\\(5,6)\\(6,1)\\(6,2)\\(6,3)\\(6,4)\\(6,5)\\(6,6)\end{array}$$

[6]Without looking at the list, can you say which 3 of the $6^2 = 36$ possible length-2 sequences are missing?

11.1.10 Further reading

Rosen [Ros12] does basic counting in Chapter 6 and more advanced counting (including solving recurrences and using generating functions) in chapter 8. Biggs [Big02] gives a basic introduction to counting in Chapters 6 and 10, with more esoteric topics in Chapters 11 and 12. Graham *et al.* [GKP94] have quite a bit on counting various things.

Combinatorics largely focuses on counting rather than efficient algorithms for constructing particular combinatorial objects. The book *Constructive Combinatorics*, by Denisses Stanton and White, [SW86] remedies this omission, and includes algorithms not only for enumerating all instances of various classes of combinatorial objects but also for finding the i-th such instance in an appropriate ordering without having to generate all previous instances (**unranking**) and the inverse operation of finding the position of a particular object in an appropriate ordering (**ranking**).

11.2 Binomial coefficients

The **binomial coefficient** "n choose k", written

$$\binom{n}{k} = \frac{(n)_k}{k!} = \frac{n!}{k! \cdot (n-k)!},$$

counts the number of k-element subsets of an n-element set.

The name arises from the **binomial theorem**, which in the following form was first proved by Isaac Newton:

Theorem 11.2.1 (Binomial theorem)**.** *For any $n \in \mathbb{R}$,*

$$(x+y)^n = \sum_{k=0}^{\infty} \binom{n}{k} x^k y^{n-k}, \tag{11.2.1}$$

provided the sum converges.

A sufficient condition for the sum converging is $|x/y| < 1$. For the general version of the theorem, $\binom{n}{k}$ is defined as $(n)_k / k!$, which works even if n is not a non-negative integer. The usual proof requires calculus.

In the common case when n is a non-negative integer, we can limit ourselves to letting k range from 0 to n. The reason is that $\binom{n}{k} = 0$ when n is a non-negative integer and $k > n$. This gives the more familiar version

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}. \tag{11.2.2}$$

The connection between (11.2.2) and counting subsets is straightforward: expanding $(x+y)^n$ using the distributive law gives 2^n terms, each of which is a unique sequence of n x's and y's. If we think of the x's in each term as labeling a subset of the n positions in the term, the terms that get added together to get $x^k y^{n-k}$ correspond one-to-one to subsets of size k. So there are $\binom{n}{k}$ such terms, accounting for the coefficient on the right-hand side.

11.2.1 Recursive definition

If we don't like computing factorials, we can also compute binomial coefficients recursively. This may actually be less efficient for large n (we need to do $\Theta(n^2)$ additions instead of $\Theta(n)$ multiplications and divisions), but the recurrence gives some insight into the structure of binomial coefficients.

Base cases:

- If $k = 0$, then there is exactly one zero-element set of our n-element set—it's the empty set—and we have $\binom{n}{0} = 1$.
- If $k > n$, then there are no k-element subsets, and we have $\forall k > n : \binom{n}{k} = 0$.

Recursive step: We'll use **Pascal's identity**, which says that

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

The easiest proof of this identity is combinatorial, which means that we will construct an explicit bijection between a set counted by the left-hand side and a set counted by the right-hand side. This is often one of the best ways of understanding simple binomial coefficient identities.

On the left-hand side, we are counting all the k-element subsets of an n-element set S. On the right hand side, we are counting two different collections of sets: the $(k-1)$-element and k-element subsets of an $(n-1)$-element set. The trick is to recognize that we get an $(n-1)$-element set S' from our original set by removing one of the elements x. When we do this, we affect the subsets in one of two ways:

1. If the subset doesn't contain x, it doesn't change. So there is a one-to-one correspondence (the identity function) between k-subsets of S that don't contain x and k-subsets of S'. This bijection accounts for the first term on the right-hand side.

2. If the subset does contain x, then we get a $(k-1)$-element subset of S' when we remove it. Since we can go back the other way by reinserting x, we get a bijection between k-subsets of S that contain x and $(k-1)$-subsets of S'. This bijection accounts for the second term on the right-hand side.

Adding the two cases together (using the sum rule), we conclude that the identity holds.

Using the base case and Pascal's identity, we can construct **Pascal's triangle**, a table of values of binomial coefficients:

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1
…					

Each row corresponds to increasing values of n, and each column to increasing values of k, with $\binom{0}{0}$ in the upper left-hand corner. To compute each entry, we add together the entry directly above it and the entry diagonally above and to the left.

11.2.1.1 Pascal's identity: algebraic proof

Using the binomial theorem plus a little bit of algebra, we can prove Pascal's identity without using a combinatorial argument (this is not necessarily an improvement). The additional fact we need is that if we have two equal series

$$\sum_{k=0}^{\infty} a_k x^k = \sum_{k=0}^{\infty} b_k x^k$$

then $a_i = b_i$ for all i.

Here's the proof of Pascal's identity:

$$\begin{aligned}
\sum_{k=0}^{n} \binom{n}{k} x^k &= (1+x)^n \\
&= (1+x)(1+x)^{n-1} \\
&= (1+x)^{n-1} + x(1+x)^{n-1} \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} x^k + x \sum_{k=0}^{n-1} \binom{n-1}{k} x^k \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} x^k + \sum_{k=0}^{n-1} \binom{n-1}{k} x^{k+1} \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} x^k + \sum_{k=1}^{n} \binom{n-1}{k-1} x^k \\
&= \sum_{k=0}^{n} \binom{n-1}{k} x^k + \sum_{k=0}^{n} \binom{n-1}{k-1} x^k \\
&= \sum_{k=0}^{n} \binom{n-1}{k} + \binom{n-1}{k-1} x^k.
\end{aligned}$$

and now we equate matching coefficients to get

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

as advertised.

11.2.2 Vandermonde's identity

Vandermonde's identity says that, provided r does not exceed m or n,

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{r-k} \binom{n}{k}.$$

11.2.2.1 Combinatorial proof

To pick r elements of an $m+n$ element set, we have to pick some of them from the first m elements and some from the second n elements. Suppose we choose k elements from the last n; there are $\binom{n}{k}$ different ways to do this, and $\binom{m}{r-k}$ different ways to choose the remaining $r-k$ from the first m. This gives (by the product rule) $\binom{m}{r-k}\binom{n}{k}$ ways to choose r elements from the whole set if we limit ourselves to choosing exactly k from the last n. The identity follow by summing over all possible values of k.

11.2.2.2 Algebraic proof

Here we use the fact that, for any sequences of coefficients $\{a_i\}$ and $\{b_i\}$,

$$\left(\sum_{i=0}^{n} a_i x^i\right)\left(\sum_{i=0}^{m} b_i x^i\right) = \sum_{i=0}^{m+n}\left(\sum_{j=0}^{i} a_j b_{i-j}\right) x^i.$$

So now consider

$$\begin{aligned}\sum_{r=0}^{m+n}\binom{m+n}{r}x^r &= (1+x)^{m+n}\\ &= (1+x)^n(1+x)^m\\ &= \left(\sum_{i=0}^{n}\binom{n}{i}x^i\right)\left(\sum_{j=0}^{m}\binom{m}{j}x^j\right)\\ &= \sum_{r=0}^{m+n}\left(\sum_{k=0}^{r}\binom{n}{k}\binom{m}{r-k}\right)x^r.\end{aligned}$$

and equate terms with matching exponents.

Is this more enlightening than the combinatorial version? It depends on what kind of enlightenment you are looking for. In this case the combinatorial and algebraic arguments are counting essentially the same things in the same way, so it's not clear what if any advantage either has over the other. But in many cases it's easier to construct an algebraic argument than a combinatorial one, in the same way that it's easier to do arithmetic using standard grade-school algorithms than by constructing explicit bijections. On the other hand, a combinatorial argument may let you carry other things you know about some structure besides just its size across the bijection, giving you more insight into the things you are counting. The best course is probably to have both techniques in your toolbox.

11.2.3 Sums of binomial coefficients

What is the sum of all binomial coefficients for a given n? We can show

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n$$

combinatorially, by observing that adding up all subsets of an n-element set of all sizes is the same as counting all subsets. Alternatively, apply the binomial theorem to $(1+1)^n$.

Here's another sum, with alternating sign. This is useful if you want to know how the even-k binomial coefficients compare to the odd-k binomial coefficients.

$$\sum_{k=0}^{n}(-1)^k\binom{n}{k} = 0.\text{(Assuming } n \neq 0.\text{)}$$

Proof: $(1-1)^n = 0^n = 0$ when n is nonzero. (When n *is* zero, the 0^n part still works, since $0^0 = 1 = \binom{0}{0}(-1)^0$.)

By now it should be obvious that

$$\sum_{k=0}^{n} 2^k \binom{n}{k} = 3^n.$$

It's not hard to construct more examples of this phenomenon.

11.2.4 The general inclusion-exclusion formula

We've previously seen that $|A \cup B| = |A| + |B| - |A \cap B|$. The generalization of this fact from two to many sets is called the **inclusion-exclusion** formula and says:

Theorem 11.2.2.

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{S\subseteq\{1\dots n\}, S\neq\emptyset} (-1)^{|S|+1}\left|\bigcap_{j\in S} A_j\right|. \tag{11.2.3}$$

This rather horrible expression means that to count the elements in the union of n sets A_1 through A_n, we start by adding up all the individual sets $|A_1| + |A_2| + \dots |A_n|$, then subtract off the overcount from elements that appear in two sets $-|A_1 \cap A_2| - |A_1 \cap A_3| - \dots$, then add back the resulting undercount from elements that appear in three sets, and so on.

Why does this work? Consider a single element x that appears in k of the sets. We'll count it as $+1$ in $\binom{k}{1}$ individual sets, as -1 in $\binom{k}{2}$ pairs, $+1$ in $\binom{k}{3}$ triples, and so on, adding up to

$$\sum_{i=1}^{k}(-1)^{k+1}\binom{k}{i} = -\left(\sum_{i=1}^{k}(-1)^k\binom{k}{i}\right) = -\left(\sum_{i=0}^{k}(-1)^k\binom{k}{i} - 1\right) = -(0-1) = 1.$$

11.2.5 Negative binomial coefficients

Though it doesn't make sense to talk about the number of k-subsets of a (-1)-element set, the binomial coefficient $\binom{n}{k}$ has a meaningful value for negative n, which works in the binomial theorem. We'll use the lower-factorial version of the definition:

$$\binom{-n}{k} = (-n)_k/k! = \left(\prod_{i=-n-k+1}^{-n} i\right)/k!.$$

Note we still demand that $k \in \mathbb{N}$; we are only allowed to do funny things with the upper index n.

So for example:

$$\binom{-1}{k} = (-1)_k/k! = \left(\prod_{i=-1-k+1}^{-1} i\right)/k! = \left(\prod_{i=-k}^{-1} i\right)/\left(\prod_{i=1}^{k} i\right) = (-1)^k.$$

An application of this fact is that

$$\frac{1}{1-z} = (1-z)^{-1} = \sum_{n=0}^{\infty}\binom{-1}{n}1^{-1-n}(-z)^n = \sum_{n=0}^{\infty}(-1)^n(-z)^n = \sum_{n=0}^{\infty} z^n.$$

In computing this sum, we had to be careful which of 1 and $-z$ got the n exponent and which got $-1-n$. If we do it the other way, we get

$$\frac{1}{1-z} = (1-z)^{-1} = \sum_{n=0}^{\infty}\binom{-1}{n}1^n(-z)^{-1-n} = -\frac{1}{z}\sum_{n=0}^{\infty}\frac{1}{z^n}$$

This turns out to actually be correct, since applying the geometric series formula turns the last line into

$$-\frac{1}{z}\cdot\frac{1}{1-1/z} = -\frac{1}{z-1} = \frac{1}{1-z},$$

but it's a lot less useful.

What happens for a larger upper index? One way to think about $(-n)_k$ is that we are really computing $(n+k-1)_k$ and then negating all the factors (which corresponds to multiplying the whole expression by $(-1)^k$. So this gives us the identity

$$\binom{-n}{k} = (-n)_k/k! = (-1)^k(n+k-1)_k/k! = (-1)^k\binom{n+k-1}{k}.$$

So, for example,

$$\frac{1}{(1-z)^2} = (1-z)^{-2} = \sum_n \binom{-2}{n} 1^{-2-n}(-z)^n = \sum_n (-1)^n \binom{n+1}{n} (-z)^n = \sum_n (n+1) z^n.$$

These facts will be useful when we look at generating functions in §11.3.

11.2.6 Fractional binomial coefficients

Yes, we can do fractional binomial coefficients, too. Exercise: Find the value of

$$\binom{1/2}{n} = \frac{(1/2)_n}{n!}.$$

Like negative binomial coefficients, these don't have an obvious combinatorial interpretation, but can be handy for computing power series of fractional binomial powers like $\sqrt{1+z} = (1+z)^{1/2}$.

11.2.7 Further reading

Graham *et al.* [GKP94] §5.1–5.3 is an excellent source for information about all sorts of facts about binomial coefficients.

11.3 Generating functions

11.3.1 Basics

The short version: A generating function represents objects of weight n with z^n, and adds all the objects you have up to get a sum $a_0z^0 + a_1z^1 + a_2z^2 + \dots$, where each a_n counts the number of different objects of weight n. If you are very lucky (or constructed your set of objects by combining simpler sets of objects in certain straightforward ways) there will be some compact expression that is expands to this horrible sum but is easier to right down. Such compact expressions are called **generating functions**, and manipulating them algebraically gives an alternative to actually knowing how to count (Chapter 11).

11.3.1.1 A simple example

We are given some initial prefixes for words: `qu`, `s`, and `t`; some vowels to put in the middle: `a`, `i`, and `oi`; and some suffixes: `d`, `ff`, and `ck`, and we want to calculate the number of words we can build of each length.

One way is to generate all 27 words[7] and sort them by length:

```
sad sid tad tid
quad quid sack saff sick siff soid tack taff tick tiff toid
quack quaff quick quiff quoid soick soiff toick toiff
quoick quoiff
```

This gives us 4 length-3 words, 12 length-4 words, 9 length-5 words, and 2 length-6 words. This is probably best done using a computer, and becomes expensive if we start looking at much larger lists.

An alternative is to solve the problem by judicious use of algebra. Pretend that each of our letters is actually a variable, and that when we concatenate `qu`, `oi`, and `ck` to make `quoick`, we are really multiplying the variables using our usual notation. Then we can express all 27 words as the product (`qu`+`s`+`t`)(`a`+`i`+`oi`)(`d`+`ff`+`ck`). But we don't care about the exact set of words, we just want to know how many we get of each length.

So now we do the magic trick: we replace every variable we've got with a single variable z. For example, this turns `quoick` into $zzzzzz = z^6$, so we can still find the length of a word by reading off the exponent on z. But we can also do this before we multiply everything out, getting

$$\begin{aligned}(zz+z+z)(z+z+zz)(z+zz+zz) &= (2z+z^2)(2z+z^2)(z+2z^2)\\ &= z^3(2+z)^2(1+2z)\\ &= z^3(4+4z+z^2)(1+2z)\\ &= z^3(4+12z+9z^2+2z^3)\\ &= 4z^3+12z^4+9z^5+2z^6.\end{aligned}$$

We can now read off the number of words of each length directly off the coefficients of this polynomial.

11.3.1.2 Why this works

In general, what we do is replace any object of **weight** 1 with z. If we have an object with weight n, we think of it as n weight-1 objects stuck together, i.e., z^n. Disjoint unions are done using addition as in simple counting: $z+z^2$ represents the choice between a weight-1 object and a weight-2 object (which might have been built out of 2 weight-1 objects), while $12z^4$ represents a

[7]We are using *word* in the combinatorial sense of a finite sequence of letters (possibly even the empty sequence) and not the usual sense of a finite, nonempty sequence of letters that actually make sense.

choice between 12 different weight-4 objects. The trick is that when we multiply two expressions like this, whenever two values z^k and z^l collide, the exponents add to give a new value z^{k+l} representing a new object with total weight $k + l$, and if we have something more complex like $(nz^k)(mz^l)$, then the coefficients multiply to give $(nm)z^{k+l}$ different weight $(k + l)$ objects.

For example, suppose we want to count the number of robots we can build given 5 choices of heads, each of weight 2, and 6 choices of bodies, each of weight 5. We represent the heads by $5z^2$ and the bodies by $6z^5$. When we multiply these expressions together, the coefficients multiply (which we want, by the product rule) and the exponents add: we get $5z^2 \cdot 6z^5 = 30z^7$ or 30 robots of weight 7 each.

The real power comes in when we consider objects of different weights. If we add to our 5 weight-2 robot heads two extra-fancy heads of weight 3, and compensate on the body side with three new lightweight weight-4 bodies, our new expression is $(5z^2 + 2z^3)(3z^4 + 6z^5) = 15z^6 + 36z^7 + 12z^8$, giving a possible 15 weight-6 robots, 36 weight-7 robots, and 12 weight-8 robots. The rules for multiplying polynomials automatically tally up all the different cases for us.

This trick even works for infinitely-long polynomials that represent infinite series (such "polynomials" are called formal power series). Even though there might be infinitely many ways to pick three natural numbers, there are only finitely many ways to pick three natural numbers whose sum is 37. By computing an appropriate formal power series and extracting the coefficient from the z^{37} term, we can figure out exactly how many ways there are. This works best, of course, when we don't have to haul around an entire infinite series, but can instead represent it by some more compact function whose expansion gives the desired series. Such a function is called a **generating function**, and manipulating generating functions can be a powerful alternative to creativity in making combinatorial arguments.

11.3.1.3 Formal definition

Given a sequence $a_0, a_1, a_2, \ldots$, its **generating function** $F(z)$ is given by the sum

$$F(z) = \sum_{i=0}^{\infty} a_i z^i.$$

A sum in this form is called a **formal power series**. It is "formal" in the sense that we don't necessarily plan to actually compute the sum, and are instead using the string of z^i terms as a long rack to store coefficients on.

In some cases, the sum has a more compact representation. For example, we have

$$\frac{1}{1-z} = \sum_{i=0}^{\infty} z^i,$$

so $1/(1-z)$ is the generating function for the sequence $1, 1, 1, \dots$. This may let us manipulate this sequence conveniently by manipulating the generating function.

Here's a simple case. If $F(z)$ generates some sequence a_i, what does sequence b_i does $F(2z)$ generate? The i-th term in the expansion of $F(2z)$ will be $a_i(2z)^i = a_i 2^i z^i$, so we have $b_i = 2^i a_i$. This means that the sequence $1, 2, 4, 8, 16, \dots$ has generating function $1/(1-2z)$. In general, if $F(z)$ represents a_i, then $F(cz)$ represents $c^i a_i$.

What else can we do to F? One useful operation is to take its derivative with respect to z. We then have

$$\frac{d}{dz}F(z) = \sum_{i=0}^{\infty} a_i \frac{d}{dz} z^i = \sum_{i=0}^{\infty} a_i i z^{i-1}.$$

This *almost* gets us the representation for the series ia_i, but the exponents on the z's are off by one. But that's easily fixed:

$$z\frac{d}{dz}F(z) = z\sum_{i=0}^{\infty} a_i i z^{i-1} = \sum_{i=0}^{\infty} a_i i z^i.$$

So the sequence $0, 1, 2, 3, 4, \dots$ has generating function

$$z\frac{d}{dz}\frac{1}{1-z} = \frac{z}{(1-z)^2},$$

and the sequence of squares $0, 1, 4, 9, 16, \dots$ has generating function

$$z\frac{d}{dz}\frac{z}{(1-z)^2} = \frac{z}{(1-z)^2} + \frac{2z^2}{(1-z)^3}.$$

As you can see, some generating functions are prettier than others.

(We can also use integration to divide each term by i, but the details are messier.)

Another way to get the sequence $0, 1, 2, 3, 4, \dots$ is to observe that it satisfies the recurrence:

- $a_0 = 0$.

- $a_{n+1} = a_n + 1 (\forall n \in \mathbb{N})$.

A standard trick in this case is to multiply each of the $\forall i$ bits by z^n, sum over all n, and see what happens. This gives $\sum a_{n+1}z^n = \sum a_n z^n + \sum z^n = \sum a_n z^n + 1/(1-z)$. The first term on the right-hand side is the generating function for a_n, which we can call $F(z)$ so we don't have to keep writing it out. The second term is just the generating function for $1, 1, 1, 1, 1, \ldots$. But what about the left-hand side? This is almost the same as $F(z)$, except the coefficients don't match up with the exponents. We can fix this by dividing $F(z)$ by z, after carefully subtracting off the a_0 term:

$$\begin{aligned}
(F(z) - a_0)/z &= \left(\sum_{n=0}^{\infty} a_n z^n - a_0\right)/z \\
&= \left(\sum_{n=1}^{\infty} a_n z^n\right)/z \\
&= \sum_{n=1}^{\infty} a_n z^{n-1} \\
&= \sum_{n=0}^{\infty} a_{n+1} z^n.
\end{aligned}$$

So this gives the equation $(F(z)-a_0)/z = F(z)+1/(1-z)$. Since $a_0 = 0$, we can rewrite this as $F(z)/z = F(z) + 1/(1-z)$. A little bit of algebra turns this into $F(z) - zF(z) = z/(1-z) or F(z) = z/(1-z)^2$.

Yet another way to get this sequence is construct a collection of objects with a simple structure such that there are exactly n objects with weight n. One way to do this is to consider strings of the form a^+b^* where we have at least one a followed by zero or more b's. This gives n strings of length n, because we get one string for each of the 1 through n a's we can put in (an example would be *abb*, *aab*, and *aaa* for $n = 3$). We can compute the generating function for this set because to generate each string we must pick in order:

- One initial a. Generating function $= z$.
- Zero or more a's. Generating function $= 1/(1-z)$.
- Zero or more b's. Generating function $= 1/(1-z)$.

Taking the product of these gives $z/(1-z)^2$, as before.

This trick is useful in general; if you are given a generating function $F(z)$ for a_n, but want a generating function for $b_n = \sum_{k\leq n} a_k$, allow yourself to pad each weight-k object out to weight n in exactly one way using $n-k$ junk objects, i.e. multiply $F(z)$ by $1/(1-z)$.

11.3.2 Some standard generating functions

Here is a table of some of the most useful generating functions.

$$\begin{aligned}
\frac{1}{1-z} &= \sum_{i=0}^{\infty} z^i \\
\frac{z}{(1-z)^2} &= \sum_{i=0}^{\infty} iz^i \\
(1+z)^n &= \sum_{i=0}^{\infty} \binom{n}{i} z^i = \sum_{i=0}^{n} \binom{n}{i} z^i \\
\frac{1}{(1-z)^n} &= \sum_{i=0}^{\infty} \binom{n+i-1}{i} z^i
\end{aligned}$$

Of these, the first is the most useful to remember (it's also handy for remembering how to sum geometric series). All of these equations can be proven using the binomial theorem.

11.3.3 More operations on formal power series and generating functions

Let $F(z) = \sum_i a_i z^i$ and $G(z) = \sum_i b_i z^i$. Then their sum $F(z) + G(z) = \sum_i (a_i + b_i) z^i$ is the generating function for the sequence $(a_i + b_i)$. What is their product $F(z)G(z)$?

To compute the i-th term of $F(z)G(z)$, we have to sum over all pairs of terms, one from F and one from G, that produce a z^i factor. Such pairs of terms are precisely those that have exponents that sum to i. So we have

$$F(z)G(z) = \sum_{i=0}^{\infty} \left(\sum_{j=0}^{i} a_j b_{j-i} \right) z^i.$$

As we've seen, this equation has a natural combinatorial interpretation. If we interpret the coefficient a_i on the i-th term of $F(z)$ as counting the number of "a-things" of weight i, and the coefficient b_i as the number of "b-things" of weight i, then the i-th coefficient of $F(z)G(z)$ counts the number of ways to make a combined thing of total weight i by gluing together an a-thing and a b-thing.

As a special case, if $F(z) = G(z)$, then the i-th coefficient of $F(z)G(z) = F^2(z)$ counts how many ways to make a thing of total weight i using two "a-things", and $F^n(z)$ counts how many ways (for each i) to make a thing of total weight i using n "a-things". This gives us an easy combinatorial proof of a special case of the binomial theorem:

$$(1+x)^n = \sum_{i=0}^{\infty} \binom{n}{i} x^i.$$

Think of the left-hand side as the generating function $F(x) = 1 + x$ raised to the n-th power. The function F by itself says that you have a choice between one weight-0 object or one weight-1 object. On the right-hand side the i-th coefficient counts how many ways you can put together a total of i weight-1 objects given n to choose from—so it's $\binom{n}{i}$.

11.3.4 Counting with generating functions

The product formula above suggests that generating functions can be used to count combinatorial objects that are built up out of other objects, where our goal is to count the number of objects of each possible non-negative integer "weight" (we put "weight" in scare quotes because we can make the "weight" be any property of the object we like, as long as it's a non-negative integer—a typical choice might be the size of a set, as in the binomial theorem example above). There are five basic operations involved in this process; we've seen two of them already, but will restate them here with the others.

Throughout this section, we assume that $F(z)$ is the generating function counting objects in some set A and $G(z)$ the generating function counting objects in some set B.

11.3.4.1 Disjoint union

Suppose $C = A \cup B$ and A and B are disjoint. Then the generating function for objects in C is $F(z) + G(z)$.

Example: Suppose that A is the set of all strings of zero or more letters x, where the weight of a string is just its length. Then $F(z) = 1/(1-z)$, since there is exactly one string of each length and the coefficient a_i on each z^i is always 1. Suppose that B is the set of all strings of zero or more letters y and/or z, so that $G(z) = 1/(1-2z)$ (since there are now 2^i choices of length-i strings). The set C of strings that are either (a) all x's or (b) made up of y's, z's, or both, has generating function $F(z) + G(z) = 1/(1-z) + 1/(1-2z)$.

11.3.4.2 Cartesian product

Now let $C = A \times B$, and let the weight of a pair $(a, b) \in C$ be the sum of the weights of a and b. Then the generating function for objects in C is $F(z)G(z)$.

Example: Let A be all-x strings and B be all-y or all-z strings, as in the previous example. Let C be the set of all strings that consist of zero or more x's followed by zero or more y's and/or z's. Then the generating function for C is $F(z)G(z) = \frac{1}{(1-z)(1-2z)}$.

11.3.4.3 Repetition

Now let C consists of all finite sequences of objects in A, with the weight of each sequence equal to the sum of the weights of its elements (0 for an empty sequence). Let $H(z)$ be the generating function for C. From the preceding rules we have

$$H = 1 + F + F^2 + F^3 + \cdots = \frac{1}{1-F}.$$

This works best when $H(0) = 0$; otherwise we get infinitely many weight-0 sequences. It's also worth noting that this is just a special case of substitution (see below), where our "outer" generating function is $1/(1-z)$.

Example: $(0|11)*$ Let $A = \{0, 11\}$, and let C be the set of all sequences of zeros and ones where ones occur only in even-length runs. Then the generating function for A is $z + z^2$ and the generating function for C is $1/(1-z-z^2)$. We can extract exact coefficients from this generating function using the techniques below.

Example: sequences of positive integers Suppose we want to know how many different ways there are to generate a particular integer as a sum of positive integers. For example, we can express 4 as 4, $3+1$, $2+2$, $2+1+1$, $1+1+1+1$, $1+1+2$, $1+2+1$, or $1+3$, giving 8 different ways.

We can solve this problem using the repetition rule. Let $F = z/(1-z)$

generate all the positive integers. Then

$$\begin{aligned}
H &= \frac{1}{1-F} \\
&= \frac{1}{1-\frac{z}{1-z}} \\
&= \frac{1-z}{(1-z)-z} \\
&= \frac{1-z}{1-2z}.
\end{aligned}$$

We can get exact coefficients by observing that

$$\begin{aligned}
\frac{1-z}{1-2z} &= \frac{1}{1-2z} - \frac{z}{1-2z} \\
&= \sum_{n=0}^{\infty} 2^n z^n - \sum_{n=0}^{\infty} 2^n z^{n+1} \\
&= \sum_{n=0}^{\infty} 2^n z^n - \sum_{n=1}^{\infty} 2^{n-1} z^n \\
&= 1 + \sum_{n=1}^{\infty} (2^n - 2^{n-1}) z^n \\
&= 1 + \sum_{n=1}^{\infty} 2^{n-1} z^n.
\end{aligned}$$

This means that there is 1 way to express 0 (the empty sum), and 2^{n-1} ways to express any larger value n (e.g. $2^{4-1} = 8$ ways to express 4).

Once we know what the right answer is, it's not terribly hard to come up with a combinatorial explanation. The quantity 2^{n-1} counts the number of subsets of an $(n-1)$-element set. So imagine that we have $n-1$ places and we mark some subset of them, plus add an extra mark at the end; this might give us a pattern like `XX-X`. Now for each sequence of places ending with a mark we replace it with the number of places (e.g. `XX-X` $= 1, 1, 2$, `X-X-X--X` $= 1, 3, 2, 4$). Then the sum of the numbers we get is equal to n, because it's just counting the total length of the sequence by dividing it up at the marks and the adding the pieces back together. The value 0 doesn't fit this pattern (we can't put in the extra mark without getting a sequence of length 1), so we have 0 as a special case again.

If we are very clever, we might come up with this combinatorial explanation from the beginning. But the generating function approach saves us from having to be clever.

11.3.4.4 Pointing

This operation is a little tricky to describe. Suppose that we can think of each weight-k object in A as consisting of k items, and that we want to count not only how many weight-k objects there are, but how many ways we can produce a weight-k object where one of its k items has a special mark on it. Since there are k different items to choose for each weight-k object, we are effectively multiplying the count of weight-k objects by k. In generating function terms, we have

$$H(z) = z\frac{d}{dz}F(z).$$

Repeating this operation allows us to mark more items (with some items possibly getting more than one mark). If we want to mark n distinct items in each object (with distinguishable marks), we can compute

$$H(z) = z^n \frac{d^n}{dz^n}F(z),$$

where the repeated derivative turns each term $a_i z^i$ into $a_i i(i-1)(i-2)\dots(i-n+1)z^{i-n}$ and the z^n factor fixes up the exponents. To make the marks indistinguishable (i.e., we don't care what order the values are marked in), divide by $n!$ to turn the extra factor into $\binom{i}{n}$.

(If you are not sure how to take a derivative, look at §F.2.)

Example: Count the number of finite sequences of zeros and ones where exactly two digits are underlined. The generating function for $\{0,1\}$ is $2z$, so the generating function for sequences of zeros and ones is $F = 1/(1-2z)$ by the repetition rule. To mark two digits with indistinguishable marks, we need to compute

$$\frac{1}{2}z^2\frac{d^2}{dz^2}\frac{1}{1-2z} = \frac{1}{2}z^2\frac{d}{dz}\frac{2}{(1-2z)^2} = \frac{1}{2}z^2\frac{8}{(1-2z)^3} = \frac{4z^2}{(1-2z)^3}.$$

11.3.4.5 Substitution

Suppose that the way to make a C-thing is to take a weight-k A-thing and attach to each its k items a B-thing, where the weight of the new C-thing is the sum of the weights of the B-things. Then the generating function for C is the composition $F(G(z))$.

Why this works: Suppose we just want to compute the number of C-things of each weight that are made from some single specific weight-k A-thing. Then the generating function for this quantity is just $(G(z))^k$. If we expand our horizons to include all a_k weight-k A-things, we have to multiply by a_k to get $a_k(G(z))^k$. If we further expand our horizons to include A-things of all different weights, we have to sum over all k:

$$\sum_{k=0}^{\infty} a_k(G(z))^k.$$

But this is just what we get if we start with $F(z)$ and substitute $G(z)$ for each occurrence of z, i.e. if we compute $F(G(z))$.

Example: bit-strings with primes Suppose we let A be all sequences of zeros and ones, with generating function $F(z) = 1/(1-2z)$. Now suppose we can attach a single or double prime to each 0 or 1, giving $0'$ or $0''$ or $1'$ or $1''$, and we want a generating function for the number of distinct primed bit-strings with n attached primes. The set $\{', ''\}$ has generating function $G(z) = z+z^2$, so the composite set has generating function $F(z) = 1/(1-2(z+z^2)) = 1/(1-2z-2z^2)$.

Example: (0|11)* again The previous example is a bit contrived. Here's one that's a little more practical, although it involves a brief digression into **multivariate generating functions**. A multivariate generating function $F(x, y)$ generates a series $\sum_{ij} a_{ij}x^iy^j$, where a_{ij} counts the number of things that have i x's and j y's. (There is also the obvious generalization to more than two variables). Consider the multivariate generating function for the set $\{0, 1\}$, where x counts zeros and y counts ones: this is just $x + y$. The multivariate generating function for sequences of zeros and ones is $1/(1-x-y)$ by the repetition rule. Now suppose that each 0 is left intact but each 1 is replaced by 11, and we want to count the total number of strings by length, using z as our series variable. So we substitute z for x and z^2 for y (since each y turns into a string of length 2), giving $1/(1-z-z^2)$. This gives another way to get the generating function for strings built by repeating 0 and 11.

11.3.5 Generating functions and recurrences

What makes generating functions particularly useful for algorithm analysis is that they directly solve recurrences of the form $T(n) = aT(n-1) + bT(n-2) + f(n)$ (or similar recurrences with more T terms on the right-hand side), provided we have a generating function $F(z)$ for $f(n)$. The idea is that there exists some generating function $G(z)$ that describes the entire sequence of values $T(0), T(1), T(2), \ldots$, and we just need to solve for it by restating the recurrence as an equation about G. The left-hand side will just turn into G. For the right-hand side, we need to shift $T(n-1)$ and $T(n-2)$ to line up right, so that the right-hand side will correctly represent the sequence $T(0), T(1), aT(0) + aT(1) + F(2)$, etc. It's not hard to see that the generating function for the sequence $0, T(0), T(1), T(2), \ldots$ (corresponding to the $T(n-1)$ term) is just $zG(z)$, and similarly the sequence $0, 0, T(1), T(2), T(3), \ldots$ (corresponding to the $T(n-2)$ term) is $z^2G(z)$. So we have (being very careful to subtract out extraneous terms at for $i = 0$ and $i = 1$):

$$G = az(G - T(0)) + bz^2G + (F - f(0) - zf(1)) + T(0) + zT(1),$$

and after expanding F we can in principle solve this for G as a function of z.

11.3.5.1 Example: A Fibonacci-like recurrence

Let's take a concrete example. The Fibonacci-like recurrence

$$T(n) = T(n-1) + T(n-2), T(0) = 1, T(1) = 1,$$

becomes

$$G = (zG - z) + z^2G + 1 + z.$$

(here $F = 0$).

Solving for G gives

$$G = 1/(1 - z - z^2).$$

Unfortunately this is not something we recognize from our table, although it has shown up in a couple of examples. (Exercise: *Why* does the recurrence $T(n) = T(n-1) + T(n-2)$ count the number of strings built from 0 and 11 of length n?) In the next section we show how to recover a closed-form expression for the coefficients of the resulting series.

11.3.6 Recovering coefficients from generating functions

There are basically three ways to recover coefficients from generating functions:

1. Recognize the generating function from a table of known generating functions, or as a simple combination of such known generating functions. This doesn't work very often but it is possible to get lucky.

2. To find the k-th coefficient of $F(z)$, compute the k-th derivative $d^k/dz^k F(z)$ and divide by $k!$ to shift a_k to the z^0 term. Then substitute 0 for z. For example, if $F(z) = 1/(1-z)$ then $a_0 = 1$ (no differentiating), $a_1 = 1/(1-0)^2 = 1$, $a_2 = 1/(1-0)^3 = 1$, etc. This usually only works if the derivatives have a particularly nice form or if you only care about the first couple of coefficients (it's particularly effective if you only want a_0).

3. If the generating function is of the form $1/Q(z)$, where Q is a polynomial with $Q(0) \neq 0$, then it is generally possible to expand the generating function out as a sum of terms of the form $P_c/(1-z/c)$ where c is a root of Q (i.e. a value such that $Q(c) = 0$). Each denominator P_c will be a constant if c is not a repeated root; if c is a repeated root, then P_c can be a polynomial of degree up to one less than the multiplicity of c. We like these expanded solutions because we recognize $1/(1-z/c) = \sum_i c^{-i}z^i$, and so we can read off the coefficients a_i generated by $1/Q(z)$ as an appropriately weighted some of c_1^{-i}, c_2^{-i}, etc., where the c_j range over the roots of Q.

Example: Take the generating function $G = 1/(1-z-z^2)$. We can simplify it by factoring the denominator: $1-z-z^2 = (1-az)(1-bz)$ where $1/a$ and $1/b$ are the solutions to the equation $1-z-z^2 = 0$; in this case $a = (1+\sqrt{5})/2$, which is approximately 1.618 and $b = (1-\sqrt{5})/2$, which is approximately -0.618. It happens to be the case that we can always expand $1/P(z)$ as $A/(1-az) + B(1-bz)$ for some constants A and B whenever P is a degree 2 polynomial with constant coefficient 1 and distinct roots a and b, so

$$G = A/(1-az) + B/(1-bz),$$

and here we can recognize the right-hand side as the sum of the generating functions for the sequences $A \cdot a^i$ and $B \cdot b^i$. The $A \cdot a^i$ term dominates, so we have that $T(n) = \Theta(a^n)$, where a is approximately 1.618. We can also solve for A and B exactly to find an exact solution if desired.

A rule of thumb that applies to recurrences of the form $T(n) = a_1T(n-1) + a_2T(n-2) + \ldots a_kT(n-k) + f(n)$ is that unless f is particularly large, the solution is usually exponential in $1/x$, where x is the smallest root of the polynomial $1 - a_1z - a_2z^2 \cdots - a_kz^k$. This can be used to get very quick estimates of the solutions to such recurrences (which can then be proved without fooling around with generating functions).

Exercise: What is the exact solution if $T(n) = T(n-1) + T(n-2) + 1$? Or if $T(n) = T(n-1) + T(n-2) + n$?

11.3.6.1 Partial fraction expansion and Heaviside's cover-up method

There is a nice trick for finding the numerators in a partial fraction expansion. Suppose we have

$$\frac{1}{(1-az)(1-bz)} = \frac{A}{1-az} + \frac{B}{1-bz}.$$

Multiply both sides by $1-az$ to get

$$\frac{1}{1-bz} = A + \frac{B(1-az)}{1-bz}.$$

Now plug in $z = 1/a$ to get

$$\frac{1}{1-b/a} = A + 0.$$

We can immediately read off A. Similarly, multiplying by $1-bz$ and then setting $1-bz$ to zero gets B. The method is known as the "cover-up method" because multiplication by $1-az$ can be simulated by covering up $1-az$ in the denominator of the left-hand side and all the terms that don't have $1-az$ in the denominator in the right hand side.

The cover-up method will work in general whenever there are no repeated roots, even if there are many of them; the idea is that setting $1-qz$ to zero knocks out all the terms on the right-hand side but one. With repeated roots we have to worry about getting numerators that aren't just a constant, so things get more complicated. We'll come back to this case below.

Example: A simple recurrence Suppose $f(0) = 0, f(1) = 1$, and for $n \geq 2$, $f(n) = f(n-1) + 2f(n-2)$. Multiplying these equations by z^n and summing over all n gives a generating function

$$F(z) = \sum_{n=0} \infty f(n)z^n = 0 \cdot z^0 + 1 \cdot z^1 + \sum_{n=2}^{\infty} f(n-1)z^n + \sum_{n=2}^{\infty} 2f(n-2)z^n.$$

With a bit of tweaking, we can get rid of the sums on the RHS by converting them into copies of F:

$$\begin{aligned}
F(z) &= z + \sum_{n=2}^{\infty} f(n-1)z^n + 2\sum_{n=2}^{\infty} f(n-2)z^n \\
&= z + \sum_{n=1}^{\infty} f(n)z^{n+1} + 2\sum_{n=0}^{\infty} f(n)z^{n+2} \\
&= z + z\sum_{n=1}^{\infty} f(n)z^n + 2z^2\sum_{n=0}^{\infty} f(n)z^n \\
&= z + z(F(z) - f(0)z^0) + 2z^2F(z) \\
&= z + zF(z) + 2z^2F(z).
\end{aligned}$$

Now solve for $F(z)$ to get $F(x) = \frac{z}{1-z-2z^2} = \frac{z}{(1+z)(1-2z)} = z\left(\frac{A}{1+z} + \frac{B}{1-2z}\right)$, where we need to solve for A and B.

We can do this directly, or we can use the cover-up method. The cover-up method is easier. Setting $z = -1$ and covering up $1 + z$ gives $A = 1/(1 - 2(-1)) = 1/3$. Setting $z = 1/2$ and covering up $1 - 2z$ gives $B = 1/(1 + z) = 1/(1 + 1/2) = 2/3$. So we have

$$\begin{aligned}
F(z) &= \frac{(1/3)z}{1+z} + \frac{(2/3)z}{1-2z} \\
&= \sum_{n=0}^{\infty} \frac{(-1)^n}{3} z^{n+1} + \sum_{n=0}^{\infty} \frac{2\cdot 2^n}{3} z^{n+1} \\
&= \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{3} z^n + \sum_{n=1}^{\infty} \frac{2^n}{3} z^n \\
&= \sum_{n=1}^{\infty} \left(\frac{2^n - (-1)^n}{3}\right) z^n.
\end{aligned}$$

This gives $f(0) = 0$ and, for $n \geq 1$, $f(n) = \frac{2^{n-(-1)n}}{3}$. It's not hard to check that this gives the same answer as the recurrence.

Example: Coughing cows Let's count the number of strings of each length of the form `(M)*(O|U)*(G|H|K)*` where `(x|y)` means we can use x or y and `*` means we can repeat the previous parenthesized expression 0 or more times (these are examples of **regular expressions**).

We start with a sequence of 0 or more M's. The generating function for this part is our old friend $1/(1-z)$. For the second part, we have two choices for each letter, giving $1/(1 - 2z)$. For the third part, we have $1/(1 - 3z)$.

Since each part can be chosen independently of the other two, the generating function for all three parts together is just the product:

$$\frac{1}{(1-z)(1-2z)(1-3z)}.$$

Let's use the cover-up method to convert this to a sum of partial fractions. We have

$$\begin{aligned}
\frac{1}{(1-z)(1-2z)(1-3z)} &= \frac{\left(\frac{1}{(1-2)(1-3)}\right)}{1-z} + \frac{\left(\frac{1}{(1-\frac{1}{2})(1-\frac{3}{2})}\right)}{1-2z} + \frac{\left(\frac{1}{(1-\frac{1}{3})(1-\frac{2}{3})}\right)}{1-3z} \\
&= \frac{\frac{1}{2}}{1-z} + \frac{-4}{1-2z} + \frac{\frac{9}{2}}{1-3z}.
\end{aligned}$$

So the exact number of length-n sequences is $(1/2) - 4 \cdot 2^n + (9/2) \cdot 3^n$. We can check this for small n:

n	Formula	Strings
0	$1/2 - 4 + 9/2 = 1$	()
1	$1/2 - 8 + 27/2 = 6$	M, O, U, G, H, K
2	$1/2 - 16 + 81/2 = 25$	$MM, MO, MU, MG, MH, MK, OO, OU, OG, OH, OK, UO, UU, UG, UH$
3	$1/2 - 32 + 243/2 = 90$	(exercise)$\ddot{\smile}$

Example: A messy recurrence Let's try to solve the recurrence $T(n) = 4T(n-1) + 12T(n-2) + 1$ with $T(0) = 0$ and $T(1) = 1$.

Let $F = \sum T(n)z^n$.

Summing over all n gives

$$\begin{aligned}
F = \sum_{n=0}^{\infty} T(n)z^n &= T(0)z^0 + T(1)z^1 + 4\sum_{n=2}^{\infty} T(n-1)z^n + 12\sum_{n=2}^{\infty} T(n-2)z^n + \sum_{n=2}^{\infty} 1 \cdot z^n \\
&= z + 4z\sum_{n=1}^{\infty} T(n)z^n + 12z^2\sum_{n=0}^{\infty} T(n)z^n + z^2\sum_{n=0}^{\infty} z^n \\
&= z + 4z(F - T(0)) + 12z^2F + \frac{z^2}{1-z} \\
&= z + 4zF + 12z^2F + \frac{z^2}{1-z}.
\end{aligned}$$

Solving for F gives

$$F = \frac{\left(z + \frac{z^2}{1-z}\right)}{1 - 4z - 12z^2}.$$

We want to solve this using partial fractions, so we need to factor $(1-4z-12z^2) = (1+2z)(1-6z)$. This gives

$$\begin{aligned}
F &= \frac{\left(z + \frac{z^2}{1-z}\right)}{(1+2z)(1-6z)} \\
&= \frac{z}{(1+2z)(1-6z)} + \frac{z^2}{(1-z)(1+2z)(1-6z)}. \\
&= z\left(\frac{1}{(1+2z)\left(1-6\left(-\frac{1}{2}\right)\right)} + \frac{1}{\left(1+2\left(\frac{1}{6}\right)\right)(1-6z)}\right) \\
&+ z^2\left(\frac{1}{(1-z)\,(1+2)\,(1-6)} + \frac{1}{\left(1-\left(-\frac{1}{2}\right)\right)(1+2z)\left(1-6\left(-\frac{1}{2}\right)\right)} + \frac{1}{\left(1-\frac{1}{6}\right)\left(1+2\left(\frac{1}{6}\right)\right)(1-6z)}\right) \\
&= \frac{\frac{1}{4}z}{1+2z} + \frac{\frac{3}{4}z}{1-6z} + \frac{-\frac{1}{15}z^2}{1-z} + \frac{\frac{1}{6}z^2}{1+2z} + \frac{\frac{9}{10}z^2}{1-6z}.
\end{aligned}$$

From this we can immediately read off the value of $T(n)$ for $n \geq 2$:

$$\begin{aligned}
T(n) &= \frac{1}{4}(-2)^{n-1} + \frac{3}{4}6^{n-1} - \frac{1}{15} + \frac{1}{6}(-2)^{n-2} + \frac{9}{10}6^{n-2} \\
&= -\frac{1}{8}(-2)^n + \frac{1}{8}6^n - \frac{1}{15} + \frac{1}{24}(-2)^n + \frac{1}{40}6^n \\
&= \frac{3}{20}6^n - \frac{1}{12}(-2)^n - \frac{1}{15}.
\end{aligned}$$

Let's check this against the solutions we get from the recurrence itself:

n	$T(n)$
0	0
1	1
2	$1 + 4 \cdot 1 + 12 \cdot 0 = 5$
3	$1 + 4 \cdot 5 + 12 \cdot 1 = 33$
4	$1 + 4 \cdot 33 + 12 \cdot 5 = 193$

We'll try $n = 3$, and get $T(3) = (3/20) \cdot 216 + 8/12 - 1/15 = (3 \cdot 3 \cdot 216 + 40 - 4)/60 = (1944 + 40 - 4)/60 = 1980/60 = 33$.

To be extra safe, let's try $T(2) = (3/20) \cdot 36 - 4/12 - 1/15 = (3 \cdot 3 \cdot 36 - 20 - 4)/60 = (324 - 20 - 4)/60 = 300/60 = 5$. This looks good too.

The moral of this exercise? Generating functions can solve ugly-looking recurrences exactly, but you have to be very very careful in doing the math.

11.3.6.2 Partial fraction expansion with repeated roots

Let $a_n = 2a_{n-1} + n$, with some constant a_0. We'd like to find a closed-form formula for a_n.

As a test, let's figure out the first few terms of the sequence:

$$\begin{aligned}
a_0 &= a_0\\
a_1 &= 2a_0 + 1\\
a_2 &= 4a_0{+}2{+}2{=}\ 4a_0 + 4\\
a_3 &= 8a_0{+}8{+}3{=}\ 8a_0 + 11\\
a_4 &= 16a_0 + 22{+}4{=}\ 16a_0 + 26
\end{aligned}$$

The a_0 terms look nice (they're $2^n a_0$), but the 0, 1, 4, 11, 26 sequence doesn't look like anything familiar. So we'll find the formula the hard way.

First we convert the recurrence into an equation over generating functions and solve for the generating function F:

$$\begin{aligned}
\sum a_n z^n &= 2\sum a_{n-1}z^n + \sum nz^n + a_0\\
F &= 2zF + \frac{z}{(1-z)^2} + a_0\\
(1-2z)F &= \frac{z}{(1-z)^2} + a_0\\
F &= \frac{z}{(1-z)^2(1-2z)} + \frac{a_0}{1-2z}.
\end{aligned}$$

Observe that the right-hand term gives us exactly the $2^n a_0$ terms we expected, since $1/(1-2z)$ generates the sequence 2^n. But what about the left-hand term? Here we need to apply a partial-fraction expansion, which is simplified because we already know how to factor the denominator but is complicated because there is a repeated root.

We can now proceed in one of two ways: we can solve directly for the partial fraction expansion, or we can use an extended version of Heaviside's cover-up method that handles repeated roots using differentiation. We'll start with the direct method.

Solving for the PFE directly Write

$$\frac{1}{(1-z)^2(1-2z)} = \frac{A}{(1-z)^2} + \frac{B}{1-2z}$$

.

We expect B to be a constant and A to be of the form $A_1 z + A_0$.

To find B, use the technique of multiplying by $1-2z$ and setting $z = 1/2$:

$$\frac{1}{(1-\frac{1}{2})^2} = \frac{A \cdot 0}{(1-z)^2} + B.$$

So $B = 1/(1-1/2)^2 = 1/(1/4) = 4$.

We can't do this for A, but we can solve for it after substituting in $B = 4$:

$$\begin{aligned}
\frac{1}{(1-z)^2(1-2z)} &= \frac{A}{(1-z)^2} + \frac{4}{1-2z} \\
1 &= A(1-2z) + 4(1-z)^2 \\
A &= \frac{1-4(1-z)^2}{1-2z} \\
&= \frac{1-4+8z-4z^2}{1-2z} \\
&= \frac{-3+8z-4z^2}{1-2z} \\
&= \frac{-(1-2z)(3-2z)}{1-2z} \\
&= 2z-3.
\end{aligned}$$

So we have the expansion

$$\frac{1}{(1-z)^2(1-2z)} = \frac{2z-3}{(1-z)^2} + \frac{4}{1-2z},$$

from which we get

$$\begin{aligned}
F &= \frac{z}{(1-z)^2(1-2z)} + \frac{a_0}{1-2z} \\
&= \frac{2z^2-3z}{(1-z)^2} + \frac{4z}{1-2z} + \frac{a_0}{1-2z}.
\end{aligned}$$

If we remember that $1/(1-z)^2$ generates the sequence $x_n = n+1$ and $1/(1-2z)$ generates $x_n = 2^n$, then we can quickly read off the solution (for large n):

$$a_n = 2(n-1) - 3n + 4 \cdot 2^{n-1} + a_0 \cdot 2^n = 2^n a_0 + 2^{n+1} - 2 - n$$

which we can check by plugging in particular values of n and comparing it to the values we got by iterating the recurrence before.

The reason for the "large n" caveat is that $z^2/(1-z)^2$ doesn't generate precisely the sequence $x_n = n-1$, since it takes on the values $0, 0, 1, 2, 3, 4, \dots$ instead of $-1, 0, 1, 2, 3, 4, \dots$. Similarly, the power series for $z/(1-2z)$ does not have the coefficient $2^{n-1} = 1/2$ when $n = 0$. Miraculously, in this particular example the formula works for $n = 0$, even though it shouldn't: $2(n-1)$ is -2 instead of 0, but $4 \cdot 2^{n-1}$ is 2 instead of 0, and the two errors cancel each other out.

Solving for the PFE using the extended cover-up method It is also possible to extend the cover-up method to handle repeated roots. Here we choose a slightly different form of the partial fraction expansion:

$$\frac{1}{(1-z)^2(1-2z)} = \frac{A}{(1-z)^2} + \frac{B}{1-z} + \frac{C}{1-2z}.$$

Here A, B, and C are all constants. We can get A and C by the cover-up method, where for A we multiply both sides by $(1-z)^2$ before setting $z = 1$; this gives $A = 1/(1-2) = -1$ and $C = 1/(1-\frac{1}{2})^2 = 4$. For B, if we multiply both sides by $(1-z)$ we are left with $A/(1-z)$ on the right-hand side and a $(1-z)$ in the denominator on the left-hand side. Clearly setting $z = 1$ in this case will not help us.

The solution is to first multiply by $(1-z)^2$ as before but then take a derivative:

$$\begin{aligned}
\frac{1}{(1-z)^2(1-2z)} &= \frac{A}{(1-z)^2} + \frac{B}{1-z} + \frac{C}{1-2z} \\
\frac{1}{1-2z} &= A + B(1-z) + \frac{C(1-z)^2}{1-2z} \\
\frac{d}{dz}\frac{1}{1-2z} &= \frac{d}{dz}\left(A + B(1-z) + \frac{C(1-z)^2}{1-2z}\right) \\
\frac{2}{(1-2z)^2} &= -B + \frac{-2C(1-z)}{1-2z} + \frac{2C(1-z)^2}{(1-2z)^2}
\end{aligned}$$

Now if we set $z = 1$, every term on the right-hand side except $-B$ becomes 0, and we get $-B = 2/(1-2)^2$ or $B = -2$.

Plugging A, B, and C into our original formula gives

$$\frac{1}{(1-z)^2(1-2z)} = \frac{-1}{(1-z)^2} + \frac{-2}{1-z} + \frac{4}{1-2z},$$

and thus

$$F = \frac{z}{(1-z)^2(1-2z)} + \frac{a_0}{1-2z} = z\left(\frac{-1}{(1-z)^2} + \frac{-2}{1-z} + \frac{4}{1-2z}\right) + \frac{a_0}{1-2z}.$$

From this we can read off (for large n):

$$a_n = 4 \cdot 2^{n-1} - n - 2 + a_0 \cdot 2^n = 2^{n+1} + 2^n a_0 - n - 2.$$

We believe this because it looks like the solution we already got.

11.3.7 Asymptotic estimates

We can simplify our life considerably if we only want an asymptotic estimate of a_n (see Chapter 7). The basic idea is that if a_n is non-negative for sufficiently large n and $\sum a_n z^n$ converges for some fixed value z, then a_n must be $o(z^{-n})$ in the limit. (Proof: otherwise, $a_n z^n$ is at least a constant for infinitely many n, giving a divergent sum.) So we can use the **radius of convergence** of a generating function $F(z)$, defined as the largest value r such that $F(z)$ is defined for all (complex) z with $|z| < r$, to get a quick estimate of the growth rate of F's coefficients: whatever they do, we have $a_n = O(r^{-n})$.

For generating functions that are **rational functions** (ratios of polynomials), we can use the partial fraction expansion to do even better. First observe that for $F(z) = \sum f_i z^n = 1/(1-az)^k$, we have $f_n = \binom{k+n-1}{n} a^n = \frac{(n+k-1)(n+k-2)\dots(k-1)}{(k-1)!} a^n = \Theta(a^n n^{k-1})$. Second, observe that the numerator is irrelevant: if $1/(1-az)^k = \Theta(a^n n^{k-1})$ then $bz^m/(1-az)^{k-1} = b\Theta(a^{n-m}(n-m)^{k-1}) = ba^{-m}(1-m/n)^{k-1}\Theta(a^n n^{k-1}) = \Theta(a^n n^{k-1})$, because everything outside the Θ disappears into the constant for sufficiently large n. Finally, observe that in a partial fraction expansion, the term $1/(1-az)^k$ with the largest coefficient a (if there is one) wins in the resulting asymptotic sum: $\Theta(a^n) + \Theta(b^n) = \Theta(a^n)$ if $|a| > |b|$. So we have:

Theorem 11.3.1. *Let $F(z) = \sum f_n z^n = P(z)/Q(z)$ where P and Q are polynomials in z. If Q has a root r with multiplicity k, and all other roots s of Q satisfy $|r| < |s|$, then $f_n = \Theta((1/r)^n n^{k-1})$.*

The requirement that r is a unique minimal root of Q is necessary; for example, $F(z) = 2/(1-z^2) = 1/(1-z) + 1/(1+z)$ generates the sequence $0, 2, 0, 2, \dots$, which is *not* $\Theta(1)$ because of all the zeros; here the problem is that $1 - z^2$ has two roots with the same absolute value, so for some values of n it is possible for them to cancel each other out.

A root in the denominator of a rational function F is called a **pole**. So another way to state the theorem is that the asymptotic value of the coefficients of a rational generating function is determined by the smallest pole.

More examples:

$F(z)$	Smallest pole	Asymptotic value
$1/(1-z)$	1	$\Theta(1)$
$1/(1-z)^2$	1, multiplicity 2	$\Theta(n)$
$1/(1-z-z^2)$	$(\sqrt{5}-1)/2 = 2/(1+\sqrt{5})$	$\Theta(((1+\sqrt{5})/2)^n)$
$1/((1-z)(1-2z)(1-3z))$	$1/3$	$\Theta(3^n)$
$(z+z^2(1-z))/(1-4z-12z^2)$	$1/6$	$\Theta(6^n)$
$1/((1-z)^2(1-2z))$	$1/2$	$\Theta(2^n)$

In each case it may be instructive to compare the asymptotic values to the exact values obtained earlier on this page.

11.3.8 Recovering the sum of all coefficients

Given a generating function for a convergent series $\sum_i a_i z^i$, we can compute the sum of all the a_i by setting z to 1. Unfortunately, for many common generating functions setting $z = 1$ yields $0/0$ (if it yields something else divided by zero then the series diverges). In this case we can recover the correct sum by taking the limit as z goes to 1 using *L'Hôpital's rule*, which says that $\lim_{x\to c} f(x)/g(x) = \lim_{x\to c} f'(x)/g'(x)$ when the latter limit exists and either $f(c) = g(c) = 0$ or $f(c) = g(c) = \infty$.[8]

11.3.8.1 Example

Let's derive the formula for $1 + 2 + \cdots + n$. We'll start with the generating function for the series $\sum_{i=0}^{n} z^i$, which is $(1 - z^n + 1)/(1 - z)$. Applying the $z\frac{d}{dz}$ method gives us

$$\begin{aligned}\sum_{i=0}^{n} iz^i &= z\frac{d}{dz}\frac{1-z^{n+1}}{1-z}\\ &= z\left(\frac{1}{(1-z)^2} - \frac{(n+1)z^n}{1-z} - \frac{z^{n+1}}{(1-z)^2}\right)\\ &= \frac{z-(n+1)z^{n+1}+nz^{n+2}}{(1-z)^2}.\end{aligned}$$

[8]The justification for doing this is that we know that a finite sequence really has a finite sum, so the "singularity" appearing at $z = 1$ in e.g. $\frac{1-z^{n+1}}{1-z}$ is an artifact of the generating-function representation rather than the original series—it's a "removable singularity" that can be replaced by the limit of $f(x)/g(x)$ as $x \to c$.

Plugging $z = 1$ into this expression gives $(1 - (n+1) + n)/(1 - 1) = 0/0$, which does not make us happy. So we go to the hospital—twice, since one application of L'Hôpital's rule doesn't get rid of our 0/0 problem:

$$\begin{aligned}
\lim_{z\to 1} \frac{z - (n+1)z^{n+1} + nz^{n+2}}{(1-z)^2} &= \lim_{z\to 1} \frac{1 - (n+1)^2 z^n + n(n+2)z^{n+1}}{-2(1-z)} \\
&= \lim_{z\to 1} \frac{-n(n+1)^2 z^{n-1} + n(n+1)(n+2)z^n}{2} \\
&= \frac{-n(n+1)^2 + n(n+1)(n+2)}{2} \\
&= \frac{-n^3 - 2n^2 - n + n^3 + 3n^2 + 2n}{2} \\
&= \frac{n^2+n}{2} = \frac{n(n+1)}{2},
\end{aligned}$$

which is our usual formula. Gauss's childhood proof is a lot quicker, but the generating-function proof is something that we could in principle automate most of the work using a computer algebra system, and it doesn't require much creativity or intelligence. So it might be the weapon of choice for nastier problems where no clever proof comes to mind.

More examples of this technique can be found in §11.2, where the binomial theorem applied to $(1 + x)^n$ (which is really just a generating function for $\sum \binom{n}{i} z^i$) is used to add up various sums of binomial coefficients.

11.3.9 A recursive generating function

Let's suppose we want to count binary trees with n internal nodes. We can obtain such a tree either by (a) choosing an empty tree (g.f.: $z^0 = 1$); or (b) choosing a root with weight 1 (g.f. $1 \cdot z^1 = z$), since we can choose it in exactly one way), and two subtrees (g.f. $= F^2$ where F is the g.f. for trees). This gives us a recursive definition

$$F = 1 + zF^2.$$

Solving for F using the quadratic formula gives

$$F = \frac{1 \pm \sqrt{1-4z}}{2z}.$$

That $2z$ in the denominator may cause us trouble later, but let's worry about that when the time comes. First we need to figure out how to extract coefficients from the square root term.

The binomial theorem says

$$\sqrt{1-4z} = (1-4z)^{1/2} = \sum_{n=0}^{\infty} \binom{1/2}{n} (-4z)^n.$$

For $n \geq 1$, we can expand out the $\binom{1/2}{n}$ terms as

$$\begin{aligned}
\binom{1/2}{n} &= \frac{(1/2)_{(n)}}{n!} \\
&= \frac{1}{n!} \cdot \prod_{k=0}^{n-1} (1/2 - k) \\
&= \frac{1}{n!} \cdot \prod_{k=0}^{n-1} \frac{1-2k}{2} \\
&= \frac{(-1)^n}{2^n n!} \cdot \prod_{k=0}^{n-1} (2k-1) \\
&= \frac{(-1)^n}{2^n n!} \cdot \frac{\prod_{k=1}^{2n-2} k}{\prod_{k=1}^{n-1} 2k} \\
&= \frac{(-1)^n}{2^n n!} \cdot \frac{(2n-2)!}{2^{n-1}(n-1)!} \\
&= \frac{(-1)^n}{2^{2n-1}} \cdot \frac{(2n-2)!}{n!(n-1)!} \\
&= \frac{(-1)^n}{2^{2n-1}(2n-1)} \cdot \frac{(2n-1)!}{n!(n-1)!} \\
&= \frac{(-1)^n}{2^{2n-1}(2n-1)} \cdot \binom{2n-1}{n}.
\end{aligned}$$

For $n = 0$, the switch from the big product of odd terms to $(2n-2)!$ divided by the even terms doesn't work, because $(2n-2)!$ is undefined. So here we just use the special case $\binom{1/2}{0} = 1$.

Now plug this nasty expression back into F to get

$$
\begin{aligned}
F &= \frac{1 \pm \sqrt{1-4z}}{2z} \\
&= \frac{1}{2z} \pm \frac{1}{2z} \sum_{n=0}^{\infty} \binom{1/2}{n} (-4z)^n \\
&= \frac{1}{2z} \pm \left(\frac{1}{2z} + \frac{1}{2z} \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{2^{2n-1}(2n-1)} \binom{2n-1}{n} (-4z)^n \right) \\
&= \frac{1}{2z} \pm \left(\frac{1}{2z} + \frac{1}{2z} \sum_{n=1}^{\infty} \frac{(-1)^{2n-1}2^{2n}}{2^{2n-1}(2n-1)} \binom{2n-1}{n} z^n \right) \\
&= \frac{1}{2z} \pm \left(\frac{1}{2z} + \frac{1}{2z} \sum_{n=1}^{\infty} \frac{-2}{(2n-1)} \binom{2n-1}{n} z^n \right) \\
&= \frac{1}{2z} \pm \left(\frac{1}{2z} - \sum_{n=1}^{\infty} \frac{1}{(2n-1)} \binom{2n-1}{n} z^{n-1} \right) \\
&= \frac{1}{2z} \pm \left(\frac{1}{2z} - \sum_{n=0}^{\infty} \frac{1}{(2n+1)} \binom{2n+1}{n+1} z^n \right) \\
&= \sum_{n=0}^{\infty} \frac{1}{(2n+1)} \binom{2n+1}{n+1} z^n \\
&= \sum_{n=0}^{\infty} \frac{1}{n+1} \binom{2n}{n} z^n.
\end{aligned}
$$

Here we choose minus for the plus-or-minus to get the right answer and then do a little bit of tidying up of the binomial coefficient.

We can check the first few values of $f(n)$:

n	$f(n)$
0	$\binom{0}{0} = 1$
1	$(1/2)\binom{2}{1} = 1$
2	$(1/3)\binom{4}{2} = 6/3 = 2$
3	$(1/4)\binom{6}{3} = 20/4 = 5$

and these are consistent with what we get if we draw all the small binary trees by hand.

The numbers $\frac{1}{n+1}\binom{2n}{n}$ show up in a lot of places in combinatorics, and are known as the **Catalan numbers**.

11.3.10 Summary of operations on generating functions

The following table describes all the nasty things we can do to a generating function. Throughout, we assume $F = \sum f_k z^k$, $G = \sum g_k z^k$, etc.

Operation	Generating functions	Coefficients	Combinatorial interpretation
Find f_0	$f_0 = F(0)$	Returns f_0	Count weight 0 objects.
Find f_k	$f_k = \frac{1}{k!}\frac{d^k}{dz^k}F(z)\vert_{z=0}$	Returns f_k	Count weight k objects.
Flatten	$F(1)$	Computes $\sum f_k$	Count all objects, ignoring weights.
Shift right	$G = zF$	$g_k = f_{k-1}$	Add 1 to all weights.
Shift left	$G = z^{-1}(F - F(0))$	$g_k = f_{k+1}$	Subtract 1 from all weights, after removing any weight-0 objects.
Pointing	$G = z\frac{d}{dz}F$	$g_k = kf_k$	A G-thing is an F-thing with a label pointing to one of its units.
Sum	$H = F + G$	$h_k = f_k + g_k$	Disjoint union.
Product	$H = FG$	$h_k = \sum_i f_i g_{k-i}$	Cartesian product.
Composition	$H = F \circ G$	$H = \sum f_k G^k$	To make an H-thing, first choose an F-thing of weight m, then bolt onto it m G-things. The weight of the H-thing is the sum of the weights of the G-things.
Repetition	$G = 1/(1 - F)$	$G = \sum F^k$	A G-thing is a sequence of zero or more F-things. Note: this is just a special case of composition.

11.3.11 Variants

The **exponential generating function** or **egf** for a sequence $a_0, \dots$ is given by $F(z) = \sum a_n z^n/n!$. For example, the egf for the sequence $1, 1, 1, \dots$ is $e^z = \sum z^n/n!$. Exponential generating functions admit a slightly different set of operations from ordinary generating functions: differentiation gives left shift (since the factorials compensate for the exponents coming down), multiplying by z gives $b_n = na_{n+1}$, etc. The main application is that the product $F(z)G(z)$ of two egf's gives the sequence whose n-th term is $\sum \binom{n}{k} a_k b_{n-k}$; so for problems where we want that binomial coefficient in the convolution (e.g. when we are building weight n objects not only by choosing a weight-k object plus a weight-$(n-k)$ object but also by arbitrarily rearranging their unit-weight pieces) we want to use an egf rather than an ogf. We won't use these in CS202, but it's worth knowing they exist.

A **probability generating function** or **pgf** is essentially an ordinary generating function where each coefficient a_n is the probability that some random variable equals n. See §12.2 for more details.

11.3.12 Further reading

Rosen [Ros12] discusses some basic facts about generating functions in §8.4. Graham *et al.* [GKP94] give a more thorough introduction. Herbert Wilf's book *generatingfunctionology*, which can be downloaded from the web, will tell you more about the subject than you probably want to know.

See this page for very detailed notes on partial fraction expansion.

Chapter 12

Probability theory

Here are two examples of questions we might ask about the likelihood of some event:

- Gambling: I throw two six-sided dice, what are my chances of seeing a 7?
- Insurance: I insure a typical resident of Smurfington-Upon-Tyne against premature baldness. How likely is it that I have to pay a claim?

Answers to these questions are summarized by a **probability**, a number in the range 0 to 1 that represents the likelihood that some event occurs. There are two dominant interpretations of this likelihood:

- The **frequentist interpretation** says that if an event occurs with probability p, then in the limit as I accumulate many examples of similar events, I will see the number of occurrences divided by the number of samples converging to p. For example, if I flip a fair coin over and over again many times, I expect that heads will come up roughly half of the times I flip it, because the probability of coming up heads is 1/2.
- The **Bayesian interpretation** says that when I say that an event occurs with probability p, that means my subjective beliefs about the event would lead me to take a bet that would be profitable on average if this were the real probability. So a Bayesian would take a double-or-nothing bet on a coin coming up heads if they believed that the probability it came up heads was at least 1/2.

Frequentists and Bayesians have historically spent a lot of time arguing with each other over which interpretation makes sense. The usual argument against frequentist probability is that it only works for repeatable experiments, and doesn't allow for statements like "the probability that it will rain tomorrow is 50%" or the even more problematic "based on what I know, there is a 50% probability that it rained yesterday." The usual argument against Bayesian probability is that it's hopelessly subjective—it's possible (even likely) that my subjective guesses about the probability that it will rain tomorrow are not the same as yours.[1]

As mathematicians, we can ignore such arguments, and treat probability axiomatically as just another form of counting, where we normalize everything so that we always end up counting to exactly 1. It happens to be the case that this approach to probability works for both frequentist interpretations (assuming that the probability of an event measures the proportion of outcomes that cause the event to occur) and Bayesian interpretations (assuming our subjective beliefs are consistent).

12.1 Events and probabilities

We'll start by describing the basic ideas of probability in terms of probabilities of events, which either occur or don't. Later we will generalize these ideas and talk about random variables, which may take on many different values in different outcomes.

12.1.1 Probability axioms

Coming up with axioms for probabilities that work in all the cases we want to consider took much longer than anybody expected, and the current set in common use only go back to the 1930's. Before presenting these, let's talk a bit about the basic ideas of probability.

An **event** A is something that might happen, or might not; it acts like a predicate over possible outcomes. The **probability** $\Pr[A]$ of an event A is a real number in the range 0 to 1, that must satisfy certain consistency rules like $\Pr[\neg A] = 1 - \Pr[A]$.

In **discrete probability**, there is a finite set of **atoms**, each with an assigned probability, and every event is a union of atoms. The probability

[1] This caricature of the debate over interpreting probability is thoroughly incomplete. For a thoroughly complete discussion, including many other interpretations, see `http://plato.stanford.edu/entries/probability-interpret/`.

assigned to an event is the sum of the probabilities assigned to the atoms it contains. For example, we could consider rolling two six-sided dice. The atoms are the pairs (i, j) that give the value on the first and second die, and we assign a probability of $1/36$ to each pair. The probability that we roll a 7 is the sum of the cases $(1, 6)$, $(2, 5)$, $(3, 4)$, $(4, 3)$, $(5, 2)$, and $(6, 1)$, or $6/36 = 1/6$.

Discrete probability doesn't work if we have infinitely many atoms. Suppose we roll a pair of dice infinitely many times (e.g., because we want to know the probability that we never accumulate more 6's than 7's in this infinite sequence). Now there are infinitely many possible outcomes: all the sequences of pairs (i, j). If we make all these outcomes equally likely, we have to assign each a probability of zero. But then how do we get back to a probability of $1/6$ that the first roll comes up 7?

12.1.1.1 The Kolmogorov axioms

A triple $(\Omega, \mathcal{F}, P)$ is a **probability space** if Ω is a set of **outcomes** (where each outcome specifies everything that ever happens, in complete detail); $\mathcal{F}$ is a **sigma-algebra**, which is a family of subsets of Ω, called **measurable sets**, that is closed under complement (i.e., if A is in $\mathcal{F}$ then $\Omega \setminus A$ is in $\mathcal{F}$) and countable union (union of $A_1, A_2, \ldots$ is in $\mathcal{F}$ if each set A_i is); and P is a **probability measure** that assigns a number in $[0, 1]$ to each set in $\mathcal{F}$. The measure P must satisfy three axioms, due to Kolmogorov [Kol33]:

1. $P(A) \geq 0$ for all $A \in \mathcal{F}$.
2. $P(\Omega) = 1$.
3. For any sequence of pairwise disjoint events $A_1, A_2, A_3, \ldots, P(\cup A_i) = \sum P(A_i)$.

From these one can derive rules like $P(\Omega \setminus A) = 1 - P(A)$ etc.

Most of the time, Ω is finite, and we can just make $\mathcal{F}$ include all subsets of Ω, and define $P(A)$ to be the sum of $P(\{x\})$ over all x in A. This gets us back to the discrete probability model we had before.

Unless we are looking at multiple probability spaces or have some particular need to examine Ω, $\mathcal{F}$, or P closely, we usually won't bother specifying the details of the probability space we are working in. So most of the time we will just refer to "the" probability $\Pr[A]$ of an event A, bearing in mind that we are implicitly treating A as a subset of some implicit Ω that is measurable with respect to an implicit $\mathcal{F}$ and whose probability is really $P(A)$ for some implicit measure P.

12.1.1.2 Examples of probability spaces

- $\Omega = \{\mathsf{H}, \mathsf{T}\}$, $\mathcal{F} = \mathcal{P}(\Omega) = \{\{,\}\{\mathsf{H}\}, \{\mathsf{T}\}, \{\mathsf{H}, \mathsf{T}\}\}$, $\Pr[A] = |A|/2$. This represents a fair coin with two outcomes H and T that each occur with probability 1/2.

- $\Omega = \{\mathsf{H}, \mathsf{T}\}$, $\mathcal{F} = \mathcal{P}(\Omega)$, $\Pr[\{\mathsf{H}\}] = p$, $\Pr[\{\mathsf{T}\}] = 1-p$. This represents a biased coin, where H comes up with probability p.

- $\Omega = \{(i,j) \mid i,j \in \{1,2,3,4,5,6\}\}$, $\mathcal{F} = \mathcal{P}(\Omega)$, $\Pr[A] = |A|/36$. Roll of two fair dice. A typical event might be "the total roll is 4", which is the set $\{(1,3),(2,2),(3,1)\}$ with probability $3/36 = 1/12$.

- $\Omega = \mathbb{N}$, $\mathcal{F} = \mathcal{P}(\Omega)$, $\Pr[A] = \sum_{n \in A} 2^{-n-1}$. This is an infinite probability space; a real-world process that might generate it is to flip a fair coin repeatedly and count how many times it comes up tails before the first time it comes up heads. Note that even though it is infinite, we can still define all probabilities by summing over atoms: $\Pr[\{0\}] = 1/2$, $\Pr[1] = 1/4$, $\Pr[2] = 1/8$, etc.

It's unusual for anybody doing probability to actually write out the details of the probability space like this. Much more often, a writer will just assert the probabilities of a few basic events (e.g. $\Pr[\{\mathsf{H}\}] = 1/2$), and claim that any other probability that can be deduced from these initial probabilities from the axioms also holds (e.g. $\Pr[\{\mathsf{T}\}] = 1 - \Pr[\{\mathsf{H}\}] = 1/2$). The main reason Kolmogorov gets his name attached to the axioms is that he was responsible for **Kolmogorov's extension theorem**, which says (speaking very informally) that as long as your initial assertions are consistent, there exists a probability space that makes them and all their consequences true.

12.1.2 Probability as counting

The easiest probability space to work with is a **uniform discrete probability space**, which has N outcomes each of which occurs with probability $1/N$. If someone announces that some quantity is "random" without specifying probabilities (especially if that someone is a computer scientist), the odds are that what they mean is that each possible value of the quantity is equally likely. If that someone is being more careful, they would say that the quantity is "drawn uniformly at random" from a particular set.

Such spaces are among the oldest studied in probability, and go back to the very early days of probability theory where randomness was almost always expressed in terms of pulling tokens out of well-mixed urns, because such "urn models" where one of the few situations where everybody agreed on what the probabilities should be.

12.1.2.1 Examples

- A **random bit** has two outcomes, 0 and 1. Each occurs with probability 1/2.
- A **die roll** has six outcomes, 1 through 6. Each occurs with probability 1/6.
- A roll of two dice has 36 outcomes (order of the dice matters). Each occurs with probability 1/36.
- A random n-bit string has 2^n outcomes. Each occurs with probability 2^{-n}. The probability that exactly one bit is a 1 is obtained by counting all strings with a single 1 and dividing by 2^n. This gives $n2^{-n}$.
- A **poker hand** consists of a subset of 5 cards drawn uniformly at random from a deck of 52 cards. Depending on whether the order of the 5 cards is considered important (usually it isn't), there are either $\binom{52}{5}$ or $(52)_5$ possible hands. The probability of getting a flush (all five cards in the hand drawn from the same suit of 13 cards) is $4\binom{13}{5}/$ $binom525$; there are 4 choices of suits, and $\binom{13}{5}$ ways to draw 5 cards from each suit.
- A **random permutation** on n items has $n!$ outcomes, one for each possible permutation. A typical event might be that the first element of a random permutation of $1 \dots n$ is 1; this occurs with probability $(n-1)!/n! = 1/n$. Another example of a random permutation might be a uniform shuffling of a 52-card deck (difficult to achieve in practice!). Here, the probability that we get a particular set of 5 cards as the first 5 in the deck is obtained by counting all the permutations that have those 5 cards in the first 5 positions (there are $5! \cdot 47!$ of them) divided by 52!. The result is the same $1/\binom{52}{5}$ that we get from the uniform poker hands.

12.1.3 Independence and the intersection of two events

Events A and B are **independent** if $\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]$. In general, a set of events $\{A_i\}$ is independent if each A_i is independent of any event defined only in terms of the other events.

It can be dangerous to assume that events are independent when they aren't, but quite often when describing a probability space we will explicitly state that certain events are independent. For example, one typically describes the space of random n-bit strings (or n coin flips) by saying that one has n independent random bits and then deriving that each particular sequence occurs with probability 2^{-n} rather than starting with each sequence occurring with probability 2^{-n} and then calculating that each particular bit is 1 with independent probability $1/2$. The first description makes much more of the structure of the probability space explicitly, and so is more directly useful in calculation.

12.1.3.1 Examples

- What is the probability of getting two heads on independent fair coin flips? Calculate it directly from the definition of independence: $\Pr[H_1 \cap H_2] = (1/2)(1/2) = 1/4$.
- Suppose the coin-flips are *not* independent (maybe the two coins are glued together). What is the probability of getting two heads? This can range anywhere from zero (coin 2 always comes up the opposite of coin 1) to $1/2$ (if coin 1 comes up heads, so does coin 2).
- What is the probability that both you and I draw a flush (all 5 cards the same suit) from the same poker deck? Since we are fighting over the same collection of same-suit subsets, we'd expect $\Pr[A \cap B] < \Pr[A] \cdot \Pr[B]$—the event that you get a flush (A) is not independent of the event that I get a flush (B), and we'd have to calculate the probability of both by counting all ways to draw two hands that are both flushes. But if we put your cards back and then shuffle the deck again, the events in this new case *are* independent, and we can just square the $\Pr[\text{flush}]$ that we calculated before.
- Suppose the Red Sox play the Yankees. What is the probability that the final score is exactly 4–4? Amazingly, it appears that it is equal to[2]

$$\begin{aligned}&\Pr[\text{Red Sox score 4 runs against the Yankees}]\\ &\cdot \Pr[\text{Yankees score 4 runs against the Red Sox}].\end{aligned}$$

[2] See `http://arXiv.org/abs/math/0509698`.

To the extent we can measure the underlying probability distribution, the score of each team in a professional baseball game appears to be independent of the score of the other team.

12.1.4 Union of events

What is the probability of $A \cup B$? If A and B are disjoint, then the axioms give $\Pr[A \cup B] = \Pr[A] + \Pr[B]$. But what if A and B are not disjoint?

By analogy to inclusion-exclusion in counting we would expect that

$$\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B].$$

Intuitively, when we sum the probabilities of A and B, we double-count the event that both occur, and must subtract it off to compensate. To prove this formally, consider the events $A \cap B$, $A \cap \neg B$, and $\neg A \cap B$. These are disjoint, so the probability of the union of any subset of this set of events is equal to the sum of its components. So in particular we have

$$\begin{aligned}
&\Pr[A] + \Pr[B] - \Pr[A \cap B] \\
&\quad = (\Pr[A \cap B] + \Pr[A \cap \neg B]) + (\Pr[A \cap B] + \Pr[\neg A \cap B]) - \Pr[A \cap B] \\
&\quad = \Pr[A \cap B] + \Pr[A \cap \neg B] + \Pr[\neg A \cap B] \\
&\quad = \Pr[A \cup B].
\end{aligned}$$

12.1.4.1 Examples

- What is the probability of getting at least one head out of two independent coin-flips? Compute $\Pr[H_1 \cap H_2] = 1/2+1/2-(1/2)(1/2) = 3/4$.
- What is the probability of getting at least one head out of two coin-flips, when the coin-flips are not independent? Here again we can get any probability from 0 to 1, because the probability of getting at least one head is just $1 - \Pr[H_1 H_2]$.

For more events, we can use a probabilistic version of the inclusion-exclusion formula (Theorem 11.2.2). The new version looks like this:

Theorem 12.1.1. *Let $A_1 \dots A_n$ be events on some probability space. Then*

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{S \subseteq \{1 \dots n\}, S \neq \emptyset} (-1)^{|S|+1} \Pr\left[\bigcap_{j \in S} A_j\right]. \tag{12.1.1}$$

For discrete probability, the proof is essentially the same as for Theorem 11.2.2; the difference is that instead of showing that we add 1 for each possible element of $\bigcap A_i$, we show that we add the probability of each outcome in $\bigcap A_i$. The result continues to hold for more general spaces, but requires a little more work.[3]

12.1.5 Conditional probability

Suppose I want to answer the question "What is the probability that my dice add up to 6 if I know that the first one is an odd number?" This question involves **conditional probability**, where we calculate a probability subject to some conditions. The probability of an event A conditioned on an event B, written $\Pr[A \mid B]$, is defined by the formula

$$\Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]}.$$

One way to think about this is that when we assert that B occurs we are in effect replacing the entire probability space with just the part that sits in B. So we have to divide all of our probabilities by $\Pr[B]$ in order to make $\Pr[B \mid B] = 1$, and we have to replace A with $A \cap B$ to exclude the part of A that can't happen any more.

Note also that conditioning on B only makes sense if $\Pr[B] > 0$. If $\Pr[B] = 0$, $\Pr[A \mid B]$ is undefined.

12.1.5.1 Conditional probabilities and intersections of non-independent events

Simple algebraic manipulation gives

$$\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B].$$

So one of the ways to compute the probability of two events occurring is to compute the probability of one of them, and the multiply by the probability that the second occurs conditioned on the first. For example, if my attempt to reach the summit of Mount Everest requires that I first learn how to climb mountains ($\Pr[B] = 0.1$) and then make it to the top safely ($\Pr[A \mid B] = 0.9$), then my chances of getting to the top are $\Pr[A \cap B \mid =](0.9)(0.1) = 0.09$.

[3]The basic idea is to chop $\bigcap A_i$ into all sets of the form $\bigcup B_i$ where each B_i is either A_i or $\neg A_i$; this reduces to the discrete case.

We can do this for sequences of events as well. Suppose that I have an urn that starts with k black balls and 1 red ball. In each of n trials I draw one ball uniformly at random from the urn. If it is red, I give up. If it is black, I put the ball back and add another black ball, thus increasing the number of balls by 1. What is the probability that on every trial I get a black ball?

Let A_i be the event that I get a black ball on the first i trials. Then $\Pr[A_0] = 1$, and for larger i we have $\Pr[A_i] = \Pr[A_i \mid A_{i-1}] \Pr[A_{i-1}]$. If A_{i-1} holds, then at the time of the i-th trial we have $k + i$ total balls in the urn, of which one is red. So the probability that we draw a black ball is $1 - \frac{1}{k+i} = \frac{k+i-1}{k+i}$. By induction we can then show that

$$\Pr[A_i] = \prod_{j=1}^{i} \frac{k+j-1}{k+j}.$$

This is an example of a collapsing product, where the denominator of each fraction cancels out the numerator of the next; we are left only with the denominator $k + i$ of the last term and the numerator k of the first, giving $\Pr[A_i] = \frac{k}{k+i}$. It follows that we make it through all n trials with probability $\Pr[A_n] = \frac{k}{k+n}$.

12.1.5.2 The law of total probability

We can use the fact that A is the disjoint union of $A \cap B$ and $A \cap \overline{B}$ to get $\Pr[A]$ by case analysis:

$$\begin{aligned} \Pr[A] &= \Pr[A \cap B] + \Pr\left[A \cap \overline{B}\right] \\ &= \Pr[A \mid B] \Pr[B] + \Pr\left[A \,\middle|\, \overline{B}\right] \Pr\left[\overline{B}\right]. \end{aligned}$$

For example, if there is a 0.2 chance I can make it to the top of Mt Everest safely without learning how to climb first, my chances of getting there go up to $(0.9)(0.1) + (0.2)(0.9) = 0.27$.

This method is sometimes given the rather grandiose name of the **law of total probability**. The most general version is that if $B_1 \dots B_n$ are all disjoint events and the sum of their probabilities is 1, then

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A] B_i \Pr[B_i]$$

12.1.5.3 Bayes's formula

If one knows $\Pr[A \mid B]$, $\Pr[A \mid \neg B]$, and $\Pr[B]$, it's possible to compute $\Pr[B \mid A]$:

$$\begin{aligned}\Pr[B \mid A] &= \frac{\Pr[A \cap B]}{\Pr[A]} \\ &= \frac{\Pr[A \mid B]\Pr[B]}{\Pr[A]} \\ &= \frac{\Pr[A \mid B]\Pr[B]}{\Pr[A \mid B]\Pr[B] + \Pr\left[A \,\middle|\, \overline{B}\right]\Pr\left[\overline{B}\right]}.\end{aligned}$$

This formula is used heavily in statistics, where it goes by the name of **Bayes's formula**. Say that you have an Airport Terrorist Detector that lights up with probability 0.75 when inserted into the nostrils of a Terrorist, but lights up with probability 0.001 when inserted into the nostrils of a non-Terrorist. Suppose that for other reasons you know that Granny has only a 0.0001 chance of being a Terrorist. What is the probability that Granny is a Terrorist if the detector lights up?

Let B be the event "Granny is a terrorist" and A the event "Detector lights up." Then $\Pr[B \mid A] = (0.75 \times 0.0001)/(0.75 \times 0.0001 + 0.001 \times 0.9999) \approx 0.0007495$. This examples show how even a small **false positive** rate can make it difficult to interpret the results of tests for rare conditions.

12.2 Random variables

A **random variable** X is a variable that takes on particular values randomly. This means that for each possible value x, there is an event $[X = x]$ with some probability of occurring that corresponds to X (the random variable, usually written as an upper-case letter) taking on the value x (some fixed value). Formally, a random variable X is really a function $X(\omega)$ of the outcome ω that occurs, but we save a lot of ink by leaving out ω.[4]

12.2.1 Examples of random variables

- Indicator variables: The **indicator variable** for an event A is a variable X that is 1 if A occurs and 0 if it doesn't (i.e., $X(\omega) = 1$ if $\omega \in A$

[4]For some spaces, not all functions $X(\omega)$ work as random variables, because the events $[X = x]$ might not be measurable with respect to $\mathcal{F}$. We will generally not run into these issues.

and 0 otherwise). There are many conventions out there for writing indicator variables. I am partial to 1_A, but you may also see them written using the Greek letter chi (e.g. χ_A) or by abusing the bracket notation for events (e.g., $[A]$, $[Y^2 > 3]$, [all six coins come up heads]).

- Functions of random variables: Any function you are likely to run across of a random variable or random variables is a random variable. So if X and Y are random variables, $X + Y$, XY, and $\log X$ are all random variables.

- Counts of events: Flip a fair coin n times and let X be the number of times it comes up heads. Then X is an integer-valued random variable.

- Random sets and structures: Suppose that we have a set T of n elements, and we pick out a subset U by flipping an independent fair coin for each element to decide whether to include it. Then U is a set-valued random variable. Or we could consider the infinite sequence $X_0, X_1, X_2, \dots$, where $X_0 = 0$ and X_{n+1} is either $X_n + 1$ or $X_n - 1$, depending on the result of independent fair coin flip. Then we can think of the entire sequence X as a sequence-valued random variable.

12.2.2 The distribution of a random variable

The **distribution** of a random variable describes the probability that it takes on various values. For real-valued random variables, the **probability distribution function** is given by a function of the form $F(x) = \Pr[X \le x]$. This allows for very general distributions—for example, a variable that is uniform on $[0, 1]$ can be specified by $F(x) = x$ when $0 \le x \le 1$, and 0 or 1 as appropriate outside this interval—but for **discrete random variables** that take on only countably many possible values, this is usually more power than we need.

For these variables, the distribution is most easily described by just giving the **probability mass function** $\Pr[X = x]$ for each possible value x. If we need to, it's not too hard to recover the distribution function from the mass function (or vice versa). So we will often cheat a bit and treat a mass function as specifying a distribution even if it isn't technically a distribution function.

Typically, if we know the distribution of a random variable, we don't bother worrying about what the underlying probability space is. The reason for this is we can just take Ω to be the range of the random variable, and define $\Pr[\omega]$ for each ω in Ω to be $\Pr[X = \omega]$. For example, a six-sided die corresponds to taking $\Omega = \{1, 2, 3, 4, 5, 6\}$, assigning $\Pr[\omega] = 1/6$ for all ω, and letting $X(\omega) = \omega$. This will give the probabilities for any events involving X that we would have gotten on whatever original probability space X might have been defined on.

The same thing works if we have multiple random variables, but now we let each point in the space be a tuple that gives the values of all of the variables. Specifying the probability in this case is done using a **joint distribution** (see below).

12.2.2.1 Some standard distributions

Here are some common distributions for a random variable X:

- **Bernoulli distribution**: $\Pr[X = 1] = p$, $\Pr[X = 0] = q$, where p is a parameter of the distribution and $q = 1 - p$. This corresponds to a single biased coin-flip.

- **Binomial distribution**: $\Pr[X = k] = \binom{n}{k} p^k q^{(n-k)}$, where n and p are parameters of the distribution and $q = 1 - p$. This corresponds to the sum of n biased coin-flips.

- **Geometric distribution**: $\Pr[X = k] = q^k p$, where p is a parameter of the distribution and q is again equal to $1 - p$. This corresponds to number of tails we flip before we get the first head in a sequence of biased coin-flips.

- **Poisson distribution**: $\Pr[X = k] = e^{-\lambda}\lambda^k/k!$. This is what happens to a binomial distribution when we make $p = \lambda/n$ and then take the limit as n goes to infinity. We can think of it as counting the number of events that occur in one time unit if the events occur at a constant continuous rate that averages λ events per time unit. The canonical example is radioactive decay.

- **Uniform distribution**: For the uniform distribution on $[a, b]$, the distribution function F of X is given by $F(x) = 0$ when $x \leq a$, $(x - a)/(b - a)$ when $a \leq x \leq b$, and 1 when $b \leq x$, where a and b are parameters of the distribution. This is a continuous random variable that has equal probability of landing anywhere in the $[a, b]$ interval.

  The term *uniform distribution* may also refer to a uniform distribution on a finite set S; this assigns $\Pr[X = x] = \frac{1}{|S|}$ when x is in S and 0 otherwise. As a distribution function, $F(x)$ is the rather discontinuous function $|\{y \in S \mid y \leq x\}| / |S|$.

- **Normal distribution**: The normal distribution function is given by

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2} dx.$$

  This corresponds to another limit of the binomial distribution, where now we fix $p = 1/2$ but compute $\frac{X-n/2}{\sqrt{n}}$ to converge to a single fixed distribution as n goes to infinity. The normal distribution shows up (possibly scaled and shifted) whenever we have a sum of many independent, identically distributed random variables: this is the **Central Limit Theorem**, and is the reason why much of statistics works, and why we can represent 0 and 1 bits using buckets of jumpy randomly-positioned electrons.

12.2.2.2 Joint distributions

Two or more random variables can be described using a **joint distribution**. For discrete random variables, we often represent this as a joint probability mass function $\Pr[X = x \wedge Y = y]$ for all fixed values x and y, or more generally $\Pr[\forall i : X_i = x_i]$. For continuous random variables, we may instead need to use a joint distribution function $F(x_1, \ldots, x_n) = \Pr[\forall i : X_i \leq x_i]$.

Given a joint distribution on X and Y, we can recover the distribution on X or Y individually by summing up cases: $\Pr[X = x] = \sum_y \Pr[X = x \wedge Y = y]$ (for discrete variables), or $\Pr[X \leq x] = \lim_{y\to\infty} \Pr[X \leq x \wedge Y \leq y]$ (for more general variables). The distribution of X obtained in this way is called a **marginal distribution** of the original joint distribution. In general, we can't go in the other direction, because just knowing the marginal distributions doesn't tell us how the random variables might be dependent on each other.

Examples

- Let X and Y be six-sided dice. Then $\Pr[X = x \wedge Y = y] = 1/36$ for all values of x and y in $\{1, 2, 3, 4, 5, 6\}$. The underlying probability space consists of all pairs (x, y) in $\{1, 2, 3, 4, 6\} \times \{1, 2, 3, 4, 5, 6\}$.

- Let X be a six-sided die and let $Y = 7-X$. Then $\Pr[X = x \wedge Y = y] = 1/6$ if $1 \leq x \leq 6$ and $y = 7-x$, and 0 otherwise. The underlying probability space is most easily described by including just six points for the X values, although we could also do $\{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}$ as in the previous case, just assigning probability 0 to most of the points. However, even though the joint distribution is very different from the previous case, the marginal distributions of X and Y are exactly the same as before: each of X and Y takes on all values in $\{1, 2, 3, 4, 5, 6\}$ with equal probability.

12.2.3 Independence of random variables

The difference between the two preceding examples is that in the first case, X and Y are independent, and in the second case, they aren't.

Two random variables X and Y are **independent** if any pair of events of the form $X \in A$, $Y \in B$ are independent. For discrete random variables, it is enough to show that $\Pr[X = x \wedge Y = y] = \Pr[X = x] \cdot \Pr[Y = y]$, or in other words that the events $[X = x]$ and $[Y = y]$ are independent for all values x and y. For continuous random variables, the corresponding equation is $\Pr[X \leq x \wedge Y \leq y] = \Pr[X \leq x] \cdot \Pr[Y \leq y]$. In practice, we will typically either be told that two random variables are independent or deduce it from the fact that they arise from separated physical processes.

12.2.3.1 Examples

- Roll two six-sided dice, and let X and Y be the values of the dice. By convention we assume that these values are independent. This means for example that $\Pr[X \in \{1, 2, 3l\} \wedge Y \in \{1, 2, 3\}] = \Pr[X \in \{1, 2, 3\}] \cdot \Pr[Y \in \{1, 2, 3\}] = (1/2)(1/2) = 1/4$, which is a slightly easier computation than counting up the 9 cases (and then arguing that each occurs with probability $(1/6)^2$, which requires knowing that X and Y are independent).

- Take the same X and Y, and let $Z = X + Y$. Now Z and X are not independent, because $\Pr[X = 1 \wedge Z = 12] = 0$, which is not equal to $\Pr[X = 1] \cdot \Pr[Z = 12] = (1/6)(1/36) = 1/216$.

- Place two radioactive sources on opposite sides of the Earth, and let X and Y be the number of radioactive decay events in each source during some 10 millisecond interval. Since the sources are 42 milliseconds away from each other at the speed of light, we can assert that either X and Y are independent, or the world doesn't behave the way the physicists think it does. This is an example of variables being independent because they are physically independent.

- Roll one six-sided die X, and let $Y = \lceil X/2 \rceil$ and $Z = X \bmod 2$. Then Y and Z are independent, even though they are generated using the same physical process.

12.2.3.2 Independence of many random variables

In general, if we have a collection of random variables X_i, we say that they are all independent if the joint distribution is the product of the marginal distributions, i.e., if $\Pr\left[\forall i : X_i \le x_i\right] = \prod_i \Pr\left[X_i \le x_i\right]$. It may be that a collection of random variables is not independent even though all subcollections are.

For example, let X and Y be fair coin-flips, and let $Z = X \oplus Y$. Then any two of X, Y, and Z are independent, but the three variables X, Y, and Z are not independent, because $\Pr\left[X = 0 \wedge Y = 0 \wedge Z = 0\right] = 1/4$ instead of $1/8$ as one would get by taking the product of the marginal probabilities.

Since we can compute the joint distribution from the marginal distributions for independent variables, we will often just specify the marginal distributions and declare that a collection of random variables are independent. This implicitly gives us an underlying probability space consisting of all sequences of values for the variables.

12.2.4 The expectation of a random variable

For a real-valued random variable X, its **expectation** $\operatorname{E}\left[X\right]$ (sometimes just $\operatorname{E} X$) is its average value, weighted by probability.[5] For discrete random variables, the expectation is defined by

$$\operatorname{E}\left[X\right] = \sum_x x \Pr\left[X = x\right].$$

For a continuous random variable with distribution function $F(x)$, the expectation is defined by

$$\operatorname{E}\left[X\right] = \int_{-\infty}^{\infty} x dF(x).$$

The integral here is a **Lebesgue-Stieltjes integral**, which generalizes the usual integral for continuous $F(x)$ by doing the right thing if $F(x)$ jumps due to some x that occurs with nonzero probability. We will avoid thinking about this by mostly worrying about expectations for discrete random variables.

Example (discrete variable) Let X be the number rolled with a fair six-sided die. Then $\operatorname{E}\left[X\right] = (1/6)(1 + 2 + 3 + 4 + 5 + 6) = 3\frac{1}{2}$.

[5] Technically, this will work for any values we can add and multiply by probabilities. So if X is actually a vector in $\mathbb{R}^3$ (for example), we can talk about the expectation of X, which in some sense will be the average position of the location given by X.

Example (unbounded discrete variable) Let X be a geometric random variable with parameter p; this means that $\Pr[X = k] = q^k p$, where as usual $q = 1 - p$. Then $\mathrm{E}[X] = \sum_{k=0}^{\infty} kq^k p = p\sum_{k=0}^{\infty} kq^k = p \cdot \frac{q}{(1-q)^2} = \frac{pq}{p^2} = \frac{q}{p} = \frac{1-p}{p} = \frac{1}{p} - 1$.

Expectation is a way to summarize the distribution of a random variable without giving all the details. If you take the average of many independent copies of a random variable, you will be likely to get a value close to the expectation. Expectations are also used in **decision theory** to compare different choices. For example, given a choice between a 50% chance of winning \$100 (expected value: \$50) and a 20% chance of winning \$1000 (expected value: \$200), a **rational decision maker** would take the second option. Whether ordinary human beings correspond to an economist's notion of a rational decision maker often depends on other details of the situation.

Terminology note: If you hear somebody say that some random variable X takes on the value z **on average**, this usually means that $\mathrm{E}[X] = z$.

12.2.4.1 Variables without expectations

If a random variable has a particularly annoying distribution, it may not have a finite expectation, even thought the variable itself takes on only finite values. This happens if the sum for the expectation diverges.

For example, suppose I start with a dollar, and double my money every time a fair coin-flip comes up heads. If the coin comes up tails, I keep whatever I have at that point. What is my expected wealth at the end of this process?

Let X be the number of times I get heads. Then X is just a geometric random variable with $p = 1/2$, so $\Pr[X = k] = (1 - (1/2))^k (1/2)^k = 2^{-k-1}$. My wealth is also a random variable: 2^X. If we try to compute $\mathrm{E}\left[2^X\right]$, we get

$$\begin{aligned}
\mathrm{E}[2^X] &= \sum_{k=0}^{\infty} 2^k \Pr[X = k] \\
&= \sum_{k=0}^{\infty} 2^k \cdot 2^{-k-1} \\
&= \sum_{k=0}^{\infty} \frac{1}{2},
\end{aligned}$$

which diverges. Typically we say that a random variable like this has no expected value, although sometimes you will see people writing $\mathrm{E}\left[2^X\right] = \infty$.

(For an even nastier case, consider what happens with $\mathrm{E}\left[(-2)^X\right]$.)

12.2.4.2 Expectation of a sum

The expectation operator is **linear**: this means that $\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y]$ and $\mathrm{E}[aX] = a\,\mathrm{E}[X]$ when a is a constant. This fact holds for all random variables X and Y, whether they are independent or not, and is not hard to prove for discrete probability spaces:

$$\begin{aligned}
\mathrm{E}[aX+Y] &= \sum_{x,y}(ax+y)\Pr[X=x \wedge Y=x] \\
&= a\sum_{x,y} x\Pr[X=x \wedge Y=x] + \sum_{x,y} y\Pr[X=x \wedge Y=x] \\
&= a\sum_{x} x\sum_{y}\Pr[X=x \wedge Y=x] + \sum_{y} y\sum_{x}\Pr[X=x \wedge Y=x] \\
&= a\sum_{x} x\Pr[X=x] + \sum_{y} y\Pr[Y=y] \\
&= a\,\mathrm{E}[X] + \mathrm{E}[Y].
\end{aligned}$$

Linearity of expectation makes computing many expectations easy. Example: Flip a fair coin n times, and let X be the number of heads. What is $\mathrm{E}[X]$? We can solve this problem by letting X_i be the indicator variable for the event "coin i came up heads." Then $X = \sum_{i=1}^n X_i$ and $\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathrm{E}[X_i] = \sum_{i=1}^n \frac{1}{2} = \frac{n}{2}$. In principle it is possible to calculate the same value from the distribution of X (this involves a lot of binomial coefficients), but linearity of expectation is much easier.

Example Choose a random permutation π, i.e., a random bijection from $\{1\ldots n\}$ to itself. What is the expected number of values i for which $\pi(i) = i$?

Let X_i be the indicator variable for the event that $\pi(i) = i$. Then we are looking for $\mathrm{E}[X_1 + X_2 + \ldots X_n] = \mathrm{E}[X_1] + \mathrm{E}[X_2] + \ldots \mathrm{E}[X_n]$. But $\mathrm{E}[X_i]$ is just $1/n$ for each i, so the sum is $n(1/n) = 1$. Calculating this by computing $\Pr\left[\sum_{i=1}^n X_i = x\right]$ first would be very painful.

12.2.4.3 Expectation of a product

For products of random variables, the situation is more complicated. Here the rule is that $\mathrm{E}[XY] = \mathrm{E}[X]\cdot\mathrm{E}[Y]$ if X and Y are independent. But if X and Y are not independent, the expectation of their product can't be computed without considering their joint distribution.

For example: Roll two dice and take their product. What value do we get on average? The product formula gives $\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y] = (7/2)^2 = (49/4) = 12\frac{1}{4}$. We could also calculate this directly by summing over all 36 cases, but it would take a while.

Alternatively, roll one die and multiply it by itself. Now what value do we get on average? Here we are no longer dealing with independent random variables, so we have to do it the hard way: $\mathrm{E}[X^2] = (1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2)/6 = 91/6 = 15\frac{1}{6}$. This is substantially higher than when the dice are uncorrelated. (Exercise: How can you rig the second die so it still comes up with each value $\frac{1}{6}$ of the time but *minimizes* $\mathrm{E}[XY]$?)

We can prove the product rule without too much trouble for discrete random variables. The easiest way is to start from the right-hand side.

$$\begin{aligned}
\mathrm{E}[X] \cdot \mathrm{E}[Y] &= \left(\sum_x x \Pr[X = x]\right)\left(\sum_y y \Pr[Y = y]\right) \\
&= \sum_{x,y} xy \Pr[X = x] \Pr[Y = y] \\
&= \sum_z z \left(\sum_{x,y,xy=z} \Pr[X = x] \Pr[Y = y]\right) \\
&= \sum_z z \left(\sum_{x,y,xy=z} \Pr[X = x \wedge Y = y]\right) \\
&= \sum_z z \Pr[XY = z] \\
&= \mathrm{E}[XY].
\end{aligned}$$

Here we use independence in going from $\Pr[X = x]\Pr[Y = y]$ to $\Pr[X = x \wedge Y = y]$ and use the union rule to convert the x, y sum into $\Pr[XY = z]$.

12.2.4.4 Conditional expectation

Like conditional probability, there is also a notion of **conditional expectation**. The simplest version of conditional expectation conditions on a single event A, is written $\mathrm{E}[X \mid A]$, and is defined for discrete random variables by

$$\mathrm{E}[X \mid A] = \sum_x x \Pr[X = x \mid A].$$

This is exactly the same as ordinary expectation except that the probabilities are now all conditioned on A.

To take a simple example, consider the expected value of a six-sided die conditioned on not rolling a 1. The conditional probability of getting 1 is now 0, and the conditional probability of each of the remaining 5 values is 1/5, so we get $(1/5)(2+3+4+5+6) = 4$.

Conditional expectation acts very much like regular expectation, so for example we have $\mathrm{E}\left[aX + bY \mid A\right] = a\,\mathrm{E}\left[X \mid A\right] + b\,\mathrm{E}\left[Y \mid A\right]$.

One of the most useful applications of conditional expectation is that it allows computing (unconditional) expectations by case analysis, using the fact that

$$\mathrm{E}\left[X\right] = \mathrm{E}\left[X \mid A\right]\Pr\left[A\right] + \mathrm{E}\left[X \mid \neg A\right]\Pr\left[\neg A\right].$$

or, more generally,

$$\mathrm{E}\left[X\right] = \sum_i \mathrm{E}\left[X \mid A_i\right]\Pr\left[A_i\right]$$

when $A_1, A_2, \ldots$ are disjoint events whose union is the entire probability space Ω. This is the expectation analog of the law of total probability.

Examples

- I have a 50% chance of reaching the top of Mt Everest, where Sir Edmund Hilary and Tenzing Norgay hid somewhere between 0 and 10 kilograms of gold (a random variable with uniform distribution). How much gold do I expect to bring home? Compute

  $$\begin{aligned}\mathrm{E}\left[X\right] &= \mathrm{E}\left[X \mid \text{reached the top}\right]\Pr\left[\text{reached the top}\right] + \mathrm{E}\left[X \mid \text{didn't}\right]\Pr\left[\text{didn't}\right]\\ &= 5\cdot 0.5 + 0\cdot 0.5 = 2.5.\end{aligned}$$

- Suppose I flip a coin that comes up heads with probability p until I get heads. How many times on average do I flip the coin?

  We'll let X be the number of coin flips. Conditioning on whether the coin comes up heads on the first flip gives $\mathrm{E}\left[X\right] = 1\cdot p + (1+\mathrm{E}\left[X'\right])\cdot(1-p)$, where X' is random variable counting the number of coin-flips needed to get heads ignoring the first coin-flip. But since X' has the same distribution as X, we get $\mathrm{E}\left[X\right] = p + (1-p)(1+\mathrm{E}\left[X\right]) or\, \mathrm{E}\left[X\right] = (p+(1-p))/p = 1/p$. So a fair coin must be flipped twice on average to get a head, which is about what we'd expect if we hadn't thought about it much.

- Suppose I have my experimental test subjects complete a task that gets scored on a scale of 0 to 100. I decide to test whether rewarding success is a better strategy for improving outcomes than punishing failure. So for any subject that scores high than 50, I give them a chocolate bar. For any subject that scores lower than 50, I give them an electric shock. (Students who score exactly 50 get nothing.) I then have them each perform the task a second time and measure the average change in their scores. What happens?

  Let's suppose that there is no effect whatsoever of my rewards and punishments, and that each test subject obtains each possible score with equal probability 1/101. Now let's calculate the average improvement for test subjects who initially score less than 50 or greater than 50. Call the outcome on the first test X and the outcome on the second test Y. The change in the score is then $Y - X$.

  In the first case, we are computing $\mathrm{E}[Y - X \mid X < 50]$. This is the same as $\mathrm{E}[Y \mid X < 50] - \mathrm{E}[X \mid X < 50] = \mathrm{E}[Y] - \mathrm{E}[X \mid X < 50] = 50 - 24.5 = +25.5$. So punishing failure produces a 25.5 point improvement on average.

  In the second case, we are computing $\mathrm{E}[Y - X \mid X > 50]$. This is the same as $\mathrm{E}[Y \mid X > 50] - \mathrm{E}[X \mid X > 50] = \mathrm{E}[Y] - \mathrm{E}[X \mid X > 50] = 50 - 75.5 = -25.5$. So rewarding success produces a 25.5 point decline on average.

  Clearly this suggests that we punish failure if we want improvements and reward success if we want backsliding. This is intuitively correct: punishing failure encourages our slacker test subjects to do better next time, while rewarding success just makes them lazy and complacent. But since the test outcomes don't depend on anything we are doing, we get exactly the same answer if we reward failure and punish success: in the former case, a $+25.5$ point average change, in the later a -25.5 point average change. This is also intuitively correct: rewarding failure makes our subjects like the test so that they will try to do better next time, while punishing success makes them feel that it isn't worth it. From this we learn that our intuitions[6] provide powerful tools for rationalizing almost any outcome in terms of the good or bad behavior of our test subjects. A more careful analysis shows that we performed the wrong comparison, and we are the victim of **regression to the mean**. This phenomenon was one of several now-notorious cognitive

[6]OK, my intuitions.

biases described in a famous paper by Tversky and Kahneman [TK74].

For a real-world example of how similar problems can arise in processing data, the United States Bureau of Labor Statistics defines a small business as any company with 500 or fewer employees. So if a company has 400 employees in 2007, 600 in 2008, and 400 in 2009, then we just saw a net creation of 200 new jobs by a small business in 2007, followed by the destruction of 200 jobs by a large business in 2008. It has been argued that this effect accounts for much of the observed fact that small businesses generate proportionally more new jobs than large ones, although the details are tricky [NWZ11].

12.2.4.5 Conditioning on a random variable

There is a more general notion of conditional expectation for random variables, where the conditioning is done on some other random variable Y. Unlike $\mathrm{E}[X \mid A]$, which is a constant, the expected value of X conditioned on Y, written $\mathrm{E}[X \mid Y]$, is itself a random variable: when $Y = y$, it takes on the value $\mathrm{E}[X \mid Y = y]$.

Here's a simple example. Let's compute $\mathrm{E}[X + Y \mid X]$ where X and Y are the values of independent six-sided dice. When $X = x$, $\mathrm{E}[\mathrm{E}[X + Y \mid X] \mid X = x] = \mathrm{E}[X + Y \mid X = x] = x + \mathrm{E}[Y] = x + 7/2$. For the full random variable we can write $\mathrm{E}[X + Y \mid X] = X + 7/2$.

Another way to get the result in the preceding example is to use some general facts about conditional expectation:

- $\mathrm{E}[aX + bY \mid Z] = a\,\mathrm{E}[X \mid Z] + b\,\mathrm{E}[Y \mid Z]$. This is the conditional-expectation version of linearity of expectation.

- $\mathrm{E}[X \mid X] = X$. This is immediate from the definition, since $\mathrm{E}[X]\, X = x = x$.

- If X and Y are independent, then $\mathrm{E}[Y \mid X] = \mathrm{E}[Y]$. The intuition is that knowing the value of X gives no information about Y, so $\mathrm{E}[Y]\, X = x = \mathrm{E}[Y]$ for any x in the range of X. (To do this formally requires using the fact that $\Pr[Y = y \mid X = x] = \frac{\Pr[Y=y \wedge X=x]}{\Pr[X=x]} = \frac{\Pr[Y=y]\Pr[X=x]}{\Pr[X=x]} = \Pr[Y = y]$, provided X and Y are independent and $\Pr[X = x] \neq 0$.)

- Also useful: $\mathrm{E}[\mathrm{E}[X \mid Y]] = \mathrm{E}[X]$. Averaging a second time removes all dependence on Y.

These in principle allow us to do very complicated calculations involving conditional expectation.

Some examples:

- Let X and Y be the values of independent six-sided dice. What is $\operatorname{E}[X \mid X+Y]$? Here we observe that $X+Y = \operatorname{E}[X+Y \mid X+Y] = \operatorname{E}[X \mid X+Y] + \operatorname{E}[Y \mid X+Y] = 2\operatorname{E}[X \mid X+Y]$ by symmetry. So $\operatorname{E}[X \mid X+Y] = (X+Y)/2$. This is pretty much what we'd expect: on average, half the total value is supplied by one of the dice. (It also works well for extreme cases like $X+Y=12$ or $X+Y=2$, giving a quick check on the formula.)

- What is $\operatorname{E}\left[(X+Y)^2 \mid X\right]$ when X and Y are independent? Compute $\operatorname{E}\left[(X+Y)^2 \mid X\right] = \operatorname{E}\left[X^2 \mid X\right] + 2\operatorname{E}[XY \mid X] + \operatorname{E}\left[Y^2 \mid X\right] = X^2 + 2X\operatorname{E}[Y] + \operatorname{E}\left[Y^2\right]$. For example, if X and Y are independent six-sided dice we have $\operatorname{E}\left[(X+Y)^2 \mid X\right] = X^2 + 7X + 91/6$, so if you are rolling the dice one at a time and the first one comes up 5, you can expect on average to get a squared total of $25 + 35 + 91/6 = 75\frac{1}{6}$. But if the first one comes up 1, you only get $1 + 7 + 91/6 = 23\frac{1}{6}$ on average.

12.2.5 Markov's inequality

Knowing the expectation of a random variable gives you some information about it, but different random variables may have the same expectation but very different behavior: consider, for example, the random variable X that is 0 with probability 1/2 and 1 with probability 1/2 and the random variable Y that is 1/2 with probability 1. In some cases we don't care about the average value of a variable so much as its likelihood of reaching some extreme value: for example, if my feet are encased in cement blocks at the beach, knowing that the average high tide is only 1 meter is not as important as knowing whether it ever gets above 2 meters. **Markov's inequality** lets us bound the probability of unusually high values of *non-negative* random variables as a function of their expectation. It says that, for any $a > 0$,

$$\Pr[X > a\operatorname{E}[X]] < 1/a.$$

This can be proved easily using conditional expectations. We have:

$$\operatorname{E}[X] = \operatorname{E}[X \mid X > a\operatorname{E}[X]]\Pr[X > a\operatorname{E}[X]] + \operatorname{E}[X] X \leq a\operatorname{E}[X]\Pr[X > a\operatorname{E}[X]].$$

Since X is non-negative, $\operatorname{E}[X \mid X \leq a\operatorname{E}[X]] \geq 0$, so subtracting out the last term on the right-hand side can only make it smaller. This gives:

$$\begin{aligned}\mathrm{E}[X] &\geq \mathrm{E}[X \mid X > a\,\mathrm{E}[X]]\Pr[X > a\,\mathrm{E}[X]] \\ &> a\,\mathrm{E}[X]\Pr[X > a\,\mathrm{E}[X]],\end{aligned}$$

and dividing both side by $a\,\mathrm{E}[X]$ gives the desired result.

Another version of Markov's inequality replaces $>$ with $\geq$:

$$\Pr[X \geq a\,\mathrm{E}[X]] \leq 1/a.$$

The proof is essentially the same.

12.2.5.1 Example

Suppose that that all you know about the high tide height X is that $\mathrm{E}[X] = 1$ meter and $X \geq 0$. What can we say about the probability that $X > 2$ meters? Using Markov's inequality, we get $\Pr[X > 2 \text{ meters}] = \Pr[X > 2\,\mathrm{E}[X]] < 1/2$.

12.2.5.2 Conditional Markov's inequality

There is, of course, a conditional version of Markov's inequality:

$$\Pr[X > a\,\mathrm{E}[X \mid A] \mid A] < 1/a.$$

This version doesn't get anywhere near as much use as the unconditioned version, but it may be worth remembering that it exists.

12.2.6 The variance of a random variable

Expectation tells you the average value of a random variable, but it doesn't tell you how far from the average the random variable typically gets: the random variables $X = 0$ and $Y = \pm 1,000,000,000,000$ with equal probability both have expectation 0, though their distributions are very different. Though it is impossible to summarize everything about the spread of a distribution in a single number, a useful approximation for many purposes is the **variance** $\mathrm{Var}[X]$ of a random variable X, which is defined as the expected square of the deviation from the expectation, or $\mathrm{E}\left[(X - \mathrm{E}[X])^2\right]$.

Example Let X be 0 or 1 with equal probability. Then $\mathrm{E}[X] = 1/2$, and $(X - \mathrm{E}[X])^2$ is always $1/4$. So $\mathrm{Var}[X] = 1/4$.

Example Let X be the value of a fair six-sided die. Then $\mathrm{E}[X] = 7/2$, and $\mathrm{E}\left[(X - \mathrm{E}[X])^2\right] = \frac{1}{6}\left((1-7/2)^2 + (2-7/2)^2 + (3-7/2)^2 + \cdots + (6-7/2)^2\right) = 35/12$.

Computing variance directly from the definition can be tedious. Often it is easier to compute it from $\mathrm{E}\left[X^2\right]$ and $\mathrm{E}[X]$:

$$\begin{aligned}
\operatorname{Var}[X] &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\
&= \mathrm{E}\left[X^2 - 2X\,\mathrm{E}[X] + (\mathrm{E}[X])^2\right] \\
&= \mathrm{E}\left[X^2\right] - 2\,\mathrm{E}[X]\,\mathrm{E}[X] + (\mathrm{E}[X])^2 \\
&= \mathrm{E}\left[X^2\right] - (\mathrm{E}[X])^2 .
\end{aligned}$$

The second-to-last step uses linearity of expectation and the fact that $\mathrm{E}[X]$ is a constant.

Example For X being 0 or 1 with equal probability, we have $\mathrm{E}\left[X^2\right] = 1/2$ and $(\mathrm{E}[X])^2 = 1/4$, so $\operatorname{Var}[X] = 1/4$.

Example Let's try the six-sided die again, except this time we'll use an n-sided die. We have

$$\begin{aligned}
\operatorname{Var}[X] &= \mathrm{E}\left[X^2\right] - (\mathrm{E}[X])^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} i^2 - \left(\frac{n+1}{2}\right)^2 \\
&= \frac{1}{n} \cdot \frac{n(n+1)(2n+1)}{6} - \frac{(n+1)^2}{4} \\
&= \frac{(n+1)(2n+1)}{6} - \frac{(n+1)^2}{4}.
\end{aligned}$$

When $n = 6$, this gives $\frac{7\cdot 13}{6} - \frac{49}{4} = \frac{35}{12}$. (Ok, maybe it isn't always easier).

12.2.6.1 Multiplication by constants

Suppose we are asked to compute the variance of cX, where c is a constant. We have

$$\begin{aligned}
\operatorname{Var}[cX] &= \mathrm{E}\left[(cX)^2\right] - \mathrm{E}[cX]^2 \\
&= c^2\,\mathrm{E}\left[X^2\right] - (c\,\mathrm{E}[X])^2 \\
&= c^2 \operatorname{Var}[X] .
\end{aligned}$$

So, for example, if X is 0 or 2 with equal probability, $\operatorname{Var}[X] = 4\cdot(1/4) = 1$. This is exactly what we expect given that $X - \operatorname{E}[X]$ is always ± 1.

Another consequence is that $\operatorname{Var}[-X] = (-1)^2 \operatorname{Var}[X] = \operatorname{Var}[X]$. So variance is not affected by negation.

12.2.6.2 The variance of a sum

What is $\operatorname{Var}[X+Y]$? Write

$$\begin{aligned}
\operatorname{Var}[X+Y] &= \operatorname{E}\left[(X+Y)^2\right] - (\operatorname{E}[X+Y])^2 \\
&= \operatorname{E}\left[X^2\right] + 2\operatorname{E}[XY] + \operatorname{E}\left[Y^2\right] - (\operatorname{E}[X])^2 - 2\operatorname{E}[X]\cdot\operatorname{E}[Y] - (\operatorname{E}[Y])^2 \\
&= (\operatorname{E}\left[X^2\right] - (\operatorname{E}[X])^2) + (\operatorname{E}\left[Y^2\right] - (\operatorname{E}[Y])^2) + 2(\operatorname{E}[XY] - \operatorname{E}[X]\cdot\operatorname{E}[Y]) \\
&= \operatorname{Var}[X] + \operatorname{Var}[Y] + 2(\operatorname{E}[XY] - \operatorname{E}[X]\cdot\operatorname{E}[Y]).
\end{aligned}$$

The quantity $\operatorname{E}[XY] - \operatorname{E}[X]\operatorname{E}[Y]$ is called the **covariance** of X and Y and is written $\operatorname{Cov}(X,Y)$. So we have just shown that

$$\operatorname{Var}[X+Y] = \operatorname{Var}[X] + \operatorname{Var}[Y] + 2\operatorname{Cov}(X,Y).$$

When $\operatorname{Cov}(X,Y) = 0$, or equivalently when $\operatorname{E}[XY] = \operatorname{E}[X]\operatorname{E}[Y]$, X and Y are said to be **uncorrelated** and their variances add. This occurs when X and Y are independent, but may also occur without X and Y being independent.

For larger sums the corresponding formula is

$$\operatorname{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \operatorname{Var}[X_i] + \sum_{i\neq j} \operatorname{Cov}(X,Y).$$

This simplifies to $\operatorname{Var}[\sum X_i] = \sum \operatorname{Var}[X_i]$ when the X_i are **pairwise independent**, so that each pair of distinct X_i and X_j are independent. Pairwise independence implied by independence (but is not equivalent to it), so this also works for fully independent random variables.

For example, we can use the simplified formula to compute the variance of the number of heads in n independent fair coin-flips. Let X_i be the indicator variable for the event that the i-th flip comes up heads and let X be the sum of the X_i. We have already seen that $\operatorname{Var}[X_i] = 1/4$, so $\operatorname{Var}[X] = n\operatorname{Var}[X_i] = n/4$.

Similarly, if c is a constant, then we can compute $\operatorname{Var}[X+c] = \operatorname{Var}[X] + \operatorname{Var}[c] = \operatorname{Var}[X]$, since (1) $\operatorname{E}[cX] = c\operatorname{E}[X] = \operatorname{E}[c]\operatorname{E}[X]$ means that c (considered as a random variable) and X are uncorrelated, and (2) $\operatorname{Var}[a] =$

$\mathrm{E}\left[(c-\mathrm{E}[c])^2\right] = \mathrm{E}[0] = 0$. So shifting a random variable up or down doesn't change its variance.

12.2.6.3 Chebyshev's inequality

Variance is an expectation, so we can use Markov's inequality on it. The result is **Chebyshev's inequality**, which like Markov's inequality comes in two versions:

$$\begin{aligned}
\Pr[|X-\mathrm{E}[X]| \geq r] &\leq \frac{\mathrm{Var}[X]}{r^2}, \\
\Pr[|X-\mathrm{E}[X]| > r] &< \frac{\mathrm{Var}[X]}{r^2}.
\end{aligned}$$

Proof. We'll do the first version. The event $|X-\mathrm{E}[X]| \geq r$ is the same as the event $(X-\mathrm{E}[X])^2 \geq r^2$. By Markov's inequality, the probability that this occurs is at most $\frac{\mathrm{E}\left[(X-\mathrm{E}[X])^2\right]}{r^2} = \frac{\mathrm{Var}[X]}{r^2}$. ∎

Application: showing that a random variable is close to its expectation This is the usual statistical application.

Example Flip a fair coin n times, and let X be the number of heads. What is the probability that $|X-n/2| > r$? Recall that $\mathrm{Var}[X] = n/4$, so $\Pr[|X-n/2| > r] < (n/4)/r^2 = n/(4r^2)$. So, for example, the chances of deviating from the average by more than 1000 after 1000000 coin-flips is less than 1/4.

Example Out of n voters in Saskaloosa County, m plan to vote for Smith for County Dogcatcher. A polling firm samples k voters (with replacement) and asks them who they plan to vote for. Suppose that $m < n/2$; compute a bound on the probability that the polling firm incorrectly polls a majority for Smith.

Solution: Let X_i be the indicator variable for a Smith vote when the i-th voter is polled and let $X = \sum X_i$ be the total number of pollees who say they will vote for Smith. Let $p = \mathrm{E}[X_i] = m/n$. Then $\mathrm{Var}[X_i] = p - p^2$, $\mathrm{E}[X] = kp$, and $\mathrm{Var}[X] = k(p-p^2)$. To get a majority in the poll, we need $X > k/2$ or $X - \mathrm{E}[X] > k/2 - kp$. Using

Chebyshev's inequality, this event occurs with probability at most

$$\begin{aligned}\frac{\operatorname{Var}[X]}{(k/2-kp)^2} &= \frac{k(p-p^2)}{(k/2-kp)^2} \\ &= \frac{1}{k}\cdot\frac{p-p^2}{(1/2-p)^2}.\end{aligned}$$

Note that the bound decreases as k grows and (for fixed p) does not depend on n.

In practice, statisticians will use a stronger result called the **central limit theorem**, which describes the shape of the distribution of the sum of many independent random variables much more accurately than the bound from Chebyshev's inequality. Designers of randomized algorithms are more likely to use **Chernoff bounds**.

Application: lower bounds on random variables Unlike Markov's inequality, which can only show that a random variable can't be too big too often, Chebyshev's inequality can be used to show that a random variable can't be too small, by showing first that its expectation is high and then that its variance is low. For example, suppose that each of the 10^{30} oxygen molecules in the room is close enough to your mouth to inhale with pairwise independent probability 10^{-4} (it's a big room). Then the expected number of oxygen molecules near your mouth is a healthy $10^{30}\cdot 10^{-4} = 10^{26}$. What is the probability that all 10^{26} of them escape your grasp?

Let X_i be the indicator variable for the event that the i-th molecule is close enough to inhale. We've effectively already used the fact that $\operatorname{E}[X_i] = 10^{-4}$. To use Chebyshev's inequality, we also need $\operatorname{Var}[X_i] = \operatorname{E}[X_i^2] - \operatorname{E}[X_i]^2 = 10^{-4} - 10^{-8} \approx 10^{-4}$. So the total variance is about $10^{30}\cdot 10^{-4} = 10^{26}$ and Chebyshev's inequality says we have $\Pr\left[|X - \operatorname{E}[X]| \ge 10^{26}\right] \le 10^{26}/(10^{26})^2 = 10^{-26}$. So death by failure of statistical mechanics is unlikely (and the real probability is much much smaller).

But wait! Even a mere 90% drop in O_2 levels is going to be enough to cause problems. What is the probability that this happens? Again we can calculate $\Pr[90\%\text{ drop}] \le \Pr\left[|X - \operatorname{E}[X]| \ge 0.9\cdot 10^{26}\right] \le 10^{26}/(0.9\cdot 10^{26})^2 \approx 1.23\cdot 10^{-26}$. So even temporary asphyxiation by statistical mechanics is not something to worry about.

12.2.7 Probability generating functions

For a discrete random variable X taking on only values in $\mathbb{N}$, we can express its distribution using a **probability generating function** or **pgf**:

$$F(z) = \sum_{n=0}^{\infty} \Pr[X = n] z^n.$$

These are essentially standard-issue generating functions (see §11.3) with the additional requirement that all coefficients are non-negative and $F(1) = 1$.

A trivial example is the pgf for a Bernoulli random variable (1 with probability p, 0 with probability $q = 1 - p$). Here the pgf is just $q + pz$.

A more complicated example is the pgf for a geometric random variable. Now we have $\sum_{n=0}^{\infty} q^n p z^n = p \sum_{n=0}^{\infty} (qz)^n = \frac{p}{1-qz}$.

12.2.7.1 Sums

A very useful property of pgf's is that the pgf of a sum of independent random variables is just the product of the pgf's of the individual random variables. The reason for this is essentially the same as for ordinary generating functions: when we multiply together two terms $(\Pr[X = n] z^n)(\Pr[Y = m] z^m)$, we get $\Pr[X = n \wedge Y = m] z^{n+m}$, and the sum over all the different ways of decomposing $n + m$ gives all the different ways to get this sum.

So, for example, the pgf of a binomial random variable equal to the sum of n independent Bernoulli random variables is $(q + pz)^n$ (hence the name "binomial").

12.2.7.2 Expectation and variance

One nice thing about pgf's is that the can be used to quickly compute expectation and variance. For expectation, we have

$$F'(z) = \sum_{n=0}^{\infty} n \Pr[X = n] z^{n-1}.$$

So

$$\begin{aligned} F'(1) &= \sum_{n=0}^{\infty} n \Pr[X = n] \\ &= \mathrm{E}[X]. \end{aligned}$$

If we take the second derivative, we get

$$F''(z) = \sum_{n=0}^{\infty} n(n-1) \Pr[X = n] z^{n-1}$$

or

$$\begin{aligned} F''(1) &= \sum_{n=0}^{\infty} n(n-1) \Pr[X = n] \\ &= \mathrm{E}[X(X-1)] \\ &== \mathrm{E}\left[X^2\right] - \mathrm{E}[X]. \end{aligned}$$

So we can recover $\mathrm{E}\left[X^2\right]$ as $F''(1) + F'(1)$ and get $\mathrm{Var}[X]$ as $F''(1) + F'(1) - (F'(1))^2$.

Example If X is a Bernoulli random variable with pgf $F = (q + pz)$, then $F' = p$ and $F'' = 0$, giving $\mathrm{E}[X] = F'(1) = p$ and $\mathrm{Var}[X] = F''(1) + F'(1) - (F'(1))^2 = 0 + p - p^2 = p(1-p) = pq$.

Example If X is a binomial random variable with pgf $F = (q + pz)^n$, then $F' = n(q + pz)^{n-1}p$ and $F'' = n(n-1)(q + pz)^{n-2}p^2$, giving $\mathrm{E}[X] = F'(1) = np$ and $\mathrm{Var}[X] = F''(1) + F'(1) - (F'(1))^2 = n(n-1)p^2 + np - n^2p^2 = np - np^2 = npq$. These values would, of course, be a lot faster to compute using the formulas for sums of independent random variables, but it's nice to see that they work.

Example If X is a geometric random variable with pgf $p/(1 - qz)$, then $F' = pq/(1 - qz)^2$ and $F'' = 2pq^2/(1 - qz)^3$. So $\mathrm{E}[X] = F'(1) = pq/(1-q)^2 = pq/p^2 = q/p$, and $\mathrm{Var}[X] = F''(1) + F'(1) - (F'(1))^2 = 2pq^2/(1-q)^3 + q/p - q^2/p^2 = 2q^2/p^2 + q/p - q^2/p^2 = q^2/p^2 + q/p$. The variance would probably be a pain to calculate by hand.

Example Let X be a Poisson random variable with rate λ. We claimed earlier that a Poisson random variable is the limit of a sequence of binomial random variables where $p = \lambda/n$ and n goes to infinity, so (cheating quite a bit) we expect that X's pgf $F = \lim_{n\to\infty}((1-\lambda/n) + (\lambda/n)z)^n = (1 + (-\lambda + \lambda z)/n)^n = \exp(-\lambda + \lambda z) = \exp(-\lambda) \sum \lambda^n z^n/n!$. We can check that the total probability $F(1) = \exp(-\lambda + \lambda) = e^0 = 1$, that the expectation $F'(1) = \lambda \exp(-\lambda + \lambda) = \lambda$, and that the variance $F''(1) + F'(1) - (F'(1))^2 = \lambda^2 \exp(-\lambda + \lambda) + \lambda - \lambda^2 = \lambda$. These last two quantities are what we'd expect if we calculated the expectation and the variance directly as the limit of taking n Bernoulli random variables with expectation λ/n and variance $(\lambda/n)(1 - \lambda/n)$ each.

12.2.8 Summary: effects of operations on expectation and variance of random variables

Operation	Effect on expectation
Multiplication by a constant	$\mathrm{E}[aX] = a\,\mathrm{E}[X]$
Addition	$\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y]$. Does not require independence.
Product	$\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y] + \mathrm{Cov}(X,Y)$. (Or just $\mathrm{E}[X]\,\mathrm{E}[Y]$ if X and Y a

12.2.9 The general case

So far we have only considered discrete random variables, which avoids a lot of nasty technical issues. In general, a **random variable** on a probability space $(\Omega, \mathcal{F}, P)$ is a function whose domain is Ω that satisfies some extra conditions on its values that make interesting events involving the random variable elements of $\mathcal{F}$. Typically the codomain will be the reals or the integers, although any set is possible. Random variables are generally written as capital letters with their arguments suppressed: rather than writing $X(\omega)$, where $\omega \in \Omega$, we write just X.

A technical condition on random variables is that the inverse image of any measurable subset of the codomain must be in $\mathcal{F}$—in simple terms, if you can't nail down ω exactly, being able to tell which element of $\mathcal{F}$ you land in should be enough to determine the value of $X(\omega)$. For a discrete random variables, this just means that $X^{-1}(x) \in \mathcal{F}$ for each possible value x. For real-valued random variables, the requirement is that the event $[X \le x]$ is in $\mathcal{F}$ for any fixed x. In each case we say that X is **measurable** with respect to $\mathcal{F}$ (or just "measurable $\mathcal{F}$").[7] Usually we will not worry about this issue too much, but it may come up if we are varying $\mathcal{F}$ to represent different amounts of information available to different observers (e.g., if X and Y are the values of two dice, X is measurable to somebody who can see both dice but not to somebody who can only see the sum of the dice).

The **distribution function** of a real-valued random variable describes the probability that it takes on each of its possible values; it is specified by giving a function $F(x) = \Pr[X \le x]$. The reason for using $\Pr[X \le x]$ instead of $\Pr[X = x]$ is that it allows specifying continuous random variables such as a random variable that is uniform in the range $[0, 1]$; this random variable has a distribution function given by $F(x) = x$ when $0 \le x \le 1$, $F(x) = 0$ for $x < 0$, and $F(x) = 1$ for $x > 1$.

[7]The detail we are sweeping under the rug here is what makes a subset of the codomain measurable. The essential idea is that we also have a σ-algebra $\mathcal{F}'$ on the codomain, and elements of this codomain σ-algebra are the measurable subsets. The rules for simple random variables and real-valued random variables come from default choices of σ-algebra.

For discrete random variables the distribution function will have discontinuous jumps at each possible value of the variable. For example, the distribution function of a variable X that is 0 or 1 with equal probability is $F(x) = 0$ for $x < 0$, $1/2$ for $0 \leq x < 1$, and 1 for $x \geq 1$.

Knowing the distribution of a random variable tells you what that variable might do by itself, but doesn't tell you how it interacts with other random variables. For example, if X is 0 or 1 with equal probability then X and $1-X$ both have the same distribution, but they are connected in a way that is not true for X and some independent variable Y with the same distribution. For multiple variables, a **joint distribution** gives the probability that each variable takes on a particular value; for example, if X and Y are two independent uniform samples from the range $[0,1]$, their distribution function $F(x,y) = \Pr[X \leq x \wedge Y \leq y] = xy$ (when $0 \leq x, y \leq 1$). If instead $Y = 1 - X$, we get the distribution function $F(x,y) = \Pr[X \leq x \wedge Y \leq y]$ equal to x when $y \geq 1 - x$ and 0 when $y < 1 - x$ (assuming $0 \leq x, y \leq 1$).

We've seen that for discrete random variables, it is more useful to look at the **probability mass function** $f(x) = \Pr[X = x]$. We can always recover the probability distribution function from the probability mass function if the latter sums to 1.

12.2.9.1 Densities

If a real-valued random variable is **continuous** in the sense of having a distribution function with no jumps (which means that it has probability 0 of landing on any particular value), we may be able to describe its distribution by giving a **density** instead. The density is the derivative of the distribution function. We can also think of it as a probability at each point defined in the limit, by taking smaller and smaller regions around the point and dividing the probability of landing in the region by the size of the region.

For example, the density of a uniform $[0,1]$ random variable is $f(x) = 1$ for x in $[0,1]$, and $f(x) = 0$ otherwise. For a uniform $[0,2]$ random variable, we get a density of $\frac{1}{2}$ throughout the $[0,2]$ interval. The density always integrates to 1.

Some distributions are easier to describe using densities than using distribution functions. The **normal distribution**, which is of central importance in statistics, has density

$$\frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Its distribution function is the integral of this quantity, which has no closed-form expression.

Joint densities also exist. The **joint density** of a pair of random variables with joint distribution function $F(x, y)$ is given by the partial derivative $f(x, y) = \frac{\partial^2}{\partial x \partial y} F(x, y)$. The intuition here again is that we are approximating the (zero) probability at a point by taking the probability of a small region around the point and dividing by the area of the region.

12.2.9.2 Independence

Independence is the same as for discrete random variables: Two random variables X and Y are **independent** if any pair of events of the form $X \in A$, $Y \in B$ are independent. For real-valued random variables it is enough to show that their joint distribution $F(x, y)$ is equal to the product of their individual distributions $F_X(x)F_Y(y)$. For real-valued random variables with densities, showing the densities multiply also works. Both methods generalize in the obvious way to sets of three or more random variables.

12.2.9.3 Expectation

If a continuous random variable has a density $f(x)$, the formula for its expectation is

$$\mathrm{E}\left[X\right] = \int x f(x) dx.$$

For example, let X be a uniform random variable in the range $[a, b]$. Then $f(x) = \frac{1}{b-a}$ when $a \le x \le b$ and 0 otherwise, giving

$$\begin{aligned}
\mathrm{E}\left[X\right] &= \int_a^b x \frac{1}{b-a} dx \\
&= \left. \frac{x^2}{2(b-a)} \right|_{x=a}^{b} \\
&= \frac{b^2 - a^2}{2(b-a)} \\
&= \frac{a+b}{2}.
\end{aligned}$$

For continuous random variables without densities, we land in a rather swampy end of integration theory. We will not talk about this case if we can help it. But in each case the expectation depends only on the distribution of X and not on its relationship to other random variables.

Chapter 13

Linear algebra

Linear algebra is the branch of mathematics that studies vector spaces and linear transformations between them.

13.1 Vectors and vector spaces

Let's start with vectors. In the simplest form, a **vector** consists of a sequence of n values from some field (see §4.1); for most purposes, this field will be $\mathbb{R}$. The number of values (called **coordinates**) in a vector is the **dimension** of the vector. The set of all vectors over a given field of a given dimension (e.g., $\mathbb{R}^n$) forms a **vector space**, which has a more general definition that we will give later.

So the idea is that a vector represents a point in an n-dimensional space represented by its coordinates in some coordinate system. For example, if we imagine the Earth is flat, we can represent positions on the surface of the Earth as a latitude and longitude, with the point $\langle 0, 0\rangle$ representing the origin of the system at the intersection between the equator (all points of the form $\langle 0, x\rangle$ and the prime meridian (all points of the form $\langle x, 0\rangle$. In this system, the location of Arthur K. Watson Hall (AKW) would be $\langle 41.31337, -72.92508\rangle$, and the location of LC 317 would be $\langle 41.30854, -72.92967\rangle$. These are both offsets (measured in degrees) from the origin point $\langle 0, 0\rangle$.

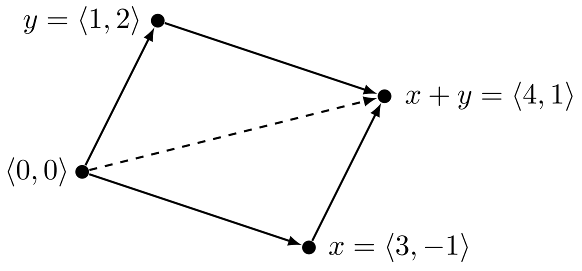

$y = \langle 1, 2\rangle$
$x + y = \langle 4, 1\rangle$
$\langle 0, 0\rangle$
$x = \langle 3, -1\rangle$

Figure 13.1: Geometric interpretation of vector addition

13.1.1 Relative positions and vector addition

What makes this a little confusing is that we will often use vectors to represent relative positions as well.[1] So if we ask the question "where do I have to go to get to LC 317 from AKW?", one answer is to travel -0.00483 degrees in latitude and -0.00459 degrees in longitude, or, in vector terms, to follow the relative vector $\langle -0.00483, -0.00459\rangle$. This works because we define vector addition coordinatewise: given two vectors x and y, their sum $x + y$ is defined by $(x + y)_i = x_i + y_i$ for each index i. In geometric terms, this has the effect of constructing a compound vector by laying vectors x and y end-to-end and drawing a new vector from the start of x to the end of y (see Figure 13.1.)

The correspondence between vectors as absolute positions and vectors as relative positions comes from fixing an origin 0. If we want to specify an absolute position (like the location of AKW), we give its position relative to the origin (the intersection of the equator and the prime meridian). Similarly, the location of LC 317 can be specified by giving its position relative to the origin, which we can compute by first going to AKW ($\langle 41.31337, -72.92508\rangle$), and then adding the offset of LC 317 from AWK ($\langle -0.00483, -0.00459\rangle$) to this vector to get the offset directly from the origin ($\langle 41.30854, -72.92967\rangle$).

More generally, we can add together as many vectors as we want, by adding them coordinate-by-coordinate.

This can be used to reduce the complexity of pirate-treasure instructions:

[1]A further complication that we will sidestep completely is that physicists will often use "vector" to mean both an absolute position and an offset from it—sort of like an edge in a graph—requiring n coordinates to represent the starting point of the vector and another n coordinates to represent the ending point. These vectors really do look like arrows at a particular position in space. Our vectors will be simpler, and always start at the origin.

1. Yargh! Start at the olde hollow tree on Dead Man's Isle, *if ye dare.*
2. Walk 10 paces north.
3. Walk 5 paces east.
4. Walk 20 paces south.
5. Walk $6\sqrt{2}$ paces northwest.
6. Dig 8 paces down.
7. Climb back up 6 paces. There be the treasure, argh!

In vector notation, this becomes:

1. $\langle 0,0,0\rangle$
2. $+\ \langle 10,0,0\rangle$
3. $+\ \langle 0,5,0\rangle$
4. $+\ \langle -20,0,0\rangle$
5. $+\ \langle 6,-6,0\rangle$
6. $+\ \langle 0,0,-8\rangle$
7. $+\ \langle 0,0,6\rangle$

which sums to $\langle -4,-1,-2\rangle$. So we can make our life easier by walking 4 paces south, 1 pace west, and digging only 2 paces down.

13.1.2 Scaling

Vectors may also be **scaled** by multiplying each of their coordinates by an element of the base field, called a **scalar**. For example, if $x = \langle -4,-1,-2\rangle$ is the number of paces north, east, and down from the olde hollow tree to the treasure in the previous example, we can scale x by 2 to get the number of paces for Short-Legged Pete. This gives

$$2\,\langle -4,-1,-2\rangle = \langle -8,-2,-4\rangle\,.$$

13.2 Abstract vector spaces

So far we have looked at vectors in $\mathbb{R}^n$, which can be added together (by adding their coordinates) and scaled (by multiplying by an element of $\mathbb{R}$). In the abstract, a **vector space** is any set that supports these operations, consistent with certain axioms that make it behave like we expect from our experience with $\mathbb{R}^n$.

Formally, a vector space consists of **vectors**, which form an additive Abelian group,[2] and **scalars**, which can be used to scale vectors through **scalar multiplication**. The scalars are assumed to be a field (see §4.1); the reals $\mathbb{R}$ and complex numbers $\mathbb{C}$ are typical choices.

Vector addition and scalar multiplication are related by a distributive law and some consistency requirements. When a and b are scalars, and x and y are vectors, we have

$$\begin{aligned} a(x+y) &= ax+ay \\ (a+b)x &= ax+bc \\ 0x &= 0 \\ 1x &= x \\ a(bx) &= (ab)x \end{aligned}$$

Note that in $0x = 0$, the 0 on the left-hand side is a scalar while the 0 on the right-hand side is a vector.

To avoid confusing between scalars, some writers will mark vectors using boldface ($\mathbf{x}$) or with a superimposed arrow ($\vec{x}$). Both are annoying enough to type that we will not use either convention.

It's not hard to see that the $\mathbb{R}^n$ vectors we defined earlier satisfy this definition. But there are other examples of vector spaces:

- The complex numbers $\mathbb{C}$ form a two-dimensional vector space over the real numbers $\mathbb{R}$. This is because we can represent any complex number $a+bi$ as a two-dimensional vector $\langle a, b\rangle$, and ordinary complex-number addition and multiplication by reals behaves just like vector addition and scalar multiplication in $\mathbb{R}^2$: $(a+bi)+(c+di) = (a+b)+(c+d)i$, $r(a+bi) = (ra)+(rb)i$.

- If F is a field and S is any set, then the set F^S of functions $f : S \to F$ is a vector space, where the scalars are the elements of F, $f+g$ is defined

[2] This means that there is an addition operation for vectors that is commutative ($x+y = y+x$), associative ($x+(y+z) = (x+y)+z$), and has an identity element 0 ($0+x = x+0 = x$) and inverses $-x$ ($x + (-x) = 0$).

by $(f+g)(x) = f(x) + g(x)$, and af is defined by $(af)(x) = a \cdot f(x)$. Our usual finite-dimensional real vector spaces are special cases of this, where $S = \{1, \dots, n\}$ and $F = \mathbb{R}$.[3]

- The set of all real-valued random variables on a probability space forms a vector space, with $X + Y$ and aX defined in the usual way. For discrete probability spaces, this is another special case of F^S, since each random variable is really an element of $\mathbb{R}^\Omega$. For general probability spaces, there are some technical conditions with measurability that mean that we don't get all of $\mathbb{R}^\Omega$, but it's still the case that $X+Y$ and aX are random variables whenever X and Y are, giving us a vector space.

- In some application areas, it's common to consider restricted classes of functions with some nice properties; for example, we might look at functions from $\mathbb{R}$ to $\mathbb{R}$ that are continuous or differentiable, or infinite sequences $\mathbb{N} \to \mathbb{R}$ that converge to a finite sum. These restricted classes of functions are all vector spaces as long as $f + g$ and af are in the class whenever f and g are.

13.3 Matrices

We've seen that a **sequence** $a_1, a_2, \dots, a_n$ is really just a function from some index set ($\{1 \dots n\}$ in this case) to some codomain, where $a_i = a(i)$ for each i. What if we have two index sets? Then we have a two-dimensional structure:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}$$

where $A_{ij} = a(i,j)$, and the domain of the function is just the cross-product of the two index sets. Such a structure is called a **matrix**. The values A_{ij} are called the **elements** or **entries** of the matrix. A sequence of elements with the same first index is called a **row** of the matrix; similarly, a sequence of elements with the same second index is called a **column**. The **dimension** of the matrix specifies the number of rows and the number of columns: the matrix above has dimension $(3,2)$, or, less formally, it is a 3×2 matrix.[4] A matrix is **square** if it has the same number of rows and columns.

[3]Note that we adopt the FORTRAN convention of indexing from 1. This is pretty much universal in linear algebra.

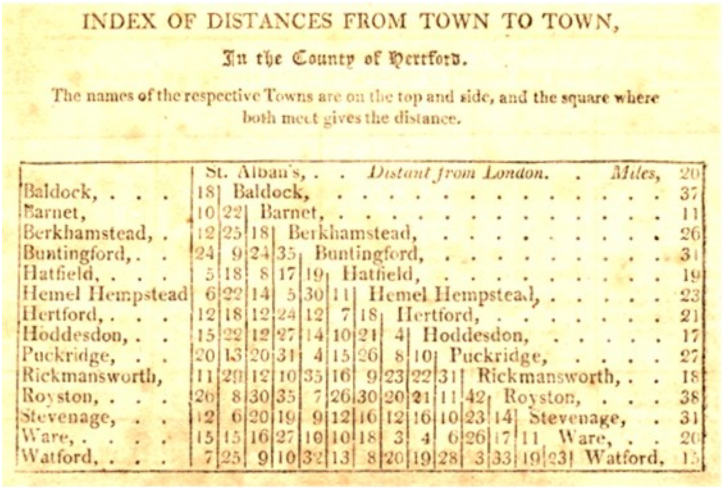
[4]The convention for both indices and dimension is that rows come before columns.

Note: The convention in matrix indices is to count from 1 rather than 0. In programming language terms, matrices are written in FORTRAN.

By convention, variables representing matrices are usually written with capital letters. This is to distinguish them from both scalars and vectors.

13.3.1 Interpretation

We can use a matrix any time we want to depict a function of two arguments (over small finite sets if we want it to fit on one page). A typical example (that predates the formal notion of a matrix by centuries) is a table of distances between cities or towns, such as this example from 1807:[5]



INDEX OF DISTANCES FROM TOWN TO TOWN,

In the County of Hertford.

The names of the respective Towns are on the top and side, and the square where both meet gives the distance.

	St. Alban's	Baldock	Barnet	Berkhamstead	Buntingford	Hatfield	Hemel Hempstead	Hertford	Hoddesdon	Puckridge	Rickmansworth	Royston	Stevenage	Ware	*Distant from London.* *Miles,*
St. Alban's,															20
Baldock,	18														37
Barnet,	10	22													11
Berkhamstead,	12	25	18												26
Buntingford,	24	9	24	35											31
Hatfield,	5	18	8	17	19										19
Hemel Hempstead	6	22	14	5	30	11									23
Hertford,	12	18	12	24	12	7	18								21
Hoddesdon,	15	22	12	27	14	10	21	4							17
Puckridge,	20	13	20	31	4	15	26	8	10						27
Rickmansworth,	11	29	12	10	35	16	9	23	22	31					18
Royston,	26	8	30	35	7	26	30	20	21	11	42				38
Stevenage,	12	6	20	19	9	12	16	12	16	10	23	14			31
Ware,	15	15	16	27	10	10	18	3	4	6	26	17	11		20
Watford,	7	25	9	10	32	13	8	20	19	28	3	33	19	23	15

Because distance matrices are *symmetric* (see below), usually only half of the matrix is actually printed.

Another example would be a matrix of counts. Suppose we have a set of destinations D and a set of origins O. For each pair $(i, j) \in D \times O$, let C_{ij} be the number of different ways to travel from j to i. For example, let origin 1 be Bass Library, origin 2 be AKW, and let destinations 1, 2, and 3 be Bass, AKW, and SML. Then there is 1 way to travel between Bass and AKW (walk), 1 way to travel from AKW to SML (walk), and 2 ways to travel from Bass to SML (walk above-ground or below-ground). If we assume that we are not allowed to stay put, there are 0 ways to go from

[5]The original image is taken from `http://www.hertfordshire-genealogy.co.uk/data/books/books-3/book-0370-cooke-1807.htm`. As an exact reproduction of a public domain document, this image is not subject to copyright in the United States.

Bass to Bass or AKW to AKW, giving the matrix

$$C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Wherever we have counts, we can also have probabilities. Suppose we have a particle that moves between positions $1 \dots n$ by flipping a coin, and moving up with probability $\frac{1}{2}$ and down with probability $\frac{1}{2}$ (staying put if it would otherwise move past the endpoints). We can describe this process by a **transition matrix** P whose entry P_{ij} gives the probability of moving to i starting from j. For example, for $n = 4$, the transition matrix is

$$P = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

Finally, the most common use of matrices in linear algebra is to represent the coefficients of a **linear transformation**, which we will describe later.

13.3.2 Operations on matrices

13.3.2.1 Transpose of a matrix

The **transpose** of a matrix A, written $A^\top$ or A', is obtained by reversing the indices of the original matrix; $(A^\top)_{ij} = A_{ji}$ for each i and j. This has the effect of turning rows into columns and vice versa:

$$A^\top = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}^\top = \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \end{bmatrix}$$

If a matrix is equal to its own transpose (i.e., if $A_{ij} = A_{ji}$ for all i and j), it is said to be **symmetric**. The transpose of an $n \times m$ matrix is an $m \times n$ matrix, so only square matrices can be symmetric.

13.3.2.2 Sum of two matrices

If we have two matrices A and B with the same dimension, we can compute their sum $A + B$ by the rule $(A + B)_{ij} = A_{ij} + B_{ij}$. Another way to say this is that matrix sums are done term-by-term: there is no interaction between entries with different indices.

For example, suppose we have the matrix of counts C above of ways of getting between two destinations on the Yale campus. Suppose that upperclassmen are allowed to also take the secret Science Hill Monorail from the sub-basement of Bass Library to the sub-basement of AKW. We can get the total number of ways an upperclassman can get from each origin to each destination by adding to C a second matrix M giving the paths involving monorail travel:

$$C + M = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & 1 \end{bmatrix}.$$

13.3.2.3 Product of two matrices

Suppose we are not content to travel once, but have a plan once we reach our destination in D to travel again to a final destination in some set F. Just as we constructed the matrix C (or $C+M$, for monorail-using upperclassmen) counting the number of ways to go from each point in O to each point in D, we can construct a matrix Q counting the number of ways to go from each point in D to each point in F. Can we combine these two matrices to compute the number of ways to travel $O \to D \to F$?

The resulting matrix is known as the **product** QC. We can compute each entry in QC by taking a sum of products of entries in Q and C. Observe that the number of ways to get from k to i via some single intermediate point j is just $Q_{ij}C_{jk}$. To get all possible routes, we have to sum over all possible intermediate points, giving $(QC)_{ik} = \sum_j Q_{ij}C_{jk}$.

This gives the rule for multiplying matrices in general: to get $(AB)_{ik}$, sum $A_{ij}B_{jk}$ over all intermediate values j. This works only when the number of columns in A is the same as the number of rows in B (since j has to vary over the same range in both matrices), i.e., when A is an $n \times m$ matrix and B is an $m \times s$ matrix for some n, m, and s. If the dimensions of the matrices don't match up like this, the matrix product is *undefined*. If the dimensions do match, they are said to be **compatible**.

For example, let $B = (C + M)$ from the sum example and let A be the number of ways of getting from each of destinations 1 = Bass, 2 = AKW, and 3 = SML to final destinations 1 = Heaven and 2 = Hell. After consulting with appropriate representatives of the Divinity School, we determine that one can get to either Heaven or Hell from any intermediate destination in one way by dying (in a state of grace or sin, respectively), but that Bass Library provides the additional option of getting to Hell by digging. This gives a matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}.$$

We can now compute the product

$$A(C+M) = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \cdot 0 + 1 \cdot 2 + 1 \cdot 2 & 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 \\ 2 \cdot 0 + 1 \cdot 2 + 1 \cdot 2 & 2 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 4 & 3 \end{bmatrix}.$$

One special matrix I (for each dimension $n \times n$) has the property that $IA = A$ and $BI = B$ for all matrices A and B with compatible dimension. This matrix is known as the **identity matrix**, and is defined by the rule $I_{ii} = 1$ and $I_{ij} = 0$ for $i \neq j$. It is not hard to see that in this case $(IA)_{ij} = \sum_k I_{ik} A_{kj} = I_{ii} A_{ij} = A_{ij}$, giving $IA = A$; a similar computation shows that $BI = B$. With a little more effort (omitted here) we can show that I is the *unique* matrix with this identity property.

13.3.2.4 The inverse of a matrix

A matrix A is **invertible** if there exists a matrix A^{-1} such that $AA^{-1} = A^{-1}A = 1$. This is only possible if A is square (because otherwise the dimensions don't work) and may not be possible even then. Note that it is enough to find a matrix such that $A^{-1}A = I$ to show that A is invertible.

To try to invert a matrix, we start with the pair of matrices A, I (where I is the identity matrix defined above) and multiply both sides of the pair from the left by a sequence of transformation matrices $B_1, B_2, \ldots B_k$ until $B_k B_{k-1} \cdots B_1 A = I$. At this point the right-hand matrix will be $B_k B_{k-1} \cdots B_1 = A^{-1}$. (We could just keep track of all the B_i, but it's easier to keep track of their product.)

How do we pick the B_i? These will be matrices that (a) multiply some row by a scalar, (b) add a multiple of one row to another row, or (c) swap two rows. We'll use the first kind to make all the diagonal entries equal one, and the second kind to get zeroes in all the off-diagonal entries. The third kind will be saved for emergencies, like getting a zero on the diagonal.

That the operations (a), (b), and (c) correspond to multiplying by a matrix is provable but tedious.[6] Given these operations, we can turn any

[6]The tedious details: To multiple row r by a, use a matrix B with $B_{ii} = 1$ when $i \neq r$, $B_{rr} = a$, and $B_{ij} = 0$ for $i \neq j$; to add a times row r to row s, use a matrix B with $B_{ii} = 1$ when $i \neq r$, $B_{rs} = a$, and $B_{ij} = 0$ for all other pairs ij; to swap rows r and s, use a matrix B with $B_{ii} = 1$ for $i \notin \{r, s\}$, $B_{rs} = B_{sr} = 1$, and $B_{ij} = 0$ for all other pairs ij.

invertible matrix A into I by working from the top down, rescaling each row i using a type (a) operation to make $A_{ii} = 1$, then using a type (b) operation to subtract A_{ji} times row i from each row $j > i$ to zero out A_{ji}, then finally repeating the same process starting at the bottom to zero out all the entries above the diagonal. The only way this can fail is if we hit some $A_{ii} = 0$, which we can swap with a nonzero A_{ji} if one exists (using a type (c) operation). If all the rows from i on down have a zero in the i column, then the original matrix A is not invertible. This entire process is known as Gauss-Jordan elimination.

This procedure can be used to solve matrix equations: if $AX = B$, and we know A and B, we can compute X by first computing A^{-1} and then multiplying $X = A^{-1}AX = A^{-1}B$. If we are not interested in A^{-1} for its own sake, we can simplify things by substituting B for I during the Gauss-Jordan elimination procedure; at the end, it will be transformed to X.

Example Original A is on the left, I on the right.

Initial matrices:

$$\begin{bmatrix} 2 & 0 & 1 \\ 1 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Divide top row by 2:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 1 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Subtract top row from middle row and 3·top row from bottom row:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 0 & 1 & 1/2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -1/2 & 1 & 0 \\ -3/2 & 0 & 1 \end{bmatrix}$$

Swap middle and bottom rows:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -1/2 & 1 & 0 \end{bmatrix}$$

Multiply bottom row by 2:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix}$$

Subtract $\frac{1}{2}$·bottom row from top and middle rows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 1 \\ -1 & 2 & 0 \end{bmatrix}$$

and we're done. (It's probably worth multiplying the original A by the alleged A^{-1} to make sure that we didn't make a mistake.)

13.3.2.5 Scalar multiplication

Suppose we have a matrix A and some constant c. The **scalar product** cA is given by the rule $(cA)_{ij} = cA_{ij}$; in other words, we multiply (or *scale*) each entry in A by c. The quantity c in this context is called a **scalar**; the term *scalar* is also used to refer to any other single number that might happen to be floating around.

Note that if we only have scalars, we can pretend that they are 1×1 matrices; $a + b = a_{11} + b_1$ and $ab = a_{11}b_{11}$. But this doesn't work if we multiply a scalar by a matrix, since cA (where c is considered to be a matrix) is only defined if A has only one row. Hence the distinction between matrices and scalars.

13.3.3 Matrix identities

For the most part, matrix operations behave like scalar operations, with a few important exceptions:

1. Matrix multiplication is only defined for matrices with compatible dimensions.

2. Matrix multiplication is *not commutative*: in general, we do not expect that $AB = BA$. This is obvious when one or both of A and B is not square (one of the products is undefined because the dimensions aren't compatible), but may also be true even if A and B are both square.

For a simple example of a non-commutative pair of matrices, consider

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \neq \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}.$$

On the other hand, matrix multiplication *is* associative: $A(BC) = (AB)C$. The proof is by expansion of the definition. First compute $(A(BC))_{ij} = \sum_k A_{ik}(BC)_{kj} = \sum_k \sum_m A_{ik}B_{km}C_{mj}$. Then compute $((AB)C)_{ij} = \sum_m (AB)_{im}C_{mj} = \sum_m \sum_k A_{ik}B_{km}C_{mj} = \sum_k \sum_m A_{ik}B_{km}C_{mj} = (A(BC))_{ij}$.

So despite the limitations due to non-compatibility and non-commutativity, we still have:

Associative laws $A+(B+C) = (A+B)+C$ (easy), $(AB)C = A(BC)$ (see above). Also works for scalars: $c(AB) = (cA)B = A(cB)and(cd)A = c(dA) = d(cA)$.

Distributive laws $A(B + C) = AB + BC$, $A(B + C) = AB + AC$. Also works for scalars: $c(A + B) = cA + cB, (c + d)A = cA + dA$.

Additive identity $A + 0 = 0 + A = A$, where 0 is the all-zero matrix of the same dimension as A.

Multiplicative identity $AI = A, IA = A, 1A = A, A1 = A$, where I is the identity matrix of appropriate dimension in each case and 1 is the scalar value 1.

Inverse of a product $(AB)^{-1} = B^{-1}A^{-1}$. Proof: $(B^{-1}A^{-1})(AB) = B^{-1}(A^{-1}A)B = B^{-1}(IB) = B^{-1}B = I$, and similarly for $(AB)(B^{-1}A^{-1})$.

Transposes $(A + B)^\top = A^\top + B^\top$ (easy), $(AB)^\top = B^\top A^\top$ (a little trickier). $(A^{-1})^\top = (A^\top)^{-1}$, provided A^{-1} exists (proof: $A^\top(A^{-1})^\top = (A^{-1}A)^\top = I^\top = I$).

Using these identities, we can do arithmetic on matrices without knowing what their actual entries are, so long as we are careful about non-commutativity. So for example we can compute

$$(A + B)^2 = (A + B)(A + B) = A^2 + AB + BA + B^2.$$

Similarly, if for square A we have

$$S = \sum_{n \in \mathbb{N}} A^n,$$

(where $A^0 = I$) we can solve the equation

$$S = I + AS$$

by first subtracting AS from both sides to get

$$IS - AS = I$$

then applying the distributive law:

$$(I - A)S = I$$

and finally multiplying both sides from the left by $(I - A)^{-1}$ to get

$$S = (I - A)^{-1},$$

assuming $I - A$ is invertible.

13.4 Vectors as matrices

Matrices give us an alternative representation for vectors, which allows us to extend matrix multiplication to vectors as well. We'll abuse terminology a bit by referring to a $1 \times n$ or $n \times 1$ matrix as a **vector**. A $1 \times n$ matrix is called a **row vector** for obvious reasons; similarly, an $n \times 1$ matrix is called a **column vector**.

Vectors defined in this way behave exactly like matrices in every respect. However, they are often written with lowercase letters to distinguish them from their taller and wider cousins. If this will cause confusion with scalars, we can disambiguate by writing vectors with a little arrow on top: $\vec{x}$ or in boldface: $\mathbf{x}$. Often we will just hope it will be obvious from context which variables represent vectors and which represent scalars, since writing all the little arrows can take a lot of time.

When extracting individual coordinates from a vector, we omit the boring index and just write x_1, x_2, etc. This is done for both row and column vectors, so rather than write $x_i^\top$ we can just write x_i.

Vector addition and scalar multiplication behave exactly the same for vectors-as-matrices as they did for our definition of vectors-as-sequences. This justifies treating vectors-as-matrices as just another representation for vectors.

13.4.1 Length

The **length** of a vector x, usually written as $\|x\|$ or sometimes just $|x|$, is defined as $\sqrt{\sum_i x_i}$; the definition follows from the Pythagorean theorem: $\|x\|^2 = \sum x_i^2$. Because the coordinates are squared, all vectors have non-negative length, and only the zero vector has length 0.

Length interacts with scalar multiplication exactly as you would expect: $\|cx\| = c\,\|x\|$. The length of the sum of two vectors depends on how the are aligned with each other, but the **triangle inequality** $\|x+y\| \leq \|x\| + \|y\|$ always holds.

A special class of vectors are the *unit vectors*, those vectors x for which $\|x\| = 1$. In geometric terms, these correspond to all the points on the surface of a radius-1 sphere centered at the origin. Any vector x can be turned into a unit vector $x/\|x\|$ by dividing by its length. In two dimensions, the unit vectors are all of the form $[xy]^\top = [\cos\theta, \sin\theta]^\top$, where by convention θ is the angle from due east measured counterclockwise; this is why traveling 9 units northwest corresponds to the vector $9[\cos 135°, \sin 135°]^\top = [-9/\sqrt{2}, 9/\sqrt{2}]^\top$. In one dimension, the unit vectors are (± 1). (There are no unit vectors in zero dimensions: the unique zero-dimensional vector has length 0.)

13.4.2 Dot products and orthogonality

Suppose we have some column vector x, and we want to know how far x sends us in a particular direction, where the direction is represented by a unit column vector e. We can compute this distance (a scalar) by taking the **dot product**

$$e \cdot x = e^\top x = \sum e_i x_i.$$

For example, if $x = [34]^\top$ and $e = [10]^\top$, then the dot product is

$$-e \cdot x = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 1 \cdot 3 + 0 \cdot 4 = 3.$$

In this case we see that the $[10]^\top$ vector conveniently extracts the first coordinate, which is about what we'd expect. But we can also find out how far x takes us in the $[1/\sqrt{2}1/\sqrt{2}]^\top$ direction: this is $[1/\sqrt{2}1/\sqrt{2}]x = 7/\sqrt{2}$.

By convention, we are allowed to take the dot product of two row vectors or of a row vector times a column vector or vice versa, provided of course that the non-boring dimensions match. In each case we transpose as appropriate to end up with a scalar when we take the matrix product.

Nothing in the definition of the dot product restricts either vector to be a unit vector. If we compute $x \cdot y$ where $x = ce$ and $\|e\| = 1$, then we are effectively multiplying $e \cdot y$ by c. It follows that the dot product is proportional to the length of both of its arguments. This often is expressed in terms of the geometric formulation, memorized by vector calculus students since time immemorial:

The dot product of x and y is equal to the product of their lengths times the cosine of the angle between them.

This formulation is a little misleading, since modern geometers will often *define* the angle between two vectors x and y as $\cos^{-1}(x \cdot y/(\|x\| \cdot \|y\|))$, but it gives a good picture of what is going on. One can also define the dot-product as the area of the parallelogram with sides x and y, with the complication that if the parallelogram is flipped upside-down we treat the area as negative. The simple version in terms of coordinates is harder to get confused about, so we'll generally stick with that.

Two vectors are **orthogonal** if their dot product is zero. In geometric terms, this occurs when either one or both vectors is the zero vector or when the angle between them is $\pm 90°$ (since $\cos(\pm 90°) = 0$). In other words, two non-zero vectors are orthogonal if and only if they are perpendicular to each other.

Orthogonal vectors satisfy the **Pythagorean theorem**: If $x \cdot y = 0$, then $\|x+y\|^2 = (x+y)\cdot(x+y) = x\cdot x + x\cdot y + y\cdot x + y\cdot y = x\cdot x + y\cdot y = \|x\|^2 + \|y\|^2$. It is not hard to see that the converse is also true: any pair of vectors for which $\|x+y\|^2 = \|x\|^2 + \|y\|^2$ must be orthogonal (at least in $\mathbb{R}^n$).

Orthogonality is also an important property of vectors used to define coordinate systems, as we will see below.

13.5 Linear combinations and subspaces

A **linear combination** of a set of vectors $x_1 \dots x_n$ is any vector that can be expressed as $\sum c_i x_i$ for some coefficients c_i. The **span** of the vectors, written $\langle x_1 \dots x_n \rangle$, is the set of all linear combinations of the x_i.[7]

The span of a set of vectors forms a **subspace** of the vector space, where a subspace is a set of vectors that is closed under linear combinations. This is a succinct way of saying that if x and y are in the subspace, so is $ax + by$ for any scalars a and b. We can prove this fact easily: if $x = \sum c_i x_i$ and $y = \sum d_i x_i$, then $ax + by = \sum (ac_i + bd_i) x_i$.

A set of vectors $x_1, x_2, \dots, x_n$ is **linearly independent** if there is no way to write one of the vectors as a linear combination of the others, i.e., if there is no choice of coefficients that makes some $x_i = \sum_{j \neq i} c_j x_j$. An equivalent definition is that there is no choice of coefficients c_i such that $\sum c_i x_i = 0$ and at least one c_i is nonzero (to see the equivalence, subtract x_i from both sides of the $x_i = \sum c_j x_j$ equation).

[7]Technical note: If the set of vectors $\{x_i\}$ is infinite, then we will only permit linear combinations with a finite number of nonzero coefficients. We will generally not consider vector spaces big enough for this to be an issue.

13.5.1 Bases

If a set of vectors is both (a) linearly independent, and (b) spans the entire vector space, then we call that set of vectors a **basis** of the vector space. An example of a basis is the **standard basis** consisting of the vectors $[10\dots00]^\top, [01\dots00]^\top, \dots, [00\dots10]^\top, [00\dots01]^\top$. This has the additional nice property of being made of of vectors that are all orthogonal to each other (making it an **orthogonal basis**) and of unit length (making it a **normal** basis).

A basis that is both orthogonal and normal is called **orthonormal**. We like orthonormal bases because we can recover the coefficients of some arbitrary vector v by taking dot-products. If $v = \sum a_i x_i$, then $v \cdot x_j = \sum a_i(x_i \cdot x_j) = a_i$, since orthogonality means that $x_i \cdot x_j = 0$ when $i \neq j$, and normality means $x_i \cdot x_i = \|x_i\|^2 = 1$.

However, even for non-orthonormal bases it is still the case that any vector can be written as a unique linear combination of basis elements. This fact is so useful we will state it as a theorem:

Theorem 13.5.1. *If $\{x_i\}$ is a basis for some vector space V, then every vector y has a unique representation $y = a_1x_1 + a_2x_2 + \dots + a_nx_n$.*

Proof. Suppose there is some y with more than one representation, i.e., there are sequences of coefficients a_i and b_i such that $y = a_1x_1 + a_2x_2 + \dots + a_nx_n = b_1x_1 + b_2x_2 + \dots + b_nx_n$. Then $0 = y - y = a_1x_1 + a_2x_2 + \dots + a_nx_n - b_1x_1 + b_2x_2 + \dots + b_nx_n = (a_1 - b_1)x_1 + (a_2 - b_2)x_2 + \dots + (a_n - b_n)x_n$. But since the x_i are independent, the only way a linear combination of the x_i can equal 0 is if all coefficients are 0, i.e., if $a_i = b_i$ for all i. ☐

Even better, we can do all of our usual vector space arithmetic in terms of the coefficients a_i. For example, if $a = \sum a_i x_i$ and $b = \sum b_i x_i$, then it can easily be verified that $a + b = \sum(a_i + b_i)x_i$ and $ca = \sum(ca_i)x_i$.

However, it may be the case that the same vector will have different representations in *different* bases. For example, in $\mathbb{R}^2$, we could have a basis $B_1 = \{(1,0),(0,1)\}$ and a basis $B_2 = \{(1,0),(1,-2)\}$. The vector $(2,3)$ would be represented as $(2,3)$ using basis B_1 but would be represented as $(5/2, -3/2)$ in basis B_2. In the standard basis $\{(1,0),(0,1)\}$, the representation of $(2,3)$ is just $(2,3)$.

Both bases above have the same size. This is not an accident; if a vector space has a finite basis, then all bases have the same size. We'll state this as a theorem, too:

Theorem 13.5.2. *Let $x_1 \dots x_n$ and $y_1 \dots y_m$ be two finite bases of the same vector space V. Then $n = m$.*

Proof. Assume without loss of generality that $n \leq m$. We will show how to replace elements of the x_i basis with elements of the y_i basis to produce a new basis consisting only of $y_1 \dots y_n$. Start by considering the sequence $y_1, x_1 \dots x_n$. This sequence is not independent since y_1 can be expressed as a linear combination of the x_i (they're a basis). So from Theorem 1 there is some x_i that can be expressed as a linear combination of $y_1, x_1 \dots x_{i-1}$. Swap this x_i out to get a new sequence $y_1, x_1 \dots x_{i-1}, x_{i+1}, \dots x_n$. This new sequence is also a basis, because (a) any z can be expressed as a linear combination of these vectors by substituting the expansion of x_i into the expansion of z in the original basis, and (b) it's independent, because if there is some nonzero linear combination that produces 0 we can substitute the expansion of x_i to get a nonzero linear combination of the original basis that produces 0 as well. Now continue by constructing the sequence $y_2, y_1, x_1 \dots x_{i-1}, x_{i+1}, \dots x_n$, and arguing that some $x_{i'}$ in this sequence must be expressible as a combination of earlier terms by Theorem 13.5.1 (it can't be y_1 because then y_2, y_1 is not independent), and drop this $x_{i'}$. By repeating this process we can eventually eliminate all the x_i, leaving the basis $y_n, \dots, y_1$. But then any y_k for $k > n$ would be a linear combination of this basis, so we must have $m = n$. $\square$

The size of any basis of a vector space is called the **dimension** of the space.

13.6 Linear transformations

When we multiply a column vector by a matrix, we transform the vector into a new vector. This transformation is **linear** in the sense that $A(x + y) = Ax + Ay$ and $A(cx) = cAx$; thus we call it a **linear transformation**. Conversely, any linear function f from column vectors to column vectors can be written as a matrix M such that $f(x) = Mx$. We can prove this by decomposing each x using the standard basis.

Theorem 13.6.1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation. Then there is a unique $n \times m$ matrix M such that $f(x) = Mx$ for all column vectors x.*

Proof. We'll use the following trick for extracting entries of a matrix by multiplication. Let M be an $n \times m$ matrix, and let e^i be a column vector with $e^i_j = 1$ if $i = j$ and 0 otherwise.[8] Now observe that $(e^i)^\top M e^j = \sum_k e^i_k (Me^j)_k = (Me^j)_i = \sum_k M_{ik} e^j_k = M_{ij}$. So given a particular linear f, we will now *define* M by the rule $M_{ij} = (e^i)^\top f(e^j)$. It is not hard to see that this gives $f(e^j) = Me^j$ for each basis vector j, since multiplying by $(e^i)^\top$ grabs the i-th coordinate in each case. To show that $Mx = f(x)$ for all x, decompose each x as $\sum_k c_k e^k$. Now compute $f(x) = f(\sum_k c_k e^k) = \sum_k c_k f(e^k) = \sum_k c_k M(e^k) = M(\sum_k c_k e^k) = Mx$. $\square$

13.6.1 Composition

What happens if we compose two linear transformations? We multiply the corresponding matrices:

$$(g \circ f)(x) = g(f(x)) = g(M_f x) = M_g(M_f x) = (M_g M_f)x.$$

This gives us another reason why the dimensions have to be compatible to take a matrix product: If multiplying by an $n \times m$ matrix A gives a map $g : \mathbb{R}^m \to \mathbb{R}^n$, and multiplying by a $k \times l$ matrix B gives a map $f : \mathbb{R}^l \to \mathbb{R}^k$, then the composition $g \circ f$ corresponding to AB only works if $m = k$.

13.6.2 Role of rows and columns of M in the product Mx

When we multiply a matrix and a column vector, we can think of the matrix as a sequence of row or column vectors and look at how the column vector operates on these sequences.

Let $M_{i\cdot}$ be the i-th row of the matrix (the "$\cdot$" is a stand-in for the missing column index). Then we have

$$(Mx)_i = \sum_k M_{ik} x_k = M_{i\cdot} \cdot x.$$

So we can think of Mx as a vector of dot-products between the rows of M and x:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} \langle 1,2,3 \rangle \cdot \langle 1,1,2 \rangle \\ \langle 4,5,6 \rangle \cdot \langle 1,1,2 \rangle \end{bmatrix} = \begin{bmatrix} 9 \\ 21 \end{bmatrix}.$$

[8] We are abusing notation by not being specific about how long e^i is; we will use the same expression to refer to any column vector with a 1 in the i-th row and zeros everywhere else. We are also moving what would normally be a subscript up into the superscript position to leave room for the row index—this is a pretty common trick with vectors and should not be confused with exponentiation.

Alternatively, we can work with the columns $M_{\cdot j}$ of M. Now we have

$$(Mx)_i = \sum_k M_{ik} x_k = \sum_k (M_{\cdot k})_i x_k.$$

From this we can conclude that Mx is a linear combination of columns of M: $Mx = \sum_k x_k M_{\cdot k}$. Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 5 \end{bmatrix} + 2 \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 7 \\ 12 \end{bmatrix} = \begin{bmatrix} 9 \\ 21 \end{bmatrix}.$$

The set $\{Mx\}$ for all x is thus equal to the span of the columns of M; it is called the **column space** of M.

For yM, where y is a row vector, similar properties hold: we can think of yM either as a row vector of dot-products of y with columns of M or as a weighted sum of rows of M; the proof follows immediately from the above facts about a product of a matrix and a column vector and the fact that $yM = (M^\top y^\top)^\top$. The span of the rows of M is called the **row space** of M, and equals the set $\{yM\}$ of all results of multiplying a row vector by M.

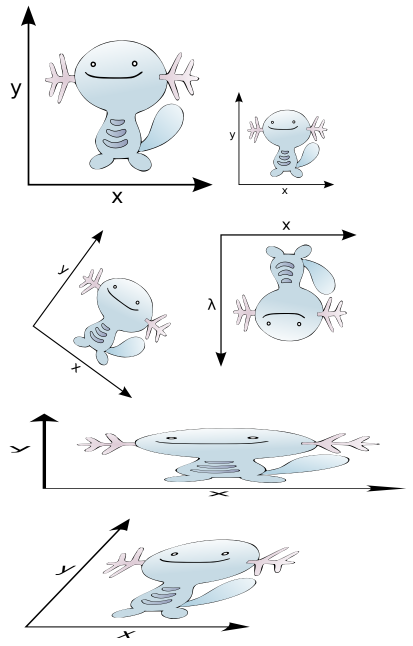
13.6.3 Geometric interpretation

Geometrically, linear transformations can be thought of as changing the basis vectors for a space: they keep the origin in the same place, move the basis vectors, and rearrange all the other vectors so that they have the same coordinates in terms of the new basis vectors. These new basis vectors are easily read off of the matrix representing the linear transformation, since they are just the columns of the matrix. So in this sense all linear transformations are transformations from some vector space to the column space of some matrix.[9]

This property makes linear transformations popular in graphics, where they can be used to represent a wide variety of transformations of images. Below is a picture of an untransformed image (top left) together with two standard basis vectors labeled x and y. In each of the other images, we have shifted the basis vectors using a linear transformation, and carried the image along with it.[10]

[9]The situation is slightly more complicated for infinite-dimensional vector spaces, but we will try to avoid them.

[10]The thing in the picture is a Pokémon known as a *Wooper*, which evolves into a *Quagsire* at level 20. This evolution is not a linear transformation.



Note that in all of these transformations, the origin stays in the same place. If you want to move an image, you need to add a vector to everything. This gives an **affine transformation**, which is any transformation that can be written as $f(x) = Ax + b$ for some matrix A and column vector b. One nifty thing about affine transformations is that—like linear transformations—they compose to produce new transformations of the same kind: $A(Cx + d) + b = (AC)x + (Ad + b)$.

Many two-dimensional linear transformations have standard names. The simplest transformation is **scaling**, where each axis is scaled by a constant, but the overall orientation of the image is preserved. In the picture above, the top right image is scaled by the same constant in both directions and the second-from-the-bottom image is scaled differently in each direction.

Recall that the product Mx corresponds to taking a weighted sum of the columns of M, with the weights supplied by the coordinates of x. So in terms of our basis vectors x and y, we can think of a linear transformation as specified by a matrix whose columns tell us what vectors for replace x and y with. In particular, a scaling transformation is represented by a matrix of the form

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix},$$

where s_x is the scale factor for the x (first) coordinate and s_y is the scale factor for the y (second) coordinate. Flips (as in the second image from the top on the right) are a special case of scaling where one or both of the scale factors is -1.

A more complicated transformation, as shown in the bottom image, is a **shear**. Here the image is shifted by some constant amount in one coordinate as the other coordinate increases. Its matrix looks like this:

$$\begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}.$$

Here the x vector is preserved: $(1, 0)$ maps to the first column $(1, 0)$, but the y vector is given a new component in the x direction of c, corresponding to the shear. If we also flipped or scaled the image at the same time that we sheared it, we could represent this by putting values other than 1 on the diagonal.

For a rotation, we will need some trigonometric functions to compute the new coordinates of the axes as a function of the angle we rotate the image by. The convention is that we rotate counterclockwise: so in the figure above, the rotated image is rotated counterclockwise approximately 315° or $-45°$. If Θ is the angle of rotation, the rotation matrix is given by

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

For example, when $\theta = 0°$, then we have $\cos\theta = 1$ and $\sin\theta = 0$, giving the identity matrix. When $\theta = 90°$, then $\cos\theta = 0$ and $\sin\theta = 1$, so we rotate the x axis to the vector $(\cos\theta, \sin\theta) = (0, 1)$ and the y axis to $(-\sin\theta, \cos\theta) = (-1, 0)$. This puts the x axis pointing north where the y axis used to be, and puts the y axis pointing due west.

13.6.4 Rank and inverses

The dimension of the column space of a matrix—or, equivalently, the dimension of the range of the corresponding linear transformation—is called the

rank. The rank of a linear transformation determines, among other things, whether it has an inverse.

Theorem 13.6.2. *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a linear transformation with an inverse f^{-1}, then we can show all of the following:*

1. *f^{-1} is also a linear transformation.*

2. *$n = m$, and f has **full rank**, i.e.,* $\operatorname{rank}(f) = \operatorname{rank}(f^{-1}) = m$.

Proof. 1. Let x and y be elements of codomain(f) and let a be a scalar. Then $f(af^{-1}(x)) = a(f(f^{-1}(x))) = ax$, implying that $f^{-1}(ax) = af^{-1}(x)$. Similarly, $f(f^{-1}(x) + f^{-1}(y)) = f(f^{-1}(x)) + f(f^{-1}(y)) = x + y$, giving $f^{-1}(x + y) = f^{-1}(x) + f^{-1}(y)$. So f^{-1} is linear.

2. Suppose $n < m$. Pick any basis e^i for $\mathbb{R}^n$, and observe that $\{f(e^i)\}$ spans range(f) (since we can always decompose x as $\sum a_i e^i$ to get $f(x) = \sum a_i f(e^i)$). So the dimension of range(f) is at most n. If $n < m$, then range(f) is a proper subset of $\mathbb{R}^m$ (otherwise it would be m-dimensional). This implies f is not surjective and thus has no inverse. Alternatively, if $m < n$, use the same argument to show that any claimed f^{-1} isn't. By the same argument, if either f or f^{-1} does not have full rank, it's not surjective.

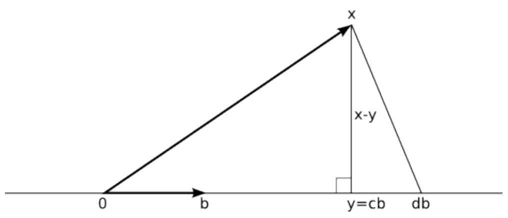
∎

The converse is also true: If $f : \mathbb{R}^n \to \mathbb{R}^n$ has full rank, it has an inverse. The proof of this is to observe that if $\dim(\operatorname{range}(f)) = n$, then $\operatorname{range}(f) = \mathbb{R}^n$ (since $\mathbb{R}^n$ has no full-dimensional subspaces). So in particular we can take any basis $\{e^i\}$ for $\mathbb{R}^n$ and find corresponding $\{x^i\}$ such that $f(x^i) = e^i$. Now the linear transformation that maps $\sum a_i e^i$ to $\sum a_i x^i$ is an inverse for f, since $f(\sum a_i x^i) = \sum a_i f(x_i) = \sum a_i e^i$.

13.6.5 Projections

Suppose we are given a low-dimensional subspace of some high-dimensional space (e.g. a line (dimension 1) passing through a plane (dimension 2)), and we want to find the closest point in the subspace to a given point in the full space. The process of doing this is called **projection**, and essentially consists of finding some point z such that $(x - z)$ is orthogonal to any vector in the subspace.

Let's look at the case of projecting onto a line first, then consider the more general case.



A line consists of all points that are scalar multiples of some fixed vector b. Given any other vector x, we want to extract all of the parts of x that lie in the direction of b and throw everything else away. In particular, we want to find a vector $y = cb$ for some scalar c, such that $(x - y) \cdot b = 0$. This is is enough information to solve for c.

We have $(x - cb) \cdot b = 0$, so $x \cdot b = c(b \cdot b)$ or $c = (x \cdot b)/(b \cdot b)$. So the projection of x onto the subspace $\{cb \mid c \in \mathbb{R}\}$ is given by $y = b(x \cdot b)/(b \cdot b)$ or $y = b(x \cdot b)/\|b\|^2$. If b is normal (i.e. if $\|b\| = 1$), then we can leave out the denominator; this is one reason we like orthonormal bases so much.

Why is this the right choice to minimize distance? Suppose we pick some other vector db instead. Then the points x, cb, and db form a right triangle with the right angle at cb, and the distance from x to db is $\|x - db\| = \sqrt{\|x - cb\|^2 + \|cb - db\|^2} \geq \|x - cb\|$.

But now what happens if we want to project onto a larger subspace? For example, suppose we have a point x in three dimensions and we want to project it onto some plane of the form $\{c_1b_1 + c_2b_2\}$, where b_1 and b_2 span the plane. Here the natural thing to try is to send x to $y = b_1(x \cdot b_1)/\|b_1\|^2 + b_2(x \cdot b_2)/\|b_2\|^2$. We then want to argue that the vector $(x - y)$ is orthogonal to any vector of the form $c_1b_1 + c_2b_2$. As before, $(x - y)$ is orthogonal to any vector in the plane, it's orthogonal to the difference between the y we picked and some other z we didn't pick, so the right-triangle argument again shows it gives the shortest distance.

Does this work? Let's calculate: $(x - y) \cdot (c_1b_1 + c_2b_2) = x \cdot (c_1b_1 + c_2b_2) - (b_1(x \cdot b_1)/\|b_1\|^2 + b_2(x \cdot b_2)/\|b_2\|^2) \cdot (c_1b_1 + c_2b_2) = c_1(x \cdot b_1 - (b_1 \cdot b_1)(x \cdot b_1)/(b_1 \cdot b_1)) + c_2(x \cdot b_2 - (b_2 \cdot b_2)(x \cdot b_2)/(b_2 \cdot b_2)) - c_1(b_1 \cdot b_2)(x \cdot b_1)/(b_1 \cdot b_1) - c_2(b_1 \cdot b_2)(x \cdot b_2)/(b_2 \cdot b_2)$.

The first two terms cancel out very nicely, just as in the one-dimensional case, but then we are left with a nasty $(b_1 \cdot b_2)$(much horrible junk) term at the end. *It didn't work!*

So what do we do? We could repeat our method for the one-dimensional case and solve for c_1 and c_2 directly. This is probably a pain in the neck. Or we can observe that the horrible extra term includes a $(b_1 \cdot b_2)$ factor, and if b_1 and b_2 are orthogonal, it disappears. The moral: We can project onto a

2-dimensional subspace by projecting independently onto the 1-dimensional subspace spanned by each basis vector, *provided the basis vectors are orthogonal*. And now we have another reason to like orthonormal bases.

This generalizes to subspaces of arbitrary high dimension: as long as the b_i are all orthogonal to each other, the projection of x onto the subspace $\langle b_i \rangle$ is given by $\sum b_i(x \cdot b_i)/\left\|b_i\right\|^2$. Note that we can always express this as matrix multiplication by making each row of a matrix B equal to one of the vectors $b_i/\left\|b_i\right\|^2$; the product Bx then gives the coefficients for the basis elements in the projection of x, since we have already seen that multiplying a matrix by a column vector corresponds to taking a dot product with each row. If we want to recover the projected vector $\sum c_i b_i$ we can do so by taking advantage of the fact that multiplying a matrix by a column vector also corresponds to taking a linear combination of columns: this gives a combined operation of $B^\top Bx$ which we can express as a single **projection matrix** $P = B^\top B$. So projection corresponds to yet another special case of a linear transformation.

One last detail: suppose we aren't given orthonormal b_i but are instead given some arbitrary non-orthogonal non-normal basis for the subspace. Then what do we do?

The trick here is to use a technique called Gram-Schmidt orthogonalization. This constructs an orthogonal basis from an arbitrary basis by induction. At each step, we have a collection of orthogonalized vectors $b_1 \dots b_k$ and some that we haven't processed yet $a_{k+1} \dots a_m$; the induction hypothesis says that the $b_1 \dots b_k$ vectors are (a) orthogonal and (b) span the same subspace as $a_1 \dots a_k$. The base case is the empty set of basis vectors, which is trivially orthogonal and also trivially spans the subspace consisting only of the 0 vector. We add one new vector to the orthogonalized set by projecting a_{k+1} to some point c on the subspace spanned by $b_1 \dots, b_k$; we then let $b_{k+1} = a_{k+1} - c$. This new vector is orthogonal to all of $b_1 \dots b_k$ by the definition of orthogonal projection, giving a new, larger orthogonal set $b_1 \dots b_{k+1}$. These vectors span the same subspace as $a_1 \dots a_{k+1}$ because we can take any vector x expressed as $\sum_{i=1}^{k+1} c_i a_i$, and rewrite it as $\sum_{i=1}^{k} c_i b_i + c_{k+1}(c + b_{k+1})$, and in the second term $c_{k+1}c$ reduces to a linear combination of $b_1 \dots b_k$; the converse essentially repeats this argument in reverse. It follows that when the process completes we have an orthogonal set of vectors $b_1 \dots b_m$ that span precisely the same subspace as $a_1 \dots a_m$, and we have our orthogonal basis. (But not orthonormal: if we want it to be orthonormal, we divide each b_i by $\|b_i\|$ as well.)

13.7 Further reading

Linear algebra is vitally important in Computer Science: it is a key tool in graphics, scientific computing, robotics, neural networks, and many other areas. If you do further work in these areas, you will quickly find that we have not covered anywhere near enough linear algebra in this course. Your best strategy for remedying this deficiency may be to take an actual linear algebra course; failing that, a very approachable introductory text is *Linear Algebra and Its Applications*, by Gilbert Strang [Str05]. You can also watch an entire course of linear algebra lectures through YouTube: `http://www.youtube.com/view_play_list?p=E7DDD91010BC51F8`.

Some other useful books on linear algebra:

- Golub and Van Loan, *Matrix Computations* [GVL12]. Picks up where Strang leaves off with practical issues in doing computation.

- Halmos, *Finite-Dimensional Vector Spaces* [Hal58]. Good introduction to abstract linear algebra: properties of vector spaces without jumping directly to matrices.

Matlab (which is available on the Zoo machines: type 'matlab' at a shell prompt) is useful for playing around with operations on matrices. There are also various non-commercial knockoffs like Scilab or Octave that are not as comprehensive as Matlab but are adequate for most purposes. Note that with any of these tools, if you find yourselves doing lots of numerical computation, it is a good idea to talk to a numerical analyst about round-off error: the floating-point numbers inside computers are not the same as real numbers, and if you aren't careful about how you use them you can get very strange answers.

Chapter 14

Finite fields

Our goal here is to find computationally-useful structures that act enough like the rational numbers $\mathbb{Q}$ or the real numbers $\mathbb{R}$ that we can do arithmetic in them that are small enough that we can describe any element of the structure uniquely with a finite number of bits. Such structures are called **finite fields**.

An example of a finite field is $\mathbb{Z}_p$, the integers mod p (see §8.4). These finite fields are inconvenient for computers, which like to count in bits and prefer numbers that look like 2^n to horrible nasty primes. So we'd really like finite fields of size 2^n for various n, particularly if the operations of addition, multiplication, etc. have a cheap implementation in terms of sequences of bits. To get these, we will show how to construct a finite field of size p^n for any prime p and positive integer n, and then let $p = 2$.

14.1 A magic trick

We will start with a magic trick. Suppose we want to generate a long sequence of bits that are hard to predict. One way to do this is using a mechanism known as a **linear-feedback shift register** or **LFSR**. There are many variants of LFSRs. Here is one that generates a sequence that repeats every 15 bits by keeping track of 4 bits of state, which we think of as a binary number $r_3r_2r_1r_0$.

To generate each new bit, we execute the following algorithm:

1. Rotate the bits of r left, to get a new number $r_2r_1r_0r_3$.

2. If the former leftmost bit was 1, flip the new leftmost bit.

3. Output the rightmost bit.

Here is the algorithm in action, starting with $r = 0001$:

r	rotated r	rotated r after flip	output
0001	0010	0010	0
0010	0100	0100	0
0100	1000	1000	0
1000	0001	1001	1
1001	0011	1011	1
1011	0111	1111	1
1111	1111	0111	1
0111	1110	1110	0
1110	1101	0101	1
0101	1010	1010	0
1010	0101	1101	1
1101	1011	0011	1
0011	0110	0110	0
0110	1100	1100	0
1100	1001	0001	1
0001	0010	0010	0

After 15 steps, we get back to 0001, having passed through all possible 4-bit values except 0000. The output sequence 000111101011001... has the property that every 4-bit sequence except 0000 appears starting at one of the 15 positions, meaning that after seeing any 3 bits (except 000), both bits are equally likely to be the next bit in the sequence. We thus get a sequence that is almost as long as possible given we have only 2^4 possible states, that is highly unpredictable, and that is cheap to generate. So unpredictable and cheap, in fact, that the governments of both the United States[1] and Russia[2] operate networks of orbital satellites that beam microwaves into our brains carrying signals generated by linear-feedback shift registers very much like this one. Similar devices are embedded at the heart of every modern computer, scrambling all communications between the motherboard and PCI cards to reduce the likelihood of accidental eavesdropping.

What horrifying deep voodoo makes this work?

14.2 Fields and rings

A **field** is a set F together with two operations $+$ and $\cdot$ that behave like addition and multiplication in the rationals or real numbers. Formally, this

[1]In the form of the Global Positioning System.

[2]In the form of GLONASS.

means that:

1. Addition is **associative**: $(x + y) + z = x + (y + z)$ for all x, y, z in F.
2. There is an **additive identity** 0 such that $0 + x = x + 0 = x$ for all x in F.
3. Every x in F has an **additive inverse** $-x$ such that $x + (-x) = (-x) + x = 0$.
4. Addition is **commutative**: $x + y = y + x$ for all x, y in F.
5. Multiplication **distributes** over addition: $x \cdot (y + z) = (x \cdot y + x \cdot z)$ and $(y + z) \cdot x = (y \cdot x + z \cdot x)$ for all x, y, z in F.
6. Multiplication is associative: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ for all x, y, z in F.
7. There is a **multiplicative identity** 1 such that $1 \cdot x = x \cdot 1 = x$ for all x in F.
8. Multiplication is commutative: $x \cdot y = y \cdot x$ for all x, y in F.
9. Every x in $F \setminus \{0\}$ has a **multiplicative inverse** x^{-1} such that $x \cdot x^{-1} = x^{-1} \cdot x = 1$.

Some structures fail to satisfy all of these axioms but are still interesting enough to be given names. A structure that satisfies 1–3 is called a **group**; 1–4 is an **abelian group** or commutative group; 1–7 is a **ring**; 1–8 is a **commutative ring**. In the case of groups and abelian groups there is only one operation $+$. There are also more exotic names for structures satisfying other subsets of the axioms.[3]

Some examples of fields: $\mathbb{R}, \mathbb{Q}, \mathbb{C}, \mathbb{Z}_p$ where p is prime. We will be particularly interested in $\mathbb{Z}_p$, since we are looking for *finite* fields that can fit inside a computer.

The integers $\mathbb{Z}$ are an example of a commutative ring, as is $\mathbb{Z}_m$ for $m > 1$. Square matrices of fixed dimension greater than 1 are an example of a non-commutative ring.

[3]A set with one operation that does not necessarily satisfy any axioms is a **magma**. If the operation is associative, it's a **semigroup**, and if there is also an identity (but not necessarily inverses), it's a **monoid**. For example, the set of nonempty strings with $+$ interpreted as concatenation form a semigroup, and throwing in the empty string as well gives a monoid.

Weaker versions of rings knock out the multiplicative identity (a **pseudo-ring** or **rng**) or negation (a **semiring** or **rig**). An example of a semiring that is actually useful is the $(\max, +)$ semiring, which uses max for addition and $+$ (which distributes over max) for multiplication; this turns out to be handy for representing scheduling problems.

14.3 Polynomials over a field

Any field F generates a **polynomial ring** $F[x]$ consisting of all polynomials in the variable x with coefficients in F. For example, if $F = \mathbb{Q}$, some elements of $\mathbb{Q}[x]$ are $3/5, (22/7)x^2 + 12, 9003x^{417} - (32/3)x^4 + x^2$, etc. Addition and multiplication are done exactly as you'd expect, by applying the distributive law and combining like terms: $(x + 1) \cdot (x^2 + 3/5) = x \cdot x^2 + x \cdot (3/5) + x^2 + (3/5) = x^3 + x^2 + (3/5)x + (3/5)$.

The **degree** $\deg(p)$ of a polynomial p in $F[x]$ is the exponent on the **leading term**, the term with a nonzero coefficient that has the largest exponent. Examples: $\deg(x^2 + 1) = 2$, $\deg(17) = 0$. For 0, which doesn't have any terms with nonzero coefficients, the degree is taken to be $-\infty$. Degrees add when multiplying polynomials: $\deg((x^2+1)(x+5)) = \deg(x^2+1) + deg(x+5) = 2+1 = 3$; this is just a consequence of the leading terms in the polynomials we are multiplying producing the leading term of the new polynomial. For addition, we have $\deg(p + q) \le \max(\deg(p), \deg(q))$, but we can't guarantee equality (maybe the leading terms cancel).

Because $F[x]$ is a ring, we can't do division the way we do it in a field like $\mathbb{R}$, but we *can* do division the way we do it in a ring like $\mathbb{Z}$, leaving a remainder. The equivalent of the integer **division algorithm** for $\mathbb{Z}$ is:

Theorem 14.3.1 (Division algorithm for polynomials)**.** *Given a polynomial f and a nonzero polynomial g in $F[x]$, there are unique polynomials q and r such that $f = q \cdot g + r$ and $\deg(r) < \deg(g)$.*

Proof. The proof is by induction on $\deg(f)$. If $\deg(f) < \deg(g)$, let $q = 0$ and $r = f$. If $\deg(f)$ is larger, let $m = \deg(f)$, $n = \deg(g)$, and $q_{m-n} = f_m g_n^{-1}$. Then $q_{m-n}x^{m-n}g$ is a degree-m polynomial with leading term f_m. Subtracting this from f gives a polynomial f' of degree at most $m - 1$, and by the induction hypothesis there exist q', r such that $f' = q' \cdot g + r$ and $\deg r < \deg g$. Let $q = q_{m-n}x^{m-n} + q'$; then $f = f' + q_{m-n}x^{m-n}g = (q_{m-n}x^{m-n}g + q'g + r = q \cdot g + r$. ∎

The essential idea of the proof is that we are finding q and r using the same process of long division as we use for integers. For example, in $\mathbb{Q}[x]$:

$$\begin{array}{rr}
 & x + 1 \\ \cline{2-2}
x + 2) & x^2 + 3x + 5 \\
 & -x^2 - 2x \phantom{{}+5} \\ \cline{2-2}
 & x + 5 \\
 & -x - 2 \\ \cline{2-2}
 & 3
\end{array}$$

From this we get $x^2 + 3x + 5 = (x + 2)(x - 1) + 3$, with $\deg(3) = 0 < \deg(x + 2) = 1$.

We are going to use division of polynomials to define finite fields by taking $F[x]$ modulo some well-chosen polynomial, analogously to the way we can turn $\mathbb{Z}$ (a ring) into a field $\mathbb{Z}_p$ by taking quotients mod p. As long as we choose the right polynomial, this works in any field.

14.4 Algebraic field extensions

Given a field F, we can make a bigger field by adding in extra elements that behave in a well-defined and consistent way. An example of this is the extension of the real numbers $\mathbb{R}$ to the complex numbers $\mathbb{C}$ by adding i.

The general name for this trick is **algebraic field extension** or just **field extension**, and it works by first constructing the ring of polynomials $F[x]$ and then smashing it down into a field by taking remainders modulo some fixed polynomial $p(x)$. For this to work, the polynomial has to to be **irreducible**, which mean that $p(x) = 0$ if and only if $x = 0$, or equivalently that p can't be factored as $(x + a)p'$ for some a and p'. This latter definition makes irreducibility sort of like being prime, and makes this construction sort of like the construction of $\mathbb{Z}_p$.

The fact that the resulting object is a field follows from inheriting all the commutative ring properties from $F[x]$, plus getting multiplicative inverses for essentially the same reason as in $\mathbb{Z}_p$: we can find them using the extended Euclidean algorithm applied to polynomials instead of integers (we won't prove this).

In the case of the complex numbers $\mathbb{C}$, the construction is $\mathbb{C} = \mathbb{R}[i]/(i^2 + 1)$. Because $i^2 + 1 = 0$ has no solution $i \in \mathbb{R}$, this makes $i^2 + 1$ an irreducible polynomial. An element of $\mathbb{C}$ is then a degree-1 or less polynomial in $\mathbb{R}[i]$, because these are the only polynomials that survive taking the remainder mod $i^2 + 1$ intact.

If you've used complex numbers before, you were probably taught to multiply them using the rule $i^2 = -1$, which is a rewriting of $i^2 + 1 = 0$. This is equivalent to taking remainders: $(i + 1)(i + 2) = (i^2 + 3i + 2) = 1 \cdot (i^2 + 1) + (3i + 1) = 3i + 1$.

The same thing works for other fields and other irreducible polynomials. For example, in $\mathbb{Z}_2$, the polynomial $x^2 + x + 1$ is irreducible, because $x^2 + x + 1 = 0$ has no solution (try plugging in 0 and 1 to see). So we can construct a new finite field $\mathbb{Z}_2[x]/(x^2 + x + 1)$ whose elements are polynomials with coefficients in $\mathbb{Z}_2$ with all operations done modulo $x^2 + x + 1$.

Addition in $\mathbb{Z}_2[x]/(x^2+x+1)$ looks like vector addition:[4] $(x+1)+(x+1) = 0\cdot x+0 = 0, (x+1)+x = 1, (1)+(x) = (x+1)$. Multiplication in $\mathbb{Z}_2[x]/(x^2+x+1)$ works by first multiplying the polynomials and taking the remainder mod (x^2+x+1): $(x+1)\cdot(x+1) = x^2+1 = 1\cdot(x^2+x+1)+x = x$. If you don't want to take remainders, you can instead substitute $x+1$ for any occurrence of x^2 (just like substituting -1 for i^2 in $\mathbb{C}$), since $x^2+x+1 = 0$ implies $x^2 = -x-1 = x+1$ (since $-1 = 1$ in $\mathbb{Z}_2$).

The full multiplication table for this field looks like this:

	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

We can see that every nonzero element has an inverse by looking for ones in the table; e.g. $1\cdot 1 = 1$ means 1 is its own inverse and $x\cdot(x+1) = x^2+x = 1$ means that x and $x+1$ are inverses of each other.

Here's the same thing for $\mathbb{Z}_2[x]/(x^3+x+1)$:

	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

Note that we now have $2^3 = 8$ elements. In general, if we take $\mathbb{Z}_p[x]$ modulo a degree-n polynomial, we will get a field with p^n elements. These turn out to be all the possible finite fields, with exactly one finite field for each number of the form p^n (up to isomorphism, which means that we consider two fields equivalent if there is a bijection between them that preserves $+$ and $\cdot$). We can refer to a finite field of size p^n abstractly as $GF(p^n)$, which is an abbreviation for the **Galois field** p^n.

[4]This is not an accident: any extension field acts like a vector space over its base field.

14.5 Applications

So what are these things good for?

On the one hand, given an irreducible polynomial $p(x)$ of degree n over $\mathbb{Z}_2(x)$, it's easy to implement arithmetic in $\mathbb{Z}_2[x]/p(x)$ (and thus $GF(2^n)$) using standard-issue binary integers. The trick is to represent each polynomial $\sum a_i x^i$ by the integer value $a = \sum a_i 2^i$, so that each coefficient a_i is just the i-th bit of a. Adding two polynomials $a + b$ represented in this way corresponds to computing the bitwise exclusive or of a and b: `a^b` in programming languages that inherit their arithmetic syntax from C (i.e., almost everything except Scheme). Multiplying polynomials is more involved, although it's easy for some special cases like multiplying by x, which becomes a left-shift (`a<<1`) followed by XORing with the representation of our modulus if we get a 1 in the n-th place. (The general case is like this but involves doing XORs of a lot of left-shifted values, depending on the bits in the polynomial we are multiplying by.)

On the other hand, knowing that we can multiply $7 \equiv x^2 + x + 1$ by $5 \equiv x^2 + 1$ and get $6 \equiv x^2 + x$ quickly using C bit operations doesn't help us much if this product doesn't mean anything. For modular arithmetic (§8.4), we at least have the consolation that $7 \cdot 5 = 6 \pmod{29}$ tells us something about remainders. In $GF(2^3)$, what this means is much more mysterious. This makes it useful—not in contexts where we want multiplication to make sense—but in contexts where we don't. These mostly come up in random number generation and cryptography.

14.5.1 Linear-feedback shift registers

Let's suppose we generate $x^0, x^1, x^2, \ldots$ in $\mathbb{Z}_2/(x^4 + x^3 + 1)$, which happens to be one of the finite fields isomorphic to $GF(2^4)$. Since there are only $2^4 - 1 = 15$ nonzero elements in $GF(2^4)$, we can predict that eventually this sequence will repeat, and in fact we can show that $p^{15} = 1$ for any nonzero p using essentially the same argument as for Fermat's Little Theorem. So we will have $x^0 = x^{15} = x^{30}$ etc. and thus will expect our sequence to repeat every 15 steps (or possibly some factor of 15, if we are unlucky).

To compute the actual sequence, we could write out full polynomials: $1, x, x^2, x^3, x^3 + 1, x^3 + x + 1, \ldots$, but this gets tiresome fast. So instead we'd like to exploit our representation of $\sum a_i x^i$ as $\sum a_i 2^i$.

Now multiplying by x is equivalent to shifting left (i.e. multiplying by 2) followed by XORing with 11001, the binary representation of $x^4 + x^3 + 1$, if we get a bit in the x^4 place that we need to get rid of. For example, we might do:

```
 1101 (initial value)
11010 (after shift)
 0011 (after XOR with 11001)
```

or

```
 0110 (initial value)
01100 (after shift)
 1100 (no XOR needed)
```

If we write our initial value as $r_3r_2r_1r_0$, the shift produces a new value $r_3r_2r_1r_00$. Then XORing with 11001 has three effects: (a) it removes a leading 1 if present; (b) it sets the rightmost bit to r_3; and (c) it flips the new leftmost bit if $r_3 = 1$. Steps (a) and (b) turn the shift into a rotation. Step (c) is the mysterious flip from our sequence generator. So in fact what our magic sequence generator was doing was just computing all the powers of x in a particular finite field.

As in $\mathbb{Z}_p$, these powers of an element bounce around unpredictably, which makes them a useful (though cryptographically very weak) pseudorandom number generator. Because high-speed linear-feedback shift registers are very cheap to implement in hardware, they are used in applications where a pre-programmed, statistically smooth sequence of bits is needed, as in the Global Positioning System and to scramble electrical signals in computers to reduce radio-frequency interference.

14.5.2 Checksums

Shifting an LFSR corresponds to multiplying by x. If we also add 1 from time to time, we can build any polynomial we like, and get the remainder mod m; for example, to compute the remainder of 100101 mod 11001 we do

```
 0000 (start with 0)
00001 (shift in 1)
 0001 (no XOR)
00010 (shift in 0)
 0010 (no XOR)
00100 (shift in 0)
 0100 (no XOR)
01001 (shift in 1)
 1001 (no XOR)
```

```
10010 (shift in 0)
 1011 (XOR with 11001)
10111 (shift in 1)
 1110 (XOR with 11001)
```

and we have computed that the remainder of $x^5 + x^3 + 1 \bmod x^4 + x^3 + 1$ is $x^3 + x^2 + x$.

This is the basis for **cyclic redundancy check** (**CRC**) checksums, which are used to detect accidental corruption of data. The idea is that we feed our data stream into the LFSR as the coefficients of some gigantic polynomial, and the checksum summarizing the data is the state when we are done. Since it's unlikely that a random sequence of flipped or otherwise damaged bits would equal 0 mod m, most non-malicious changes to the data will be visible by producing an incorrect checksum.

14.5.3 Cryptography

$GF(2^n)$ can also substitute for $\mathbb{Z}_p$ in some cryptographic protocols. An example would be the function $f(s) = x^s \pmod{m}$, which is fairly easy to compute in $\mathbb{Z}_p$ and even easier to compute in $GF(2^n)$, but which seems to be hard to invert in both cases. Here we can take advantage of the fast remainder operation provided by LFSRs to avoid having to do expensive division in $\mathbb{Z}$.

Appendix A

Assignments

Assignments are typically due Thursdays at 5:00 pm. Assignments may be turned in by placing them in Huamin Li's mailbox. This is labeled 015–016 and can be found with the rest of the mailboxes in the short hallway just to the left of the Prospect Street entrance to Arthur K. Watson Hall.

Sample solutions will appear in this appendix after the assignment is due. For questions about grading, consult the grader (indicated in the assignment title).

A.1 Assignment 1: due Thursday, 2013-09-12, at 5:00 pm (Huamin)

Bureaucratic part

Send me email! My address is `james.aspnes@gmail.com`.

In your message, include:

1. Your name.
2. Your status: whether you are an undergraduate, grad student, auditor, etc.
3. Anything else you'd like to say.

(You will not be graded on the bureaucratic part, but you should do it anyway.)

A.1.1 Tautologies

Show that each of the following propositions is a tautology using a truth table, following the examples in §2.2.2. Each of your truth tables should include columns for all sub-expressions of the proposition.

1. $(\neg P \rightarrow P) \leftrightarrow P$.
2. $P \vee (Q \rightarrow \neg(P \leftrightarrow Q))$.
3. $(P \vee Q) \leftrightarrow (Q \vee (P \leftrightarrow (Q \rightarrow R)))$.

Solution

For each solution, we give the required truth-table solution first, and then attempt to give some intuition for why it works. The intuition is merely an explanation of what is going on and is not required for your solutions.

1. Here is the truth table:

P	$\neg P$	$\neg P \rightarrow P$	$(\neg P \rightarrow P) \leftrightarrow P$
0	1	0	1
1	0	1	1

   Intuitively, the only way for an implication to be false is if it has a true premise and a false conclusion, so to make $\neg P \rightarrow P$ false we need P to be false, which is what the tautology says.

2. Here we just evaluate the expression completely and see that it is always true:

P	Q	$P \leftrightarrow Q$	$\neg(P \leftrightarrow Q)$	$Q \rightarrow (\neg(P \leftrightarrow Q))$	$P \vee (Q \rightarrow (\neg(P \leftrightarrow Q)))$
0	0	1	0	1	1
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	0	0	1

   This is a little less intuitive than the first case. A reasonable story might be that the proposition is true if P is true, so for it to be false, P must be false. But then $\neg(P \leftrightarrow Q)$ reduces to Q, and $Q \leftrightarrow Q$ is true.

3. $(P \vee Q) \leftrightarrow (Q \vee (P \leftrightarrow (Q \rightarrow R)))$.

P	Q	R	$P \vee Q$	$Q \rightarrow R$	$P \leftrightarrow (Q \rightarrow R)$	$Q \vee (P \leftrightarrow (Q \rightarrow R))$	$(P \vee Q) \leftrightarrow (Q \vee (P \leftrightarrow (Q \rightarrow R)))$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1

I have no intuition whatsoever for why this is true. In fact, all three of these tautologies were plucked from long lists of machine-generated tautologies, and three variables is enough to start getting tautologies that don't have good stories.

It's possible that one could prove this more succinctly by arguing by cases that if Q is true, both sides of the biconditional are true, and if Q is not true, then $Q \rightarrow R$ is always true so $P \leftrightarrow (Q \rightarrow R)$ becomes just P, making both sides equal. But sometimes it is more direct (and possibly less error-prone) just to "shut up and calculate."

A.1.2 Positively equivalent

Show how each of the following propositions can be simplified using equivalences from Table 2.2 to a single operation applied directly to P and Q.

1. $\neg(P \rightarrow \neg Q)$.
2. $\neg((P \wedge \neg Q) \vee (\neg P \wedge Q))$.

Solution

1.

$$\begin{aligned}\neg(P \rightarrow \neg Q) &\equiv \neg(\neg P \vee \neg Q)\\ &\equiv \neg\neg P \wedge \neg\neg Q\\ &\equiv P \wedge Q.\end{aligned}$$

2.

$$\begin{aligned}
\neg((P \wedge \neg Q) \vee (\neg P \wedge Q)) \\
&\equiv \neg(P \wedge \neg Q) \wedge \neg(\neg P \wedge Q) \\
&\equiv (\neg P \vee \neg\neg Q) \wedge (\neg\neg P \vee \neg Q) \\
&\equiv (\neg P \vee Q) \wedge (P \vee \neg Q) \\
&\equiv (\neg P \vee Q) \wedge (\neg Q \vee P) \\
&\equiv (P \rightarrow Q) \wedge (Q \rightarrow P) \\
&\equiv P \leftrightarrow Q.
\end{aligned}$$

A.1.3 A theory of leadership

Suppose we are trying to write down, in predicate logic, a theory explaining the success or failure of various historical leaders. The universe consists of historical leaders; when writing $\forall x$ we are implicitly limiting ourselves to historical leaders under consideration, and similarly when writing $\exists x$. We have two predicates taller(x, y) ("x was taller than y") and successful(x) ("x was successful as a leader), as well as all the usual tools of predicate logic $\forall$, $\exists$, $=$, and so forth, and can refer to specific leaders by name.

Express each of the following statements in mathematical form. Note that these statements are not connected, and no guarantees are made about whether any of them are actually true.

1. Lincoln was the tallest leader.
2. Napoleon was at least as tall as any unsuccessful leader.
3. No two leaders had the same height.

Solution

1. The easiest way to write this is probably $\forall x : \text{taller}(\text{Lincoln}, x)$. There is a possible issue here, since this version says that nobody is taller than Lincoln, but it may be that somebody is the same height.[1] A stronger claim is $\forall x : (x \neq \text{Lincoln}) \rightarrow \text{taller(Lincoln, x)}$. Both solutions (and their various logical equivalents) are acceptable.

[1]At least one respected English-language novelist [Say33] has made the claim that it is well understood that stating that a particular brand of toothpaste is "the most effective" is not a falsehood even if it is equally effective with other brands—which are also "the most effective"—but this understanding is not universal. The use of "the" also suggests that Lincoln is unique among tallest leaders.

2. $\forall x : \neg \operatorname{successful}(x) \rightarrow \neg \operatorname{taller}(x, \text{Napoleon})$.

3. $\forall x\, \forall y : (x = y) \vee \operatorname{taller}(x, y) \vee \operatorname{taller}(y, x)$. Equivalently, $\forall x\, \forall y : x \neq y \rightarrow (\operatorname{taller}(x, y) \vee \operatorname{taller}(y, x))$. If we assume that $\operatorname{taller}(x, y)$ and $\operatorname{taller}(y, x)$ are mutually exclusive, then $\forall x\, \forall y : (x = y) \vee (\operatorname{taller}(x, y) \oplus \operatorname{taller}(y, x))$ also works.

A.2 Assignment 2: due Thursday, 2013-09-19, at 5:00 pm (Xiao)

A.2.1 Subsets

Let R, S, and T be sets. Using the definitions of $\subseteq$, $\cup$, $\cap$, and $\setminus$ given in §3.2, prove or disprove each of the following statements:

1. R is a subset of S if and only if $R \subseteq (S \setminus T) \cup (R \cap T)$.

2. R is a subset of S if and only if $R \subseteq R \cap S$.

3. R is a subset of $S \setminus R$ if and only if $R = \emptyset$.

Solution

1. Disproof: Consider $R = T = \{1\}$, $S = \emptyset$. Then R is not a subset of S, but

$$\begin{aligned}
(S \setminus T) \cup (R \cap T) &= (\emptyset \setminus \{1\}) \cup (\{1\} \cap \{1\}) \\
&= \emptyset \cup \{1\} \\
&= \{1\} \\
&\supseteq R.
\end{aligned}$$

2. Proof: We need to show this in both directions: first, that if R is a subset of S, then $R \subseteq R \cap S$; then, that if $R \subseteq R \cap S$, R is a subset of S.

   Suppose that R is a subset of S. Let x be an element of R. Then x is also an element of S. Since it is an element of both R and S, x is an element of $R \cap S$. It follows that $R \subseteq R \cap S$.

   Conversely, suppose that $R \subseteq R \cap S$, and let x be an element of R. Then x is an element of $R \cap S$, implying that it is an element of S. Since x was arbitrary, this gives that every element of R is an element of S, or $R \subseteq S$.

3. Proof: If $R = \emptyset$, then R is a subset of any set, including $S \setminus R$. Alternatively, if $R \neq \emptyset$, then R has at least one element x. But $S \setminus R$ contains only those elements y that are in S but not R; since x is in R, it isn't in $S \setminus R$, and $R \not\subseteq S \setminus R$.

A.2.2 A distributive law

Show that the following identities hold for all sets A, B, and C:

1. $A \times (B \cup C) = (A \times B) \cup (A \times C)$.
2. $A \times (B \cap C) = (A \times B) \cap (A \times C)$.

Solution

1. Let $(a, x) \in A \times (B \cup C)$. Then $a \in A$ and $x \in B \cup C$. If $x \in B$, then $(a, x) \in A \times B$; alternatively, if $x \in C$, then $(a, x) \in A \times C$. In either case, $(a, x) \in (A \times B) \cup (A \times C)$.

   Conversely, if $(a, x) \in (A \times B) \cup (A \times C)$, then either $(a, x) \in A \times B$ or $(a, x) \in A \times C$. In either case, $a \in A$. In the first case, $x \in B$, and in the second $x \in C$, giving $x \in B \cup C$ in either case as well. So $(a, x) \in A \times (B \cup C)$.

2. Let $(a, x) \in A \times (B \cap C)$. Then $a \in A$ and $x \in B \cap C$, giving $x \in B$ and $x \in C$. From $a \in A$ and $x \in B$ we have $(a, x) \in A \times B$; similarly from $a \in A$ and $x \in C$ we have $(a, x) \in A \times C$. So $(a, x) \in (A \times B) \cap (A \times C)$.

   Conversly, if $(a, x) \in (A \times B) \cap (A \times C)$, then $(a, x) \in A \times B$ and $(a, x) \in A \times C$. Both give $a \in A$, and the first gives $x \in B$ while the second gives $x \in C$. From this we can conclude that $x \in B \cap C$, so $(a, x) \in A \times (B \cap C)$.

A.2.3 Exponents

Let A be a set with $|A| = n > 0$. What is the size of each of the following sets of functions? Justify your answers.

1. $A^{\emptyset}$.
2. $\emptyset^{A}$.
3. $\emptyset^{\emptyset}$.

Solution

1. $\left|A^{\emptyset}\right| = 1$. Proof: There is exactly one function from $\emptyset$ to A (the empty function).

2. $\left|\emptyset^{A}\right| = 0$. Proof: There is no function from A to $\emptyset$, because A contains at least one element x, and any function $f : A \to \emptyset$ would have to map x to some $f(x) \in \emptyset$, a contradiction.

3. $\left|\emptyset^{\emptyset}\right| = 1$. Proof: This is just a special case of $A^{\emptyset}$; the empty function is the only function with $\emptyset$ as a domain. Note that this doesn't contradict the $\emptyset^{A}$ result, because there is no $x \in \emptyset$ that we fail to send anywhere.

A.3 Assignment 3: due Thursday, 2013-09-26, at 5:00 pm (Mike)

A.3.1 Surjections

Let $f : S \to T$ be surjective. Let $S' \subseteq S$, and let $T' = f(S') = \{f(x) \mid x \in S'\}$.

Prove or disprove: For any f, S, T, S' as above, there exists a surjection $g : S \setminus S' \to T \setminus T'$.

Solution

Disproof: Suppose $S \neq S'$ but $T = T'$; this can occur, for example, if $S = \{a, b\}$, $T = \{z\}$, $f(a) = f(b) = z$, and $S' = \{a\}$. In this case, $T' = T = \{z\}$, giving $T \setminus T' = \emptyset$. But $S \setminus S' = \{b\} \neq \emptyset$, and since there are no functions from a nonempty set to the empty set, there can't be a surjection $g : S \setminus S' \to T \setminus T'$.

The solution that got away

I will confess that when I wrote this problem, I forgot about the empty set case.[2]

Here is a proof that $g : S \setminus S' \to T \setminus T'$ exists when $T' \neq T$, which, alas, is not the actual claim:

Let $g(x) = f(x)$ for each x in $S \setminus S'$; in other words, g is the restriction of f to $S \setminus S'$.

[2]I am grateful to Josh Rosenfeld for noticing this issue.

Let $y \in T \setminus T'$. Because f is surjective, there exists x in S with $f(x) = y$. Fix some such x. We have $f(x) = y \notin T'$, so $x \notin S'$. It follows that x is in $S \setminus S'$.

We've just shown that for any $y \in T \setminus T'$, there is some $x \in S \setminus S'$ such that $f(x) = y$. But then $g(x) = f(x) = y$ as well, so g is surjective.

A.3.2 Proving an axiom the hard way

Recall that, if a, b, and c are all in $\mathbb{N}$, and $a \le b$, then $a + c \le b + c$ (Axiom 4.2.4).

For any two sets S and T, define $S \rightarrowtail T$ if there exists an injection $f : S \to T$. We can think of $\rightarrowtail$ as analogous to $\le$ for sets, because $|S| \le |T|$ if and only if $S \rightarrowtail T$.

Show that if $A \cap C = B \cap C = \emptyset$, and $A \rightarrowtail B$, then

$$A \cup C \rightarrowtail B \cup C.$$

Clarification added 2013-09-25 It's probably best not to try using the statement $|S| \le |T|$ if and only if $S \rightarrowtail T$ in your proof. While this is one way to define $\le$ for arbitrary cardinals, the odds are that your next step is to assert $|A| + |C| \le |B| + |C|$, and while we know that this works when A, B, and C are all finite (Axiom 4.2.4), that it works for arbitrary sets is what we are asking you to prove.

Solution

We'll construct an explicit injection $g : A \cup C \to B \cup C$. For each x in $A \cup C$, let

$$g(x) = \begin{cases} f(x) & \text{if } x \in A \text{, and} \\ x & \text{if } x \in C. \end{cases}$$

Observe that g is well-defined because every x in $A \cup C$ is in A or C but not both. Observe also that, since $B \cap C = \emptyset$, $g(x) \in B$ if and only if $x \in A$, and similarly $g(x) \in C$ if and only if $x \in C$.

We now show that g is injective. Let $g(x) = g(y)$. If $g(x) = g(y) \in B$, then x and y are both in A. In this case, $f(x) = g(x) = g(y) = f(y)$ and $x = y$ because f is injective. Alternatively, if $g(x) = g(y) \in C$, then $x = g(x) = g(y) = y$. In either case we have $x = y$, so g is injective.

A.3.3 Squares and bigger squares

Show directly from the axioms in Chapter 4 that $0 \leq a \leq b$ implies $a \cdot a \leq b \cdot b$, when a and b are real numbers.

Solution

Apply scaling invariance (Axiom 4.2.5) to $0 \leq a$ and $a \leq b$ to get $a \cdot a \leq a \cdot b$. Now apply scaling again to $0 \leq b$ and $a \leq b$ to get $a \cdot b \leq b \cdot b$. Finally, apply transitivity (Axiom 4.2.3) to combine $a \cdot a \leq a \cdot b$ and $a \cdot b \leq b \cdot b$ to get $a \cdot a \leq b \cdot b$.

A.4 Assignment 4: due Thursday, 2013-10-03, at 5:00 pm (Huamin)

A.4.1 A fast-growing function

Let $f : \mathbb{N} \to \mathbb{N}$ be defined by

$$\begin{aligned} f(0) &= 2, \\ f(n+1) &= f(n) \cdot f(n) - 1. \end{aligned}$$

Show that $f(n) > 2^n$ for all $n \in \mathbb{N}$.

Solution

The proof is by induction on n, but we have to be a little careful for small values. We'll treat $n = 0$ and $n = 1$ as special cases, and start the induction at 2.

For $n = 0$, we have $f(0) = 2 > 1 = 2^0$.

For $n = 1$, we have $f(1) = f(0) \cdot f(0) - 1 = 2 \cdot 2 - 1 = 3 > 2 = 2^1$.

For $n = 2$, we have $f(2) = f(1) \cdot f(1) - 1 = 3 \cdot 3 - 1 = 8 > 4 = 2^2$.

For the induction step, we want to show that, for all $n \geq 2$, if $f(n) > 2^n$, then $f(n+1) = f(n) \cdot f(n) - 1 > 2^{n+1}$. Compute

$$\begin{aligned} f(n+1) &= f(n) \cdot f(n) - 1 \\ &> 2^n \cdot 2^n - 1 \\ &= 2^n \cdot 4 - 1 \\ &= 2^{n+1} + 2^{n+1} - 1 \\ &> 2^{n+1}. \end{aligned}$$

The principle of induction gives us that $f(n) > 2^n$ for all $n \geq 2$, and we've already covered $n = 0$ and $n = 1$ as special cases, so $f(n) > 2^n$ for all $n \in \mathbb{N}$.

A.4.2 A slow-growing set

Let

$$\begin{aligned} A_0 &= \{3, 4, 5\}, \\ A_{n+1} &= A_n \cup \sum_{x \in A_n} x. \end{aligned}$$

Give a closed-form expression for $S_n = \sum_{x \in A_n} x$. Justify your answer.

Clarification added 2013-10-01 For the purpose of this problem, you may assume that $\sum_{x \in A} x + \sum_{x \in B} x = \sum_{x \in A \cup B} x$ if A and B are disjoint. (This is provable for finite sets in a straightforward way using induction on the size of B, but the proof is pretty tedious.)

Solution

Looking at the first couple of values, we see:

$$\begin{aligned} S_0 &= 3 + 4 + 5 = 12 \\ S_1 &= 3 + 4 + 5 + 12 = 24 \\ S_2 &= 3 + 4 + 5 + 12 + 24 = 48 \end{aligned}$$

It's pretty clear that the sum is doubling at each step. This suggests a reasonable guess would be $\sum_{x \in A_n} x = 12 \cdot 2^n$, which we've shown works for $n = 0$.

For the induction step, we need to show that when constructing $A_{n+1} = A_n \cup \{S_n\}$, we are in fact doubling the sum. There is a tiny trick here in that we have to be careful that S_n isn't already an element of A_n.

Lemma A.4.1. *For all n, $S_n \notin A_n$.*

Proof. First, we'll show by induction that $|A_n| > 1$ and that every element of A_n is positive.

For the first part, $|A_0| = 3 > 1$, and by construction $A_{n+1} \supseteq A_n$. It follows that $A_n \supseteq A_0$ for all n, and so $|A_n| \geq |A_0| > 1$ for all n.

For the second part, every element of A_0 is positive, and if every element of A_n is positive, then so is $S_n = \sum_{x \in A_n} x$. Since each element x of A_{n+1} is either an element of A_n or equal to S_n, it must be positive as well.

Now suppose $S_n \in A_n$. Then $S_n = S_n + \sum_{x \in A_n \setminus \{S_n\}} x$, but the sum is a sum of at least one positive value, so we get $S_n > S_n$, a contradiction. It follows that $S_n \notin A_n$. □

Having determined that $S_n \notin A_n$, we can compute, under the assumption that $S_n = 12 \cdot 2^n$,

$$\begin{aligned}
S_{n+1} &= \sum_{x \in A_{n+1}} x \\
&= \sum_{x \in A_n} x + S_n \\
&= S_n + S_n \\
&= 12 \cdot 2^n + 12 \cdot 2^n \\
&= 12 \cdot (2^n + 2^n) \\
&= 12 \cdot 2^{n+1}.
\end{aligned}$$

This completes the induction argument and the proof.

A.4.3 Double factorials

Recall that the **factorial** of n, written $n!$, is defined by

$$n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot 3 \cdot \ldots n. \tag{A.4.1}$$

The **double factorial** of n, written $n!!$, is defined by

$$n!! = \prod_{i=0}^{\lfloor (n-1)/2 \rfloor} (n - 2i). \tag{A.4.2}$$

For even n, this expands to $n \cdot (n-2) \cdot (n-4) \cdot \ldots 2$. For odd n, it expands to $n \cdot (n-2) \cdot (n-4) \cdot \ldots 1$.

Show that there exists some n_0, such that for all n in $\mathbb{N}$ with $n \geq n_0$,

$$(2n)!! \leq (n!)^2. \tag{A.4.3}$$

Solution

First let's figure out what n_0 has to be.

We have

$$\begin{aligned}
(2\cdot 0)!! &= 1 & (0!)^2 &= 1\cdot 1 = 1\\
(2\cdot 1)!! &= 2 & (1!)^2 &= 1\cdot 1 = 1\\
(2\cdot 2)!! &= 4\cdot 2 = 8 & (2!)^2 &= 2\cdot 2 = 4\\
(2\cdot 3)!! &= 6\cdot 4\cdot 2 = 48 & (3!)^2 &= 6\cdot 6 = 36\\
(2\cdot 4)!! &= 8\cdot 6\cdot 4\cdot 2 = 384 & (4!)^2 &= 24\cdot 24 = 576
\end{aligned}$$

So we might reasonably guess $n_0 = 4$ works.

Let's show by induction that $(2n)!! \le (n!)^2$ for all $n \ge 4$. We've already done the base case.

For the induction step, it will be helpful to simplify the expression for $(2n)!!$:

$$\begin{aligned}
(2n)!! &= \prod_{i=0}^{\lfloor (2n-1)/2\rfloor} (2n-2i)\\
&= \prod_{i=0}^{n-1} (2n-2i)\\
&= \prod_{i=1}^{n} 2i.
\end{aligned}$$

(The last step does a change of variables.)

Now we can compute, assuming $n \ge 4$ and that the induction hypothesis holds for n:

$$\begin{aligned}
(2(n+1))!! &= \prod_{i=1}^{n+1} 2i \\
&= \left(\prod_{i=1}^{n} 2i\right) \cdot (2(n+1)) \\
&= ((2n)!!) \cdot 2 \cdot (n+1) \\
&\le (n!)^2 \cdot (n+1) \cdot (n+1) \\
&= (n! \cdot (n+1))^2 \\
&= \left(\left(\prod_{i=1}^{n} i\right) \cdot (n+1)\right)^2 \\
&= ((n+1)!)^2 .
\end{aligned}$$

A.5 Assignment 5: due Thursday, 2013-10-10, at 5:00 pm (Xiao)

A.5.1 A bouncy function

Let $f : \mathbb{N} \to \mathbb{N}$ be defined by

$$f(n) = \begin{cases} 1 & \text{if } n \text{ is odd, and} \\ n & \text{if } n \text{ is even.} \end{cases}$$

1. Prove or disprove: $f(n)$ is $O(n)$.
2. Prove or disprove: $f(n)$ is $\Omega(n)$.

Solution

1. Proof: Recall that $f(n)$ is $O(n)$ if there exist constants $c > 0$ and N, such that $|f(n)| \le c \cdot |n|$ for $n \ge N$. Let $c = 1$ and $N = 1$. For any $n \ge 1$, either (a) $f(n) = 1 \le 1 \cdot n$, or (b) $f(n) = n \le 1 \cdot n$. So the definition is satisfied and $f(n)$ is $O(n)$.
2. Disproof: To show that $f(n)$ is not $\Omega(n)$, we need to show that for any choice of $c > 0$ and N, there exists some $n \ge N$ with $|f(n)| < c \cdot |n|$.

   Fix c and N. Let n be the smallest odd number greater than $\max(1/c, N)$ (such a number exists by the well-ordering principle). Then $n \ge N$, and since n is odd, we have $f(n) = 1$. But $c \cdot n > c \cdot \max(1/c, N) \geq c \cdot (1/c) = 1$. So $c \cdot n > f(n)$, concluding the disproof.

A.5.2 Least common multiples of greatest common divisors

Prove or disprove: For all $a, b, c \in \mathbb{Z}^+$,

$$\operatorname{lcm}(a, \gcd(b, c)) \mid \gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(a, c)).$$

Solution

Proof: Write r for the right-hand side. Observe that

$$\begin{aligned}
&a \mid \operatorname{lcm}(a, b), \text{and} \\
&a \mid \operatorname{lcm}(a, c), \text{so} \\
&a \mid \gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(a, c)) = r.
\end{aligned}$$

Similarly

$$\begin{aligned}
&\gcd(b, c) \mid b, \text{implying} \\
&\gcd(b, c) \mid \operatorname{lcm}(a, b), \text{and} \\
&\gcd(b, c) \mid c, \text{implying} \\
&\gcd(b, c) \mid \operatorname{lcm}(a, c), \text{which together give} \\
&\gcd(b, c) \mid \gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(a, c)) = r.
\end{aligned}$$

Since $a|r$ and $\gcd(b, c)|r$, from the definition of lcm we get $\operatorname{lcm}(a, \gcd(b, c))|r$.

A.5.3 Adding and subtracting

Let $a, b \in \mathbb{N}$ with $0 < a < b$.

1. Prove or disprove: $\gcd(a, b) = \gcd(b - a, b)$.
2. Prove or disprove: $\operatorname{lcm}(a, b) = \operatorname{lcm}(a, b - a)$.

Solution

1. Proof: Let $g = \gcd(a, b)$. Then $g|a$ and $g|b$, so $g|(b - a)$ as well. So g is a common divisor of $b - a$ and b. To show that it is the greatest common divisor, let $h|b$ and $h|(b - a)$. Then $h|a$ since $a = b + (b - a)$. It follows that $h| \gcd(a, b)$, which is g.
2. Disproof: Let $a = 2$ and $b = 5$. Then $\operatorname{lcm}(2, 5) = 10$ but $\operatorname{lcm}(5-2, 5) = \operatorname{lcm}(3, 5) = 15 \neq 10$.

A.6 Assignment 6: due Thursday, 2013-10-31, at 5:00 pm (Mike)

A.6.1 Factorials mod n

Let $n \in \mathbb{N}$. Recall that $n! = \prod_{i=1}^{n} i$. Show that, if n is composite and $n > 9$, then

$$\left\lfloor \frac{n}{2} \right\rfloor ! = 0 \pmod{n}.$$

Solution

Let n be composite. Then there exist natural numbers $a, b \geq 2$ such that $n = ab$. Assume without loss of generality that $a \leq b$.

For convenience, let $k = \lfloor n/2 \rfloor$. Since $b = n/a$ and $a \geq 2$, $b \leq n/2$; but b is an integer, so $b \leq n/2$ implies $b \leq \lfloor n/2 \rfloor = k$. It follows that both a and b are at most k.

We now consider two cases:

1. If $a \neq b$, then both a and b appear as factors in $k!$. So $k! = ab \prod_{1 \leq i \leq k, i \notin \{a,b\}} i$, giving $ab|k!$, which means $n|k!$ and $k! = 0 \pmod{n}$.

2. If $a = b$, then $n = a^2$. Since $n > 9$, we have $a > 3$, which means $a \geq 4$ since a is a natural number. It follows that $n \geq 4a$ and $k \geq 2a$. So a and $2a$ both appear in the product expansion of $k!$, giving $k! \bmod 2a^2 = 0$. But then $k! \bmod n = k! \bmod a^2 = (k! \bmod 2a^2) \bmod a^2 = 0$.

A.6.2 Indivisible and divisible

Prove or disprove: If A and B are non-empty, finite, disjoint sets of prime numbers, then there is a natural number n such that

$$n \bmod a = 1$$

for every a in A, and

$$n \bmod b = 0$$

for every b in B.

Solution

Proof: Let $m_1 = \prod_{a \in A} a$ and $m_2 = \prod_{b \in B} b$. Because A and B are disjoint, m_1 and m_2 have no common prime factors, and $\gcd(m_1, m_2) = 1$. So by the Chinese Remainder Theorem, there exists some n with $0 \leq n < m_1 m_2$ such that

$$\begin{aligned} n \bmod m_1 &= 1 \\ n \bmod m_2 &= 0 \end{aligned}$$

Then $n \bmod a = (n \bmod m_1) \bmod a = 1 \bmod a = 1$ for any a in A, and similarly $n \bmod b = (n \bmod m_2) \bmod b = 0 \bmod b = 0$ for any b in B.

(It's also possible to do this by applying the more general version of the CRT directly, since each pair of elements of A and B are relatively prime.)

A.6.3 Equivalence relations

Let A be a set, and let R and S be relations on A. Let T be a relation on A defined by xTy if and only if xRy and xSy.

Prove or disprove: If R and S are equivalence relations, then T is also an equivalence relation.

Solution

Proof: The direct approach is to show that T is reflexive, symmetric, and transitive:

1. Reflexive: For any x, xRx and xSx, so xTx.
2. Symmetric: Suppose xTy. Then xRy and xSy. Since R and S are symmetric, yRx and ySx. But then yTx.
3. Transitive: Let xTy and yTz. Then xRy and yRz implies xRz, and similarly xSy and ySz implies xSz. So xRz and xSz, giving xTz.

Alternative proof: It's also possible to show this using one of the alternative characterizations of an equivalence relation from Theorem 9.4.1.

Since R and S are equivalence relations, there exist sets B and C and functions $f : A \to B$ and $g : A \to C$ such that xRy if and only if $f(x) = f(y)$ and xSy if and only if $g(x) = g(y)$. Now consider the function $h : A \to B \times C$ defined by $h(x) = (f(x), g(x))$. Then $h(x) = h(y)$ if and only if $(f(x), g(x)) = (f(y), g(y))$, which holds if and only if $f(x) = f(y)$ and

$g(x) = g(y)$. But this last condition holds if and only if xRy and xSy, the definition of xTy. So we have $h(x) = h(y)$ if and only if xTy, and T is an equivalence relation.

A.7 Assignment 7: due Thursday, 2013-11-07, at 5:00 pm (Mike)

A.7.1 Flipping lattices with a function

Prove or disprove: For all lattices S and T, and all functions $f : S \to T$, if

$$f(x \vee y) = f(x) \wedge f(y) \qquad \text{for all } x, y \in S,$$

then

$$x \le y \to f(y) \le f(x) \qquad \text{for all } x, y \in S.$$

Solution

Let S, T, f be such that $f(x \vee y) = f(x) \wedge f(y)$ for all $x, y \in S$.

Now suppose that we are given some $x, y \in S$ with $x \le y$.

Recall that $x \vee y$ is the minimum z greater than or equal to both x and y; so when $x \le y$, $y \ge x$ and $y \ge y$, and for any z with $z \ge x$ and $z \ge y$, $z \ge y$, and $y = x \vee y$. From the assumption on f we have $f(y) = f(x \vee y) = f(x) \wedge f(y)$.

Now use the fact that $f(x) \wedge f(y)$ is less than or equal to both $f(x)$ and $f(y)$ to get $f(y) = f(x) \wedge f(y) \le f(x)$.

A.7.2 Splitting graphs with a mountain

Recall that a graph $G = (V, E)$ is **bipartite** if V can be partitioned into disjoint sets L and R such that every edge uv has one endpoint in L and the other in R.

A graph homomorphism $f : G \to G'$ from a graph $G = (V, E)$ to a graph $G' = (V', E')$ is a function $f : V \to V'$ such that, for every $uv \in E$, $f(u)f(v) \in E'$.

Prove or disprove: A graph G is bipartite if and only if there exists a graph homomorphism $f : G \to K_2$.

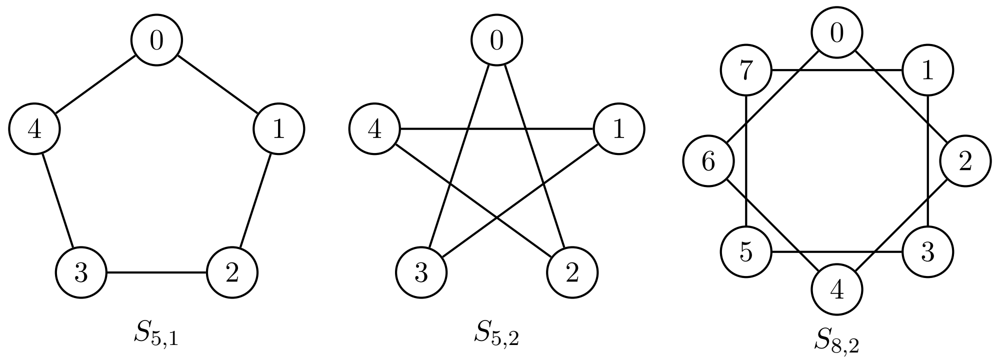


Figure A.1: Examples of $S_{m,k}$ for Problem A.7.3

Solution

Denote the vertices of K_2 by ℓ and r.

If G is bipartite, let L, R be a partition of V such that every edge has one endpoint in L and one in R, and let $f(x) = \ell$ if x is in L and $f(x) = r$ if x is in R.

Then if $uv \in E$, either $u \in L$ and $v \in R$ or vice versa; In either case, $f(u)f(v) = \ell r \in K_2$.

Conversely, suppose $f : V \to \{\ell, r\}$ is a homomorphism. Define $L = f^{-1}(\ell)$ and $R = f^{-1}(r)$; then L, R partition V. Furthermore, for any edge $uv \in E$, because $f(u)f(v)$ must be the unique edge ℓr, either $f(u) = \ell$ and $f(v) = r$ or vice versa. In either case, one of u, v is in L and the other is in R, so G is bipartite.

A.7.3 Drawing stars with modular arithmetic

For each pair of natural numbers m and k with $m \geq 2$ and $0 < k < m$, let $S_{m,k}$ be the graph whose vertices are the m elements of $\mathbb{Z}_m$ and whose edges consist of all pairs $(i, i + k)$, where the addition is performed mod m. Some examples are given in Figure A.1

Give a simple rule for determining, based on m and k, whether or not $S_{m,k}$ is connected, and prove that your rule works.

Solution

The rule is that $S_{m,k}$ is connected if and only if $\gcd(m, k) = 1$.

To show that this is the case, consider the connected component that contains 0; in other words, the set of all nodes v for which there is a path from 0 to v.

Lemma A.7.1. *There is a path from* 0 *to* v *in* $S_{m,k}$*, if and only if there is a number* a *such that* $v = ak \pmod{m}$*.*

Proof. To show that a exists when a path exists, we'll do induction on the length of the path. If the path has length 0, then $v = 0 = 0 \cdot k \pmod{m}$. If the path has length $n > 0$, let u be the last vertex on the path before v. By the induction hypothesis, $u = bk \pmod{m}$ for some b. There is an edge from u to v if and only if $v = u \pm k \pmod{m}$. So $v = bk \pm k = (bk \pm 1) \pmod{m}$.

Conversely, if there is some a such that $v = ak \pmod{m}$, then there is a path $0, k, \dots, ak$ from 0 to v in $S_{m,k}$. ☐

Now suppose $\gcd(m, k) = 1$. Then k has a multiplicative inverse mod m, so for any vertex v, letting $a = k^{-1}v \pmod{m}$ gives an a for which $ak = k^{-1}vk = v \pmod{m}$. So in this case, there is a path from 0 to every vertex in $S_{m,k}$, showing that $S_{m,k}$ is connected.

Alternatively, suppose that $\gcd(m, k) \neq 1$. Let $g = \gcd(m, k)$. Then if $v = ak \pmod{m}$, $v = ak - qm$ for some q, and since g divides both k and m it also divides $ak - qm$ and thus v. So there is no path from 0 to 1, since $g > 1$ implies g does not divide 1. This gives at least two connected components, and $S_{m,k}$ is not connected.

A.8 Assignment 8: due Thursday, 2013-11-14, at 5:00 pm (Huamin)

A.8.1 Two-path graphs

Define a **two-path graph** to be a graph consisting of exactly two disjoint paths, each containing one or more nodes. Given a particular vertex set of size n, we can consider the set of all two-path graphs on those vertices. For example, when $n = 2$, there is exactly one two-path graph, with one vertex in each length-0 path. When $n = 3$, there are three: each puts one vertex in a path by itself and the other two in a length-1 path. For larger n, the number of two-path graphs on a given set of n vertices grows quickly. For example, there are 15 two-path graphs on four vertices and 90 two-path graphs on five vertices (see Figure A.2).

Let $n \geq 3$. How many two-path graphs are there on n vertices?

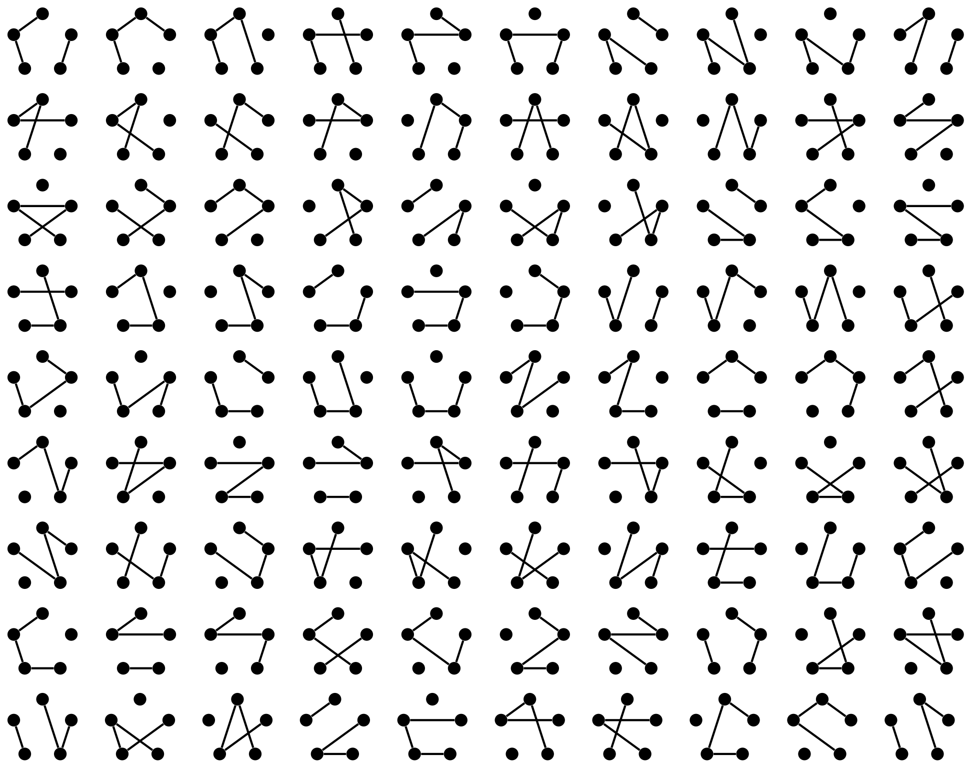


Figure A.2: All 90 two-path graphs on five vertices

Give a closed-form expression, and justify your answer.

A.8.2 Even teams

A group of sports astronomers on Gliese 667 Cc are trying to reconstruct American football. Based on sketchy radio transmissions from Earth, they have so far determined that it (a) involves ritualized violence, (b) sometimes involves tailgate parties, and (c) works best with even teams. Unfortunately, they are still confused about what even teams means.

Suppose that you have n candidate football players, that you pick k of them, and then split the k players into two teams (crimson and blue, say). As a function of n, how many different ways are there to do this with k being an even number?

For example, when $n = 2$, there are five possibilities for the pairs of teams: $(\emptyset, \emptyset), (\emptyset, \{a, b\}), (\{a\}, \{b\}), (\{b\}, \{a\})$, and $(\{a, b\}, \emptyset)$.

(Hint: Consider the difference between the number of ways to make k even and the number of ways to make k odd.)

Clarification added 2013-11-13: Ideally, your answer should be in closed form.

A.8.3 Inflected sequences

Call a sequence of three natural numbers a_0, a_1, a_2 **inflected** if $a_0 \geq a_1 \leq a_2$ or $a_0 \leq a_1 \geq a_2$.

As a function of n, how many such inflected sequences are there with $a_1, a_2, a_3 \in [n]$?

A.9 Assignment 9: due Thursday, 2013-11-21, at 5:00 pm (Xiao)

For problems that ask you to compute a value, closed-form expressions are preferred, and you should justify your answers.

A.9.1 Guessing the median

The median of a set S of n distinct numbers, when n is odd, is the element x of S with the property that exactly $(n-1)/2$ elements of S are less than x (which also implies that exactly $(n-1)/2$ elements of S are greater than x).

Consider the following speedy but inaccurate algorithm for guessing the median of a set S, when $n = |S|$ is odd and $n \geq 3$: choose a three-element subset R of S uniformly at random, then return the median of R. What is the probability, as a function of n, that the median of R is in fact the median of S?

Solution

There are $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$ choices for R, all of which are equally likely. So we want to count the number of sets R for which $\operatorname{median}(R) = \operatorname{median}(S)$.

Each such set contains $\operatorname{median}(S)$, one of the $(n-1)/2$ elements of S less than $\operatorname{median}(S)$, and one of the $(n-1)/2$ elements of S greater than $\operatorname{median}(S)$. So there are $\frac{(n-1)^2}{4}$ choices of R that cause the algorithm to work. The probability of picking one of these good sets is

$$\frac{(n-1)^2/4}{n(n-1)(n-2)/6} = \frac{3}{2} \cdot \frac{n-1}{n(n-2)}.$$

As a quick test, when $n = 3$, this evaluates to $\frac{3}{2} \cdot \frac{2}{3 \cdot 1} = 1$, which is what we'd expect given that there is only one three-element subset of S in this case. This is also a case where the algorithm works much better than the even dumber algorithm of just picking a single element of S at random, which succeeds with probability $1/n$, or $1/3$ in this case. For larger n the performance of median-of-three is less convincing, converging to a $\frac{3}{2} \cdot (1/n)$ probability of success in the limit.

A.9.2 Two flushes

A standard **poker deck** has 52 cards, which are divided into 4 suits of 13 cards each. Suppose that we shuffle a poker deck so that all 52! permutations are equally likely.[3] We then deal the top 5 cards to you and the next 5 cards to me.

Define a **flush** to be five cards from the same suit. Let A be the event that you get a flush, and let B be the event that I get a flush.

1. What is $\Pr[B \mid A]$?

2. Is this more or less than $\Pr[B]$?

[3]This turns out to be pretty hard to do in practice [BD92], but we'll suppose that we can actually do it.

Solution

Recall that

$$\Pr[B \mid A] = \frac{\Pr[B \cap A]}{\Pr[A]}.$$

Let's start by calculating $\Pr[A]$. For any single suit s, there are $(13)_5$ ways to give you 5 cards from s, out of $(52)_5$ ways to give you 5 cards, assuming in both cases that we keep track of the order of the cards.[4] So the event A_s that you get only cards in s has probability

$$\frac{(13)_5}{(52)_5}.$$

Since there are four suits, and the events A_s are disjoint for each suit, we get

$$\Pr[A] = \sum_s \Pr[A_s] = 4 \cdot \frac{(13)_5}{(52)_5}.$$

For $\Pr[A \cap B]$, let C_{st} be the event that your cards are all from suit s and mine are all from suit t. Then

$$\Pr[C_{st}] = \begin{cases} \frac{(13)_{10}}{(52)_{10}} & \text{if } s = t\text{, and} \\ \frac{(13)_5 \cdot (13)_5}{(52)_{10}} & \text{if } s \neq t. \end{cases}$$

Summing up all 16 cases gives

$$\Pr[A \cap B] = \frac{4 \cdot (13)_{10} + 12 \cdot (13)_5 \cdot (13)_5}{(52)_{10}}.$$

Now divide to get

$$\begin{aligned} \Pr[B \mid A] &= \frac{\left(\frac{4 \cdot (13)_{10} + 12 \cdot (13)_5 \cdot (13)_5}{(52)_{10}}\right)}{\left(\frac{4 \cdot (13)_5}{(52)_5}\right)} \\ &= \frac{\left(\frac{(13)_{10}}{(13)_5} + 3 \cdot (13)_5\right)}{\left(\frac{(52)_{10}}{(52)_5}\right)} \\ &= \frac{(8)_5 + 3 \cdot (13)_5}{(47)_5}. \end{aligned} \tag{A.9.1}$$

[4] If we don't keep track of the order, we get $\binom{13}{5}$ choices out of $\binom{52}{5}$ possibilities; these divide out to the same value.

Another way to get (A.9.1) is to argue that once you have five cards of a particular suit, there are $(47)_5$ equally probable choices for my five cards, of which $(8)_5$ give me five cards from your suit and $3 \cdot (13)_5$ give me five cards from one of the three other suits.

However we arrive at (A.9.1), we can evaluate it numerically to get

$$\begin{aligned}
\Pr[B \mid A] &= \frac{(8)_5 + 3 \cdot (13)_5}{(47)_5} \\
&= \frac{8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 + 3 \cdot (13 \cdot 12 \cdot 11 \cdot 10 \cdot 8)}{47 \cdot 46 \cdot 45 \cdot 44 \cdot 43} \\
&= \frac{6720 + 3 \cdot 154440}{184072680} \\
&= \frac{470040}{184072680} \\
&= \frac{3917}{1533939} \\
&\approx 0.00255356.
\end{aligned}$$

This turns out to be slightly larger than the probability that I get a flush without conditioning, which is

$$\begin{aligned}
\Pr[B] &= \frac{4 \cdot (13)_5}{(52)_5} \\
&= \frac{4 \cdot 154400}{311875200} \\
&= \frac{33}{16660} \\
&\approx 0.00198079.
\end{aligned}$$

So $\Pr[B \mid A]$ is greater than $\Pr[B]$.

This is a little surprising. If you have a flush, it seems like there should be fewer ways for me to make a flush. But what happens is that your flush in spades (say) actually makes my flush in a non-spade suit more likely—because there are fewer spades to dilute my potential heart, diamond, or club flush—and this adds more to my chances than the unlikelihood of getting a second flush in spaces subtracts.

A.9.3 Dice and more dice

Let n be a positive integer, and let $D_0, D_1, \ldots D_n$ be independent random variables representing n-sided dice. Formally, $\Pr[D_i = k] = 1/n$ for each i and each k in $\{1, \ldots, n\}$.

Let

$$S = \sum_{i=1}^{D_0} D_i.$$

As a function of n, what is $\mathrm{E}[S]$?

Solution

Expand

$$\begin{aligned}
\mathrm{E}[S] &= \mathrm{E}\left[\sum_{i=1}^{D_0} D_i\right] \\
&= \sum_{j=1}^{n} \left(\mathrm{E}\left[\sum_{i=1}^{D_0} D_i \,\middle|\, D_0 = j\right] \cdot \Pr[D_0 = j] \right) \\
&= \frac{1}{n} \sum_{j=1}^{n} \mathrm{E}\left[\sum_{i=1}^{j} D_i\right] \\
&= \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{j} \mathrm{E}[D_i] \\
&= \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{j} \left(\frac{n+1}{2}\right) \\
&= \frac{1}{n} \sum_{j=1}^{n} \left(j \cdot \frac{n+1}{2}\right) \\
&= \frac{n+1}{2n} \sum_{j=1}^{n} j \\
&= \frac{n+1}{2n} \cdot \frac{n(n+1)}{2} \\
&= \frac{(n+1)^2}{4}.
\end{aligned}$$

Appendix B

Exams

B.1 CS202 Exam 1, October 17th, 2013

Write your answers on the exam. Justify your answers. Work alone. Do not use any notes or books.

There are four problems on this exam, each worth 20 points, for a total of 80 points. You have approximately 75 minutes to complete this exam.

B.1.1 A tautology (20 points)

Use a truth table to show that the following is a tautology:

$$P \vee (P \leftrightarrow Q) \vee Q$$

Solution

P	Q	$P \leftrightarrow Q$	$P \vee (P \leftrightarrow Q)$	$P \vee (P \leftrightarrow Q) \vee Q$
0	0	1	1	1
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

B.1.2 A system of equations (20 points)

Show that the equations

$$\begin{aligned} x + y &= 0 \pmod{m} \\ x - y &= 0 \pmod{m} \end{aligned}$$

have exactly one solution with $0 \leq x, y < m$ if and only if m is odd.

Solution

Add the equations together to get

$$2x = 0 \pmod{m}$$

If m is odd, then $\gcd(m, 2) = 0$ and 2 has a multiplicative inverse in $\mathbb{Z}_m$ (which happens to be $(m+1)/2$). So we can multiply both sides of the equation by this inverse to get $x = 0 \pmod{m}$. Having determined x, we can then use either of the given equations to compute $y = 0 \pmod{m}$, giving the unique solution.

If m is even, then $x = y = 0$ and $x = y = m/2$ are two distinct solutions that satisfy the given equations.

B.1.3 A sum of products (20 points)

Let $n \in \mathbb{N}$. Give a closed-form expression for

$$\sum_{i=1}^{n} \prod_{j=1}^{i} 2.$$

Justify your answer.

Solution

Using the definition of exponentiation and the geometric series formula, we can compute

$$\begin{aligned}
\sum_{i=1}^{n} \prod_{j=1}^{i} 2 &= \sum_{i=1}^{n} 2^i \\
&= \sum_{i=0}^{n-1} 2^{i+1} \\
&= 2 \sum_{i=0}^{n-1} 2^i \\
&= 2 \cdot \frac{2^n - 1}{2 - 1} \\
&= 2 \cdot (2^n - 1) \\
&= 2^{n+1} - 2.
\end{aligned}$$

B.1.4 A subset problem (20 points)

Let A and B be sets. Show that if $A \cap C \subseteq B \cap C$ for all sets C, then $A \subseteq B$.

Solution

Suppose that for all C, $A \cap C \subseteq B \cap C$. In particular, let $C = A$. Then $A = A \cap A \subseteq B \cap A$. If $x \in A$, then $x \in B \cap A$, giving $x \in B$. So $A \subseteq B$. (Other choices for C also work.)

An alternative proof proceeds by contraposition: Suppose $A \not\subseteq B$. Then there is some x in A that is not in B. But then $A \cap \{x\} = \{x\}$ and $B \cap \{x\} = \emptyset$, so $A \cap \{x\} \not\subseteq B \cap \{x\}$.

Appendix C

Midterm exams from previous semesters

Note that topics covered may vary from semester to semester, so the appearance of a particular topic on one of these sample midterms does not necessarily mean that it may appear on a current exam.

C.1 Midterm Exam, October 12th, 2005

Write your answers on the exam. Justify your answers. Work alone. Do not use any notes or books.

There are four problems on this exam, each worth 20 points, for a total of 80 points. You have approximately 50 minutes to complete this exam.

C.1.1 A recurrence (20 points)

Give a simple formula for $T(n)$, where:

$$\begin{aligned} &T(0) = 1. \\ &T(n) = 3T(n-1) + 2^n, \text{when } n > 0. \end{aligned}$$

Solution

Using generating functions

Let $F(z) = \sum_{n=0}^{\infty} T(n) z^n$, then

$$F(z) = 3zF(z) + \frac{1}{1-2z}.$$

Solving for F gives

$$\begin{aligned} F(z) &= \frac{1}{(1-2z)(1-3z)} \\ &= \frac{-2}{1-2z} + \frac{3}{1-3z}. \end{aligned}$$

From the generating function we can immediately read off

$$T(n) = 3 \cdot 3^n - 2 \cdot 2^n = 3^{n+1} - 2^{n+1}.$$

Without using generating functions

It is possible to solve this problem without generating functions, but it's harder. Here's one approach based on forward induction. Start by computing the first few values of $T(n)$. We'll avoid reducing the expressions to make it easier to spot a pattern.

$$\begin{aligned} T(0) &= 1 \\ T(1) &= 3 + 2 \\ T(2) &= 3^2 + 3 \cdot 2 + 2^2 \\ T(3) &= 3^3 + 3^2 \cdot 2 + 3 \cdot 2^2 + 2^3 \\ T(4) &= 3^4 + 3^3 \cdot 2 + 3^2 \cdot 2^2 + 3 \cdot 2^3 + 2^4 \end{aligned}$$

At this point we might guess that

$$T(n) = \sum_{k=0}^{n} 3^{n-k} 2^k = 3^n \sum_{k=0}^{n} (2/3)^k = 3^n \left(\frac{1-(2/3)^{n+1}}{1-(2/3)} \right) = 3^{n+1} - 2^{n+1}.$$

A guess is not a proof; to prove that this guess works we verify $T(0) = 3^1 - 2^1 = 3 - 2 = 1$ and $T(n) = 3T(n-1) + 2^n = 3(3^n - 2^n) + 2^n = 3^{n+1} - 2 \cdot 2^n = 3^{n+1} - 2^{n+1}$.

C.1.2 An induction proof (20 points)

Prove by induction on n that $n! > 2^n$ for all integers $n \geq n_0$, where n_0 is an integer chosen to be as small as possible.

Solution

Trying small values of n gives $0! = 1 = 2^0$ (bad), $1! = 1 < 2^1$ (bad), $2! = 2 < 2^2$ (bad), $3! = 6 < 2^3$ (bad), $4! = 24 > 2^4 = 16$ (good). So we'll guess $n_0 = 4$ and use the $n = 4$ case as a basis.

For larger n, we have $n! = n(n-1)! > n2^{n-1} > 2 \cdot 2^{n-1} = 2^n$.

C.1.3 Some binomial coefficients (20 points)

Prove that $k\binom{n}{k} = n\binom{n-1}{k-1}$ when $1 \leq k \leq n$.

Solution

There are several ways to do this. The algebraic version is probably cleanest.

Combinatorial version

The LHS counts the way to choose k of n elements and then specially mark one of the k. Alternatively, we could choose the marked element first (n choices) and then choose the remaining $k-1$ elements from the remaining $n-1$ elements ($\binom{n-1}{k-1}$ choices); this gives the RHS.

Algebraic version

Compute $k\binom{n}{k} = k \cdot \frac{n!}{k!(n-k)!} = \frac{n!}{(k-1)!(n-k)!} = n \cdot \frac{(n-1)!}{(k-1)!((n-1)-(k-1))!} = n\binom{n-1}{k-1}$.

Generating function version

Observe that $\sum_{k=0}^{n} k\binom{n}{k}z^k = z\frac{d}{dz}(1+z)^n = zn(1+z)^{n-1} = \sum_{k=0}^{n-1} n\binom{n-1}{k}z^{k+1} = \sum_{k=1}^{n} n\binom{n-1}{k-1}z^k$. Now match z^k coefficients to get the desired result.

C.1.4 A probability problem (20 points)

Suppose you flip a fair coin n times, where $n \geq 1$. What is the probability of the event that both of the following hold: (a) the coin comes up heads at least once and (b) once it comes up heads, it never comes up tails on any later flip?

Solution

For each $i \in \{1 \ldots n\}$, let A_i be the event that the coin comes up heads for the first time on flip i and continues to come up heads thereafter. Then the desired event is the disjoint union of the A_i. Since each A_i is a single sequence of coin-flips, each occurs with probability 2^{-n}. Summing over all i gives a total probability of $n2^{-n}$.

C.2 Midterm Exam, October 24th, 2007

Write your answers on the exam. Justify your answers. Work alone. Do not use any notes or books.

There are four problems on this exam, each worth 20 points, for a total of 80 points. You have approximately 50 minutes to complete this exam.

C.2.1 Dueling recurrences (20 points)

Let $0 \leq S(0) \leq T(0)$, and suppose we have the recurrences

$$\begin{aligned} S(n+1) &= aS(n) + f(n) \\ T(n+1) &= bT(n) + g(n), \end{aligned}$$

where $0 \leq a \leq b$ and $0 \leq f(n) \leq g(n)$ for all $n \in \mathbb{N}$.

Prove that $S(n) \leq T(n)$ for all $n \in \mathbb{N}$.

Solution

We'll show the slightly stronger statement $0 \leq S(n) \leq T(n)$ by induction on n. The base case $n = 0$ is given.

Now suppose $0 \leq S(n) \leq T(n)$; we will show the same holds for $n+1$. First observe $S(n+1) = aS(n)+f(n) \geq 0$ as each variable on the right-hand side is non-negative. To show $T(n+1) \geq S(n+1)$, observe

$$\begin{aligned} T(n+1) &= bT(n) + g(n) \\ &\geq aT(n) + f(n) \\ &\geq aS(n) + f(n) \\ &= S(n+1). \end{aligned}$$

Note that we use the fact that $0 \leq T(n)$ (from the induction hypothesis) in the first step and $0 \leq a$ in the second. The claim does not go through without these assumptions, which is why using $S(n) \leq T(n)$ by itself as the induction hypothesis is not enough to make the proof work.

C.2.2 Seating arrangements (20 points)

A group of k students sit in a row of n seats. The students can choose whatever seats they wish, provided: (a) from left to right, they are seated in alphabetical order; and (b) each student has an empty seat immediately to his or her right.

For example, with 3 students `A`, `B`, and `C` and 7 seats, there are exactly 4 ways to seat the students: `A-B-C-`, `A-B-C-`, `A-B-C-`, and `-A-B-C-`.

Give a formula that gives the number of ways to seat k students in n seats according to the rules given above.

Solution

The basic idea is that we can think of each student and the adjacent empty space as a single width-2 unit. Together, these units take up $2k$ seats, leaving $n-2k$ extra empty seats to distribute between the students. There are a couple of ways to count how to do this.

Combinatorial approach

Treat each of the k student-seat blocks and $n-2k$ extra seats as filling one of $k+(n-2k)=n-k$ slots. There are exactly $\binom{n-k}{k}$ ways to do this.

Generating function approach

Write $z+z^2$ for the choice between a width-1 extra seat and a width-2 student-seat block. For a row of $n-k$ such things, we get the generating function $(z+z^2)^{n-k} = z^{n-k}(1+z)^{n-k} = z^{n-k}\sum_{i=0}^{n-k}\binom{n-k}{i}z^i = \sum_{i=0}^{n-k}\binom{n-k}{i}z^{n-k+i}$. The z^n coefficient is obtained when $i=k$, giving $\binom{n-k}{k}$ ways to fill out exactly n seats.

C.2.3 Non-attacking rooks (20 points)

Place n rooks at random on an $n\times n$ chessboard (i.e., an $n\times n$ grid), so that all $\binom{n^2}{n}$ placements are equally likely. What is the probability of the event that every row and every column of the chessboard contains exactly one rook?

Solution

We need to count how many placements of rooks there are that put exactly one rook per row and exactly one rook per column. Since we know that there is one rook per row, we can specify where these rooks go by choosing a unique column for each row. There are n choices for the first row, $n-1$ remaining for the second row, and so on, giving $n(n-1)\cdots 1 = n!$ choices altogether. So the probability of the event is $n!/\binom{n^2}{n} = (n^2-n)!/(n^2)!$.

C.2.4 Subsets (20 points)

Let $A \subseteq B$.

1. Prove or disprove: There exists an injection $f : A \to B$.
2. Prove or disprove: There exists a surjection $g : B \to A$.

Solution

1. Proof: Let $f(x) = x$. Then $f(x) = f(y)$ implies $x = y$ and f is injective.
2. Disproof: Let B be nonempty and let $A = \emptyset$. Then there is no function at all from B to A, surjective or not.

C.3 Midterm Exam, October 24th, 2008

Write your answers on the exam. Justify your answers. Work alone. Do not use any notes or books.

There are four problems on this exam, each worth 20 points, for a total of 80 points. You have approximately 50 minutes to complete this exam.

C.3.1 Some sums (20 points)

Let $a_0, a_1, \dots$ and $b_0, b_1, \dots$ be sequences such that for i in $\mathbb{N}$, $a_i \leq b_i$.

Let $A_i = \sum_{j=0}^{i} a_j$ and let $B_i = \sum_{j=0}^{i} b_j$.

Prove or disprove: For all i in $\mathbb{N}$, $A_i \leq B_i$.

Solution

Proof: By induction on i. For $i = 0$ we have $A_0 = a_0 \leq b_0 = B_0$. Now suppose $A_i \leq B_i$. Then $A_{i+1} = \sum_{j=0}^{i+1} a_j = \sum_{j=0}^{i} a_j + a_{i+1} = A_i + a_{i+1} \leq B_i + b_{i+1} = \sum_{j=0}^{i} b_j + b_{j+1} = \sum_{j=0}^{i+1} b_j = B_j$.

C.3.2 Nested ranks (20 points)

You are recruiting people for a secret organization, from a population of n possible recruits. Out of these n possible recruits, some subset M will be members. Out of this subset M, some further subset C will be members of the inner circle. Out of this subset C, some further subset X will be Exalted

Grand High Maharajaraja Panjandrums of Indifference. It is possible that any or all of these sets will be empty.

If the roster of your organization gives the members of the sets M, C, and X, and if (as usual) order doesn't matter within the sets, how many different possible rosters can you have?

Solution

There is an easy way to solve this, and a hard way to solve this.

Easy way: For each possible recruit x, we can assign x one of four states: non-member, member but not inner circle member, inner circle member but not EGHMPoI, or EGHMPoI. If we know the state of each possible recruit, that determines the contents of M, C, X and vice versa. It follows that there is a one-to-one mapping between these two representations, and that the number of rosters is equal to the number of assignments of states to all n potential recruits, which is 4^n.

Hard way: By repeated application of the binomial theorem. Expressing the selection process in terms of choosing nested subsets of m, c, and x members, the number of possible rosters is

$$\begin{aligned}
\sum_{m=0}^{n}\left\{\binom{n}{m}\sum_{c=0}^{m}\left[\binom{m}{c}\sum_{x=0}^{c}\binom{c}{x}\right]\right\} &= \sum_{m=0}^{n}\left\{\binom{n}{m}\sum_{c=0}^{m}\binom{m}{c}2^c\right\}\\
&= \sum_{m=0}^{n}\binom{n}{m}(1+2)^m\\
&= \sum_{m=0}^{n}\binom{n}{m}3^m\\
&= (1+3)^n\\
&= 4^n.
\end{aligned}$$

C.3.3 Nested sets (20 points)

Let A, B, and C be sets.

1. Prove or disprove: If $A \in B$, and $B \subseteq C$, then $A \subseteq C$.
2. Prove or disprove: If $A \subseteq B$, and $B \subseteq C$, then $A \subseteq C$.

Solution

1. Disproof: Let $A = \{\emptyset\}$, $B = \{A\} = \{\{\emptyset\}\}$, and $C = B$. Then $A \in B$ and $B \subseteq C$, but $A \not\subseteq C$, because $\emptyset \in A$ but $\emptyset \notin C$.

2. Proof: Let $x \in A$. Then since $A \subseteq B$, we have $x \in B$, and since $B \subseteq C$, we have $x \in C$. It follows that every x in A is also in C, and that A is a subset of C.

C.3.4 An efficient grading method (20 points)

A test is graded on a scale of 0 to 80 points. Because the grading is completely random, your grade can be represented by a random variable X with $0 \leq X \leq 80$ and $\mathrm{E}[X] = 60$.

1. What is the maximum possible probability that $X = 80$?
2. Suppose that we change the bounds to $20 \leq X \leq 80$, but $\mathrm{E}[X]$ is still 60. Now what is the maximum possible probability that $X = 80$?

Solution

1. Here we apply Markov's inequality: since $X \geq 0$, we have $\Pr[X \geq 80] \leq \frac{\mathrm{E}[X]}{80} = \frac{60}{80} = 3/4$. This maximum is achieved exactly by letting $X = 0$ with probability $1/4$ and 80 with probability $3/4$, giving $\mathrm{E}[X] = (1/4) \cdot 0 + (3/4) \cdot 80 = 60$.
2. Raising the minimum grade to 20 knocks out the possibility of getting 0, so our previous distribution doesn't work. In this new case we can apply Markov's inequality to $Y = X - 20 \geq 0$, to get $\Pr[X \geq 80] = \Pr[Y \geq 60] \leq \frac{\mathrm{E}[Y]}{60} = \frac{40}{60} = 2/3$. So the extreme case would seem to be that we get 20 with probability $1/3$ and 80 with probability $2/3$. It's easy to check that we then get $\mathrm{E}[X] = (1/3) \cdot 20 + (2/3) \cdot 80 = 180/3 = 60$. So in fact the best we can do now is a probability of $2/3$ of getting 80, less than we had before.

C.4 Midterm exam, October 21st, 2010

Write your answers on the exam. Justify your answers. Work alone. Do not use any notes or books.

There are four problems on this exam, each worth 20 points, for a total of 80 points. You have approximately 75 minutes to complete this exam.

C.4.1 A partial order (20 points)

Let $S \subseteq \mathbb{N}$, and for any $x, y \in \mathbb{N}$, define $x \preceq y$ if and only if there exists $z \in S$ such that $x + z = y$.

Show that if $\preceq$ is a partial order, then (a) 0 is in S and (b) for any x, y in S, $x + y$ is in S.

Solution

If $\preceq$ is a partial order, then by reflexivity we have $x \preceq x$ for any x. But then there exists $z \in S$ such that $x + z = x$, which can only happen if $z = 0$. Thus $0 \in S$.

Now suppose x and y are both in S. Then $0 + x = x$ implies $0 \preceq x$, and $x + y = x + y$ implies $x \preceq x + y$. Transitivity of $\preceq$ gives $0 \preceq x + y$, which occurs only if some z such that $0 + z = x + y$ is in S. The only such z is $x + y$, so $x + y$ is in S.

C.4.2 Big exponents (20 points)

Let p be a prime, and let $0 \leq a < p$. Show that $a^{2p-1} = a \pmod{p}$.

Solution

Write $a^{2p-1} = a^{p-1}a^{p-1}a$. If $a \neq 0$, Euler's Theorem (or Fermat's Little Theorem) says $a^{p-1} = 1 \pmod{p}$, so in this case $a^{p-1}a^{p-1}a = a \pmod{p}$. If $a = 0$, then (since $2p - 1 \neq 0$), $a^{2p-1} = 0 = a \pmod{p}$.

C.4.3 At the playground (20 points)

Let $L(x, y)$ represent the statement "x likes y" and let $T(x)$ represent the statement "x is tall," where x and y range over a universe consisting of all children on a playground. Let m be "Mary," one of the children.

1. Translate the following statement into predicate logic: "If x is tall, then Mary likes x if and only if x does not like x."

2. Show that if the previous statement holds, Mary is not tall.

Solution

1. $\forall x\,(T(x) \Rightarrow (L(m, x) \Leftrightarrow \neg L(x, x)))$.

2. Suppose the previous statement is true. Let $x = m$, then $T(m) \Rightarrow (L(m,m) \Leftrightarrow \neg L(m,m))$. But $L(m,m) \Leftrightarrow \neg L(m,m)$ is false, so $T(m)$ must also be false.

C.4.4 Gauss strikes back (20 points)

Give a closed-form formula for $\sum_{k=a}^{b} k$, assuming $0 \le a \le b$.

Solution

Here are three ways to do this:

1. Write $\sum_{k=a}^{b} k$ as $\sum_{k=1}^{b} k - \sum_{k=1}^{a-1} k$ and then use the formula $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ to get

$$\begin{aligned}\sum_{k=a}^{b} k &= \sum_{k=1}^{b} k - \sum_{k=1}^{a-1} k \\ &= \frac{b(b+1)}{2} - \frac{(a-1)a}{2} \\ &= \frac{b(b+1) - a(a-1)}{2}.\end{aligned}$$

2. Use Gauss's trick, and compute

$$\begin{aligned}2\sum_{k=a}^{b} k &= \sum_{k=a}^{b} k + \sum_{k=a}^{b} (b+a-k) \\ &= \sum_{k=a}^{b} (k+b+a-k) \\ &= \sum_{k=a}^{b} (b+a) \\ &= (b-a+1)(b+a).\end{aligned}$$

   Dividing both sides by 2 gives $\frac{(b-a+1)(b+a)}{2}$.

3. Write $\sum_{k=a}^{b} k$ as $\sum_{k=0}^{b-a} (a+k) = (b-a+1)a + \sum_{k=0}^{b-a} k$. Then use the sum formula as before to turn this into $(b-a+1)a + \frac{(b-a)(b-a+1)}{2}$.

Though these solutions appear different, all of them can be expanded to $\frac{b^2-a^2+a+b}{2}$.

Appendix D

Final exams from previous semesters

Note that topics may vary from semester to semester, so the appearance of a particular topic on one of these exams does not necessarily indicate that it will appear on the second exam for the current semester. Note also that these exams were designed for a longer time slot—and were weighted higher—than the current semester's exams; the current semester's exams are likely to be substantially shorter.

D.1 CS202 Final Exam, December 15th, 2004

Write your answers in the blue book(s). Justify your answers. Work alone. Do not use any notes or books.

There are seven problems on this exam, each worth 20 points, for a total of 140 points. You have approximately three hours to complete this exam.

D.1.1 A multiplicative game (20 points)

Consider the following game: A player starts with a score of 0. On each turn, the player rolls two dice, each of which is equally likely to come up 1, 2, 3, 4, 5, or 6. They then take the product xy of the two numbers on the dice. If the product is greater than 20, the game ends. Otherwise, they add the product to their score and take a new turn. The player's score at the end of the game is thus the sum of the products of the dice for all turns before the first turn on which they get a product greater than 20.

1. What is the probability that the player's score at the end of the game is zero?
2. What is the expectation of the player's score at the end of the game?

Solution

1. The only way to get a score of zero is to lose on the first roll. There are 36 equally probable outcomes for the first roll, and of these the six outcomes (4,6), (5,5), (5,6), (6,4), (6,5), and (6,6) yield a product greater than 20. So the probability of getting zero is $6/36 = 1/6$.

2. To compute the total expected score, let us first compute the expected score for a single turn. This is

$$\frac{1}{36}\sum_{i=1}^{6}\sum_{j=1}^{6} ij[ij \leq 20].$$

where $[ij \leq 20]$ is the indicator random variable for the event that $ij \leq 20$.

I don't know of a really clean way to evaluate the sum, but we can expand it as

$$\begin{aligned}
&\left(\sum_{i=1}^{3} i\right)\left(\sum_{j=1}^{6} j\right) + 4\sum_{j=1}^{5} j + 5\sum_{j=1}^{4} j + 6\sum_{j=1}^{3} j \\
&= 6 \cdot 21 + 4 \cdot 15 + 5 \cdot 10 + 6 \cdot 6 \\
&= 126 + 60 + 50 + 36 \\
&= 272.
\end{aligned}$$

So the expected score per turn is $272/36 = 68/9$.

Now we need to calculate the expected total score; call this value S. Assuming we continue after the first turn, the expected total score for the second and subsequent turns is also S, since the structure of the tail of the game is identical to the game as a whole. So we have

$$S = 68/9 + (5/6)S,$$

which we can solve to get $S = (6 \cdot 68)/9 = 136/3$.

D.1.2 An equivalence in space (20 points)

Let V be a k-dimensional vector space over the real numbers $\mathbf{R}$ with a standard basis $\vec{x}_i$. Recall that any vector $\vec{z}$ in V can be represented uniquely as $\sum_{i=1}^k z_i\vec{x}_i$. Let $f : V \to \mathbf{R}$ be defined by $f(\vec{z}) = \sum_{i=1}^k |z_i|$, where the z_i are the coefficients of $\vec{z}$ in the standard representation. Define a relation $\sim$ on $V \times V$ by $\vec{z}_1 \sim \vec{z}_2$ if and only if $f(\vec{z}_1) = f(\vec{z}_2)$. Show that $\sim$ is an equivalence relation, i.e., that it is reflexive, symmetric, and transitive.

Solution

Both the structure of the vector space and the definition of f are irrelevant; the only fact we need is that $\vec{z}_1 \sim \vec{z}_2$ if and only if $f(\vec{z}_1) = f(\vec{z}_2)$. Thus for all $\vec{z}$, $\vec{z} \sim \vec{z}$ since $f(\vec{z}) = f(\vec{z}$ (reflexivity); for all $\vec{y}$ and $\vec{z}$, if $\vec{y} \sim \vec{z}$, then $f(\vec{y}) = f(\vec{z})$ implies $f(\vec{z}) = f(\vec{y})$ implies $\vec{z} \sim \vec{y}$ (symmetry); and for all $\vec{x}$, $\vec{y}$, and $\vec{z}$, if $\vec{x} \sim \vec{y}$ and $\vec{y} \sim \vec{z}$, then $f(\vec{x}) = f(\vec{y})$ and $f(\vec{y}) = f(\vec{z})$, so $f(\vec{x}) = f(\vec{z})$ and $\vec{x} \sim \vec{z}$ (transitivity).

D.1.3 A very big fraction (20 points)

Use the fact that $p = 2^{24036583} - 1$ is prime to show that

$$\frac{9^{2^{24036582}} - 9}{2^{24036583} - 1}$$

is an integer.

Solution

Let's save ourselves a lot of writing by letting $x = 24036583$, so that $p = 2^x - 1$ and the fraction becomes

$$\frac{9^{2^{x-1}} - 9}{p}.$$

To show that this is an integer, we need to show that p divides the denominator, i.e., that

$$9^{2^{x-1}} - 9 = 0 \pmod{p}.$$

We'd like to attack this with Fermat's Little Theorem, so we need to get the exponent to look something like $p - 1 = 2^x - 2$. Observe that $9 = 3^2$, so

$$9^{2^{x-1}} = (3^2)^{2^{x-1}} = 3^{2^x} = 3^{2^x-2} \cdot 3^2 = 3^{p-1} \cdot 3^2.$$

But $3^{p-1} = 1 \pmod{p}$, so we get $9^{2^{x-1}} = 3^2 = 9 \pmod{p}$, and thus $9^{2^{x-1}} - 9 = 0 \pmod{p}$ as desired.

D.1.4 A pair of odd vertices (20 points)

Let G be a simple undirected graph (i.e., one with no self-loops or parallel edges), and let u be a vertex in G with odd degree. Show that there is another vertex $v \neq u$ in G such that (a) v also has odd degree, and (b) there is a path from u to v in G.

Solution

Let G' be the connected component of u in G. Then G' is itself a graph, and the degree of any vertex is the same in G' as in G. Since the sum of all the degrees of vertices in G' must be even by the Handshaking Lemma, there cannot be an odd number of odd-degree vertices in G', and so there is some v in G' not equal to u that also has odd degree. Since G' is connected, there exists a path from u to v.

D.1.5 How many magmas? (20 points)

Recall that a *magma* is an algebra consisting of a set of elements and one binary operation, which is not required to satisfy any constraints whatsoever except closure. Consider a set S of n elements. How many distinct magmas are there that have S as their set of elements?

Solution

Since the carrier is fixed, we have to count the number of different ways of defining the binary operation. Let's call the operation f. For each ordered pair of elements $(x, y) \in S \times S$, we can pick any element $z \in S$ for the value of $f(x, y)$. This gives n choices for each of the n^2 pairs, which gives n^{n^2} magmas on S.

D.1.6 A powerful relationship (20 points)

Recall that the powerset $\mathcal{P}(S)$ of a set S is the set of sets $\{A : A \subseteq S\}$. Prove that if $S \subseteq T$, then $\mathcal{P}(S) \subseteq \mathcal{P}(T)$.

Solution

Let $A \in \mathcal{P}(S)$; then by the definition of $\mathcal{P}(S)$ we have $A \subseteq S$. But then $A \subseteq S \subseteq T$ implies $A \subseteq T$, and so $A \in \mathcal{P}(T)$. Since A was arbitrary, $A \in \mathcal{P}(T)$ holds for all A in $\mathcal{P}(S)$, and we have $\mathcal{P}(S) \subseteq \mathcal{P}(T)$.

D.1.7 A group of archaeologists (20 points)

Archaeologists working deep in the Upper Nile Valley have discovered a curious machine, consisting of a large box with three levers painted red, yellow, and blue. Atop the box is a display that shows one of set of n hieroglyphs. Each lever can be pushed up or down, and pushing a lever changes the displayed hieroglyph to some other hieroglyph. The archaeologists have determined by extensive experimentation that for each hieroglyph x, pushing the red lever *up* when x is displayed always changes the display to the same hieroglyph $f(x)$, and pushing the red lever *down* always changes hieroglyph $f(x)$ to x. A similar property holds for the yellow and blue levers: pushing yellow up sends x to $g(x)$ and down sends $g(x)$ to x; and pushing blue up sends x to $h(x)$ and down sends $h(x)$ to x.

Prove that there is a finite number k such that no matter which hieroglyph is displayed initially, pushing any one of the levers up k times leaves the display with the same hieroglyph at the end.

Clarification added during exam: $k > 0$.

Solution

Let H be the set of hieroglyphs, and observe that the map $f : H \to H$ corresponding to pushing the red lever up is invertible and thus a permutation. Similarly, the maps g and h corresponding to yellow or blue up-pushes are also permutations, as are the inverses f^-1, g^-1, and h^-1 corresponding to red, yellow, or blue down-pushes. Repeated pushes of one or more levers correspond to compositions of permutations, so the set of all permutations obtained by sequences of zero or more pushes is the subgroup G of the permutation group $S_{|H|}$ generated by f, g, and h.

Now consider the cyclic subgroup $\langle f \rangle$ of G generated by f alone. Since G is finite, there is some index m such that $f^m = e$. Similarly there are indices n and p such that $g^n = e$ and $h^p = e$. So pushing the red lever up any multiple of k times restores the initial state, as does pushing the yellow lever up any multiple of n times or the blue lever up any multiple of p times. Let $k = mnp$. Then k is a multiple of m, n, and p, and pushing any single lever up k times leaves the display in the same state.

D.2 CS202 Final Exam, December 16th, 2005

Write your answers in the blue book(s). Justify your answers. Work alone. Do not use any notes or books.

There are six problems on this exam, each worth 20 points, for a total of 120 points. You have approximately three hours to complete this exam.

D.2.1 Order (20 points)

Recall that the *order* of an element x of a group is the least positive integer k such that $x^k = e$, where e is the identity, or ∞ if no such k exists.

Prove or disprove: In the symmetric group S_n of permutations on n elements, the order of any permutation is at most $\binom{n}{2}$.

Clarifications added during exam

- Assume $n > 2$.

Solution

Disproof: Consider the permutation (1 2)(3 4 5)(6 7 8 9 10)(11 12 13 14 15 16 17) in S_{17}. This has order $2 \cdot 3 \cdot 5 \cdot 7 = 210$ but $\binom{17}{2} = \frac{17 \cdot 16}{2} = 136$.

D.2.2 Count the subgroups (20 points)

Recall that the *free group* over a singleton set $\{a\}$ consists of all words of the form a^k, where k is an integer, with multiplication defined by $a^k a^m = a^{k+m}$.

Prove or disprove: The free group over $\{a\}$ has exactly one finite subgroup.

Solution

Proof: Let F be the free group defined above and let S be a subgroup of F. Suppose S contains a^k for some $k \neq 0$. Then S contains $a^{2k}, a^{3k}, \ldots$ because it is closed under multiplication. Since these elements are all distinct, S is infinite.

The alternative is that S does not contain a^k for any $k \neq 0$; this leaves only a^0 as possible element of S, and there is only one such subgroup: the trivial subgroup $\{a^0\}$.

D.2.3 Two exits (20 points)

Let $G = (V, E)$ be a nonempty connected undirected graph with no self-loops or parallel edges, in which every vertex has degree 4. Prove or disprove: For any partition of the vertices V into two nonempty non-overlapping subsets

S and T, there are at least two edges that have one endpoint in S and one in T.

Solution

Proof: Because G is connected and every vertex has even degree, there is an Euler tour of the graph (a cycle that uses every edge exactly once). Fix some particular tour and consider a partition of V into two sets S and T. There must be at least one edge between S and T, or G is not connected; but if there is only one, then the tour can't return to S or T once it leaves. It follows that there are at least 2 edges between S and T as claimed.

D.2.4 Victory (20 points)

A sabermetrician wishes to test the hypothesis that a set of n baseball teams are stricty ranked, so that no two teams have the same rank and if some team A has a higher rank than some team B, A will always beat B in a 7-game series. To test this hypothesis, the sabermetrician has each team play a 7-game series against each other team.

Suppose that the teams are in fact all equally incompetent and that the winner of each series is chosen by an independent fair coin-flip. What is the probability that the results will nonetheless be consistent with some strict ranking?

Solution

Each ranking is a total order on the n teams, and we can describe such a ranking by giving one of the $n!$ permutations of the teams. These in turn generate $n!$ distinct outcomes of the experiment that will cause the sabermetrician to believe the hypothesis. To compute the probability that one of these outcomes occurs, we must divide by the total number of outcomes, giving

$$\Pr[\text{strict ranking}] = \frac{n!}{2^{\binom{n}{2}}}.$$

D.2.5 An aggressive aquarium (20 points)

A large number of juvenile piranha, weighing 1 unit each, are placed in an aquarium. Each day, each piranha attempts to eat one other piranha. If successful, the eater increases its weight to the sum of its previous weight and the weight of its meal (and the eaten piranha is gone); if unsuccessful, the piranha remains at the same weight.

Prove that after k days, no surviving piranha weighs more than 2^k units.

Clarifications added during exam

- It is not possible for a piranha to eat and be eaten on the same day.

Solution

By induction on k. The base case is $k = 0$, when all piranha weigh exactly $2^0 = 1$ unit. Suppose some piranha has weight $x \leq 2^k$ after k days. Then either its weight stays the same, or it successfully eats another piranha of weight $y \leq 2^k$ increases its weight to $x + y \leq 2^k + 2^k = 2^{k+1}$. In either case the claim follows for $k + 1$.

D.2.6 A subspace of matrices (20 points)

Recall that a subspace of a vector space is a set that is closed under vector addition and scalar multiplication. Recall further that the subspace *generated* by a set of vector space elements is the smallest such subspace, and its *dimension* is the size of any basis of the subspace.

Let A be the 2-by-2 matrix

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

over the reals, and consider the subspace S of the vector space of 2-by-2 real matrices generated by the set $\{A, A^2, A^3, \ldots\}$. What is the dimension of S?

Solution

First let's see what A^k looks like. We have

$$A^2 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$$

and in general we can show by induction that

$$A^k = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & k-1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}.$$

Observe now that for any k,

$$A^k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix} = (k-1)\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} - (k-2)\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (k-1)A^2 - (k-2)A.$$

It follows that $\{A, A^2\}$ generates all the A^k and thus generates any linear combination of the A^k as well. It is easy to see that A and A^2 are linearly independent: if $c_1A + c_2A^2 = 0$, we must have (a) $c_1 + c_2 = 0$ (to cancel out the diagonal entries) and (b) $c_1 + 2c_2 = 0$ (to cancel out the nonzero off-diagonal entry). The only solution to both equations is $c_1 = c_2 = 0$.

Because $\{A, A^2\}$ is a linearly independent set that generates S, it is a basis, and S has dimension 2.

D.3 CS202 Final Exam, December 20th, 2007

Write your answers in the blue book(s). Justify your answers. Work alone. Do not use any notes or books.

There are six problems on this exam, each worth 20 points, for a total of 120 points. You have approximately three hours to complete this exam.

D.3.1 A coin-flipping problem (20 points)

A particularly thick and lopsided coin comes up heads with probability p_H, tails with probability p_T, and lands on its side with probability $p_S = 1 - (p_H + p_T)$. Suppose you flip the coin repeatedly. What is the probability that it comes up heads twice in a row at least once before the first time it comes up tails?

Solution

Let p be the probability of the event W that the coin comes up heads twice before coming up tails. Consider the following mutually-exclusive events for the first one or two coin-flips:

Event A	$\Pr[A]$	$\Pr[W\mid A]$
HH	p_H^2	1
HT	$p_H p_T$	0
HS	$p_H p_S$	p
T	p_T	0
S	p_S	p

Summing over all cases gives

$$p = p_H^2 + p_H p_S p + p_S p,$$

which we can solve for p to get

$$p = \frac{p_H^2}{1 - p_H p_S - p_S} = \frac{p_H^2}{p_H + p_T - p_H p_S} = \frac{p_H^2}{p_T + p_H(p_H + p_T)} = \frac{p_H^2}{p_T + p_H p_T + p_H^2}.$$

(Any of these is an acceptable answer.)

D.3.2 An ordered group (20 points)

Let G be a group and $\leq$ a partial order on the elements of G such that for all x, y in G, $x \leq xy$. How many elements does G have?

Solution

The group G has exactly one element.

First observe that G has at least one element, because it contains an identity element e.

Now let x and y be any two elements of G. We can show $x \leq y$, because $y = x(x^{-1}y)$. Similarly, $y \leq x = y(y^{-1}x)$. But then $x = y$ by antisymmetry. It follows that all elements of G are equal, i.e., that G has at most one element.

D.3.3 Weighty vectors (20 points)

Let the *weight* $w(x)$ of an $n \times 1$ column vector x be the number of nonzero elements of x. Call an $n \times n$ matrix A *near-diagonal* if it has at most one nonzero off-diagonal element; i.e., if there is at most one pair of indices i, j such that $i \neq j$ and $A_{ij} \neq 0$.

Given n, what is the smallest value k such that there exists an $n \times 1$ column vector x with $w(x) = 1$ and a sequence of k $n \times n$ near-diagonal matrices $A_1, A_2, \ldots A_k$ such that $w(A_1 A_2 \cdots A_k x) = n$?

Solution

Let's look at the effect of multiplying a vector of known weight by just one near-diagonal matrix. We will show: (a) for any near-diagonal A and any x, $w(Ax) \leq w(x)+1$, and (b) for any $n \times 1$ column vector x with $0 < w(x) < n$, there exists a near-diagonal matrix A with $w(Ax) \geq w(x) + 1$.

To prove (a), observe that $(Ax)_i = \sum_{j=1}^{n} A_{ij} x_j$. For $(Ax)_i$ to be nonzero, there must be some index j such that $A_{ij} x_j$ is nonzero. This can occur in two ways: $j = i$, and A_{ii} and x_i are both nonzero, or $j \neq i$, and A_{ij} and x_j are both nonzero. The first case can occur for at most $w(x)$ different values of i (because there are only $w(x)$ nonzero entries x_i). The second can occur for at most one value of i (because there is at most one nonzero entry A_{ij} with $i \neq j$). It follows that Ax has at most $w(x) + 1$ nonzero entries, i.e., that $w(Ax) \leq w(x) + 1$.

To prove (b), choose k and m such that $x_k = 0$ and $x_m \neq 0$, and let A be the matrix with $A_{ii} = 1$ for all i, $A_{km} = 1$, and all other entries equal to zero. Now consider $(Ax)_i$. If $i \neq k$, then $(Ax)_i = \sum_{j=1}^{n} A_{ij} x_j = A_{ii} x_i = x_i$. If $i = k$, then $(Ai)_k = \sum_{j=1}^{n} A_{ij} x_j = A_{kk} x_k + A_{km} x_m = x_m \neq 0$, since we chose k so that $a_k = 0$ and chose m so that $a_m \neq 0$. So $(Ax)_i$ is nonzero if either x_i is nonzero or $i = k$, giving $w(Ax) \geq w(x) + 1$.

Now proceed by induction:

For any k, if $A_1 \dots A_k$ are near-diagonal matrices, then $w(A_1 \cdots A_k x) \leq w(x)+k$. Proof: The base case of $k = 0$ is trivial. For larger k, $w(A_1 \cdots A_k x) = w(A_1(A_2 \cdots A_k x)) \leq w(A_2 \cdots A_k x) + 1 \leq w(x) + (k-1) + 1 = w(x) + k$.

Fix x with $w(x) = 1$. Then for any $k < n$, there exists a sequence of near-diagonal matrices $A_1 \dots A_k$ such that $w(A_1 \cdots A_k x) = k + 1$. Proof: Again the base case of $k = 0$ is trivial. For larger $k < n$, we have from the induction hypothesis that there exists a sequence of $k - 1$ near-diagonal matrices $A_2 \dots A_k$ such that $w(A_2 \dots A_k x) = k < n$. From claim (b) above we then get that there exists a near-diagonal matrix A_1 such that $w(A_1(A_2 \dots A_k x)) = w(A_2 \dots A_k x) + 1 = k + 1$.

Applying both these facts, setting $k = n - 1$ is necessary and sufficient for $w(A_1 \dots A_k x) = n$, and so $k = n - 1$ is the smallest value of k for which this works.

D.3.4 A dialectical problem (20 points)

Let S be a set with n elements. Recall that a relation R is *symmetric* if xRy implies yRx, *antisymmetric* if xRy and yRx implies $x = y$, *reflexive* if xRx for all x, and *irreflexive* if $\neg(xRx)$ for all x.

1. How many relations on S are symmetric, antisymmetric, and reflexive?
2. How many relations on S are symmetric, antisymmetric, and irreflexive?
3. How many relations on S are symmetric and antisymmetric?

Solution

Since in all three cases we are considering symmetric antisymmetric relations, we observe first that if R is such a relation, then xRy implies yRx which in turn implies $x = y$. So any such R can have xRy only if $x = y$.

1. Let R be symmetric, antisymmetric, and reflexive. We have already established that xRy implies $x = y$. Reflexivity says $x = y$ implies xRy, so we have xRy iff $x = y$. Since this fully determines R, there is exactly 1 such relation.
2. Now let R be symmetric, antisymmetric, and irreflexive. For $x \neq y$ we have $\neg(xRy)$ (from symmetry+antisymmetry); but for $x = y$, we again have $\neg(xRy)$ (from irreflexivivity). So R is the empty relation, and again there is exactly 1 such relation.
3. Now for each x there is no constraint on whether xRx holds or not, but we still have $\neg(xRy)$ for $x \neq y$. Since we can choose whether xRx holds independently for each x, we have n binary choices giving 2^n possible relations.

D.3.5 A predictable pseudorandom generator (20 points)

Suppose you are given a pseudorandom number generator that generates a sequence of values $x_0, x_1, x_2, \dots$ by the rule $x_{i+1} = (ax_i + b) \bmod p$, where p is a prime and a, b, and x_0 are arbitrary integers in the range $0 \dots p-1$. Suppose further that you know the value of p but that a, b, and x_0 are secret.

1. Prove that given any three consecutive values x_i, x_{i+1}, x_{i+2}, it is possible to compute both a and b, provided $x_i \neq x_{i+1}$.
2. Prove that given only two consecutive values x_i and x_{i+1}, it is impossible to determine a.

Solution

1. We have two equations in two unknowns:

$$\begin{aligned} ax_i + b &= x_{i+1} \pmod p \\ ax_{i+1} + b &= x_{i+2} \pmod p. \end{aligned}$$

   Subtracting the second from the first gives

$$a(x_i - x_{i+1}) = x_{i+1} - x_{i+2} \pmod p.$$

   If $x_i \neq x_{i+1}$, then we can multiply both sides by $(x_i - x_{i+1})^{-1}$ to get

$$a = (x_{i+1} - x_{i+2})(x_i - x_{i+1})^{-1} \pmod p.$$

   Now we have a. To find b, plug our value for a into either equation and solve for b.

2. We will show that for any observed values of x_i and x_{i+1}, there are at least two different values for a that are consistent with our observation; in fact, we'll show the even stronger fact that for *any* value of a, x_i and x_{i+1} are consistent with that choice of a. Proof: Fix a, and let $b = x_{i+1} - ax_i \pmod p$. Then $x_{i+1} = ax_i + b \pmod p$.

D.3.6 At the robot factory (20 points)

Each robot built by Rossum's Combinatorial Robots consists of a head and a body, each weighing a non-negative integer number of units. If there are exactly 3^n different ways to build a robot with total weight n, and exactly 2^n different bodies with weight n, exactly how many different heads are there with weight n?

Solution

This is a job for generating functions!

Let $R = \sum 3^n z^n = \frac{1}{1-3z}$ be the generating function for the number of robots of each weight, and let $B = \sum 2^n z^n = \frac{1}{1-2z}$ be the generating function for the number of bodies of each weight. Let $H = \sum h_n z^n$ be the generating function for the number of heads. Then we have $R = BH$, or

$$H = \frac{R}{B} = \frac{1-2z}{1-3z} = \frac{1}{1-3z} - \frac{2z}{1-3z}.$$

So $h_0 = 3^0 = 1$, and for $n > 0$, we have $h_n = 3^n - 2 \cdot 3^{n-1} = (3-2)3^{n-1} = 3^{n-1}$.

D.4 CS202 Final Exam, December 19th, 2008

Write your answers in the blue book(s). Justify your answers. Work alone. Do not use any notes or books.

There are five problems on this exam, each worth 20 points, for a total of 100 points. You have approximately three hours to complete this exam.

D.4.1 Some logical sets (20 points)

Let A, B, and C be sets.

Prove or disprove: If, for all x, $x \in A \to (x \in B \to x \in C)$, then $A \cap B \subseteq C$.

Solution

Proof: Rewrite $x \in A \to (x \in B \to x \in C)$ as $x \notin A \vee (x \notin B \vee x \in C)$ or $(x \notin A \vee x \notin B) \vee x \in C$. Applying De Morgan's law we can convert the first OR into an AND to get $\neg(x \in A \wedge x \in B) \vee x \in C$. This can further be rewritten as $(x \in A \wedge x \in B) \to x \in C$.

Now suppose that this expression is true for all x and consider some x in $A \cap B$. Then $x \in A \wedge x \in B$ is true. It follows that $x \in C$ is also true. Since this holds for every element x of $A \cap B$, we have $A \cap B \subseteq C$.

D.4.2 Modularity (20 points)

Let m be an integer greater than or equal to 2. For each a in $\mathbb{Z}_m$, let $f_a : \mathbb{Z}_m \to \mathbb{Z}_m$ be the function defined by the rule $f_a(x) = ax$.

Show that f_a is a bijection if and only if $\gcd(a, m) = 1$.

Solution

From the extended Euclidean algorithm we have that if $\gcd(a, m) = 1$, then there exists a multiplicative inverse a^{-1} such that $a^{-1}ax = x \pmod{m}$ for all x in $\mathbb{Z}_m$. It follows that f_a has an inverse function $f_{a^{-1}}$, and is thus a bijection.

Alternatively, suppose $\gcd(a, m) = g \neq 1$. Then $f_a(m/g) = am/g = m(a/g) = 0 = a \cdot 0 = f_a(0) \pmod{m}$ but $m/g \neq 0 \pmod{m}$ since $0 < m/g < m$. It follows that f_a is not injective and thus not a bijection.

D.4.3 Coin flipping (20 points)

Take a biased coin that comes up heads with probability p and flip it $2n$ times.

What is the probability that at some time during this experiment two consecutive coin-flips come up both heads or both tails?

Solution

It's easier to calculate the probability of the event that we never get two consecutive heads or tails, since in this case there are only two possible patterns of coin-flips: $HTHT\ldots$ or $THTH\ldots$. Since each of these patterns contains exactly n heads and n tails, they occur with probability $p^n(1-p)^n$, giving a total probability of $2p^n(1-p)^n$. The probability that neither sequence occurs is then $1-2p^n(1-p)^n$.

D.4.4 A transitive graph (20 points)

Let G be a graph with n vertices on which the adjacency relation is transitive: whenever there is an edge uv and an edge vw, there is also an edge uw. Suppose further that G is connected. How many edges does G have?

Solution

The graph G has exactly $\binom{n}{2}$ edges. The reason is that under the stated conditions, G is a complete graph.

Consider any two vertices u and v. Because G is connected, there is a path $u = v_0v_1 \ldots v_k = v$ starting at u and ending at v. We can easily prove by induction that there is an edge uv_i for each $1 \leq i \leq k$. The existence of the first such edge is immediate from its presence in the path. For later edges, we have from the induction hypothesis that there is an edge uv_i, from the path that there is an edge v_iv_{i+1}, and thus from the transitivity condition that there is and edge uv_{i+1}. When $i = k$, we have that there is an edge uv.

D.4.5 A possible matrix identity (20 points)

Prove or disprove: If A and B are symmetric matrices of the same dimension, then $A^2 - B^2 = (A-B)(A+B)$.

Solution

Observe first that $(A - B)(A + B) = A^2 + AB - BA + B^2$. The question then is whether $AB = BA$. Because A and B are symmetric, we have that $BA = B^\top A^\top = (AB)'$. So if we can show that AB is also symmetric, then we have $AB = (AB)' = BA$. Alternatively, if we can find symmetric matrices A and B such that AB is *not* symmetric, then $A^2 - B^2 \neq (A - B)(A + B)$.

Let's try multiplying two generic symmetric 2-by-2 matrices:

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} d & e \\ e & f \end{pmatrix} = \begin{pmatrix} ad + be & ae + bf \\ bd + ce & be + cf \end{pmatrix}$$

The product doesn't look very symmetric, and in fact we can assign variables to make it not so. We need $ae + bf \neq bd + ce$. Let's set $b = 0$ to make the bf and bd terms drop out, and $e = 1$ to leave just a and c. Setting $a = 0$ and $c = 1$ gives an asymmetric product. Note that we didn't determine d or f, so let's just set them to zero as well to make things as simple as possible. The result is:

$$AB = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

Which is clearly not symmetric. So for these particular matrices we have $A^2 - B^2 \neq (A - B)(A + B)$, disproving the claim.

D.5 CS202 Final Exam, December 14th, 2010

Write your answers in the blue book(s). Justify your answers. Give closed-form solutions when possible. Work alone. Do not use any notes or books.

There are five problems on this exam, each worth 20 points, for a total of 100 points. You have approximately three hours to complete this exam.

D.5.1 Backwards and forwards (20 points)

Let $\{0, 1\}^n$ be the set of all binary strings $x_1 x_2 \dots x_n$ of length n.

For any string x in $\{0, 1\}^n$, let $r(x) = x_n x_{n-1} \dots x_1$ be the reversal of x. Let $x \sim y$ if $x = y$ or $x = r(y)$.

Given a string x in $\{0, 1\}^n$ and a permutation π of $\{1, \dots, n\}$, let $\pi(x)$ be the string $x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)}$. Let $x \approx y$ if there exists some π such that $x = \pi(y)$.

Both $\sim$ and $\approx$ are equivalence relations. Let $\{0,1\}^n/\sim$ and $\{0,1\}^n/\approx$ be the corresponding sets of equivalence classes.

1. What is $|\{0,1\}^n/\sim|$ as a function of n?
2. What is $|\{0,1\}^n/\approx|$ as a function of n?

Solution

1. Given a string x, the equivalent class $[x] = \{x, r(x)\}$ has either one element (if $x = r(x)$) or two elements (if $x \neq r(x)$). Let m_1 be the number of one-element classes and m_2 the number of two-element classes. Then $|\{0,1\}^n| = 2^n = m_1 + 2m_2$ and the number we are looking for is $m_1 + m_2 = \frac{2m_1+2m_2}{2} = \frac{2^n+m_1}{2} = 2^{n-1} + \frac{m_1}{2}$. To find m_1, we must count the number of strings $x_1, \dots x_n$ with $x_1 = x_n$, $x_2 = x_{n-1}$, etc. If n is even, there are exactly $2^{n/2}$ such strings, since we can specify one by giving the first $n/2$ bits (which determine the rest uniquely). If n is odd, there are exactly $2^{(n+1)/2}$ such strings, since the middle bit can be set freely. We can write both alternatives as $2^{\lceil n/2 \rceil}$, giving $|\{0,1\}^n/\sim| = 2^{n-1} + 2^{\lceil n/2 \rceil}$.
2. In this case, observe that $x \approx y$ if and only if x and y contain the same number of 1 bits. There are $n+1$ different possible values $0, 1, \dots, n$ for this number. So $|\{0,1\}^n/\approx| = n + 1$.

The solution to the first part assumes $n > 0$; otherwise it produces the nonsensical result $3/2$. The problem does not specify whether $n = 0$ should be considered; if it is, we get exactly one equivalence class for both parts (the empty set).

D.5.2 Linear transformations (20 points)

Show whether each of the following functions from $\mathbb{R}^2$ to $\mathbb{R}$ is a linear transformation or not.

$$\begin{aligned}
f_1(x) &= x_1 - x_2. \\
f_2(x) &= x_1 x_2. \\
f_3(x) &= x_1 + x_2 + 1. \\
f_4(x) &= \frac{x_1^2 - x_2^2 + x_1 - x_2}{x_1 + x_2 + 1}.
\end{aligned}$$

Clarification added during the exam: You may assume that $x_1 + x_2 \neq -1$ for f_4.

Solution

1. Linear: $f_1(ax) = ax_1 - ax_2 = a(x_1 - x_2) = af_1(x)$ and $f_1(x + y) = (x_1 + y_1) - (x_2 + y_2) = (x_1 - x_2) + (y_1 - y_2) = f_1(x) + f_1(y)$.

2. Not linear: $f_2(2x) = (2x_1)(2x_2) = 4x_1x_2 = 4f_2(x) \neq 2f_2(x)$ when $f_2(x) \neq 0$.

3. Not linear: $f_3(2x) = 2x_1 + 2x_1 + 1$ but $2f_3(x) = 2x_1 + 2x_2 + 2$. These are never equal.

4. Linear:

$$
\begin{aligned}
f_4(x) &= \frac{x_1^2 - x_2^2 + x_1 - x_2}{x_1 + x_2 + 1} \\
&= \frac{(x_1 + x_2)(x_1 - x_2) + (x_1 - x_2)}{x_1 + x_2 + 1} \\
&= \frac{(x_1 + x_2 + 1)(x_1 - x_2)}{x_1 + x_2 + 1} \\
&= x_1 - x_2 \\
&= f_1(x).
\end{aligned}
$$

Since we've already shown f_1 is linear, $f_4 = f_1$ is also linear.

A better answer is that f_4 is not a linear transformation from $\mathbb{R}^2$ to $\mathbb{R}$ because it's not defined when $x_1 + x_2 - 1 = 0$. The clarification added during the exam tries to work around this, but doesn't really work. A better clarification would have defined f_4 as above for most x, but have $f_4(x) = x_1 - x_2$ when $x_1 + x_2 = -1$. Since I was being foolish about this myself, I gave full credit for any solution that either did the division or noticed the dividing-by-zero issue.

D.5.3 Flipping coins (20 points)

Flip n independent fair coins, and let X be a random variable that counts how many of the coins come up heads. Let a be a constant. What is $\mathrm{E}[a^X]$?

Solution

To compute $\mathrm{E}[a^X]$, we need to sum over all possible values of a^X weighted by their probabilities. The variable X itself takes on each value $k \in \{0 \dots n\}$ with probability $\binom{n}{k}2^{-n}$, so a^X takes on each corresponding value a^k with the same probability. We thus have:

$$\begin{aligned}
\mathrm{E}[a^X] &= \sum_{k=0}^{n} a^k \binom{n}{k} 2^{-n} \\
&= 2^{-n} \sum_{k=0}^{n} \binom{n}{k} a^k 1^{n-k} \\
&= 2^{-n}(a+1)^n \\
&= \left(\frac{a+1}{2}\right)^n.
\end{aligned}$$

The second to last step uses the Binomial Theorem.

As a quick check, some easy cases are $a = 0$ with $\mathrm{E}[a^X] = (1/2)^n$, which is consistent with the fact that $a^X = 1$ if and only if $X = 0$; and $a = 1$ with $\mathrm{E}[a^X] = 1^n = 1$, which is consistent with $a^X = 1^X = 1$ being constant. Another easy case is $n = 1$, in which we can compute directly $\mathrm{E}[a^X] = (1/2)a^0 + (1/2)a^1 = \frac{a+1}{2}$ as given by the formula. So we can have some confidence that we didn't mess up in the algebra somewhere.

Note also that $\mathrm{E}[a^X] = \left(\frac{a+1}{2}\right)^n$ is generally not the same as $a^{\mathrm{E}[X]} = a^{n/2}$.

D.5.4 Subtracting dice (20 points)

Let X and Y represent independent 6-sided dice, and let $Z = |X - Y|$ be their difference. (For example, if $X = 4$ and $Y = 3$, then $Z = 1$, and similarly when $X = 2$ and $Y = 5$, then $Z = 3$.)

1. What is $\Pr[Z = 1]$?
2. What is $\mathrm{E}[Z]$?
3. What is $\mathrm{E}[Z|Z \neq 0]$?

Solution

1. There are five cases where $Z = 1$ with $Y = X + 1$ (because X can range from 1 to 5), and five more cases where $Z = 1$ with $X = Y + 1$. So $\Pr[Z = 1] = \frac{10}{36} = \frac{5}{18}$.

2. Here we count 10 cases where $Z = 1$, 8 cases where $Z = 2$ (using essentially the same argument as above; here the lower die can range up to 4), 6 where $Z = 3$, 4 where $Z = 4$, and 2 where $Z = 5$. The cases where $Z = 0$ we don't care about. Summing up, we get $\mathrm{E}[Z] = (10 \cdot 1 + 8 \cdot 2 + 6 \cdot 3 + 4 \cdot 4 + 2 \cdot 5)/36 = 70/36 = 35/18$.

3. We can avoid recomputing all the cases by observing that $\mathrm{E}[Z] = \mathrm{E}[Z|Z \neq 0] \Pr[Z \neq 0] + \mathrm{E}[Z|Z = 0] \Pr[Z = 0]$. Since $\mathrm{E}[Z|Z = 0] = 0$, the second term disappears and we can solve for $\mathrm{E}[Z|Z \neq 0] = \mathrm{E}[Z]/\Pr[Z \neq 0]$. We can easily calculate $\Pr[Z = 0] = 1/6$ (since both dice are equal in this case, giving 6 out of 36 possible rolls), from which we get $\Pr[Z \neq 0] = 1 - \Pr[Z = 0] = 5/6$. Plugging this into our previous formula gives $\mathrm{E}[Z|Z \neq 0] = \frac{(35/18)}{(5/6)} = 7/3$.

It is also possible (and acceptable) to solve this problem by building a table of all 36 cases and summing up the appropriate values.

D.5.5 Scanning an array (20 points)

Suppose you have an $m \times m$ array in some programming language, that is, an data structure A holding a value $A[i, j]$ for each $0 \leq i < m$ and $0 \leq j < m$. You'd like to write a program that sets every element of the array to zero.

The usual way to do this is to start with $i = 0$ and $j = 0$, increment j until it reaches m, then start over with $i = 1$ and $j = 0$, and repeat until all m^2 elements of A have been reached. But this requires two counters. Instead, a clever programmer suggests using one counter k that runs from 0 up to $m^2 - 1$, and at each iteration setting $A[3k \bmod m, 7k \bmod m]$ to zero.

For what values of $m > 0$ does this approach actually reach all m^2 locations in the array?

Solution

Any two inputs k that are equal mod m give the same pair $(3k \bmod m, 7k \bmod m)$. So no matter how many iterations we do, we only reach m distinct locations. This equals m^2 only if $m = 1$ or $m = 0$. The problem statement excludes $m = 0$, so we are left with $m = 1$ as the only value of m for which this method works.

Appendix E

How to write mathematics

Suppose you want to write down some mathematics. How do you do it?

E.1 By hand

This method is recommended for CS202 assignments.

Advantages Don't need to learn any special formatting tools: any symbol you can see you can copy. Very hard to make typographical errors.

Disadvantages Not so good for publishing. Results may be ugly if you have bad handwriting. Results may be even worse if you copy somebody else's bad handwriting.

Example $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

E.2 LaTeX

This is what these notes are written in. It's also standard for writing papers in most technical fields.

Advantages Very nice formatting. De facto standard for mathematics publishing. Free.

Disadvantages You have to install it and learn it. Can't tell what something looks like until you run it through a program. Cryptic and uninformative 1970's-era error messages. The underlying system TEX is possibly the worst programming language in widespread use.

Example

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}.$$

The text above was generated by this source code:

```
\begin{displaymath}
    \sum_{i=1}^n i = \frac{n(n+1)}{2}.
\end{displaymath}
```

although a real LATEX document would also include some boilerplate around it.

You can find many introductions to LATEX at `http://www.latex-project.org/guides/`. LATEX runs on the computers in the Zoo, and can be made to run on just about anything.

There are front-ends to LATEX like Lyx `http://www.lyx.org` that try to make it WYSIWYG, with varying results. I don't use any of them personally.

E.3 Microsoft Word equation editor

This is probably a bad habit to get into.

Advantages There's a good chance you already type things up in Word.

Disadvantages Ugly formatting. Unpopular with conferences and journals, if you end up in the paper-writing business.

I don't use Word much, so no example.

E.4 ASCII and/or Unicode art

This is the method originally used to format these notes, back when they lived at `http://pine.cs.yale.edu/pinewiki/CS202`.

Advantages Everybody can read ASCII and most people can read Unicode. No special formatting required. Results are mostly machine-readable.

Disadvantages Very ugly formatting. Writing Unicode on a computer is a bit like writing Chinese—you need to learn how to input each possible character using whatever system you've got. May render strangely on some browsers.

Example `sum[i=1 to n] i = n(n+1)/2` (ASCII). Or a fancy version:

```
 n
---
\       n(n+1)
/   i = ------
---       2
i=1
```

Amazingly enough, many mathematics papers from the typewriter era (pre-1980 or thereabouts) were written like this, often with the more obscure symbols inked in by hand. Fortunately (for readers at least), we don't have to do this any more.

Appendix F

Tools from calculus

Calculus is not a prerequisite for this course, and it is possible to have a perfectly happy career as a computer scientist without learning any calculus at all. But for some tasks, calculus is much too useful a tool to ignore. Fortunately, even though typical high-school calculus courses run a full academic hear, the good parts can be understood with a few hours of practice.

F.1 Limits

The fundamental tool used in calculus is the idea of a **limit**. This is an approximation by nearby values to the value of an expression that we can't calculate exactly, typically because it involves division by zero.

The formal definition is that the limit as x goes to a of $f(x)$ is c, written

$$\lim_{x \to a} f(x) = c,$$

if for any constant $\epsilon > 0$ there exists a constant $\delta > 0$ such that

$$|f(y) - c| \leq \epsilon$$

whenever

$$|y - x| \leq \delta.$$

The intuition is that as y gets closer to x, $f(y)$ gets closer to c.

The formal definition has three layers of quantifiers, so as with all quantified expressions it helps to think of it as a game played between you and some adversary that controls all the universal quantifiers. So to show that $\lim_{x \to a} = c$, we have three steps:

- Some malevolent jackass picks ϵ, and says "oh yeah, smart guy, I bet you can't force $f(y)$ to be within ϵ of c."
- After looking at ϵ, you respond with δ, limiting the possible values of y to the range $[x - \delta, x + \delta]$.
- Your opponent wins if he can find a nonzero y in this range with $f(y)$ outside $[c - \epsilon, c + \epsilon]$. Otherwise you win.

For example, in the next section we will want to show that

$$\lim_{z \to 0} \frac{(x+z)^2 - x^2}{z} = 2x.$$

We need to take a limit here because the left-hand side isn't defined when $z = 0$.

Before playing the game, it helps to use algebra to rewrite the left-hand side a bit:

$$\begin{aligned}
\lim_{z \to 0} \frac{(x+z)^2 - x^2}{z} &= \lim_{z \to 0} \frac{x^2 + 2x(z) + (z)^2 - x^2}{z} \\
&= \lim_{z \to 0} \frac{2x(z) + (z)^2}{z} \\
&= \lim_{z \to 0} 2x + z.
\end{aligned}$$

So now the adversary says "make $|(2x + z) - 2x| < \epsilon$," and we say "that's easy, let $\delta = \epsilon$, then no matter what z you pick, as long as $|z - 0| < \delta$, we get $|(2x + z) - 2x| = |z| < \delta = \epsilon$, QED." And the adversary slinks off with its tail between its legs to plot some terrible future revenge.

Of course, a definition only really makes sense if it doesn't work if we pick a different limit. If we try to show

$$\lim_{z \to 0} \frac{(x+z)^2 - x^2}{z} = 12,$$

(assuming $x \neq 6$), then the adversary picks $\epsilon < |12 - 2x|$. Now we are out of luck: no matter what δ we pick, the adversary can respond with some value very close to 0 (say, $\min(\delta/2, |12 - 2x|/2)$), and we land inside $\pm\delta$ but outside $12 \pm \epsilon$.

We can also take the limit as a variable goes to infinity. This has a slightly different definition:

$$\lim_{x \to \infty} f(x) = c$$

holds if for any $\epsilon > 0$, there exists an $N > 0$, such that for all $x > N$, $|f(x) - c| < \epsilon$. Structurally, this is the same 3-step game as before, except now after we see ϵ instead of constraining x to be very close to a, we constraint x to be very big. Limits as x goes to infinity are sometimes handy for evaluating asymptotic notation.

[[[**divergence, needed for little ω**]]]

F.2 Derivatives

[[[**motivation: tells us about the shape of a function. Increasing if $f' > 0$, decreasing if $f' = 0$, extremum (or inflection point) at $f' = 0$**]]]

The **derivative** or **differential** of a function measures how much the function changes if we make a very small change to its input. One way to think about this is that for most functions, if you blow up a plot of them enough, you don't see any curvature any more, and the function looks like a line that we can approximate as $ax + b$ for some coefficients a and b.

The derivative $f'(x)$ just gives this coefficient a for each particular x. The notation f' is due to Leibnitz and is convenient for functions that have names but not so convenient for something like $x^2 + 3$. For more general functions, a different notation due to Newton is used. The derivative of f with respect to x is written as $\frac{df}{dx}$ or $\frac{d}{dx}f$, and its value for a particular value $x = c$ is written using the somewhat horrendous notation

$$\left.\frac{d}{dx}f\right|_{x=c}.$$

There is a formal definitions of $f'(x)$, which nobody ever uses, given by

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x},$$

where Δx is a single two-letter variable (not the product of Δ and x!) that represents the change in x. In the preceding section, we calculated an example of this kind of limit and showed that $\frac{d}{dx}x^2 = 2x$.

Using the formal definition to calculate derivatives is painful and almost always unnecessary. The reason is that the derivatives for most standard functions are well-known, and by memorizing a few simple rules you can combine these to compute the derivative of just about anything completely mechanically. A handy cheat sheet is given in Table F.1

$f(x)$	$f'(x)$	
c	0	
x^n	nx^{n-1}	
e^x	e^x	
a^x	$a^x \ln a$	follows from $a^x = e^{x \ln a}$
$\ln x$	$1/x$	
$cg(x)$	$cg'(x)$	multiplication by a constant
$g(x)+h(x)$	$g'(x)+h'(x)$	sum rule
$g(x)h(x)$	$g(x)h'(x)+g'(x)h(x)$	product rule
$g(h(x))$	$g'(h(x))h'(x)$	chain rule

Table F.1: Table of derivatives

Example:

$$\begin{aligned}
\frac{d}{dx}\frac{x^2}{\ln x} &= x^2\left(\frac{d}{dx}\frac{1}{\ln x}\right) + \frac{1}{\ln x}\cdot\frac{d}{dx}x^2[\text{product rule}] \\
&= x^2 \cdot -1 \cdot (\ln x)^-2 \cdot \frac{d}{dx}\ln x[\text{chain rule}] + \frac{1}{\ln x}\cdot 2x \\
&= -\frac{x^2}{\ln^2 x}\cdot\frac{1}{x} + \frac{2x}{\ln x} \\
&= \frac{-x}{\ln^2 x} + \frac{2x}{\ln x}.
\end{aligned}$$

The idea is that whatever the outermost operation in an expression is, you can apply one of the rules above to move the differential inside it, until there is nothing left. Even computers can be programmed to do this. You can do it too.

F.3 Integrals

[[[**explain what integration is: area under curve, anti-differentiation**]]]

First you have to know how to differentiate (see previous section). Having learned how to differentiate, your goal in integrating some function $f(x)$ is to find another function $F(x)$ such that $F'(x) = f(x)$. You can then write that the **indefinite integral** $\int f(x)dx$ of $f(x)$ is $F(x) + C$ (any constant C

$f(x)$	$F(x)$	
$f(x)+g(x)$	$F(x)+G(x)$	
$af(x)$	$aF(x)$	a is constant
$f(ax)$	$\frac{F(ax)}{a}$	a is constant
x^n	$\frac{x^{n+1}}{n+1}$	n constant, $n \neq 1$
x^{-1}	$\ln x$	
e^x	e^x	
a^x	$\frac{a^x}{\ln a}$	a constant
$\ln x$	$x \ln x - x$	

Table F.2: Table of integrals

works), and compute **definite integrals** with the rule

$$\int_a^b f(x)dx = F(b) - F(a).$$

How do you find this magic $F(x)$? Some possibilities:

- Memorize some standard integral formulas. Some useful ones are given in Table F.2.
- Guess but verify. Guess $F(x)$ and compute $F'(x)$ to see if it's $f(x)$. May be time-consuming unless you are good at guessing, and can put enough parameters in $F(x)$ to let you adjust $F'(x)$ to equal $f(x)$. Example: if $f(x) = 2/x$, you may remember the $1/x$ formula and try $F(x) = a \ln bx$. Then $F'(x) = ab/(bx) = a/x$ and you can set $a = 2$, quietly forget you ever put in b, and astound your friends (who also forgot the $af(x)$ rule) by announcing that the integral is $2 \ln x$. Sometimes if the answer comes out wrong you can see how to fudge $F(x)$ to make it work: if for $f(x) = \ln x$ you guess $F(x) = x \ln x$, then $F'(x) = \ln x + 1$ and you can notice that you need to add a $-x$ term (the integral of -1) to get rid of the 1. This gives $\int \ln x = x \ln x - x$.
- There's a technique called **integration by parts**, which is the integral version of the $duv = udv + vdu$ formula, but it doesn't work as often as one might like. The rule is that

  $$\int udv = uv - \int vdu.$$

  An example is $\int \ln x dx = x \ln x - \int x d(\ln x) = x \ln x - \int x(1/x)dx = x \ln x - \int 1 dx = x \ln x - x$. You probably shouldn't bother memorizing this unless you need to pass AP Calculus again, although you can rederive it from the $duv = udv + vdu$ formula. [[[**figure out how to square this with rewriting** $udv + vdu$ **as** $FG' + F'G$**.**]]]

- Use a computer algebra system like Mathematica, Maple, or Maxima. Mathematica's integration routine is available on-line at `http://integrals.wolfram.com`.

- Look your function up in a big book of integrals. This is actually less effective that using Mathematica, but may continue to work during power failures.

Appendix G

The natural numbers

Here we give an example of how we can encode simple mathematics using predicate logic, and then prove theorems about the resulting structure. Our goal is to represent the **natural numbers**: 0, 1, 2, etc.[1]

G.1 The Peano axioms

Peano represents natural numbers using a special 0 constant and a function symbol S (for "successor"; think of it as $+1$). Repeatedly applying S to 0 generates increasingly large natural numbers: $S0 = 1, SS0 = 2, SSS0 = 3$, etc. (Note that 1, 2, 3, etc., are not part of the language, although we might use them sometimes as shorthand for long strings of S's.) For convenience, we don't bother writing parentheses for the function applications unless we need to do so to avoid ambiguity: read $SSSS0$ as $S(S(S(S(0))))$.

The usual interpretation of function symbols implies that 0 and its successors exist, but it doesn't guarantee that they aren't all equal to each other. The first Peano axiom[2] prevents this:

$$\forall x : Sx \neq 0. \tag{P1}$$

In English, 0 is not the successor of any number.

[1]Some people define the natural numbers as starting at 1. Those people are generally (a) wrong, (b) number theorists, (c) extremely conservative, or (d) citizens of the United Kingdom of Great Britain and Northern Ireland. As computer scientists, we will count from 0 as the gods intended.

[2]This is not actually the first axiom that Peano defined. The original Peano axioms [Pea89, §1] included some axioms on existence of Sx and the properties of equality that have since been absorbed as standard rules of first-order logic. The axioms we are presenting here correspond to Peano's axioms 8, 7, and 9.

This still allows for any number of nasty little models in which 0 is nobody's successor, but we still stop before getting all of the naturals. For example, let $SS0 = S0$; then we only have two elements in our model (0 and $S0$, because once we get to $S0$, any further applications of S keep us where we are.

To avoid this, we need to prevent S from looping back round to some number we've already produced. It can't get to 0 because of the first axiom, and to prevent it from looping back to a later number, we take advantage of the fact that they already have one successor:

$$\forall x : \forall y : Sx = Sy \to x = y. \tag{P2}$$

If we take the contrapositive in the middle, we get $x \neq y \to Sx \neq Sy$. In other words, we can't have a single number z that is the successor of two different numbers x and y.

Now we get all of $\mathbb{N}$, but we may get some extra elements as well. There is nothing in the first two axioms that prevents us from having something like this:

$$0 \to S0 \to SS0 \to SSS0 \to \dots B \to SB \to SSB \to SSSB \to \dots$$

where B stands for "bogus."

The hard part of coming up with the Peano axioms was to prevent the model from sneaking in extra bogus values (that still have successors and at most one predecessor each). This is (almost) done using the third Peano axiom, which in first-order logic—where we can't write $\forall P$—is written as an **axiom schema**. This is a pattern that generates an infinite list of an axioms, one for each choice of predicate P:

$$(P(0) \wedge (\forall x : P(x) \to P(Sx))) \to \forall x : P(x). \tag{P3}$$

This is known as the **induction schema**, and says that, for any predicate P, if we can prove that P holds for 0, and we can prove that $P(x)$ implies $P(x + 1)$, then P holds for all x in $\mathbb{N}$. The intuition is that even though we haven't bothered to write out a proof of, say $P(1337)$, we know that we can generate one by starting with $P(0)$ and modus-pwning our way out to $P(1337)$ using $P(0) \to P(1)$, then $P(1) \to P(2)$, then $P(2) \to P(3)$, etc. Since this works for any number (eventually), there can't be some number that we missed.

In particular, this lets us throw out the bogus numbers in the bad example above. Let $B(x)$ be true if x is bogus (i.e., it's equal to B or one of the other values in its chain of successors). Let $P(x) \equiv \neg B(x)$. Then $P(0)$ holds (0 is not bogus), and if $P(x)$ holds (x is not bogus) then so does $P(Sx)$. It follows from the induction axiom that $\forall x P(x)$: there are no bogus numbers.[3]

G.2 A simple proof

Let's use the Peano axioms to prove something that we know to be true about the natural numbers we learned about in grade school but that might not be obvious from the axioms themselves. (This will give us some confidence that the axioms are not bogus.) We want to show that 0 is the only number that is not a successor:

Claim G.2.1. $\forall x : (x \neq 0) \rightarrow (\exists y : x = Sy)$.

To find a proof of this, we start by looking at the structure of what we are trying to prove. It's a universal statement about elements of $\mathbb{N}$ (implicitly, the $\forall x$ is really $\forall x \in \mathbb{N}$, since our axioms exclude anything that isn't in $\mathbb{N}$), so our table of proof techniques suggests using an induction argument, which in this case means finding some predicate we can plug into the induction schema.

If we strip off the $\forall x$, we are left with

$$(x \neq 0) \rightarrow (\exists y : x = Sy).$$

Here a direct proof is suggested: assuming $x \neq 0$, and try to prove $\exists y : x = Sy$. But our axioms don't tell us much about numbers that aren't 0, so it's not clear what to do with the assumption. This turns out to be a dead end.

Recalling that $A \rightarrow B$ is the same thing as $\neg A \vee B$, we can rewrite our goal as

$$x = 0 \vee \exists y : x = Sy.$$

[3]There is a complication here. Peano's original axioms were formulated in terms of **second-order logic**, which allows quantification over all possible predicates (you can write things like $\forall P : P(x) \rightarrow P(Sx)$). So the *bogus* predicate we defined is implicitly included in that for-all. But if there is no first-order predicate that distinguishes bogus numbers from legitimate ones, the induction axiom won't kick them out. This means that the Peano axioms (in first-order logic) actually *do* allow bogus numbers to sneak in somewhere around infinity. But they have to be *very polite* bogus numbers that never do anything different from ordinary numbers. This is probably not a problem except for philosophers. Similar problems show up for any model with infinitely many elements, due to something called the Löwenheim-Skolem theorem.

This seems like a good candidate for P (our *induction hypothesis*), because we do know a few things about 0. Let's see what happens if we try plugging this into the induction schema:

- $P(0) \equiv 0 = 0 \vee \exists y : 0 = Sy$. The right-hand term looks false because of our first axiom, but the left-hand term is just the reflexive axiom for equality. $P(0)$ is true.
- $\forall x P(x) \to P(Sx)$. We can drop the $\forall x$ if we fix an arbitrary x. Expand the right-hand side $P(Sx) \equiv Sx = 0 \vee \exists y Sx = Sy$. We can be pretty confident that $Sx \neq 0$ (it's an axiom), so if this is true, we had better show $\exists y Sx = Sy$. The first thing to try for $\exists$ statements is instantiation: pick a good value for y. Picking $y = x$ works.

Since we showed $P(0)$ and $\forall x P(x) \to P(Sx)$, the induction schema tells us $\forall x P(x)$. This finishes the proof.

Having figured the proof out, we might go back and clean up any false starts to produce a compact version. A typical mathematician might write the preceding argument as:

Proof. By induction on x. For x = 0, the premise fails. For Sx, let $y = x$. □

A really lazy mathematician would write:

Proof. Induction on x. □

Though laziness is generally a virtue, you probably shouldn't be quite this lazy when writing up homework assignments.

G.3 Defining addition

Because of our restricted language, we do not yet have the ability to state valuable facts like $1+1 = 2$ (which we would have to write as $S0+S0 = SS0$). Let's fix this, by adding a two-argument function symbol $+$ which we will define using the axioms

- $x + 0 = x$.
- $x + Sy = S(x + y)$.

(We are omitting some $\forall$ quantifiers, since unbounded variables are implicitly universally quantified.)

This definition is essentially a recursive program for computing $x+y$ using only successor, and there are some programming languages (e.g. Haskell) that will allow you to define addition using almost exactly this notation. If the definition works for all inputs to $+$, we say that $+$ is **well-defined**. Not working would include giving different answers depending on which parts of the definitions we applied first, or giving no answer for some particular inputs. These bad outcomes correspond to writing a buggy program. Though we can in principle prove that this particular definition is well-defined (using induction on y), we won't bother. Instead, we will try to prove things about our new concept of *addition* that will, among other things, tell us that the definition gives the *correct* answers.

We start with a **lemma**, which is Greek for a result that is not especially useful by itself but is handy for proving other results.[4]

Lemma G.3.1. *$0 + x = x$.*

Proof. By induction on x. When $x = 0$, we have $0 + 0 = 0$, which is true from the first case of the definition. Now suppose $0 + x = x$ and consider what happens with Sx. We want to show $0 + Sx = Sx$. Rewrite $0 + Sx$ as $S(0+x)$ [second case of the definition], and use the induction hypothesis to show $S(0 + x) = S(x)$. $\square$

(We could do a lot of QED-ish jumping around in the end zone there, but it is more refined—and *lazier*—to leave off the end of the proof once it's clear we've satisifed all of our obligations.)

Here's another lemma, which looks equally useless:

Lemma G.3.2. *$x + Sy = Sx + y$.*

Proof. By induction on y. If $y = 0$, then $x + S0 = S(x + 0) = Sx = Sx + 0$. Now suppose the result holds for y and show $x + SSy = Sx + Sy$. We have $x + SSy = S(x + Sy) = S(Sx + y)$[ind. hyp.] $= Sx + Sy$. $\square$

Now we can prove a **theorem**: this is a result that we think of as useful in its own right. (In programming terms, it's something we export from a module instead of hiding inside it as an internal procedure.)

Theorem G.3.3. *$x + y = y + x$. (Commutativity of addition.)*

[4]It really means *fork*.

Proof. By induction on x. If $x = 0$, then $0+y = y+0$ (see previous lemma). Now suppose $x + y = y + x$, and we want to show $Sx + y = y + Sx$. But $y+Sx = S(y+x)$[axiom] $= S(x+y)$[induction hypothesis] $= x+Sy$[axiom] $= Sx + y$[lemma]. $\square$

This sort of definition-lemma-lemma-theorem structure is typical of written mathematical proofs. Breaking things down into small pieces (just like breaking big subroutines into small subroutines) makes debugging easier, since you can check if some intermediate lemma is true or false without having to look through the entire argument at once.

Question: How do you know which lemmas to prove? Answer: As when writing code, you start by trying to prove the main theorem, and whenever you come across something you need and can't prove immediately, you fork it off as a lemma. Conclusion: The preceding notes were not originally written in order.

G.3.1 Other useful properties of addition

So far we have shown that $x + y = y + x$, also known as **commutativity of addition**. Another familiar property is **associativity of addition**: $x + (y + z) = (x + y) + z$. This is easily proven by induction (try it!)

We don't have subtraction in $\mathbb{N}$ (what's $3 - 5$?)[5] The closest we can get is **cancellation**:

Lemma G.3.4. $x + y = x + z \rightarrow y = z$.

We can define $\leq$ for $\mathbb{N}$ directly from addition: Let $x \leq y \equiv \exists z x + z = y$. Then we can easily prove each of the following (possibly using our previous results about addition having commutativity, associativity, and cancellation).

- $0 \leq x$.
- $x \leq Sx$.
- $x \leq y \wedge y \leq z \rightarrow x \leq z$.

[5]This actually came up on a subtraction test I got in the first grade from the terrifying Mrs Garrison at Mountain Park Elementary School in Berkeley Heights, New Jersey. She insisted that -2 was not the correct answer, and that we should have recognized it as a trick question. She also made us black out the arrow the left of the zero on the number-line stickers we had all been given to put on the top of our desks. Mrs Garrison was, on the whole, a fine teacher, but she did not believe in New Math.

- $a \leq b \wedge c \leq d \rightarrow a + c \leq b + d$.
- $x \leq y \wedge y \leq x \rightarrow x = y$.

(The actual proofs will be left as an exercise for the reader.)

G.4 A scary induction proof involving even numbers

Let's define the predicate Even(x) $\equiv \exists y x = y+y$. (The use of $\equiv$ here signals that Even(x) is syntactic sugar, and we should think of any occurrence of Even(x) as expanding to $\exists y x = y + y$.)

It's pretty easy to see that 0 = 0+0 is even. Can we show that S0 is not even?

Lemma G.4.1. $\neg Even(S0)$.

Proof. Expand the claim as $\neg\exists y S0 = y + y \equiv \forall y S0 \neq y + y$. Since we are working over $\mathbb{N}$, it's tempting to try to prove the $\forall y$ bit using induction. But it's not clear why $S0 \neq y + y$ would tell us anything about $S0 \neq Sy + Sy$. So instead we do a case analysis, using our earlier observation that every number is either 0 or Sz for some z.

Case 1 $y = 0$. Then $S0 \neq 0 + 0$ since $0 + 0 = 0$ (by the definition of +) and $0 \neq S0$ (by the first axiom).

Case 2 $y = Sz$. Then $y+y = Sz+Sz = S(Sz+z) = S(z+Sz) = SS(z+z)$.[6] Suppose $S0 = SS(z + z)$ [Note: "Suppose" usually means we are starting a proof by contradiction]. Then $0 = S(z + z)$ [second axiom], violating $\forall x 0 \neq Sx$ [first axiom]. So $S0 \neq SS(z + z) = y + y$.

Since we have $S0 \neq y + y$ in either case, it follows that $S0$ is not even. $\square$

Maybe we can generalize this lemma! If we recall the pattern of non-even numbers we may have learned long ago, each of them $(1, 3, 5, 7, \dots)$ happens to be the successor of some even number $(0, 2, 4, 6, \dots)$. So maybe it holds that:

Theorem G.4.2. $Even(x) \rightarrow \neg Even(Sx)$.

[6] What justifies that middle step?

Proof. Expanding the definitions gives $(\exists y x = y + y) \rightarrow (\neg \exists z Sx = z + z)$. This is an implication at top-level, which calls for a direct proof. The assumption we make is $\exists y x = y + y$. Let's pick some particular y that makes this true (in fact, there is only one, but we don't need this). Then we can rewrite the right-hand side as $\neg \exists z S(y + y) = z + z$. There doesn't seem to be any obvious way to show this (remember that we haven't invented subtraction or division yet, and we probably don't want to).

We are rescued by showing the stronger statement $\forall y \neg \exists z S(y+y) = z+z$: this is something we can prove by induction (on y, since that's the variable inside the non-disguised universal quantifier). Our previous lemma gives the base case $\neg \exists z S(0 + 0) = z + z$, so we just need to show that $\neg \exists z S(y + y) = z + z implies \neg \exists z S(Sy + Sy) = z + z$. Suppose that $S(Sy + Sy) = z + z$ for some z ["suppose" = proof by contradiction again: we are going to drive this assumption into a ditch]. Rewrite $S(Sy + Sy)$ to get $SSS(y + y) = z + z$. Now consider two cases:

Case 1 $z = 0$. Then $SSS(y+y) = 0+0 = 0$, contradicting our first axiom.

Case 2 $z = Sw$. Then $SSS(y + y) = Sw + Sw = SS(w + w)$. Applying the second axiom twice gives $S(y + y) = w + w$. But this contradicts the induction hypothesis.

Since both cases fail, our assumption must have been false. It follows that $S(Sy + Sy)$ is not even, and the induction goes through. □

G.5 Defining more operations

Let's define multiplication $(\cdot)$ by the axioms:

- $0 \cdot y = 0$.
- $Sx \cdot y = y + x \cdot y$.

Some properties of multiplication:

- $x \cdot 0 = 0$.
- $1 \cdot x = x$.
- $x \cdot 1 = x$.
- $x \cdot y = y \cdot x$.
- $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.

- $x \neq 0 \wedge x \cdot y = x \cdot z \to y = z$.
- $x \cdot (y + z) = x \cdot y + x \cdot z$.
- $x \leq y \to z \cdot x \leq z \cdot y$.
- $z \neq 0 \wedge z \cdot x \leq z \cdot y \to x \leq y$.

(Note we are using 1 as an abbreviation for $S0$.)

The first few of these are all proved pretty much the same way as for addition. Note that we can't divide in $\mathbb{N}$ any more than we can subtract, which is why we have to be content with multiplicative cancellation.

Exercise: Show that the $\text{Even}(x)$ predicate, defined previously as $\exists y y = x + x$, is equivalent to $\text{Even}'(x) \equiv \exists y x = 2 \cdot y$, where $2 = SS0$. Does this definition make it easier or harder to prove $\neg\text{Even}'(S0)$?

Bibliography

[BD92] Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *Annals of Applied Probability*, 2(2):294–313, 1992.

[Ber34] George Berkeley. *THE ANALYST; OR, A DISCOURSE Addressed to an Infidel MATHEMATICIAN. WHEREIN It is examined whether the Object, Principles, and Inferences of the modern Analysis are more distinctly conceived, or more evidently deduced, than Religious Mysteries and Points of Faith.* Printed for J. Tonson, London, 1734.

[Big02] Norman L. Biggs. *Discrete Mathematics.* Oxford University Press, second edition, 2002.

[Bou70] N. Bourbaki. *Théorie des Ensembles.* Hermann, Paris, 1970.

[Ded01] Richard Dedekind. *Essays on the Theory of Numbers.* The Open Court Publishing Company, Chicago, 1901. Translated by Wooster Woodruff Beman.

[Die10] R. Diestel. *Graph Theory.* Graduate Texts in Mathematics. Springer, 2010.

[Fer08] Kevin Ferland. *Discrete Mathematics.* Cengage Learning, 2008.

[Gen35a] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.

[Gen35b] Gerhard Gentzen. Untersuchungen über das logische schließen. ii. *Mathematische Zeitschrift*, 39(1):405–431, 1935.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[GVL12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 4th edition, 2012.

[Hal58] Paul R. Halmos. *Finite-Dimensional Vector Spaces.* Van Nostrand, 1958.

[Hau14] Felix Hausdorff. *Grundzüge der Mengenlehre.* Veit and Company, Leipzig, 1914.

[Kol33] A.N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung.* Springer, 1933.

[Kur21] Casimir Kuratowski. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae*, 2(1):161–171, 1921.

[Mad88a] Penelope Maddy. Believing the axioms i. *Journal of Symbolic Logic*, 53(2):48–511, June 1988.

[Mad88b] Penelope Maddy. Believing the axioms ii. *Journal of Symbolic Logic*, 53(3):736–764, September 1988.

[NWZ11] David Neumark, Brandon Wall, and Junfu Zhang. Do small businesses create more jobs? new evidence for the united states from the national establishment time series. *The Review of Economics and Statistics*, 93(1):16–29, February 2011.

[Pea89] Ioseph Peano. *Arithmetices Principia: Nova Methodo Exposita.* Fratres Bocca, Rome, 1889.

[Pel99] Francis Jeffrey Pelletier. A history of natural deduction rules and elementary logic textbooks. Available at `http://www.sfu.ca/~jeffpell/papers/pelletierNDtexts.pdf`, 1999.

[Pol04] Goerge Polya. *How to Solve it: A New Aspect of Mathematical Method.* Princeton University Press, 2004.

[Ros12] Kenneth Rosen. *Discrete Mathematics and Its Applications.* McGraw-Hill Higher Education, seventh edition, 2012.

[Say33] Dorothy Sayers. *Murder Must Advertise.* Victor Gollancz, 1933.

[Sch01] Eric Schechter. Constructivism is difficult. *American Mathematical Monthly*, 108:50–54, 2001.

[Sol05] Daniel Solow. *How to Read and Do Proofs: An Introduction to Mathematical Thought Processes.* Wiley, 2005.

[Sta97] Richard P. Stanley. *Enumerative Combinatorics, Volume 1.* Number 49 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1997.

[Str05] Gilbert Strang. *Linear Algebra and its Applications.* Cengage Learning, 4th edition, 2005.

[SW86] Dennis Stanton and Dennis White. *Constructive Combinatorics.* Undergraduate Texts in Mathematics. Springer, 1986.

[TK74] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, September 1974.

[Wil95] Andrew John Wiles. Modular elliptic curves and Fermat's Last Theorem. *Annals of Mathematics*, 141(3):443–551, 1995.

Index

$O(f(n))$, 100
$\Omega(f(n))$, 100
$\Theta(f(n))$, 100
$\leftrightarrow$, 14
$\mathbb{N}$, 64, 74
$\mathbb{Q}$, 64
$\mathbb{R}$, 64
$\mathbb{Z}$, 74
$\neg$, 13
$\omega(f(n))$, 100
$\rightarrow$, 14
$\vee$, 13
$\wedge$, 14
$\mathbb{D}$, 74
k-permutation, 170
n-ary relation, 122
$o(f(n))$, 100

1, 18

abelian group, 65, 270
absolute value, 76
absorption law, 22
abuse of notation, 66, 105
accepting state, 59
acyclic, 155
addition, 65
addition (inference rule), 36
additive identity, 270
additive inverse, 270
adjacency matrix, 124
adjacent, 145
adversary, 28
affine transformation, 262
aleph-nought, 60
aleph-null, 60
aleph-one, 60
aleph-zero, 60
algebra, 74
    linear, 243
algebraic field extension, 272
algebraically closed, 73
alphabet, 59
and, 14
annihilator, 68
ansatz, 96
antisymmetric, 125
antisymmetry, 70
arc, 141
arguments, 25
associative, 65, 270
associativity, 65
    of addition, 65
    of multiplication, 67
associativity of addition, 349
asymptotic notation, 100
atom, 212
automorphism, 152
average, 226
axiom, 9, 34
Axiom of Extensionality, 48
axiom schema, 52, 345
axiomatic set theory, 52
axioms

    field, 65
    for the real numbers, 64
    Peano, 344

base case, 78
basis, 258
Bayes's formula, 220
Bayesian interpretation, 211
Bernoulli distribution, 222
biconditional, 14
big O, 100
big Omega, 100
big Theta, 100
bijection, 57
bijective, 56, 57
binary relation, 122
bind, 26
binomial coefficient, 171, 177
binomial distribution, 222
binomial theorem, 177
bipartite, 293
bipartite graph, 143
blackboard bold, 64
block, 126

cancellation, 349
Cantor pairing function, 61
Cantor's diagonalization argument, 62
cardinal arithmetic, 60
cardinal numbers, 60
cardinality, 57, 163
Cartesian product, 54
case analysis, 21
Catalan number, 208
categorical graph product, 151
Cauchy sequence, 59
ceiling, 55, 76
Central Limit Theorem, 223
central limit theorem, 237
chain rule, 341
Chebyshev's inequality, 236
Chernoff bounds, 237
Chinese Remainder Theorem, 116
class
    equivalence, 126
closed, 74
closed form, 93
closed walk, 154
closure, 137
    reflexive, 137
    reflexive symmetric transitive, 137
    reflexive transitive, 137
    symmetric, 137
    transitive, 137
CNF, 23
codomain, 54
column, 247
column space, 261
column vector, 255
combinatorics, 163
    enumerative, 163
commutative, 65, 270
commutative group, 65
commutative ring, 270
commutativity, 65
    of addition, 65
    of multiplication, 67
commutativity of addition, 349
comparability, 70
comparable, 128, 130
compatible, 250
complement, 49
complete bipartite, 147
complete graph, 145
completeness, 34
complex, 64
complex number, 64
component
    strongly-connected, 139
composed, 56

composite, 106
composition, 56
  of relations, 124
compound proposition, 14
comprehension
  restricted, 52
conclusion, 34
conditional expectation, 228
conditional probability, 218
congruence mod m, 113
conjuctive normal form, 23
conjunction (inference rule), 36
connected, 153
connected components, 153
connected to, 153
consistent, 10
constants, 11
constructive, 41
contained in, 49
continuous random variable, 241
contraction, 150
contradiction, 17
contrapositive, 19
converges, 91
converse, 19
coordinate, 243
coprime, 108
countable, 62
counting two ways, 171
covariance, 235
CRC, 276
cross product, 151
cube, 148
Curry-Howard isomorphism, 21
cycle, 147, 154
  simple, 154
cyclic redundancy check, 276

DAG, 123, 155
decision theory, 226
Dedekind cut, 59
deducible, 34
Deduction Theorem, 37
definite integral, 342
definition
  recursive, 77, 78
degree, 145, 271
dense, 72
density, 241
  joint, 242
derivative, 340
deterministic finite state machine, 59
diagonalization, 62
die roll, 215
differential, 340
dimension, 243, 247, 259
  of a matrix, 124
directed acyclic graph, 123, 139, 155
directed graph, 122, 141, 142
directed multigraph, 142
discrete probability, 212
discrete random variable, 221
disjoint, 60
disjoint union, 60
disjunctive normal form, 24
disjunctive syllogism, 36
distribution, 221
  Bernoulli, 222
  binomial, 222
  geometric, 222
  joint, 223
  marginal, 223
  normal, 223
  Poisson, 222
  uniform, 222
distribution function, 240
distributive, 270
divided by, 67
divides, 106
divisibility, 128

division, 67
division algorithm, 83, 107
    for polynomials, 271
divisor, 106, 107
DNF, 24
domain, 25, 54
dot product, 256
double factorial, 287
downward closed, 59
dyadics, 74

edge, 122
    parallel, 122, 141
edges, 141
egf, 210
Einstein summation convention, 91
elements, 47, 247
empty set, 47
endpoint, 142
entries, 247
enumerative combinatorics, 163
equality, 31
equivalence class, 126, 127
equivalence relation, 125
Euclidean algorithm, 108
    extended, 109
Euler's Theorem, 119
Euler's totient function, 119
Eulerian cycle, 155
Eulerian tour, 155
even, 113
event, 212
exclusive or, 13
existential quantification, 26
existential quantifier, 26
expectation, 225
exponential generating function, 210
extended, 132
extended Euclidean algorithm, 109
extended real line, 72
extension
    of a partial order to a total order, 132

factor, 106
factorial, 169, 287
    double, 287
false positive, 220
Fermat's Little Theorem, 120
field, 65, 269
    finite, 113, 268
    Galois, 273
    ordered, 70
field axioms, 65
field extension, 272
field of fractions, 74
finite field, 113, 268
finite simple undirected graph, 141
first-order logic, 25
floating-point number, 65
floor, 55, 76, 107
flush, 298
formal power series, 186
fraction, 64, 74
fractional part, 76
frequentist interpretation, 211
full rank, 264
function symbol, 31
functions, 54
Fundamental Theorem of Arithmetic, 112

Galois field, 273
gcd, 108
Generalized Continuum Hypothesis, 60
generating function, 186
    probability, 238
generating functions, 184
geometric distribution, 222

Goldbach's conjecture, 31
graph, 141
  bipartite, 123, 143
  directed, 122
  directed acyclic, 123, 139
  simple, 122, 141
  two-path, 295
  undirected, 141
graph Cartesian product, 150
Graph Isomorphism Problem, 151
greatest common divisor, 108
greatest lower bound, 72, 131
group, 65, 270
  abelian, 270
  commutative, 65, 270

Hamiltonian cycle, 155
Hamiltonian tour, 156
Hasse diagram, 130
head, 142
homomorphism, 115, 152
  graph, 293
hyperedges, 143
hypergraph, 143
hypothesis, 34
hypothetic syllogism, 36

identity, 66
  additive, 270
  for addition, 66
  for multiplication, 67
  multiplicative, 270
identity element, 92
identity matrix, 251
if and only if, 14
immediate predecessor, 130
immediate successor, 130
implication, 14
in-degree, 145
incident, 145
inclusion-exclusion, 182
inclusion-exclusion formula, 167
inclusive or, 13
incomparable, 130
incompleteness theorem, 35
indefinite integral, 341
independent, 215, 224, 242
  pairwise, 235
index of summation, 86
indicator variable, 220
indirect proof, 19
induced subgraph, 148
induction, 77
induction hypothesis, 78
induction schema, 78, 345
induction step, 78
inequality
  Chebyshev's, 236
inference rule, 35
inference rules, 9, 34
infimum, 72
infinite descending chain, 136
infinitesimal, 72
infix notation, 122
initial state, 59
initial vertex, 122, 142
injection, 57
injective, 56, 57
integer, 64
integers, 74
integers mod m, 113
integral
  definite, 342
  indefinite, 341
  Lebesgue-Stieltjes, 225
integration by parts, 342
intersection, 49
intuitionistic logic, 21
invariance
  scaling, 70

    translation, 70
inverse, 19, 57, 67
    additive, 270
    multiplicative, 270
    of a relation, 125
invertible, 251
irrational, 64
irreducible, 272
isomorphic, 56, 75, 126, 151
isomorphism, 151

join, 108, 131
joint density, 242
joint distribution, 222, 223, 241

Kolmogorov's extension theorem, 214

labeled, 144
lambda calculus, 63
lattice, 108, 131
law of non-contradiction, 21
law of the excluded middle, 21
law of total probability, 219
lcm, 108
least common multiple, 108
least upper bound, 71, 131
Lebesgue-Stieltjes integral, 225
lemma, 34, 348
length, 147, 153, 255
lex order, 129
lexicographic order, 129
LFSR, 268
limit, 91, 338
linear, 87, 227, 259
linear algebra, 243
linear combination, 257
linear transformation, 249, 259
linear-feedback shift register, 268
linearly independent, 257
little o, 100
little omega, 100
logical equivalence, 17
logically equivalent, 17
loops, 142
lower bound, 72, 86
lower limit, 86
lower-factorial, 170

magma, 270
marginal distribution, 223
Markov's inequality, 232
mathematical maturity, 1, 3
matrix, 123, 247
    adjacency, 124
maximal, 131
maximum, 131
measurable, 240
measurable set, 213
meet, 108, 131
mesh, 151
method of infinite descent, 81
minimal, 131
minimum, 131
minor, 150
minus, 66
model, 10, 16, 33
model checking, 16
modulus, 107
modus ponens, 24, 35, 36
modus tollens, 36
monoid, 270
multigraph, 143
multinomial coefficient, 172
multiplicative identity, 270
multiplicative inverse, 109, 115, 270
multiset, 128
multivariate generating functions, 194

naive set theory, 47
natural, 64
natural deduction, 38

natural number, 64
natural numbers, 47, 74, 344
negation, 13, 66
negative, 66, 70
neighborhood, 145
node, 141
non-constructive, 21, 41, 53
non-negative, 70
non-positive, 70
normal, 258
normal distribution, 223, 241
notation
  abuse of, 105
  asymptotic, 100
number
  complex, 64
  floating-point, 65
  natural, 64
  rational, 64
  real, 64
number theory, 106

O
  big, 100
o
  little, 100
octonion, 64
odd, 113
odd numbers, 61
Omega
  big, 100
omega
  little, 100
on average, 226
one-to-one, 57
one-to-one correspondence, 57
onto, 57
or, 13
  exclusive, 13
  inclusive, 13
order
  lexicographic, 129
  partial, 128
  pre-, 129
  quasi-, 129
  total, 70, 128, 132
ordered field, 70
ordered pair, 53
orthogonal, 257
orthogonal basis, 258
orthonormal, 258
out-degree, 145
outcome, 213
over, 67

pairwise disjoint, 166
pairwise independent, 235
parallel edge, 122
parallel edges, 141
partial order, 128
  strict, 128
partially ordered set, 128
partition, 59, 126
Pascal's identity, 178
Pascal's triangle, 179
path, 147, 153
Peano axioms, 344
peer-to-peer, 145
permutation, 170
  k-, 170
pgf, 210, 238
plus, 65
Poisson distribution, 222
poker deck, 298
poker hand, 215
pole, 204
polynomial ring, 271
poset, 128
  product, 129
positive, 70

power, 154
predecessor, 130
predicate logic, 11
predicates, 11, 25
prefix, 129
premise, 34
preorder, 129
prime, 82, 106
prime factorization, 112
probability, 211, 212
  conditional, 218
  discrete, 212
probability distribution function, 221
probability generating function, 210, 238
probability mass function, 221, 241
probability measure, 213
probability space, 213
  uniform, 214
product, 250
product poset, 129
product rule, 168
projection, 264
projection matrix, 266
proof
  by contraposition, 19
proof by construction, 41
proof by example, 41
proposition, 12
  compound, 14
propositional logic, 11, 12
provable, 34
pseudo-ring, 270
Pythagorean theorem, 257

quantifiers, 26
quantify, 11
quasiorder, 129
quaternion, 64
quotient, 107
quotient set, 126

radius of convergence, 204
random bit, 215
random permutation, 215
random variable, 220, 240
  continuous, 241
  discrete, 221
range, 54, 57
rank, 264
ranking, 177
rational, 64
rational decision maker, 226
rational functions, 204
rational number, 64
reachable, 153
real number, 64
recursion, 77
recursive, 84
recursive definition, 77, 78
recursively-defined, 83
reflexive, 125
reflexive closure, 137
reflexive symmetric transitive closure, 137
reflexive transitive closure, 137
reflexivity, 70
reflexivity axiom, 32
regression to the mean, 230
regular expression, 198
relation, 122
  n-ary, 122
  binary, 122
  equivalence, 125
  on a set, 122
relatively prime, 108
remainder, 107
representative, 114, 127
residue class, 113
resolution, 23, 36

resolution proof, 24
resolving, 23
restricted comprehension, 52
restriction, 170
rig, 270
ring, 68, 70, 270
    commutative, 270
    polynomial, 271
rng, 270
round-off error, 65
row, 247
row space, 261
row vector, 255
RSA encryption, 120
Russell's paradox, 48

scalar, 245, 246, 253
scalar multiplication, 246
scalar product, 253
scaling, 245, 262
scaling invariance, 70
second-order logic, 346
selection sort, 132
self-loops, 141
semigroup, 270
semiring, 70, 270
sequence, 247
set comprehension, 48
set difference, 49
set theory
    axiomatic, 52
    naive, 47
set-builder notation, 48
shear, 263
sigma-algebra, 213
signature, 33
signum, 76
simple, 142, 153
simple cycle, 154
simple induction, 77
simple undirected graph, 142
simplification (inference rule), 36
sink, 122, 142
size, 57
sort
    selection, 132
    topological, 132
soundness, 34
source, 122, 142
span, 257
spanning tree, 161
square, 247
square product, 150
standard basis, 258
star graph, 147
state space, 59
statement, 11
Stirling number, 171
strict partial order, 128
strong induction, 81
strongly connected, 153, 154
strongly-connected component, 139
strongly-connected components, 154
structure, 32
sub-algebra, 74
subgraph, 148
    induced, 148
sublattice, 113
subscript, 55
subset, 49
subspace, 257
substitution (inference rule), 39
substitution axiom schema, 32
substitution rule, 32, 39
subtraction, 66
successor, 130
sum rule, 165
supremum, 72
surjection, 57
surjective, 56, 57

symmetric, 125, 249
symmetric closure, 137
symmetric difference, 49
symmetry, 32
syntactic sugar, 55

tail, 142
tautology, 17
terminal vertex, 122, 142
theorem, 9, 34, 348
    Wagner's, 150
theory, 9, 33
Theta
    big, 100
Three Stooges, 47
topological sort, 132
topologically sorted, 155
total order, 70, 128, 132
totally ordered, 128
totient, 119
transition function, 59
transition matrix, 249
transitive, 125
transitive closure, 137, 154
transitivity, 32, 70
translation invariance, 70
transpose, 249
tree, 155, 157
triangle inequality, 256
trichotomy, 70
truth table, 16
    proof using, 17
tuples, 55
turnstile, 34
two-path graph, 295

uncorrelated, 235
uncountable, 63
undirected graph, 141, 142
uniform discrete probability space, 214
uniform distribution, 222
union, 49
unit, 106
universal quantification, 26
universal quantifier, 26
universe, 49
universe of discourse, 25
unranking, 177
upper bound, 71, 86
upper limit, 86

valid, 35, 36
Vandermonde's identity, 180
variable
    indicator, 220
variance, 233
vector, 243, 246, 255
vector space, 243, 246
vertex, 122
    initial, 122
    terminal, 122
vertices, 141
Von Neumann ordinals, 57

Wagner's theorem, 150
weakly connected, 154
web graph, 145
weight, 185
well order, 135
well-defined, 348
well-ordered, 58, 80

Zermelo-Fraenkel set theory with choice, 52
ZFC, 52
Zorn's lemma, 135