Design and Analysis of Algorithms

Chapter 1

What is a Computer Algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

Computer Algorithm or Program?

A computer algorithm is

A program is
- an implementation of one or more algorithms.

Notion of algorithm

```
problem
  ↓
algorithm
  ↓
input
  ↓
“computer”
  ↓
output
```

Algorithmic solution
Example of computational problem: sorting

Statement of problem:
• Input: A sequence of $n$ numbers $<a_1, a_2, \ldots, a_n>$
• Output: A reordering of the input sequence $<a_1, a_2, \ldots, a_n>$
  so that $a_i \leq a_j$ whenever $i < j$

Instance: The sequence $<5, 3, 2, 8, 3>$

Algorithms:
• Selection sort
• Insertion sort
• Merge sort
• (many others)

Selection Sort

Input: array $a[1]$, $\ldots$, $a[n]$
Output: array $a$ sorted in non-decreasing order

Algorithm:

for $i=1$ to $n$
  swap $a[i]$ with smallest of $a[i], \ldots, a[n]$

* see also pseudocode, section 3.1

Some Well-known Computational Problems

• Sorting
• Searching
• Shortest paths in a graph
• Minimum spanning tree
• Primality testing
• Traveling salesman problem
• Knapsack problem
• Chess
• Towers of Hanoi
• Program termination

Basic Issues Related to Algorithms

• How to design algorithms
• How to express algorithms
• Proving correctness
• Efficiency
  • Theoretical analysis
  • Empirical analysis
• Optimality

Algorithm design strategies

• Brute force
• Divide and conquer
• Decrease and conquer
• Transform and conquer
• Greedy approach
• Dynamic programming
• Backtracking and Branch and bound
• Space and time tradeoffs

Analysis of Algorithms

• How good is the algorithm?
  • Correctness
  • Time efficiency
  • Space efficiency
• Does there exist a better algorithm?
  • Lower bounds
  • Optimality
Correctness

- Termination
  - Well-founded sets: find a quantity that is never negative and that always decreases as the algorithm is executed
- Partial Correctness
  - For recursive algorithms: induction
  - For iterative algorithms: axiomatic semantics, loop invariants

Complexity

- Space complexity
- Time complexity
  - For iterative algorithms: sums
  - For recursive algorithms: recurrence relations

What is an algorithm?

- Recipe, process, method, technique, procedure, routine, ...
  - with following requirements:
    1. Finiteness
      - terminates after a finite number of steps
    2. Definiteness
      - rigorously and unambiguously specified
    3. Input
      - valid inputs are clearly specified
    4. Output
      - can be proved to produce the correct output given a valid input
    5. Effectiveness
      - steps are sufficiently simple and basic

Why study algorithms?

- Theoretical importance
  - the core of computer science
- Practical importance
  - A practitioner's toolkit of known algorithms
  - Framework for designing and analyzing algorithms for new problems