A Manifesto for the 21st Century: Computer Science for All!

Bob Sloan
Credits

- Almost exclusively Survey and Opinion talk
- Most work not mine.
- Intro CS work that is mine is joint with UIC Lecturer Pat Troy; privacy preliminary ideas that are mine joint with Kent Law School Prof. Richard Warner.
Core Academic Beliefs and Values
I. The most significant impact we will have on society is educating our students.
II. A research university is a great place to get an undergraduate education and to be a faculty member.
III. Every well-educated college graduate should have mathematical reasoning and problem-solving ability that is normally obtained by studying at least some post-high-school mathematics.
IV. In the early 21\textsuperscript{st} century, it is much more important that every well-educated college graduate study some computer science than mathematics.
Outline

I. Why CS/computational thinking is vital for today’s non-CS, liberal arts majors.
   1. Intellectual ideas, especially process
   2. Understanding how world works
   3. Computational thinking skill set
   4. Citizenship

II. How to teach it, & maybe also certain CS majors
What is computer science?

- Biology = study of living things; chemistry = study of structure and composition of matter
- What is computer science the study of?
- Computer Science ≠ study of computers!
  - Also NOT study of how to use IT tools (Word, Matlab, write a web page, etc.)
- Computer Science is the study of:
  - Processes ≈ Algorithms ≈ Programs
What about IT Fluency?

• IT skills are important, but:
  – Very widely needed trade school skill, not part of liberal higher education
  – May be completely appropriate some places to have a whole college course on but that’s something different
  – Should eventually migrate to K–12 world
  – Easy to mix in few weeks’ worth of this with what I’m discussing today
The Two Cultures
1961 MIT Sloan School Symposium

Computers and the world of the future

Speaker: Sir Charles Percy Snow
Author: London, England

Discussants: Elting E. Morison
Professor of Industrial History
Massachusetts Institute of Technology

Norbert Wiener
Institute Professor, Emeritus
Massachusetts Institute of Technology

Moderator: Howard W. Johnson
Dean and Professor of Industrial Management
Massachusetts Institute of Technology

Speaker: Alan J. Perlis
Director of the Computation Center
Carnegie Institute of Technology

Discussants: Peter Elias
Head, Department of Electrical Engineering
Professor of Electrical Engineering
Massachusetts Institute of Technology

J. C. R. Licklider
Vice President
Bolt Beranek & Newman Inc.

Moderator: Donald G. Marquis
Professor of Industrial Management
Massachusetts Institute of Technology
Learn Programming to Re-Think Process Everywhere

• Alan Perlis argued that *computer science should be part of a liberal education.*
  – Explicitly, he argued that all students should learn to program.
• Why?
  – *Because Computer Science is the study of process.*
  – Automated execution of process changes *everything*
    • Including how we think about things
  – *Process:* an even more important notion than rate of change
Problems as object of study

- Can the **problem** of sorting N words into alphabetical order be solved in a # of steps proportional to N? To N * log N?
- Can the **problem** of determining if an N-line computer program terminates on specific input be solved in # of steps proportional to N? To $2^N$? At All?
Undecidability

• Determining whether program halts is **Unsolvable** (undecidable) problem.
• Fundamental result of Alan Turing, Father of CS, in 1930’s.
• Philosophical, mathematical, practical implications.
The Power and Fear of Algorithms

• The Economist (Sept., 2007) spoke to the algorithms that control us, yet we don’t understand.
  – Credit Ratings, Adjustable Rate Mortgages, Search Rankings

“A handful of people, having no relation to the will of society, having no communication with the rest of society, will be taking decisions in secret which are going to affect our lives in the deepest sense.”

And this is that decisions which are going to affect a great deal of our lives, indeed whether we live at all, will have to be taken or actually are being taken by extremely small numbers of people, who are normally scientists. The execution of these decisions has to be entrusted to people who do not quite understand what the depth of the argument is. That is one of the consequences of the lapse or gulf in communication between scientists and nonscientists.

There it is. A handful of people, having no relation to the will of society, having no communication with the rest of society, will be taking decisions in secret which are going to affect our lives in the deepest sense.
Computational Thinking

• Algorithms are one important example of Computational Thinking—Ways of thought common to computer scientists.
• Other examples
  – Order of growth
  – Layering/abstraction to manage complexity

Computational Thinking
It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Jeanette M. Wing

Computational thinking builds on the power and limitations of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking connects the middle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the questions: What is computation? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To read, write, and do arithmetic, we should add computational thinking to every child’s analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incremenal about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? What is the best way to solve it? Computer science rests on solid theoretical understandings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine’s instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is essential in formulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting, code as data and data as code. It is type checking at the generalization of dimensional analysis. It is recognizing both the vertex and the dangers of slaming, or giving someone or something more than one name. It is recognizing both the cost and power of indexed addressing and procedure call. It is judging a program not just for correctness and efficiency but also for architecture, and a system’s design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using abstractions to describe a system’s behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is
Order of growth

- Before I die, U.S. undergraduate tuition and U.S. medical care cost rates of increase will change.
- Order of growth studies whether growth in resources to solve size-N problem grows
  - Proportionally to $N$ or $N \log N$
  - Proportionally to $N^2$ (or $N^3$)
  - Proportionally to $2^N$
Abstraction managing Complexity

• Coffee mug is porcelain made of mixture of molecules composed of atoms composed of electrons orbiting nucleus of protons and neutrons, each composed of quarks.
• Molecular vibration, quantum effects, etc., at various levels
• Humans evolved to pour morning coffee without worrying about quantum effects!
CS Abstraction Example: Encodings

• But we can interpret these numbers any way we want.
  – We can encode information in those numbers
• Even the notion that the computer understands numbers is an interpretation
  – We encode the voltages on wires as 0’s and 1’s, eight of these defining a byte
  – Which we can, in turn, interpret as a decimal number
Layer the encodings as deep as you want

- One encoding, ASCII, defines an “A” as 65
  - If there’s a byte with a 65 in it, and we decide that it’s a string, POOF! It’s an “A”!

- We can string together lots of these numbers together to make usable text
  - “77, 97, 114, 107” is “Mark”
  - “60, 97, 32, 104, 114, 101, 102, 61” is “<a href=” (HTML)
What do we mean by *layered* encodings?

- A number is just a number is just a number
- If you have to treat it as a letter, there’s a piece of software that does it
  - For example, that associates 65 with the graphical representation for “A”
- If you have to treat it as part of an HTML document, there’s a piece of software that does it
  - That understands that “<a href=” is the beginning of a link
- That part that knows HTML communicates with the part that knows that 65 is an “A” & 97 an “a”
Multimedia is unimedia

• But that same byte with a 97 in it might be interpreted as...
  – A very small piece of sound (e.g., 1/44100-th of a second)
  – The amount of redness in a single dot in a larger picture
  – The amount of redness in a single dot in a larger picture which is a single frame in a full-length motion picture
Software defines and manipulates encodings

• Computer programs manage all these layers
  – How do you decide what a number should mean, and how you should organize your numbers to represent all the data you want?
  – That’s data structures

• If that sounds like a lot of data, it is
  – To represent all the dots on your screen probably takes more than 3,145,728 bytes
  – Each second of sound on a CD takes 44,100 bytes
Our World Rests on Software

• Who in room used a computer outside of school, school work, etc. or email, web browsing, etc. in past 4 days?
• Who in the room has a computer with you right now?
• “Simply because of its inevitable large size, the software capable of performing the battle management task for strategic defense will contain errors. All systems of useful complexity contain software errors.” —Eastport Study Group Report (1985), page 14
  – I.e., No software system big enough to be useful in real world is bug free.
Programming to learn how world is

• I.e., No software system big enough to be useful in real world is bug free.
• After about 4 weeks of programming, you feel this and other key truths about world of 2008 deep in your bones
• A little programming is essential for all to understand what goes wrong and how it goes wrong with our digital world and just how it works.
Policy: Learn some CS TODAY!

• This is a special moment in time for
  – Digital public policy issues
    • Digital IP issues, e.g., music and movies
    • How elections are conducted
    • Etc.
  – Privacy
• In 10 years, the issues will be settled
• Supreme Court Justices, local judges, voters need some real CS understanding to make wise choices
Policy: Analogies fail 1

- What is this like?
- And is it okay?
Jammie Thomas

- RIAA has now brought 30,000 file sharing cases
- Head of litigation for Sony suggested that ripping your own CD to computer illegal: just “a nice way of saying ‘steals one copy’” in Oct. 2007 Jammie Thomas trial.
  - MN single mother of 2 earning $36K/year fined $222,000 for 24 songs on Kazaa in first RIAA lawsuit to go to jury trial.
  - Mistrial declared in September
- Unreasonable, yet we need to have business model for people to make living creating and selling software, music.
- Maybe better policy if more knew bits and bits?
Democracy

• If you have zero knowledge of programming, you do not understand the issues concerning how we just voted
Analogies 2: Porn, child predators

• Want to maximize open communication yet keep unwanted information away from children.
• Is, e.g., MySpace, and Web in general like:
  – Publishing a book?
  – Making a phone call? (Conference call?)
  – Being the trucking firm delivering boxes of books?
  – Television?

• Whose liable if something awful happens when people meet on MySpace, for defamation on line?
• What porn standards apply? *Miller* test involves location!
Privacy: Clearly in flux

• How long does a posting to a web site (Facebook, .Mac account, whatever) remain available?
  – Probably forever

• How can you really know the answer?

• Should you care? (I say yes but I’m over 35.)

• “There oughta be a law!”
  – But creating laws works well only when there are societal norms, and we have none.
Summary

• Computer Science, including algorithmic notions & at least some programming is necessary for a decent liberal education
  – Because of fundamental ideas
  – To understand how things work today
  – To reap benefits of computational thinking
  – To participate in this moment of important policy making.
Part II

• How might we teach computer science to all?

• Two promising approaches:
  1. Media Computation approach, developed by Guzdial at Georgia Tech, tweaked at UIC
  2. Alice, developed by Randy Pausch (deceased), Wanda Dann, Steve Cooper
Media Computation: Teaching in a Relevant Context

- Presenting CS topics with media projects and examples
  - Iteration as creating negative and grayscale images
  - Indexing in a range as removing redeye
  - Algorithms for blending both images and sounds
  - Linked lists as song fragments woven to make music
  - Information encodings as sound visualizations
UIC Approach: MediaComp CS 0.5

- Georgia Tech uses Media Comp as its Liberal Arts CS course

- UIC:
  - Has students that, alas, are not as strong as Georgia Tech’s students
  - Has, like most of country, issues with decline in # of CS majors while demand still huge (really!)
  - Trying to kill several birds with one stone
UIC Media Comp CS 0.5: Two different parts

- Splitting incoming majors into different starting points: CS 0.5 and CS 1.
- CS 0.5 appropriate for non-majors as well, also have toned-down variant of CS 0.5 for only non-majors
- Material taught in CS 0.5: Mild variation on Guzdial’s Media Computation in Python
Problems in Computer Science

• High Attrition Rate at Freshman and Sophomore level
  – 19% National Average
  – As high as 66%
  – 30%-40% at UIC
  – Worse for Female Students

• Not attracting students into the Program
Reasons for High Attrition

- “The traditional approach to CS1 has been found to discourage many prospective computing majors”
- Introductory CS often fail to engage students
- Class are described as
  - too boring
  - overly technical
  - lack creativity
- Don’t even think of using traditional CS1 for general non-majors course!
Reasons for High Attrition

• (Brief commercial for Sloan & Troy CS 0.5 approach)
• Students have wide variations in background and experiences:
  – Slower pace bores those with Greater Experience. Students lose interest!
  – Faster pace loses those with Lesser Experience. Students feel incompetent!
Our Solution

- Divide and Instruct
- Divide Incoming CS major students into two groups
  - Those with greater experience
  - Those with lesser experience
- Use placement exam to determine experience level
Division of Students

• Students not taking or not passing the placement exam follow normal route of CS 0.5 course followed by an aggressive CS 1 course
  – CS 0.5 is not “remedial” but “normal”

• Students passing placement exam are
  – advanced into the aggressive CS 1 course
  – receive credit for our CS 0.5 course
  – “Free” credit motivates the students
def negative(picture):
    for px in getPixels(picture):
        red=getRed(px)
        green=getGreen(px)
        blue=getBlue(px)
        negColor=makeColor(255-red,255-green,255-blue)
        setColor(px,negColor)

def clearRed(picture):
    for pixel in getPixels(picture):
        setRed(pixel,0)

def greyscale(picture):
    for p in getPixels(picture):
        redness=getRed(p)
        greenness=getGreen(p)
        blueness=getBlue(p)
        luminance=(redness+blueness+greenness)/3
        setColor(p, makeColor(luminance,luminance,luminance))
def turnRed():
    brown = makeColor(48,20,17)
    file = "~/Users/sloan/MediaSources/emma.jpg"
    picture = makePicture(file)
    for px in getPixels(picture):
        color = getColor(px)
        if distance(color,brown) < 25.0:
            redness = getRed(px)*1.5
            setRed(px,redness)
    show(picture)
    return(picture)
Posterizing: Reducing range of colors
3D Images: Combining 2 images
Not only programming: How do we compare algorithms?

• There’s more than one way to sample.
  – How do we compare algorithms to say that one is faster than another?

• Computer scientists use something called *Big-O notation*
  – It’s the *order of magnitude* of the algorithm
  – The goal is to describe what happens to the *running time* of the algorithm as the *size of the input grows*

• Big-O notation tries to ignore differences between languages, even between compiled vs. interpreted, and focus on the number of *steps* to be executed.
Advantages of Python

• Fun, relatively easy language
• “Real world” (e.g., heavy use at Google); motivation for some students
• Not terribly Java-like, so not overlapping with Java students will learn later
• Easy to do Internet things, like downloading web pages, and string things, like writing web pages
## UIC Results Summary

<table>
<thead>
<tr>
<th></th>
<th>Average Success Rate (ABC)</th>
<th>Average Retention Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OLD Version</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>75.9%</td>
<td>38.1%</td>
</tr>
<tr>
<td><strong>NEW Version</strong></td>
<td>84.1%</td>
<td>61.4%</td>
</tr>
<tr>
<td></td>
<td>4 semesters of data</td>
<td>3 semesters of data</td>
</tr>
</tbody>
</table>
Diversity Remarks

- **Gender**
  - Male: 95%
  - Female: 5%

- **Ethnicity**
  - African American: 11%
  - Asian: 28%
  - Caucasian: 44%
  - Hispanic: 17%

Our results give another data point to the Media Computation approach showing a different student demographic than earlier work.
Contexualized Pedagogy

• Guzdial would say reason Media Comp approach successful is learning is contextualized by things students know and care about: Digital Media.
  – All collect some
  – Many use Photoshop
  – Some manipulate sound (e.g., GarageBand)
Sloan would add

- Media manipulation good because you can see or hear your errors!
- If you looped over wrong pixels of picture, your “filter” is wrong and resulting picture looks wrong; if you looped over wrong sound samples, your frequency manipulation sounds wrong
Alice Approach

- Drag-and-Drop special-purpose programming language/environment created by Alice team (free)
- Programs are 3D animations (“movies” or games)
BRIEF LIVE DEMO OF ALICE HERE
Advantages of Alice

• Syntax of programming—major source of frustration—goes away
• Very easy
• Same or stronger advantage of Media comp seeing errors—in the animation data highly visible
• Pretty Cool Animation in current Alice 2.0; Very Cool Sims coming soon in Alice 3.0
Alice vs. Media Comp.

• Both have multiple papers showing serious learning gains
• Alice’s advantages are its disadvantages: very easy, drag-and-drop instead of control structures.
• Both are great ways to go; depends on audience and instructor interests.
Conclusions

• Computing is a vitally important liberal arts subject today.
  – A good liberal arts education should include some basics of algorithms, programming for all students.
  – The future of all scholarship (IP in digital age) and all privacy just might depend on it.

• There are two appealing approaches to teaching the programming piece.
Sources

• Problems with metaphors in digital policy:

• Media Comp:
  – [http://coweb.cc.gatech.edu/mediaComp-plan](http://coweb.cc.gatech.edu/mediaComp-plan)
  – Sloan & Troy, CS0.5: A Better Approach to Introductory Computer Science for Majors, *Proc. SIGCSE 2008*.

• Alice:
  – [http://www.alice.org](http://www.alice.org)
Thank you for listening!

Questions?