What Is Computer Science?

Molly A. O’Neil
CS 2308 – Spring 2016
But first: what is CS *not*?

Computer Science $\neq$ Programming

*(Really.)*
Still *not* computer science…

Software development & testing

Server administration

Information technology (IT) management

Building a computer hardware

Like programming, computer scientists may do some or all of these things
CS & programming

“Computer science is no more about computers than astronomy is about telescopes.”

[Edsger Dijkstra (maybe)]

Programming is a tool used to do CS

Why do we write code?
- To automate the solving of a problem (or the transforming of information from one form to another) that would be difficult or impossible to solve (transform) by hand
Computer Science, defined:

My definition:
Computer Science is the study of computation and its applications to problem-solving

More formally:
“Computer science is the systematic study of the feasibility, structure, expression, and mechanization of the algorithms that underlie the acquisition, representation, processing, storage, communication of, and access to information.” [Wikipedia]
An algorithm is…

“A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer” [Oxford Dictionaries]

“A procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation; broadly: a step-by-step procedure for solving a problem” [Merriam-Webster]
Algorithms & problem-solving

An algorithm is a *well-defined sequence of computational steps* to transform some input value (or set of values) into an output value (or set of values) using *finite resources*.

In order to automate the solving of a problem with a computer, we first must:

- Understand the problem
- Develop a step-by-step procedure by which the problem can be solved
- Evaluate whether the procedure consumes a reasonable amount of resources
Complexity & some questions

Computer Science is also the study of complexity

- How can this problem be solved?
  - Can it be solved at all? (Some problems can be proven non-computable; some solutions have intractable running times)

- Can you create an automatable, step-by-step process to solve the problem?

- Is this the “best” solution?
  (And what does “best” even mean?)
  - Is it the fastest solution?
  - The solution that requires the least memory?
  - That consumes the least power or energy?
Complexity & some questions

Computer Science is also the study of complexity

- How can this problem be solved?
  - Can it be solved at all? (Some problems can be proven non-computable; some solutions have intractable running times)

- Can you create an automatable, step-by-step process to solve the problem? 😡 Only this one is programming!

- Is this the “best” solution?
  - (And what does “best” even mean?)
    - Is it the fastest solution?
    - The solution that requires the least memory?
    - That consumes the least power or energy?
There are usually many algorithms to solve a particular problem. Some may be “good”, some may be “bad”. How to choose the “best”? Sometimes a single algorithm can be applied to solve multiple problems.
What this means for you in CS1 or 2

Learn the syntax and semantics of C++, but…

Always *start with the problem-solving!*

- Every assignment is a problem to solve
- Start with a whiteboard or pencil/paper
- What are the steps required to solve the problem?
- What does your code need to do?
- Eventually: Will your code perform well enough?
A (very brief) early history

1804: Joseph-Marie Jacquard invents a loom that can weave complicated patterns described by holes punched in cards.

1837: Charles Babbage describes an Analytical Engine with memory, an arithmetic unit, and ability to interpret a programming language with loops and conditionals. His friend, the mathematician and writer Ada Lovelace, writes a report describing his machine and designing the first algorithm intended to be executed on it.

1936: Alan Turing constructs the first formal model of a computer and shows that there are problems his machine cannot solve (e.g., the halting problem).
A (very brief) early history, part II

1940s: Military code-breaking and ballistics calculations drive the development of several computational machines, including the ENIAC (one of the first general-purpose computers)

1947: The invention of the transistor

1951: Grace Hopper invents the notion of a compiler, allowing the development of high-level languages
What’s the common thread between these inventions?

The general-purpose computer as we know it hadn’t been *invented* yet
What’s the common thread between these inventions?

The general-purpose computer as we know it hadn’t been *invented* yet

You don’t need a computer to do computer science!
CS is a huge field

In rough order from *theoretical* to *applied*…

- **Algorithms, Computation, & Complexity Theory**
  - Focused on answering fundamental questions about what problems are computable and what resources are required to perform those computations

- **Information & Coding Theory**
  - Studies the properties of systems that convert information from one form to another and their theoretical limits (e.g., data compression, cryptography, error detection and correction, data transmission over a network)

- **Programming Languages**
  - Concerned with the design, implementation, and formal properties of languages that allow humans to express algorithmic instructions to computers
CS is a huge field (continued)

- **Artificial Intelligence & Machine Learning**
  - Devises processes by which computers can perform tasks requiring decision-making, learning, environmental adaptation, and/or communication

- **Databases, Data Mining, & Big Data**
  - Studies methods for storing and efficiently searching, retrieving, and discovering patterns in (increasingly vast amounts of) data

- **Computer Security & Cryptography**
  - Attempts to protect information from unauthorized access or modification while retaining accessibility for authorized users
CS is a huge field (continued)

- **Computational Science**
  - Concerned with constructing techniques by which computers can solve scientific problems across many domains, often through computer simulation (e.g., computational physics, bioinformatics)

- **Computer Graphics & Visualization**
  - Studies methods for synthesizing and manipulating visual content as well as processing and analyzing images to convert them to information (e.g., computer vision)

- **Operating Systems & Compilers**
  - Concerned with the development and structure of programs that manage the interactions of a computer’s hardware and software resources, and with the development of programs to translate high-level code into optimized machine instructions
CS is a huge field (continued)

- **Parallel & Distributed Systems**
  - Studies the properties and design of computational tasks that can be broken down into several independent subtasks, and the design and implementation of systems capable of executing these sub-tasks concurrently while allowing inter-process communication

- **Computer Architecture & Computer Engineering**
  - Focuses on improving the design and implementation of computer systems to enable broader classes of problem to be solved

- **Software Engineering**
  - Applies rigorous engineering techniques to the design, development, testing, and maintenance of large software projects
CS is a huge field (continued)

- **Parallel & Distributed Systems**
  - Studies the properties and design of computational tasks that can be broken down into several independent subtasks, and the design and implementation of systems capable of executing these sub-tasks concurrently while allowing inter-process communication.

- **Computer Architecture & Computer Engineering**
  - Focuses on improving the design and implementation of computer systems to enable broader classes of problem to be solved.

- **Software Engineering**
  - Applies rigorous engineering techniques to the design, development, testing, and maintenance of large software projects.

...this wasn’t even close to an exhaustive list.
A few cool algorithms…

(...described much too quickly)
PageRank [Sergey Brin & Larry Page, 1998]

**The problem:** How to determine the importance of webpages so search results can be ordered

**The key insight:** If a lot of other webpages link to a particular page, that page is likely important

**The algorithm:**
- Assign each page an initial equal rank, essentially a probability of a person who is clicking on random links arriving at the page
- Over many iterations…
  - Each pages transfers a fraction of its rank to all the outbound linked pages
  - Stop iterating when the page ranks (probabilities) converge, i.e. stop changing
The problem: Simulating the interactions of stars in a galaxy requires computing the force applied to every star by every other star.

The key insight: Two stars that are very far from one another don’t exert much force on one another individually, but rather as one part of a nearby cluster of stars.

The algorithm:

- Divide the space around all the stars into a hierarchy of cubic cells, where cells higher up the hierarchy represent groupings of multiple stars or sub-cells.
  - Compute the center of gravity of each cell.

- When calculating star-pair force interactions, only nearby stars need to be treated individually. Stars in distant cells can be treated as the center of gravity of the entire cluster.
Ant Colony Optimization [Marco Dorigo, 1992]

**The problem:** Find a good path through a graph (e.g., find the order in which to visit N cities and return to starting city, traveling the shortest total distance)

**The key insight:** Ants travel randomly looking for food. When they find some, they return to their colony, laying pheromone on the route. If another ant finds the trail, it is likely to follow it rather than wandering randomly. Pheromone evaporates over time, so the longer the trail, the more pheromone will evaporate in the time it takes an ant to move back and forth. Pheromone density will accumulate on short paths more than on long ones.

**The algorithm:**
- With many code threads running in parallel, simulate this behavior by having each thread traverse the graph looking for better routes.
- Threads increase the weight on graph edges that appear in good solutions.
Some recent problems CS has solved...

(“Solved”? Solved well enough for now?)
Diagnosing cancer

**IBM Watson:** supercomputer capable of natural language processing, information retrieval from TB of data, and machine learning

Current project: Find personalized treatments for cancer patients by comparing treatment and disease histories, symptoms, scans, and genetic data with huge quantities of medical literature

- Watson can make treatment recommendations in minutes based on data analysis that would take weeks for a team of human researchers
- Success rate of 90% in diagnosing lung cancer, compared to 50% for human doctors

Image location sleuthing

Graphics researchers developed an algorithm that can analyze the distribution of texture, color, and lines in a photo and identify where the photo was taken (by searching for photos with a similar appearance in GPS-tagged images on Flickr)

Applications:

- Image search
- Forensics

Terrorist network mapping

Social Network Analysis (SNA) applies network and graph theory to social structures, such as friend and contact networks, disease transmission, etc.

Algorithms have been developed to identify nodes with high centrality: nodes connected to the most other nodes, nodes that are on the most shortest paths between other nodes, critical nodes that bridge otherwise unconnected sub-networks, etc.

For example, if you can identify central nodes using the communication records of suspected terrorists, you may be able to identify the leadership and whom to target to dismantle the network

- Has been retroactively applied to identify the 9/11 hijackers
- What the NSA is doing with all that communications data it collects…?

[http://www.slate.com/articles/news_and_politics/explainer/2006/05/how_the_nsa_does_social_network_analysis.html]
Acknowledgments

Introduction to CS, CS 5, Harvey Mudd
http://www.cs.hmc.edu/~cs5grad/cs5/LectureSlides/Lec1_15_gold_morn.pptx

What is Computer Science?
https://www.cs.mtu.edu/~john/whatiscs.html

A Very Brief History of Computer Science, Jeffrey Shallit
https://cs.uwaterloo.ca/~shallit/Courses/134/history.html

Top 10 Wicked Cool Algorithms, Michael Cooney