

What is an Algorithm?
From Merriam-Webster.com:

a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation

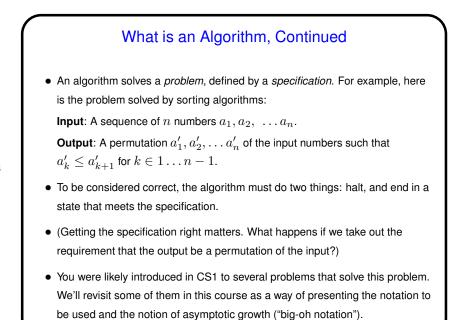
Slide 2

From Donald Knuth:

An algorithm is a finite, definite, effective procedure, with some input and some output.

May be worthwhile (textbook thinks so) to talk a few minutes in general about algorithms and why study them.

1



Slide 3

Applications of Algorithms
Internet: Web search, packet routing, ...
Security: Cell phones, e-commerce, voting machines, ...
Biology: Mapping the human genome, protein folding, ... (Dr. Hibbs's research is in this category.)
Computer graphics: Movies, video games, virtual reality, ...
Physics: Large-scale simulation. (Dr. Lewis's research is in this category.)
And many more ...

Slide 4

Slide 5

"Algorithm" in Popular Usage

- A bit of personal history: My dad spent most of his career in IT (after starting out as an aerospace engineer). Back then, in the mainframe days, keeping track of computer resource allocation and using the records to bill appropriately required a complicated algorithm. My dad spent the last few years of his working life as "keeper of the algorithm", which seemed to involve a lot of complicated calculation and some politics. My mother consequently thinks an algorithm is something extremely mysterious and arcane.
- In the popular press, "algorithm" also has (I think) something of an aura of the arcane, as in: What determines what's in our social media news feed?
 "Algorithms" (meaning something mysterious and arcane and likely to make decisions that sometimes are biased and sometimes not productive of good results). They're not wrong!
- Both of these uses of "algorithm" show how there's a political(?) side to the idea of an algorithm or maybe to the idea of a problem specification.

Broadly-Applicable Problems

- There are a number of CS-y-sounding problems whose solutions are broadly applicable. Examples:
- Finding shortest path through a connected system useful for in map applications, e-commerce to plan delivery schedules, ...

Slide 6

 Finding ordering of operations that sorts on some criterion while maintaining ordering ("topological sort").

"Hard" Problems

• For some problems, the naive or brute-force solution involves examining each of the possible solutions to a problem (e.g., finding the shortest path through a connected system — the classic traveling-salesperson problem). But sometimes that's a lot of possible solutions (on the order of *N*! for some problems). This is computationally inefficient!

Slide 7

• There are many problems in this category; it is thought that there are no efficient solutions to any of them — but if one of them can be solved efficiently, then they all can be. These are the so-called "NP-complete" problems, and one of the things we will look at is determining whether a problem you're trying to solve is in that class. If it is, no need to waste time looking for an efficient solution! (What to do? Often it *is* possible to find an efficient way to produce a good approximation.)

Algorithms as Technology

- Algorithms can be considered fundamental technology, just like hardware, compilers, GUIs, ...
- Some applications probably don't require much in the way of algorithms, but they almost for sure rely for implementation on applications of algorithms (e.g., in the compiler that translates the program into something machine-readable).
- Even techniques that seem to allow for solving complex problems without algorithms— data science, machine learning — are themsolves collections of algorithms.

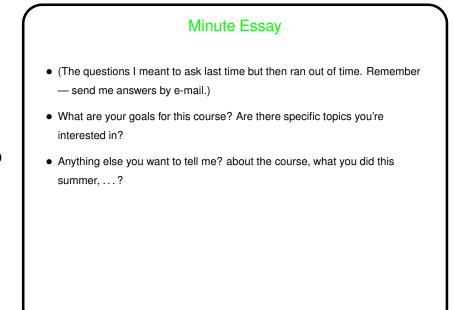
Slide 8

Efficiency of Algorithms

• We'll look at this more mathematically in Chapter 3, but for now we'll say that often we can estimate running time of an algorithm as some constant times a function f(n), where n is the size of the problem, such as size of list in sorting.

Slide 9

- The constant factor is important, and depends on details of the implementation, the compiler used to translate it for machine execution, the hardware used to run it, and many other details that do matter.
- But what matters more, especially for large inputs, is the function if you plot any function whose largest term is *n* and any whose largest term is *n*², over a large range of problem sizes, you'll notice something ...
- (Little story from my grad-school days.)



Slide 10