Slide 1

Administrivia

- (None really, except to note links to presentation on “schedule” page.)
- (This is where I will announce upcoming assignments.)

Slide 2

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

- 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.

Slide 3

What is an Algorithm, Continued

- An algorithm solves a *problem*, defined by a *specification*. For example, here is the problem solved by sorting algorithms:

  Input: A sequence of n numbers $a_1, a_2, \ldots a_n$.

  Output: A permutation $a'_1, a'_2, \ldots a'_n$ of the input numbers such that $a'_k \leq a'_{k+1}$ for $k \in 1 \ldots n-1$.

- To be considered correct, the algorithm must do two things: halt, and end in a state that meets the specification.
- (Getting the specification right matters. What happens if we take out the requirement that the output be a permutation of the input?)
- You were likely introduced in CS1 to several problems that solve this problem. We'll revisit some of them in this course as a way of presenting the notation to be used and the notion of asymptotic growth ("big-oh notation").

Slide 4

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 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.

Slide 6

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

Slide 7

"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!
- 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.)

Slide 8

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 9

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.
- 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

Minute Essay

- (The questions I meant to ask last time but then ran out of time. Remember — send me answers by e-mail.)
- What are your goals for this course? Are there specific topics you're interested in?
- Anything else you want to tell me? about the course, what you did this summer, . . . ?