The Beauty and Joy of Computing

Lecture #6 Algorithms

UC Berkeley EECS
Sr Lecturer SOE
Dan Garcia

Quest (first exam) in **in 7 days!!**

ALAN TURING, FATHER OF CS @ 100

Alan Turing (1912-1954) would have turned 100 this year. He was a brilliant British mathematician (before there was Computer Science), and formalized the concept of "Algorithm". Turing test, Turing completeness, Turing machine, etc.

en.wikipedia.org/wiki/Alan_Turing

What is an algorithm?

- An algorithm is any well-defined computational procedure that takes some value or set of values as input and produces some value or set of values as output.
- The concept of algorithms, however, is far older than computers.

Early Algorithms

- Dances, ceremonies, recipes, and building instructions are all conceptually similar to algorithms.
- Babylonians defined some fundamental mathematical procedures ~3,600 years ago.

Photo credit: Daniel Niles

Algorithms You've Seen

- Addition algorithm (for humans)

```
                          1
  187       187         187
+  53     +  53       +  53
-----     -----       -----
              0           0
```

→

Algorithms You've Seen in CS10

- Length of word
- Whether a word appears in a list
- Whether a list is sorted
- Sort a list
- Pick a random word of length *x* from list

Commonly-Used Algorithms

Luhn algorithm
Credit card number validation

Deflate
Lossless data compression

PageRank
Google's way of measuring "reputation" of web pages

EdgeRank
Facebook's method for determining what is in your news feed

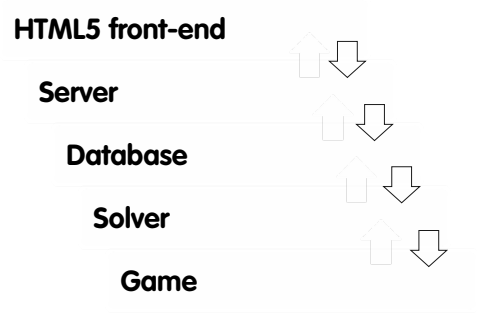
Choosing a Technique

- Most problems can be solved in more than one way, i.e., multiple algorithms exist to describe how to find the solution.
- Not all of these algorithms are created equal. Very often we have to make some trade-offs when we select a particular one.
- We'll talk more about this next time.

Ways to Attack Problems

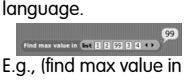
- There are many different categories of algorithms. Two common methods:
- **Top-down**
  - Starting from the top, divide the full problem up into smaller subproblems, working your way down.
  - You often write "stubs" for missing things below to test
- **Bottom-up**
  - Starting from the bottom (smallest thing you need to do), work your way up, building your way up.
  - Your system always "works" as you build layers on top.

Top-down vs Bottom-up example



Algorithms vs. Functions & Procedures

- Algorithms are conceptual definitions of how to accomplish a task and are language agnostic, usually written in pseudo-code.
- E.g., (find max value in list)
  - Set (a temporary variable) the max as the first element
  - Go through every element, compare to max, and if it's bigger, replace the max
  - Return the max
- A function or procedure is an implementation of an algorithm, in a particular language.



- E.g., (find max value in list)

Algorithm Correctness

We don't only want algorithms to be fast and efficient; we want them to be *correct!*

TOTAL Correctness
Always reports, and the answer is always correct.

PARTIAL Correctness
Sometimes reports, and the answer is always correct *when it reports*.

We also have probabilistic algorithms that have a certain *probability* of returning the right answer.

Summary

- The concept of an algorithm has been around forever, and is an integral topic in CS.
- Algorithms are well-defined procedures that can take inputs and produce output (or have side-effects).
- We're constantly dealing with trade-offs when selecting / building algorithms.
- Correctness is particularly important and testing is the most practical strategy to ensure it.
  - Many write tests first!