



UC Berkeley EECS  
Sr Lecturer SOE  
Dan Garcia

# The Beauty and Joy of Computing

## Lecture #6 Algorithms

Quest REVIEW in 8 days!

Quest (first exam) in in 9 days!



UC Berkeley CS  
CS10 TA  
Michael Ball

## PREDICTING THE FUTURE?

MIT researchers recently created an algorithm which they say will be able to predict what topics will “trend” or go viral on Twitter hours before they do. Its accuracy expected to get better with time. They are using Artificial Intelligence (Machine Learning) to get better results.



<http://web.mit.edu/newsoffice/2012/predicting-twitter-trending-topics-1101.html>



# World record for solving a 3x3x3 Rubik's cube?

---

- a) 12 minutes, 3 seconds
- b) 58.1 seconds
- c) 7.96 seconds
- d) 5.66 seconds
- e) 3.31 seconds





[www.youtube.com/watch?v=3v\\_Km6cv6DU](http://www.youtube.com/watch?v=3v_Km6cv6DU)

# Rubik's Cube Champion

**Feliks Zemdegs (b 1995)**  
**5.66 seconds, Melbourne Winter Open**





# 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

- Multiplication algorithm (for humans)

$$\begin{array}{r} 187 \\ \times 53 \\ \hline \end{array}$$
$$\begin{array}{r} 187 \\ \times 53 \\ \hline 1 \end{array}$$
$$\begin{array}{r} 187 \\ \times 53 \\ \hline 2 \\ 187 \\ \hline 1 \end{array}$$





# Algorithms You've Seen in CS10

---

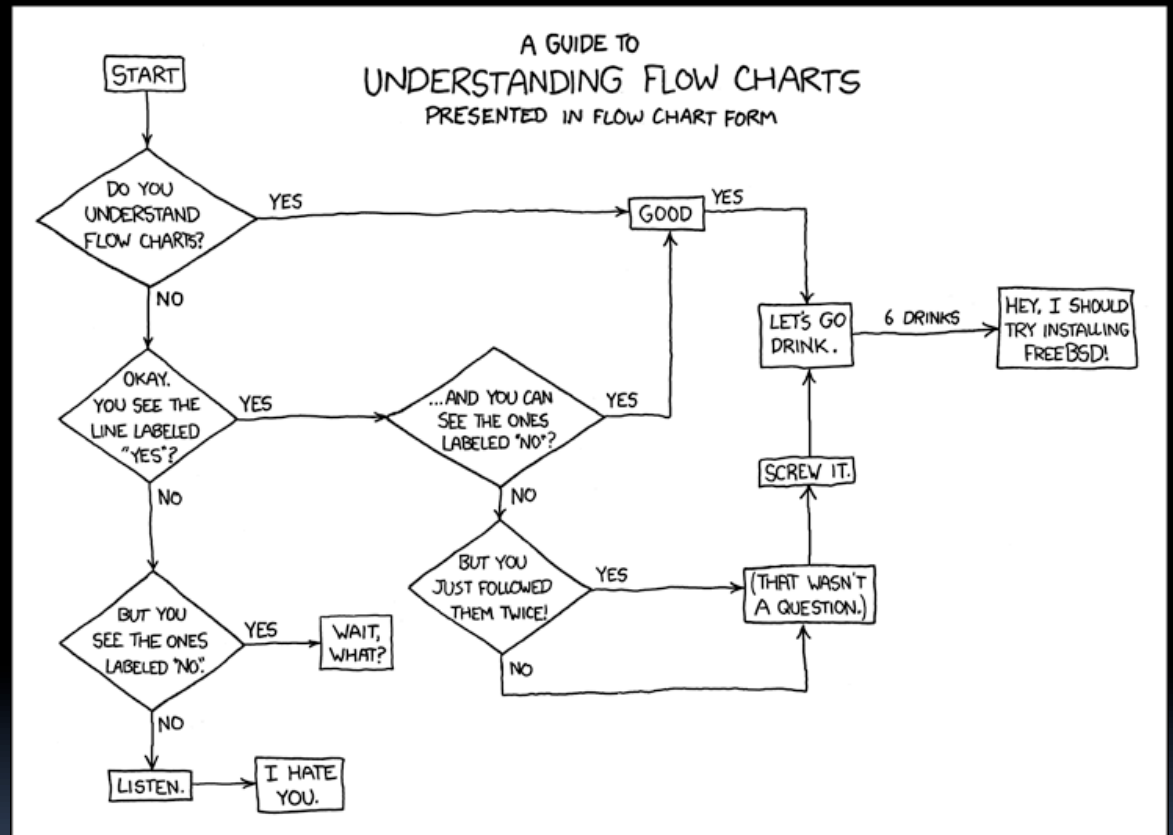
- Length of word
- Whether a word appears in a list
- Whether a list is sorted
- **Greet a List of People (Last Lab)**
- Word Comparisons (You wrote one for HW1!)
- Sort a List (Next lab!)





# How much money is an Algorithm worth?

- a) Nothing!
- b) \$1000 or less
- c) \$1 Million
- d) \$1 Billion
- e) More than \$1 Billion!







# 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 highest up on 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 these next time.





# Ways to Attack Problems

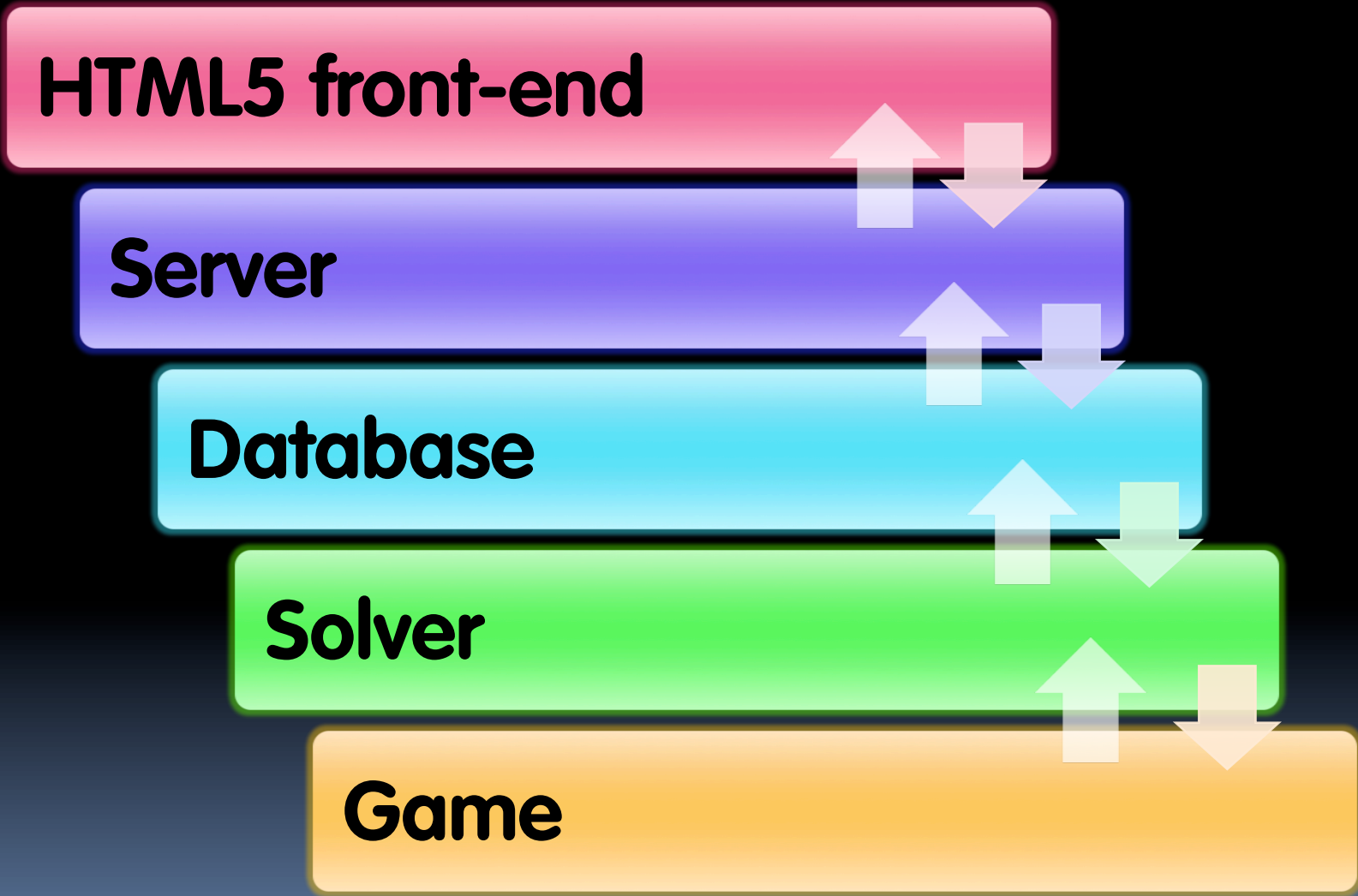
---

- Many ways to approach an algorithm!
- **Top-down**
  - Starting from the top, divide the full problem up into smaller subproblems, working your way down.
  - You often write “stubs” for missing parts below to test your code before it’s done
- **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 of working (smaller) pieces.





# 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**.

- 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.



- Find max value in list





# Algorithm Correctness

---

We don't only want algorithms to be fast and efficient; we also 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

---

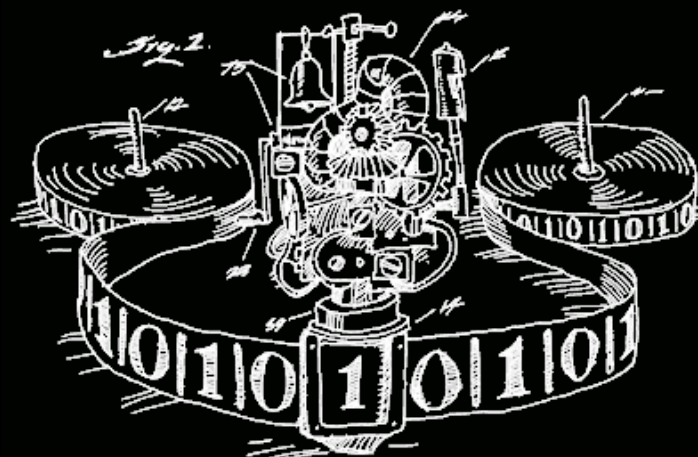
- The concept of an algorithm has been around forever, and is an integral topic in CS.
- Algorithms are **well-defined procedures** that can take inputs and produce output (or have side-effects).
- We're constantly dealing with **trade-offs** when selecting / building algorithms.
- **Correctness** is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



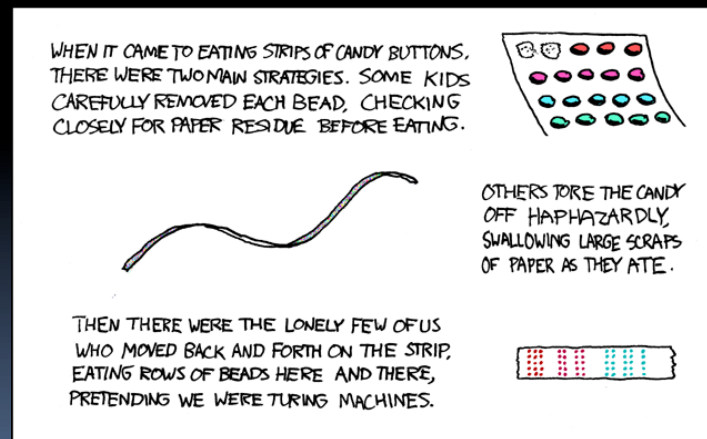


# Turing Completeness

- A Turing Machine has an infinite tape of 1s and 0s and instructions that say whether to move the tape left, right, read, or write it
  - Can simulate any computer algorithm!
- A Universal Turing Machine is one that can simulate a Turing machine on any input
- A language is considered Turing Complete if it can simulate a Universal Turing Machine
  - A way to decide that one programming language or paradigm is just as powerful as another



Turing Machine by Tom Dunne



Xkcd comic "Candy Button Paper"

Garcia & Ball







# Are Scratch & BYOB Turing Complete?

---

- a) Nope, not a chance!
- b) Scratch Only
- c) BYOB Only
- d) Of course!

