Computer Science 50
Fall 2009
Scribe Notes

Week 0 Wednesday: September 2, 2009
Andrew Sellergren

Contents

1 Introduction (0:00–9:00)

- This is CS 50.
- CS 50 stands out amongst Harvard courses primarily because you, the students, will actually learn something after taking it.
- David took CS 50 in 1996 as a sophomore which makes him, what, like 45?
- CS 50 used to have a reputation for rigor which scared a lot of people away. We'd like to dispel some of this fear by saying that while rigorous, CS 50 is certainly doable.
- One of David's first projects as a CS geek (which has since been enhanced) was Shuttleboy, which allows a user to find out the next scheduled shuttle between his origin and destination.
- A more recent project of David's is HarvardEvents which aggregates campus events into a more digestible format.
- In the case of both Shuttleboy and HarvardEvents, a well-defined problem was met with a practical yet creative programming solution.
- CS 50 teaches you not just how to solve problems, but how to solve problems *well*.

2 The Famous Phonebook Example (9:00–21:00)

- Let's say we want to look up Mike Smith in the phonebook. How, as a human, would we do it? Most likely, we would flip about halfway.
- Having done so, we find ourselves in the M's. Now we know that Smith will be in the second half of the book, so we can throw away the first half and continue our search.
- Now we take the second half of the phonebook and divide it in half as well. This brings us to the T's. So we'll throw away the second half this time and continue our search.
- In this way, we are recursively cutting down the scope of the problem. Is this faster than leafing through the phonebook one page at a time? Certainly yes. But how much faster? If the phonebook is 1024 pages long, we only need to split the phonebook in half 10 times. What if the phonebook is 4 billion pages long? Only 32 steps ($\log_2(4 \times 10^9)$).
- Here's what ripping a phonebook in half *actually* looks like.
- Ultimately, our goal is not to turn you all into computer scientists, but into psychologists, biologists, and English. . . ists who know how to think like computer scientists.

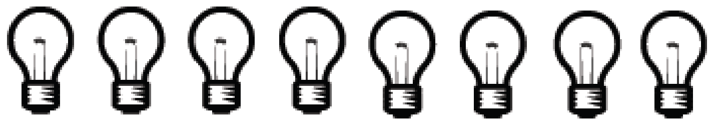
- A sampling of comments from previous students:
  - "This course will take a fair amount of your time, but if you put the work in, a whole new world will open up to you."
  - "You better take this or you're gonna regret it."
  - "It will kick your ass and leave you feeling great."
  - "OMG! CS50 is teh AWESOMENESS!!1!11!!oneoneone (exactly like that, with 'teh' and all)."
  - Speaking of teh awesomeness, check out the History of LOLcats.

3 More Examples (21:00–30:00)

- Suppose we wish to determine the number of students in the lecture hall.
- The naive way would be to have one person stand in front of the lecture hall and point to each student once, saying "1, 2, 3,. . . " until each student has been counted.
- This will take as many steps as there are students, say, 350.
- Here's a better algorithm:
  1. Stand up.
  2. Think to yourself: "I am #1."
  3. Pair off with someone standing, add your numbers together, and adopt the sum as your new number.
  4. One of you should sit down, the other should go back to step 3.
- With this approach, we very quickly get an answer of 386. Here again we see that the "divide and conquer" approach is much more efficient than the linear approach.
- So who are these 386 students? One of your fears might be that everyone to the left and everyone to the right of you is smarter than you. Actually, most of you are probably somewhere in between "those more comfortable" and "those less comfortable" with computer science and technology in general. Last year, the majority of students had never taken a CS course before.
- Last year, female enrollment was up to 38% from 29% the year before. Here's hoping it evens out to 50-50 this year!
- Also, happy 20th birthday to CS 50! We have cake for you, in both the n00b and l33t varieties.

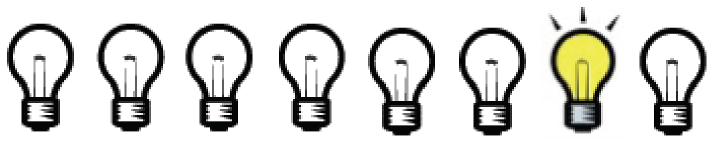
4 Binary (30:00–40:00)

- Everything that goes on underneath the hood of a computer is done in binary, the language of 0's and 1's. Think of each binary digit as a switch or a lightbulb which can be either on or off. By convention, 0 is thought of as "off" whereas 1 is "on."
- So let's say we have 8 lightbulbs in a row to assist with our binary counting. How do we represent 0? If all of the lightbulbs are off, it's equivalent to writing 00000000, or just 0 if we chop off the leading zeroes:

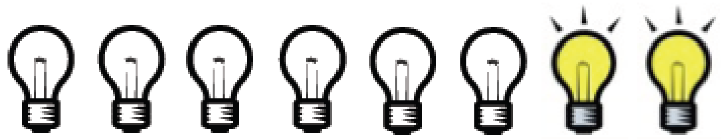


- Now if we want to represent 1, we just turn on that first lightbulb:

- And now, since the second lightbulb represents the 2's column, we turn on only the second lightbulb to represent the number 2:



- Representing the number 3 requires both the 2's column and the 1's column:



- And that's all there is to binary!

- In order to use binary to express alphabetic characters, we'll need some kind of mapping between numbers and letters. This is where ASCII comes in. To see the complete mapping, check out this ASCII table. However, you only really need to know that uppercase A is 65 and lowercase a is 97. This will help with conversions.

5 Course Overview (40:00–74:00)

- Computer scientists count starting from 0, so we will too in enumerating the weeks of the course. Thus we start with Week 0 and go to Week 12, making for 13 weeks total.
- In Week 0, we'll introduce a framework called Scratch which will ease you into programming. Instead of typing snippets of code, you'll be dragging and dropping puzzle pieces that represent programming statements such as the beginning of a loop. Using these puzzle pieces along with Sprites, Scratch's name for objects, you can create fun animations and games. One cute example is Dancing Cookies.
- In Week 1, we'll dive into C, a low-level language that gives us access to manipulating hardware and memory, for better or for worse. A lot of hacks these days are the result of programmers making mistakes in C, especially ones which allow for a buffer overrun attack (don't worry about this now!). Speaking of security, we'll also begin talking about the field of cryptography.
- In Week 3, we'll look at some fundamentals of computer science, including searching, sorting, and recursion. Week 4 will be about memory management.
- In Week 5, we'll take a closer look at how data is stored on disk. We'll get into some of the forensics which David actually used to help catch criminals when he assisted at the Massachusetts DA's office.
- In Week 7, we'll explore more sophisticated data structures. We'll focus on faster, less-wasteful programs.
- In Week 8, we'll go to a higher level and begin learning PHP. This is the language which will allow you to create websites. Keep in mind, though, that the focus of this course is not to teach you a specific programming language, but to teach you how to think as a programmer. Once you learn this, you can learn any programming language you want.
- In Week 9, we'll delve into JavaScript and the aspects of user interface design. In Week 10, we'll finally look at the hardware that's making all of this computing possible.
- The breakdown of grades is as follows:

  - Problem Sets (best 8 out of 9) 60%
  - Quizzes 30%
  - Final Project 10%

- You are welcome to take the course pass/fail, but you'll find that this course is more about your own personal improvement than it is about absolutely ranking students. Even if you consider yourself among "those less comfortable," you won't be starting the course in the bottom third. David sits down with the TFs at semester's end and discusses each and every student to be sure that his or her grade is representative of the work invested. Problem sets are graded along three axes, from 1 to 5 on each. 3 is in fact very good on this scale!

- It is our hope that the course website will prove tremendously useful to you this semester. The bulletin board is where you should direct all questions which don't involve a large portion of your code. All of the handouts we give out in class as well as lecture videos and these scribe notes will also be available for download on the course website.

- Textbooks, which aren't required, but are only there to supplement lectures:

  - Absolute Beginner's Guide to C
  - Programming in C
  - How Computers Work
  - Hacker's Delight

- We'll meet for lecture this Friday and next Friday because of shopping period and Labor Day, but otherwise this course meets only on Mondays and Wednesdays from 1 to 2:30 p.m.

- One final project from last year has actually become an official iPhone App–it's called Rover and is a digital version of the Unofficial Guide. But what other technologies are there down the road? Check out the latest from Apple.

- Sections will be geared toward each of the three groups of students: "those less comfortable," "those more comfortable," and those in between. There will also be a code walkthrough every week which is meant to get you bootstrapped for the problem set each week.

- Our staff is enormous! Check us all out here. We encourage you to get to know us in office hours and virtual office hours. Office hours start next week in the Science Center and virtual office hours start in two weeks in the Virtual Terminal Room thanks to the Elluminate software.

- Here's an overview of **last year's** problem sets:

- *Problem Set 0: Scratch.* Implement your very own program in Scratch.
- *Problem Set 1: C.* Implement your first program in C.
- *Problem Set 2: Crypto.* Encrypt a message using a specific cipher.
- *Problem Set 3: Game of Fifteen.* The interactive puzzle.
- *Problem Set 4: Sudoku.* A step up in UI design.
- *Problem Set 5: Forensics.* Recover a series of JPEGs from a formatted flash drive. Also solve a murder mystery.
- *Problem Set 6: Mispellings.* Write the most efficient spellchecker you can!
- *Problem Set 7: C$50 Finance.* Implement a website that allows users to manage an online stock portfolio.
- *Problem Set 8: Mashup.* Combine Google News and Google Maps.
- *Final Project.* The best part of the course! Showcase your masterpiece at the CS 50 Fair.

• Welcome to CS 50! Come back on Friday!