Programming Abstractions

CS106B

Cynthia Lee
Today’s Topics

1. Quick final exam discussion
   - Details/logistics, topics, sources for practice problems
2. Quarter wrap-up
   - Putting it all together: what have we accomplished together this quarter?
3. What next?
   - Options for continuing your passion for CS after this quarter is done
   - Preview of CS107: security exploits in C/C++
Final Exam
Final Exam Topics

- ADTs
- Recursion
- Backtracking
- Big-O analysis
- Pointers and dynamic memory, linked lists
- Classes and objects
- Binary heaps
- Binary search trees (BSTs), tree traversals
- Light coverage: Sorting algorithms, Graphs, Hashing, Lexicon
Final Exam Study Strategy

- Don’t memorize things—either write it in notes, or learn the concept
  - If you’ve got flash cards, you’re approaching this with the wrong mindset
  - No big multiple choice/true-false section where memorized facts would be tested

- **Read the book** (but only in a targeted way)
  - Computer science is about creating things, so do some practice problems
  - Re-do questions from lecture, do old section problems, do CS106B practice exams
  - Look at lecture videos or book as needed for review of things you identify as weak points when solving problems

- Do the practice exams
  - Study their format—there are templates for questions that I reuse often and it will help you to be familiar with the template, its rules, etc.
## Big O Quick Reference (see also http://bigocheatsheet.com/)

<table>
<thead>
<tr>
<th>What</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Hash table average case (good design)</td>
<td>O(1)</td>
</tr>
<tr>
<td>• Balanced trees</td>
<td>O(logn)</td>
</tr>
<tr>
<td>• Heap, BST with balancing such as Red-Black</td>
<td></td>
</tr>
<tr>
<td>• Binary search on sorted array</td>
<td></td>
</tr>
<tr>
<td>• Linked list find</td>
<td>O(n)</td>
</tr>
<tr>
<td>• Inserting into beginning of array/Vector</td>
<td></td>
</tr>
<tr>
<td>• Hash table worst case</td>
<td></td>
</tr>
<tr>
<td>• Unbalanced tree (e.g. BST) worst case</td>
<td></td>
</tr>
<tr>
<td>• Good sorting</td>
<td>O(nlogn)</td>
</tr>
<tr>
<td>• Mergesort, Heapsort, Quicksort (expected)</td>
<td></td>
</tr>
<tr>
<td>• Bad sorting</td>
<td>O(n²)</td>
</tr>
<tr>
<td>• Insertion, Bubble, Selection, Quicksort (worst case)</td>
<td></td>
</tr>
</tbody>
</table>
Quarter Wrap-Up

What did we set out to do in the beginning?
Where are we now?
Goals for this Course

- Learn how to model and solve complex problems with computers.
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Bring added rigor to your understanding of algorithmic performance, so you can quantitatively compare approaches for solving problems.
From here on out, there are no obvious answers to any problem worth your hourly rate. 😊

- Programming is all about exploring new ways to **model** and **solve** problems.

- There are **CHOICES** and **TRADEOFFS** in how we model these and how we implement them! (array or linked list? BST or hash table?)

- Skilled computer scientists recognize that any problem worth tackling has **many** possible models and **many** possible solutions, often none of which is clearly better than the others in all dimensions—**tradeoffs**!
That’s a lot of material to cover in 10 weeks

You are part of a very challenging course, in the best CS department in the world, and you are so, so close to completing this course!

Congratulations!! You’ve almost made it through CS106B!

- So…what next?
The CS Core

CS106B/X
Programming Abstractions

CS107
Computer Organization and Systems

CS103
Mathematical Foundations of Computing

CS109
Intro to Probability for Computer Scientists

CS110
Principles of Computer Systems

CS161
Design and Analysis of Algorithms

Systems
Theory

Stanford University
Can computers solve all problems?
Spoiler: no!

Why are some problems harder than others?
We can do find in an unsorted array in $O(N)$, and we can sort an unsorted array in $O(N\log N)$. Is sorting just inherently a harder problem, or are there better $O(N)$ sorting algorithms yet to be discovered?

How can we be certain about this?
How do we encode text, numbers, programs, etc. using just 0s and 1s?

Where does memory come from? How is it managed?

How do compilers, debuggers, etc. work?
CS107 in the news: Heartbleed

- In April 2014, security experts warned that users of thousands of major websites needed to change their passwords due to potential exposure caused by the “Heartbleed” vulnerability.

- Heartbleed exploited a buffer overrun bug in OpenSSL.
  - SSL is the layer that secures web interactions, i.e., it’s what make the “s” in “https://” mean something.
CS107 in the news: Heartbleed

- The protocol allows you to send “heartbeat” messages, which basically say:
  - Are you still there? If you are, repeat this message back to me: "hello" [0x0005 bytes].
  - Each char is one byte, so 5 letters

- Unfortunately, the software also let you send messages like this:
  - Are you still there? If you are, repeat this message back to me: "hello" [0xFFFF bytes].
  - That's 65535 bytes—much more than the length of "hello"!
  - So the software would continue for-looping past the end of the "hello" array, sending information back
  - Which causes an error, right? RIGHT??
    Turns out, no.
CS107 in the news: Chrome

- On Oct 31, 2019, Google disclosed that there was a bug in Chrome that caused a security breach

- The bug was that the program accesses memory after it has already been freed/deleted
  - Usually works, but incorrect and sometimes causes the bug
What CS107 Isn't

- CS107 is **not** a litmus test for whether you can be a computer scientist.
  - You can be a *great* computer scientist without enjoying low-level systems programming.

- CS107 is **not** indicative of what programming is “really like.”
  - CS107 does a lot of low-level programming. You don't have to do low-level programming to be a good computer scientist.
CS107E
Computer Organization and Systems—*Embedded*

- Counts for prerequisites etc. the same as regular CS107, but covers the topics with a new twist: embedded work on Raspberry Pi
Other CS Courses
CS106L
Learning the Standard Template Library (STL)

- In CS106B/X, we learn the Stanford Library containers
- *Now learn the industrial-strength ones!*
CS181
Computers, Ethics, and Public Policy

- Some sample news headlines recently:
  - Edward Snowden reveals that NSA knows more about you than your parents do
  - How should AirBnB be taxed?
  - Is it ethical for Facebook to manipulate your emotions with the newsfeed algorithm?

*We have the power to control and create technology, but how should we use it?*
Current Efforts - Google

GFE = Google Front End Server
SSL Added and removed here! 🚨
Traffic in clear text here.
CS108
Object-Oriented Systems Design

- How do you build large software systems in a team?
- Introduction to things you need to know for work in the “real world”:
  - Unit-testing frameworks
  - Object-oriented design
  - Multithreaded applications
  - Databases and web applications
  - Source control
CS193
Language-specific courses

- Misc. offerings throughout the year, focused on specific technologies:
  - CS193A: Android Programming
  - CS193C: Client-Side Web Technologies
  - CS193I: iOS Programming
  - CS193L: Lua Programming
  - CS193P: iPhone and iPad programming
Options besides CS Major

CS Minor: only 5 more classes!
- 103, 107, 109, two your choice—fun!

CS Coterminal MS degree
- Earn an MS in CS while you are here earning your BS
- Possible for CS majors and other majors
  > ex: Math major, CS co-term