Life after CS106B

What are you up to next in life?

PollEv.com/cs106bpolls

Roadmap

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

Object-Oriented Programming

Implementation

arrays

dynamic memory management

linked data structures

real-world algorithms

Midterm

Life after CS106B!

Core Tools

testing

algorithmic analysis

recursive problem-solving

Roadmap

C++ basics

User/client

vectors + grids

stacks + queues

sets + maps

Core
Tools

testing

**Object-Oriented
Programming**

Midterm

**algorithmic
analysis**

**recursive
problem-solving**

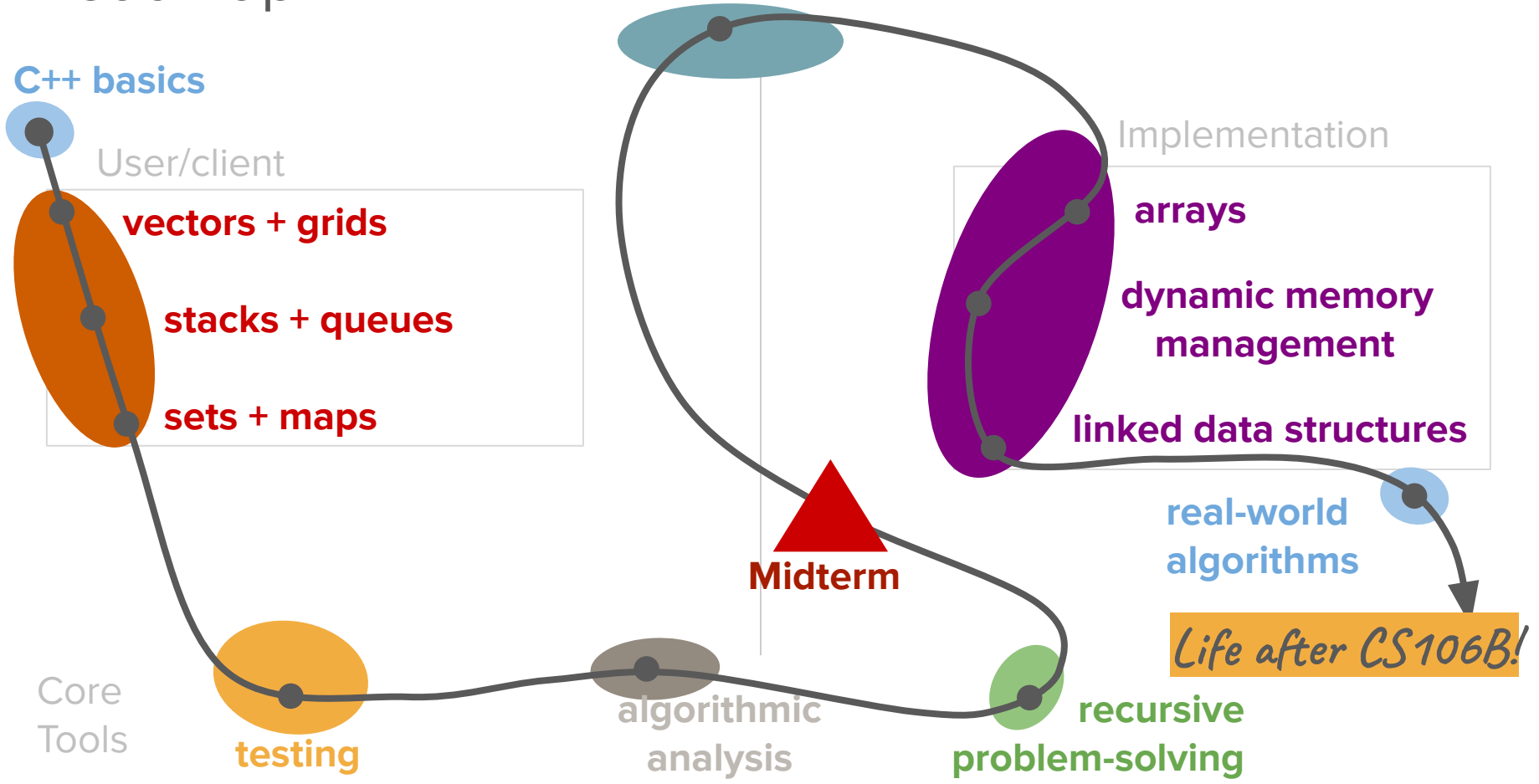Implementation

arrays

**dynamic memory
management**

linked data structures

**real-world
algorithms**

Life after CS106B!

Today's question

What does life after CS106B look like?

Today's topics

1. Course recap

2. Future areas of exploration

3. Ask us anything!

Course Recap

[everything!]

Who were you?

Who were you?

- You came from these hometowns!

Our CS106B Hometowns

chase detroit)
verne ut venice seattle (metro
maryland bend tx shandong canada
nebraska oregon ca cupertino houston
toronto jose la china san colorado redmond
california washington beijing logan
fremont troy tn chevy mi
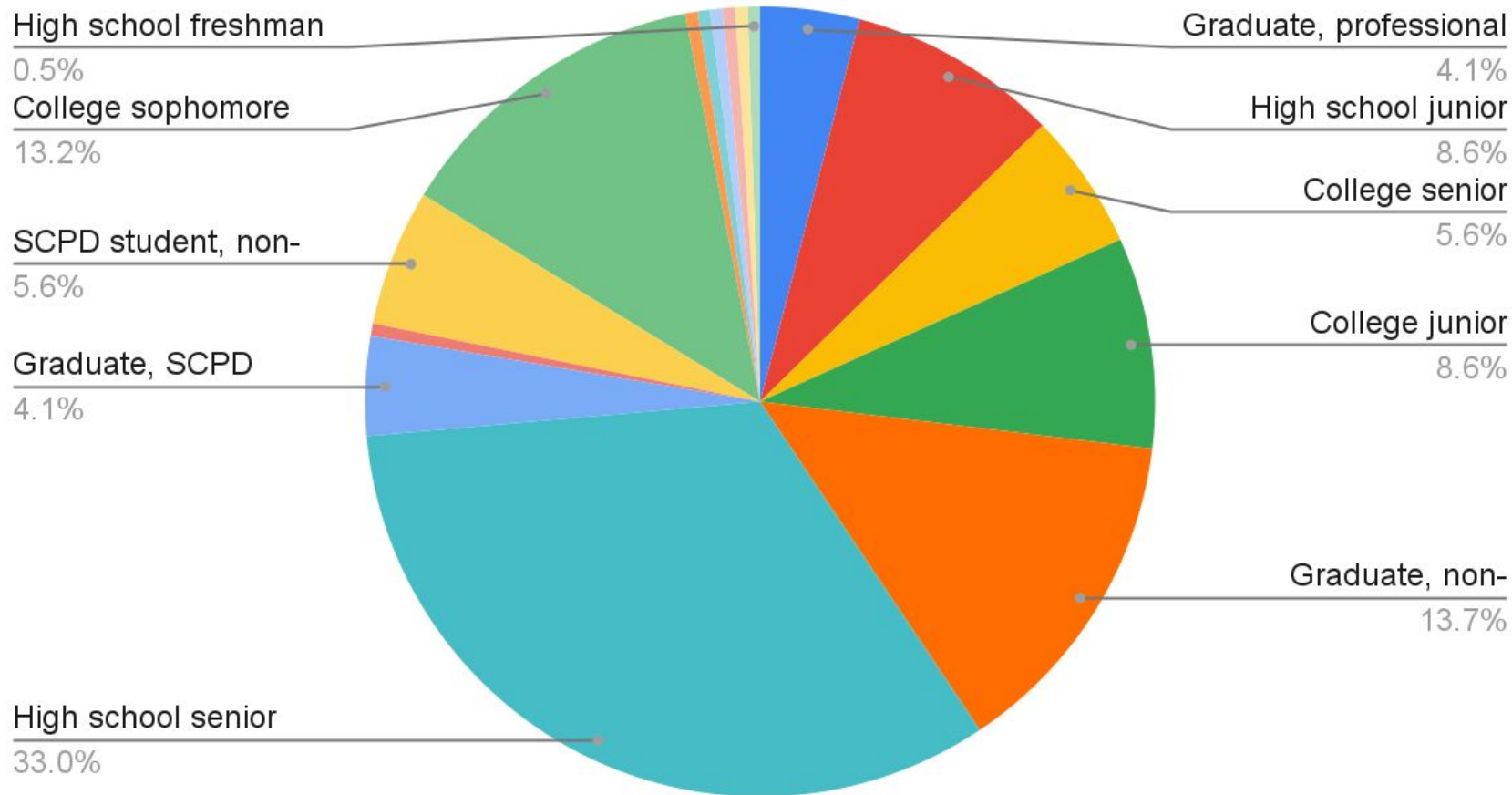galveston ramon shenzhen cleveland ohio francisco memphis omaha sunnyvale

Who were you?

- You spanned many different hometowns!

- You ranged from high school freshmen to career professionals!

What is your class year (at Stanford / other)?

- High school freshman 0.5%
- College sophomore 13.2%
- SCPD student, non- 5.6%
- Graduate, SCPD 4.1%
- High school senior 33.0%
- Graduate, professional 4.1%
- High school junior 8.6%
- College senior 5.6%
- College junior 8.6%
- Graduate, non- 13.7%

Who were you?

- You spanned many different hometowns!

- You ranged from freshmen to career professionals!

- You were most looking forward to learning …

type of data

overall greater understanding

aspects of computer   computer science project

computer science concept   stronger algorithm skills

algorithm   fundamental of c

recursion   comp science framework   software engineer way   understanding of computer

real world application   future career path

class   shazam app   understanding of data

real world problem

new people

quality of people   understanding of c

memory management   real life   complex computer science

higher level computer   knowledge   complex problem

better programmer

level programming language   best c professor

computer science class   new programming language

basic of c   kind of data

computer science   programming

skills   learning   programming knowledge

data structure   final project   taken ap csa

real life application   solid programming style   new things   better software developer

programming skills   new language   general programing skills

advanced computational class

Our learning goals for the quarter

- I am excited to use programming to solve real-world problems I encounter outside class.

- I recognize and understand common abstractions in computer science.

- I can identify programmatic concepts present in everyday technologies because I understand how computers process and organize information.

- I can break down complex problems into smaller subproblems by applying my algorithmic reasoning and recursive problem-solving skills.

- I can evaluate design tradeoffs when creating data structures and algorithms or utilizing them to implement technological solutions.

What you've learned this quarter

You've learned…
- C++ fundamentals (types, structs, pass by value/reference)
- Good testing practices
- Console programs and C++ strings
- ADTs: Vectors, Queues, Stacks, Maps, Sets, Priority Queues, HashMaps, HashSets
- Breadth-First and Depth-First Search
- Big-O notation and algorithmic analysis
- Recursion: Recursive problem-solving, fractals, backtracking

- Classes and Object-Oriented Programming
- Dynamic memory and pointers
- Arrays and linked data structures (trees, linked lists, graphs)
- Binary heaps
- Sorting algorithms (mergesort, quicksort, selection sort, insertion sort)
- Binary trees and Huffman encoding trees
- Hashing
- Parallel computing
- Programming languages and graphs
- STL ADTs!

What you've made this quarter

Assignment 1: C++ legs

NATIONAL ARCHIVES

Search Archives.gov [Search]

RESEARCH OUR RECORDS | VETERANS' SERVICE RECORDS | EDUCATOR RESOURCES | VISIT US | AMERICA'S FOUNDING DOCUMENTS

Research Our Records

Census

- About Census Records
- Search Census Records Online
- 1940 Census
- 1930 Census FAQs
- 1850-1930 Census Clues
- 1790-1840 Census Clues
- Nonpopulation Census
- 1935 Business Census
- Indian Census Rolls
- Presidents in the US Census Records
- Census Links by Year

Resources

- Soundex Coding System
- Blank Census forms and charts
- Order Census Copies Online
- Using Census Microfilm

Soundex System

The Soundex Indexing System

Updated May 30, 2007

To use the census soundex to locate information about a person, you must know his or her full name and the state or territory in which he or she lived at the time of the census. It is also helpful to know the full name of the *head of the household* in which the person lived because census takers recorded information under that name.

The soundex is a coded surname (last name) index based on the way a surname sounds rather than the way it is spelled. Surnames that sound the same, but are spelled differently, like SMITH and SMYTH, have the same code and are filed together. The soundex coding system was developed so that you can find a surname even though it may have been recorded under various spellings.

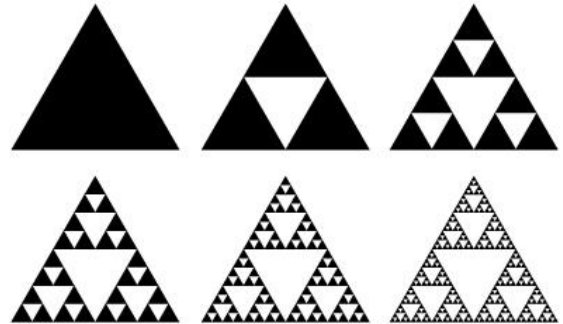To search for a particular surname, you must first work out its code.

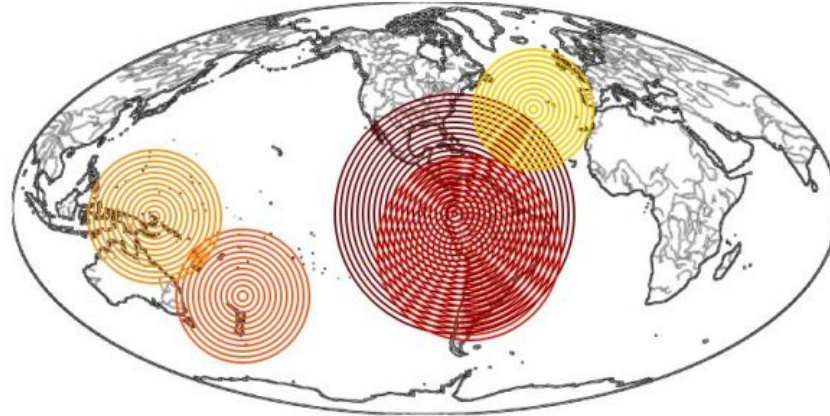# Assignment 2: Fun with Collections
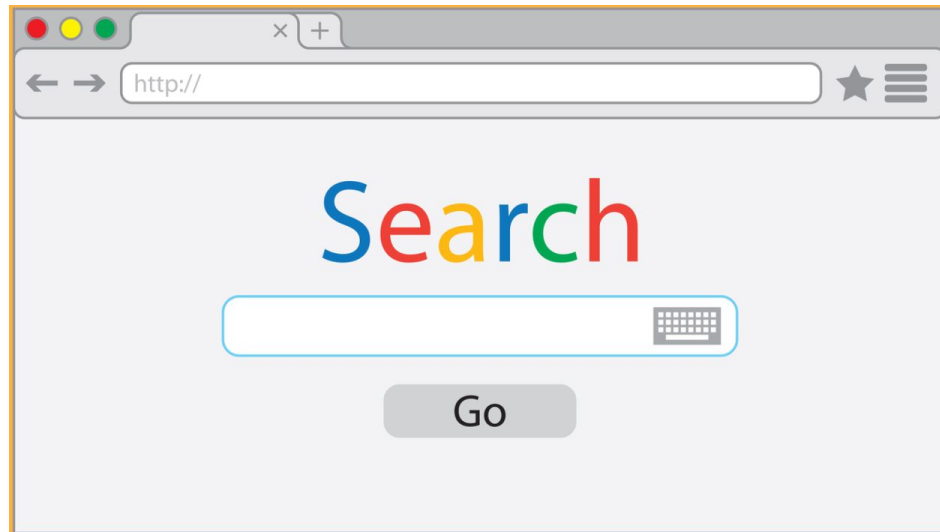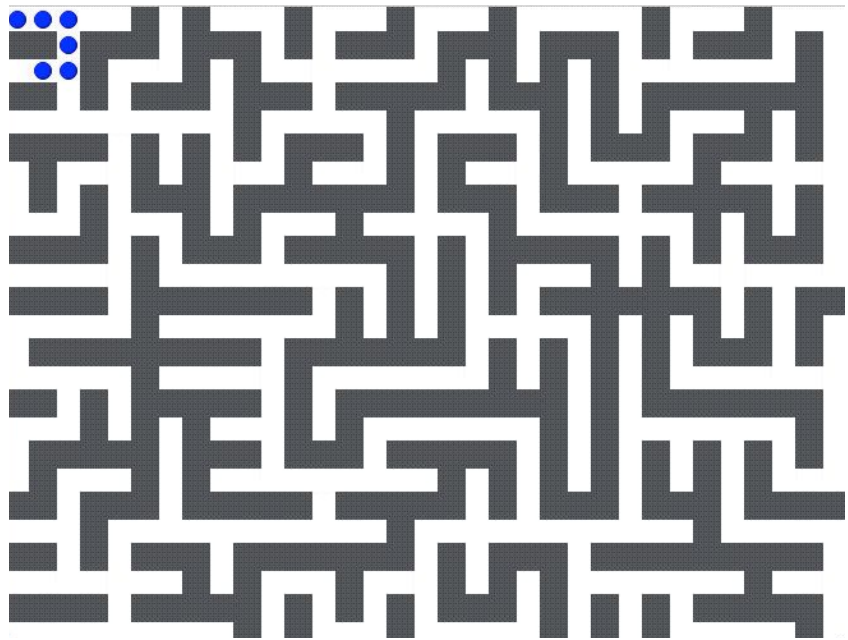
Assignment 3: Recursion

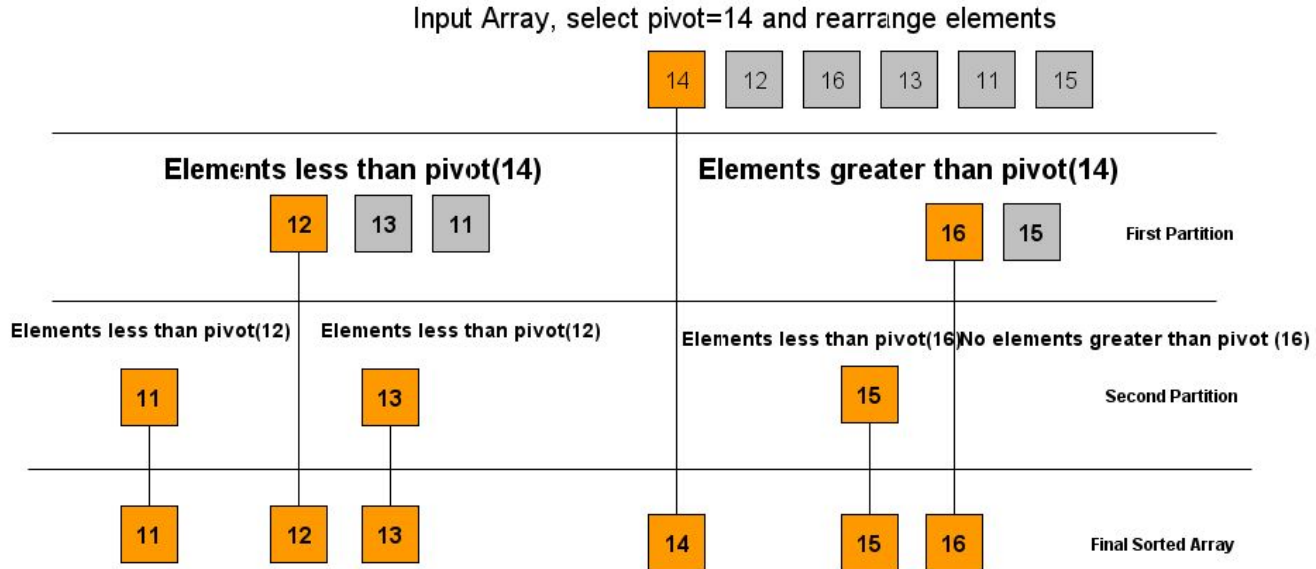{ ( ) [ ] { } ( ) ( ) }

Assignment 4: Priority Queue

This tool displays the strongest recent earthquakes reported by the US Geological Survey. You can use the controls on the side of the window to select the time interval you're interested in. This visualizer will show the 5 strongest earthquakes within that interval.
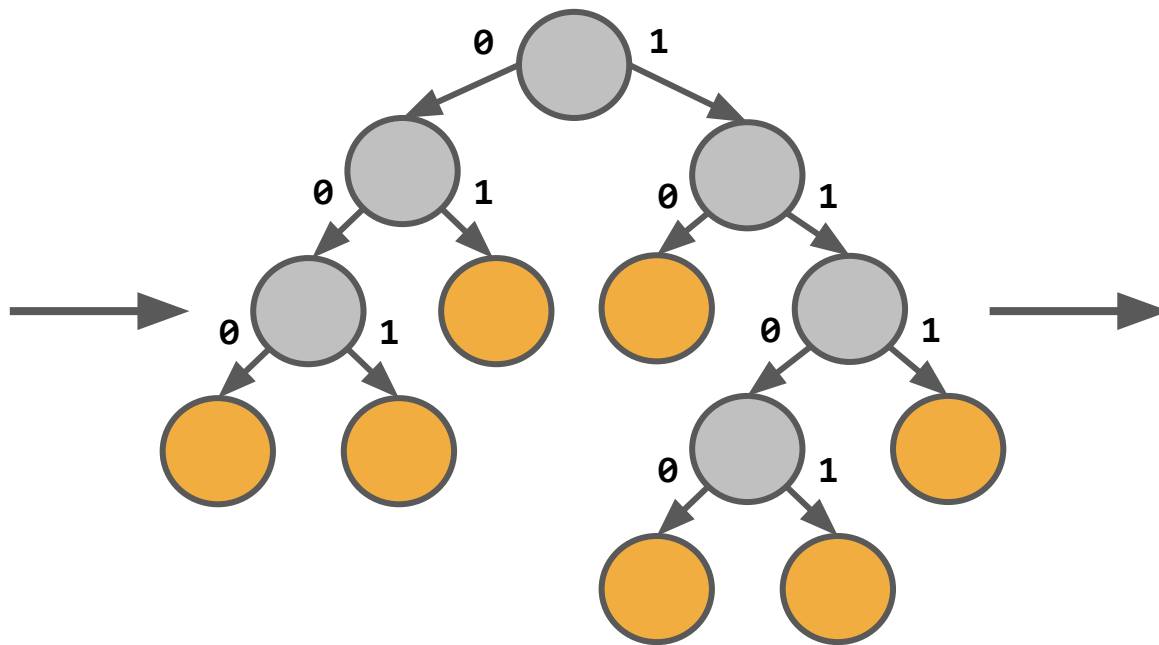
Remember that the earthquake magnitude scale is logarithmic. An earthquake that is on magnitude in strength higher than another releases around 32 times as much energy.

■ Magnitude 7.5 115km ESE of Palora, Ecuador at 02:17:22 AM on Feb 22, 2019
■ Magnitude 7 27km NNE of Azangaro, Peru at 12:50:41 AM on Mar 01, 2019
■ Magnitude 6.4 116km SE of L'Esperance Rock, New Zealand at 07:46:14 AM on Mar 06, 2019
■ Magnitude 6.4 49km NW of Namatanai, Papua New Guinea at 06:35:55 AM on Feb 17, 2019
■ Magnitude 6.2 Northern Mid-Atlantic Ridge at 11:57:05 AM on Feb 14, 2019

Assignment 5: Linked Lists and Sorting

Input Array, select pivot=14 and rearrange elements

14 | 12 | 16 | 13 | 11 | 15

Elements less than pivot(14)   **Elements greater than pivot(14)**

12 | 13 | 11   16 | 15   **First Partition**

Elements less than pivot(12)   Elements less than pivot(12)   Elements less than pivot(16) No elements greater than pivot (16)

11   13   15   **Second Partition**

11 | 12 | 13   14   15 | 16   **Final Sorted Array**

Assignment 6: Huffman Encoding

Questions you can start to answer

- What is possible with technology and code? What isn't possible?

- How can I use programming to solve problems that I otherwise would not be able to?

- What makes for a "good" algorithm or data structure? Why?

- Which problems should I solve with algorithms and data structures? What does a responsible programmer do when using data about real people?

What does life look like after CS106B?

Computer science is more than just programming!

Computer science is more than just programming!

These skills will make you better at whatever you choose to do in life!

What can I study within computer science?

(concentrations within CS)

Concentrations in computer science (at Stanford)

- Artificial Intelligence

**Possible applications for Artificial Intelligence**

Virtual Personal Assistant · Natural language processing · Deep Learning · Simulation Modelling · Machine Translation · Visualization · Artificial Intelligence · Social Network Analysis · Audio Analytics · Graph Analytics · Image Analytics · Internet of Things · Robotics & Soft Robotics · Machine Learning
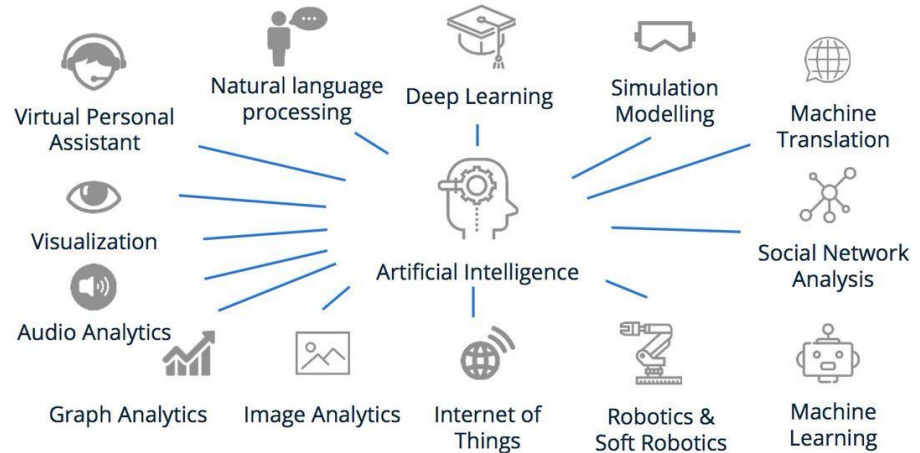
source statista via @mikequindazzi

Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
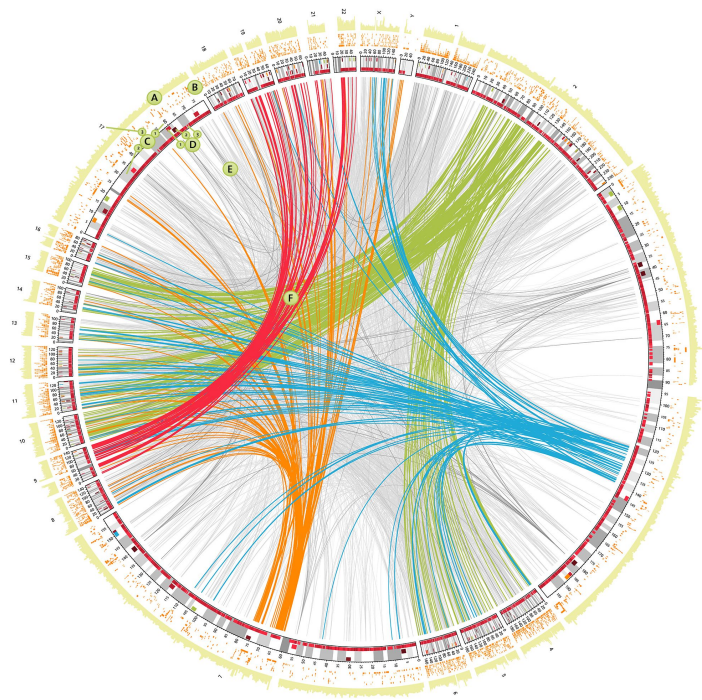
Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
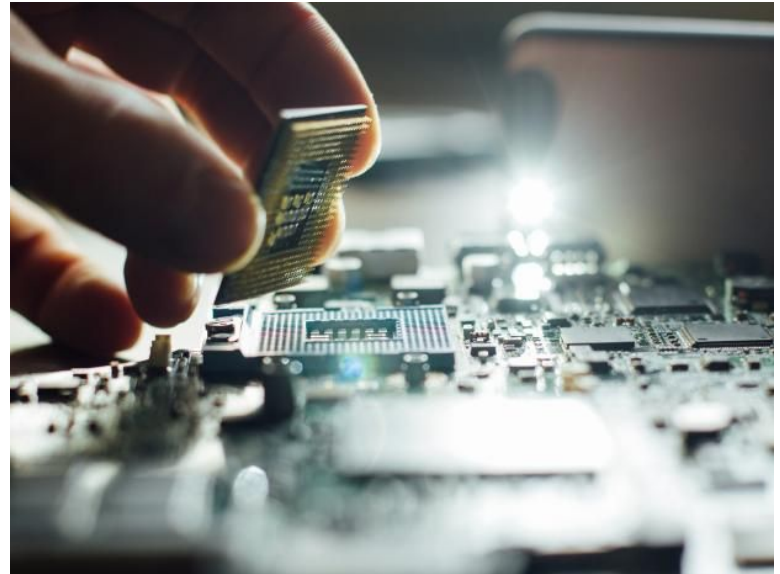
Concentrations in computer science (at Stanford)

- Artificial Intelligence
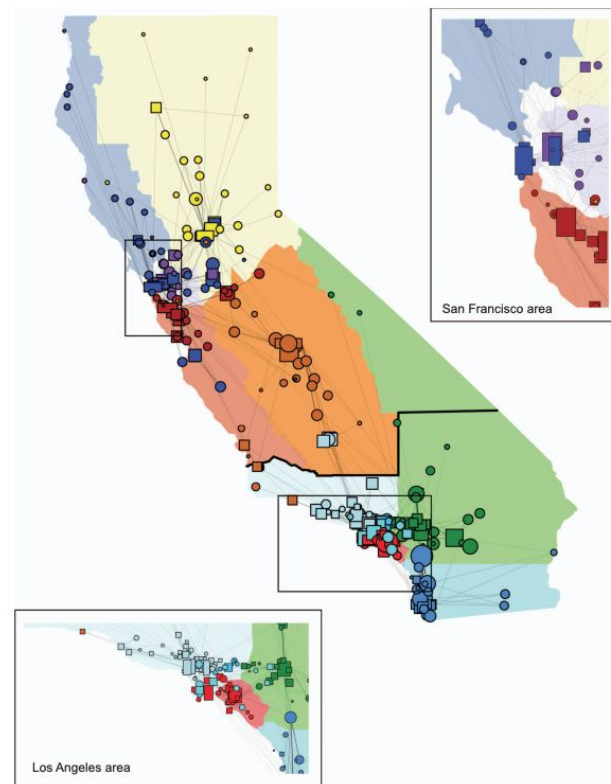- Biocomputation
- Computer Engineering
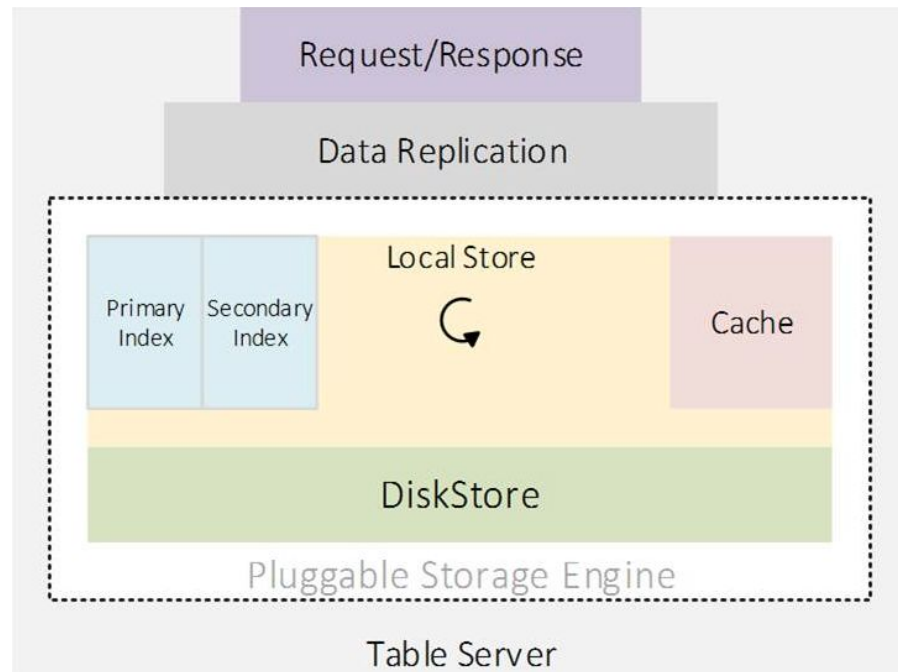- Graphics

Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics
- Human-Computer Interaction

"Element: Heading, level 2.
Content: Project 2.
Text alignment: left. "
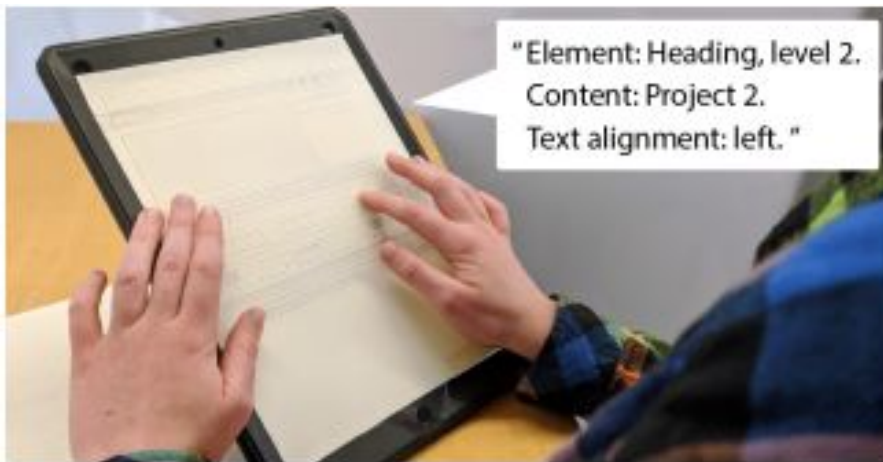
Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics
- Human-Computer Interaction
- Information

Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics
- Human-Computer Interaction
- Information
- Systems
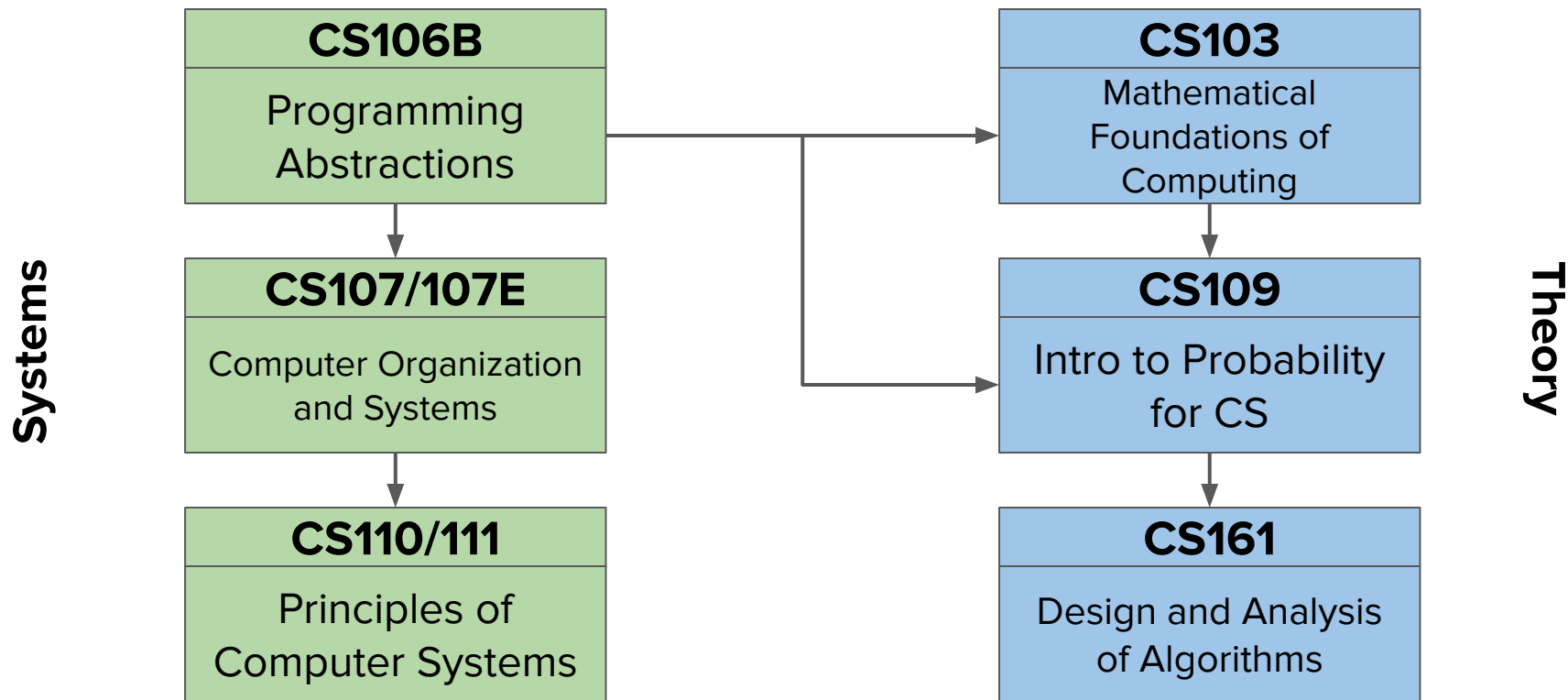
Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics
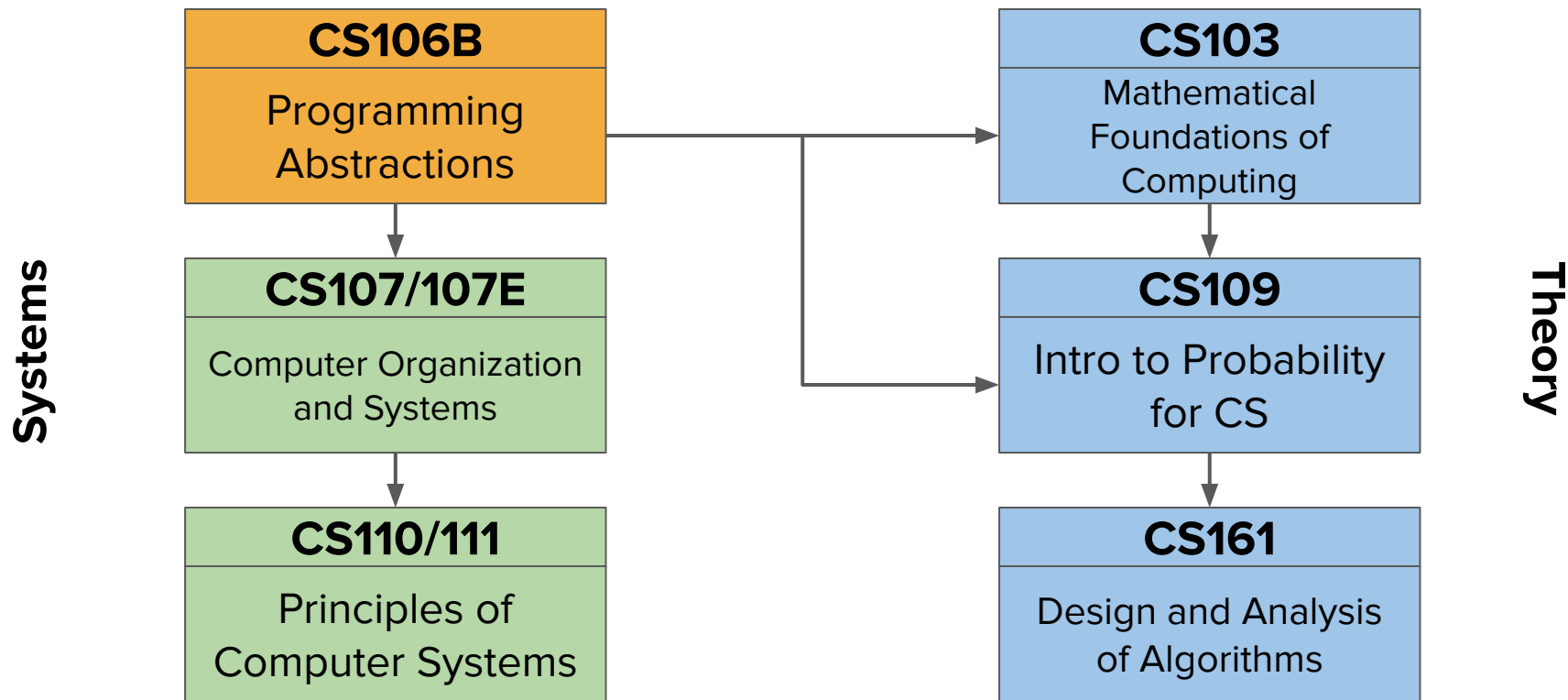- Human-Computer Interaction
- Information
- Systems
- Theory
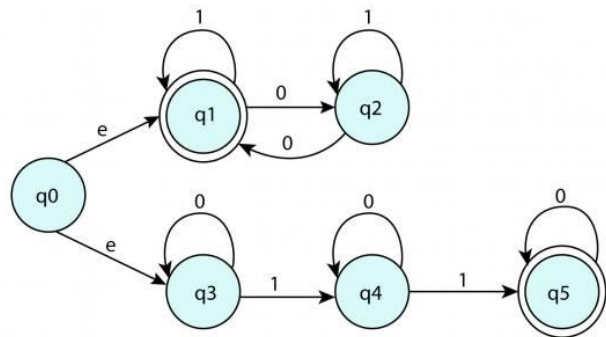
What should I study next?
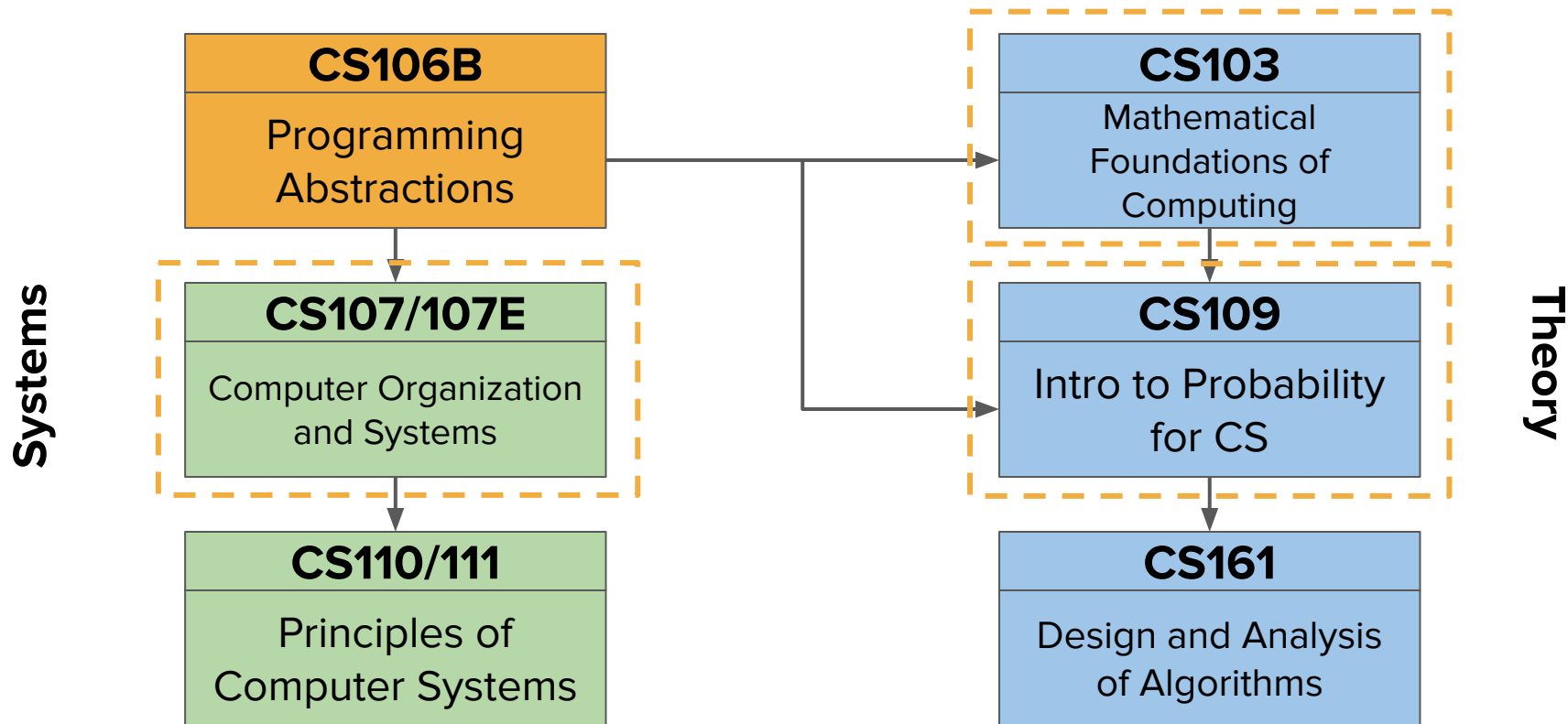
(based on areas you expressed interested in!)
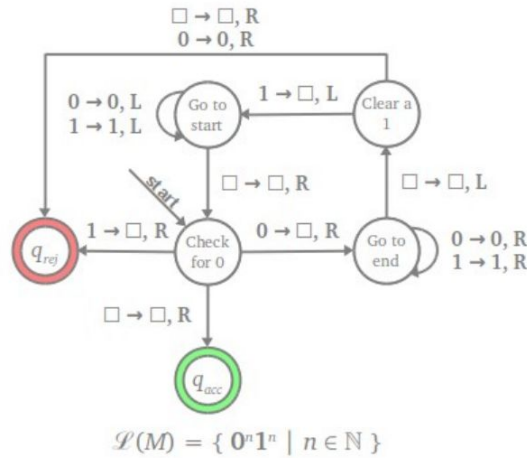
The CS Core (at Stanford)

The CS Core (at Stanford)

$$\mathscr{L}(M) = \{ \, \mathbf{0}^n \mathbf{1}^n \mid n \in \mathbb{N} \, \}$$

What are the fundamental limits of computing power?

How can we be certain about this?

Important Ideas in CS103

- Some infinities are bigger than other infinities, and this has practical consequences.

- Tropes from Ancient Greek mythology can be made mathematically rigorous to prove limits on computing power.

- Abstract models of computation have applications in network drivers, user interfaces, compiler design, and text processing.

CS107: Computer Organization and Systems

What is the internal organization of memory in a computer?

How do we bridge the dichotomy between high-level problem-solving and voltages in wires?

And why is this important to know?

Important Ideas in CS107

- The nature of memory layout explains why computer security is so hard to get right.

- Computers are physical devices whose inner workings are visible even in higher-level languages.

- Compilers can sometimes rewrite recursive functions iteratively, giving you the best of both worlds.

- Bonus: Consider CS107E as well!

CS109: Intro to Probability for Computer Scientists

- Why are hash tables fast? Why are random binary search trees probably good?

- How do we encode data so that if bits get flipped in transit, the message still arrives?

- How do I explore big data sets and make sense of them?

- What is this whole machine learning thing, how does it work, and how do I do it?

Other CS Courses

CS193: Practical Programming Technologies

- Many offerings throughout the year, focused on specific technologies:
  - **CS193A:** Android Programming
  - **CS193C:** Client-Side Web Technologies
  - **CS193P:** iPhone and iPad Programming
  - **CS193Q:** Accelerated Intro to Python
  - **CS193X:** Web Programming Fundamentals

- Great for learning particular technologies

CS106L: Stanford C++ Programming Lab

● Explore what C++ programming looks like outside CS106B.

● Get exposure to the standard libraries and some really, really cool techniques beyond what we saw here.

● Excellent next step if you'd like to work in C++ going forward.

Talk to Haven (who guest lectured yesterday) if you're interested!

CS147: Intro to Human-Computer Interaction

- How do you design software to be usable?

- What are the elements of a good design?

- How do you prototype and test out systems?

- Prerequisite: CS106B!

Areas you asked about!

- Graphics: CS148, CS248
  - Augmented Reality (AR): COMM 280
  - Languages/technologies: OpenGL, C++ (!)

- Artificial Intelligence: CS221, CS229, CS231N, CS224[N|U|W]
  - Technologies: TensorFlow, PyTorch, GCP/AWS/Azure

- Systems (compilers, memory, parallel programming): CS140, CS149, CS143
  - Languages: C, C++, Go, Rust

- Building websites/web programming: CS142, CS193X
  - Languages: HTML, CSS, JavaScript, SQL
  - Technologies: React, NodeJS, Vue, etc. (but none of these are necessary for getting started!)

Learning Beyond Stanford

- Some online resources (mostly in the form of free courses)
    - Codecademy
    - Coursera, edX, Udemy, and other MOOCs
    - Khan Academy
    - MIT Open Courseware
    - Scrimba.com
- Strategies for programming self-improvement
    - Write lots of code!
    - Work on a project you find inspiring
    - Find other people to collaborate with
    - Join an open-source project!

What should my academic path look like?

Thinking about studying more CS?

- Good reasons to think about doing CS:
    - I like the courses and what I'm doing in them.
    - I like the people I'm working with.
    - I like the impact of what I'm doing.
    - I like the community.

- Bad reasons to think about not doing CS:
    - I'm good at this, but other people are even better.
    - The material is fun, but there's nothing philosophically deep about it.
    - I heard you have to pick a track, and I don't know what I want to do yet.
    - What if 20 years later I'm just working in a cubicle all day and it's not fun and I have an Existential Crisis?

The CS Major (at Stanford)

- A common timetable:
    - Aim to complete *most* of the core by the end of your sophomore year (~4/6 classes).
    - Explore different tracks in your junior year and see which one you like the most.
    - Spend your senior year completing it.

- It's okay if you start "late"!
    - The latest time you can *comfortably* start a CS major would be to take CS106A in winter quarter of sophomore year.
    - And the coterm is always an option!

Ask us anything!

Pre-submitted Questions

How do the skills we've learned translate to how CS is used in the technology industry?

How do the skills we've learned translate to how CS is used in the technology industry?

Kevin Naughton · 2nd
Software Engineer at Google
9h · 🌐

+ Follow

my typical day at Google:
9am – reverse a linked list
11am – count unique ways to climb a staircase with dp
12pm – lunch
3pm – help animal escape NxM matrix efficiently
4pm – invert a binary tree
5pm – commute home using Dijkstra's

(Just kidding)

#coding #programming #softwareengineering

ifunny.co

How do the skills we've learned translate to how CS is used in the technology industry?

There are *many different* jobs in CS.

- Product managers: Understanding how to scale technology (Big O matters!)

- UI/UX Designers: Understanding how a user might interact with your system/software as an abstraction

- Technical writers: Being able to communicate about technology and software to a particular audience (your final projects!)

- Data scientists: Recognizing patterns in data and taking advantage of those patterns to efficiently process and store the info

How do I prepare for technical/job interviews?

How do I prepare for technical/job interviews?

- Technical interviews are arbitrary tests in that a lot of the interview questions are not related to the day-to-day skills you'll need!

- CS106B covers most of the topics that will be on programming interviews, but you'll still need to prepare for the specific format, types of question, etc.

- Practice, practice, practice!

- Many resources:
  - http://web.stanford.edu/class/cs9/ (course)
  - Cassidoo's blog, CodePath Guide, Tutorials Point, Cracking the Coding Interview (book)

whats your favorite sandwich

Why did you choose to study CS?

Why did you choose to study CS?

(if applicable) How did you start combining it with other subjects such as education and art?

What tips/advice would you give for wanting to do CS but mixing it with other topics and finding the right intersection?

What did you struggle with the most in CS106B?

What did you struggle with the most in CS106B?

- Kylie: Big O and recursion
  - I remember taking one 106B exam that asked for the Big O for several of the coding problems, and I felt like I was guessing on almost every one.

- Jenny: Linked Lists
  - My partner and I were up until the wee hours of the morning drawing so many pictures of linked lists, over and over again…

do you g uys have games on ur phone

What's the coolest CS project you've made and why do you like it?

domino
## smartphones as accessible microcontrollers

Needs

- Only 5% of low-income families globally have access to computers
- This denies low-income students equal opportunity for computational literacy
- However, 60% of low-income families have smartphones at home

Goals

domino allows students to...

- Use their phones as microcontrollers
- Deepen understanding of technology
- Democratize expression and creativity
- Practice computational thinking
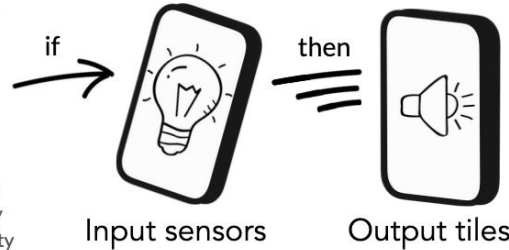
Features

- No external hardware required
- Graphical-based programming
- Quickstart for beginners without coding experience

Future Work

- Extend program functionality to a Turing Complete language: design for inTile and outTile chaining and variables
- Assess learners' understandings of their phones
- Encourage creativity by fostering a community of *domino* users

if → Input sensors then Output tiles

Key screens

Play with and understand a variety of input sensors

Sensor Playground
Choose a sensor to play with

Light | Temperature | Magnetometer
GPS | Barometer | Microphone

Discover | Play | Create | My Dominoes

Create custom dominoes

My Workspace

→ inTile →

Sensors

Discover | Play | Create | My Dominoes

Discover and remix other users' dominoes

< Treasure Detector
Created by [subUser17]

If Magnetometer is above 200 μT, then Speakers beep.

Copy to My Dominoes

Keep engaged by accepting challenges

< Challenges

Brightness Flux
Security Alarm

Lights Out
Where's the Sensor?
Play

Light Sensor

Paulo Blikstein, Aashna Shroff, Jenny Han, Kylie Jue | aashna@stanford.edu

After learning Python (CS106A) and C++(CS106B), do you have any recommendations on what coding language might be the best to learn next?

What were your favorite Stanford (CS) courses?

Will you be teaching CS classes at Stanford for the upcoming Fall/Winter/Spring quarters? (What are you doing next?)

Open Q&A

What questions do you have for us?

What's next?

Where are you now?

- You now have a wide array of tools you can use to solve a variety of problems.

Where are you now?

- You now have a wide array of tools you can use to solve a variety of problems.

- You have the skills to compare and contrast those solutions.

Where are you now?

- You now have a wide array of tools you can use to solve a variety of problems.

- You have the skills to compare and contrast those solutions.

- You have expressive mental models for teasing apart those problems. (abstractions!)

Closing thoughts

- How is the technology we use made and who makes it?

Closing thoughts

- How is the technology we use made and who makes it?

- Who benefits from the technology?

Closing thoughts

- How is the technology we use made and who makes it?

- Who benefits from the technology?

- Who might not benefit from the technology?

Closing thoughts

- How is the technology we use made and who makes it?

- Who benefits from the technology?

- Who might not benefit from the technology?

- What problems will **you** choose to solve with technology?

Roadmap

C++ basics

Object-Oriented
Programming

User/client

Implementation

vectors + grids

arrays

stacks + queues

**dynamic memory
management**

sets + maps

linked data structures

Diagnostic

**real-world
algorithms**

Core
Tools

testing

algorithmic
analysis

**recursive
problem-solving**

Life after CS106B!

Thank you!!!