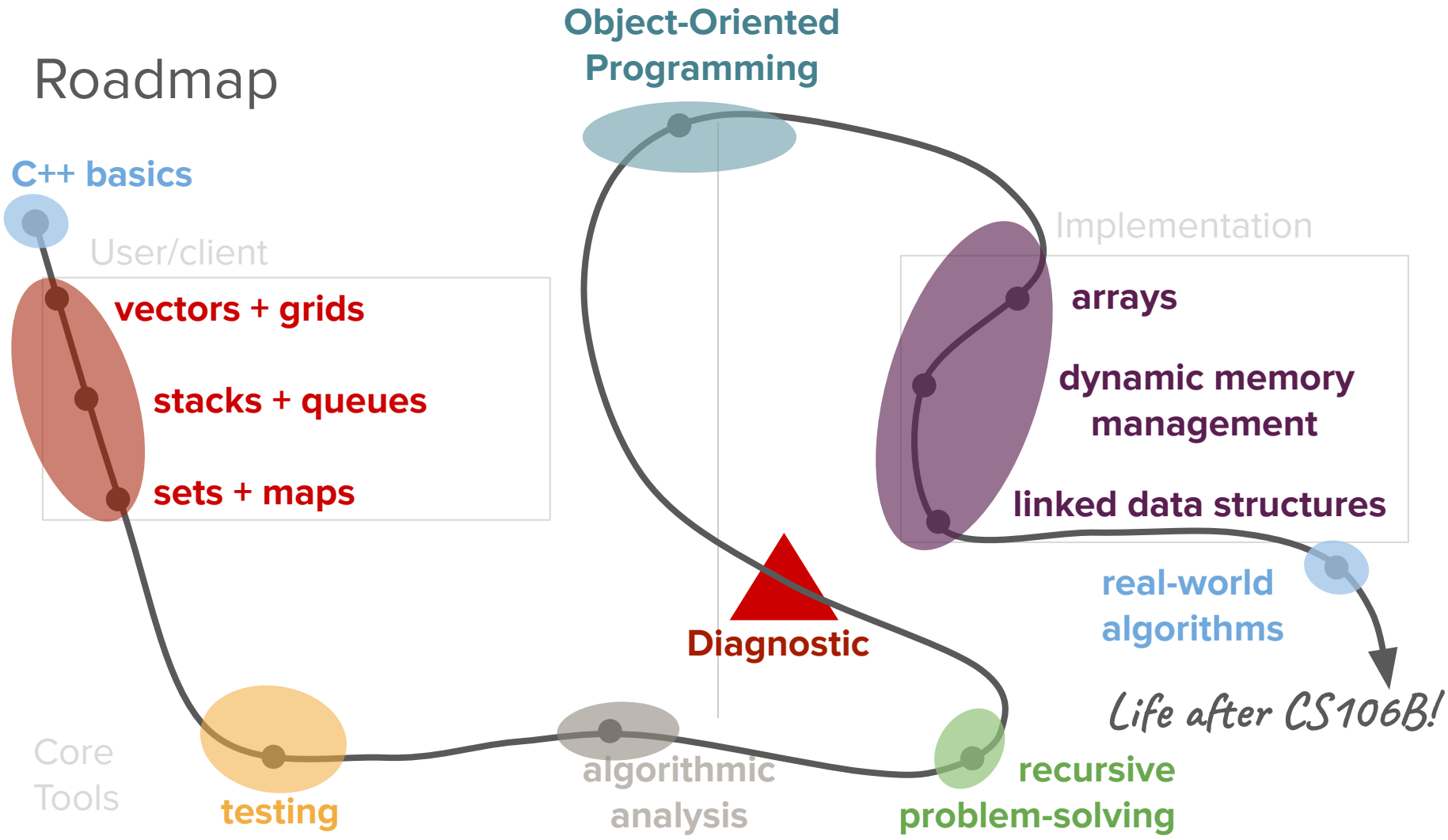
Life after CS106B

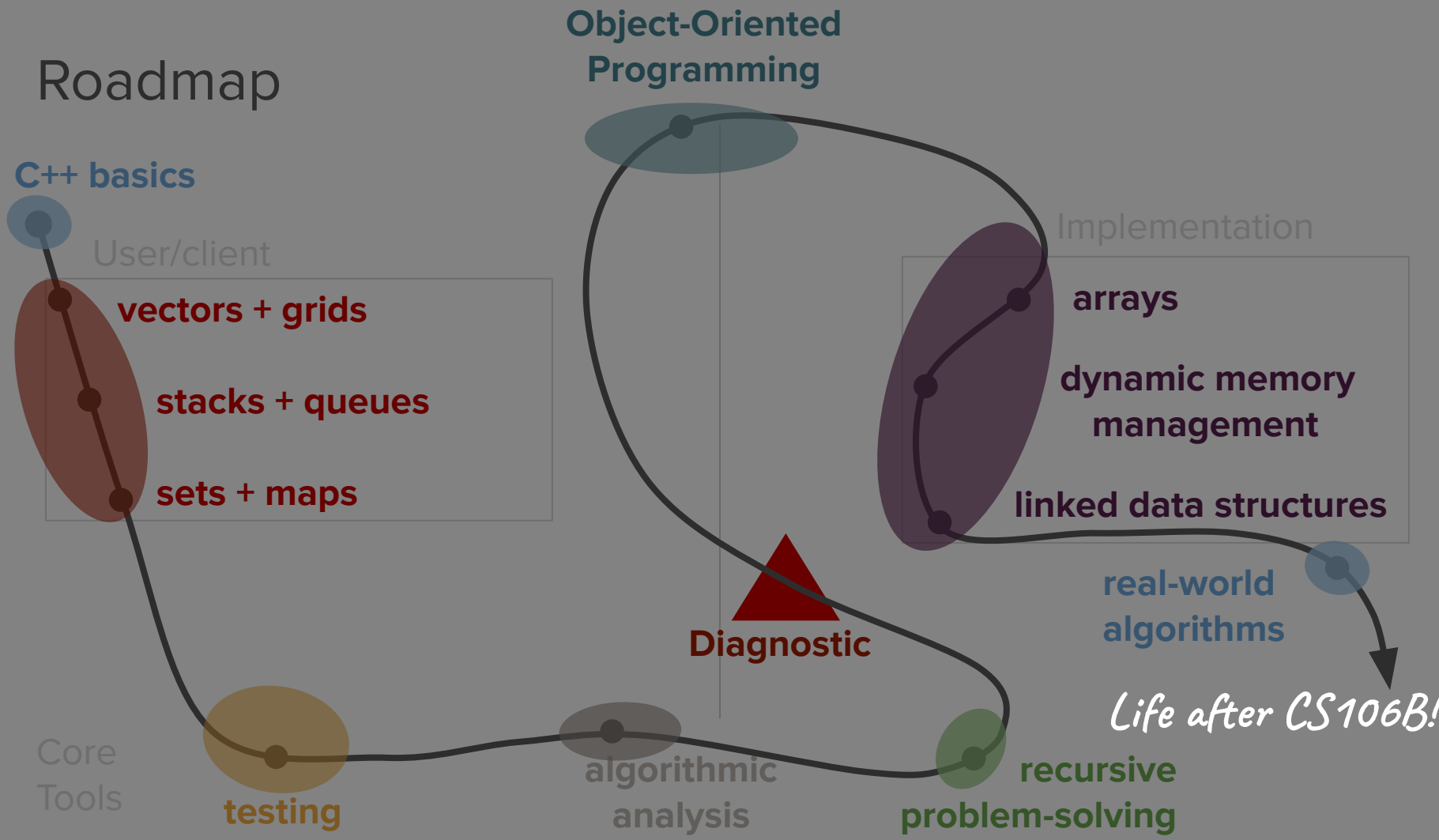
What are you up to next in life?

(put your answers the chat)

Roadmap


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Life after CS106B!
Diagnostic
Core Tools
testing
algorithmic analysis
recursive problem-solving

Roadmap


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Diagnostic
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving

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!

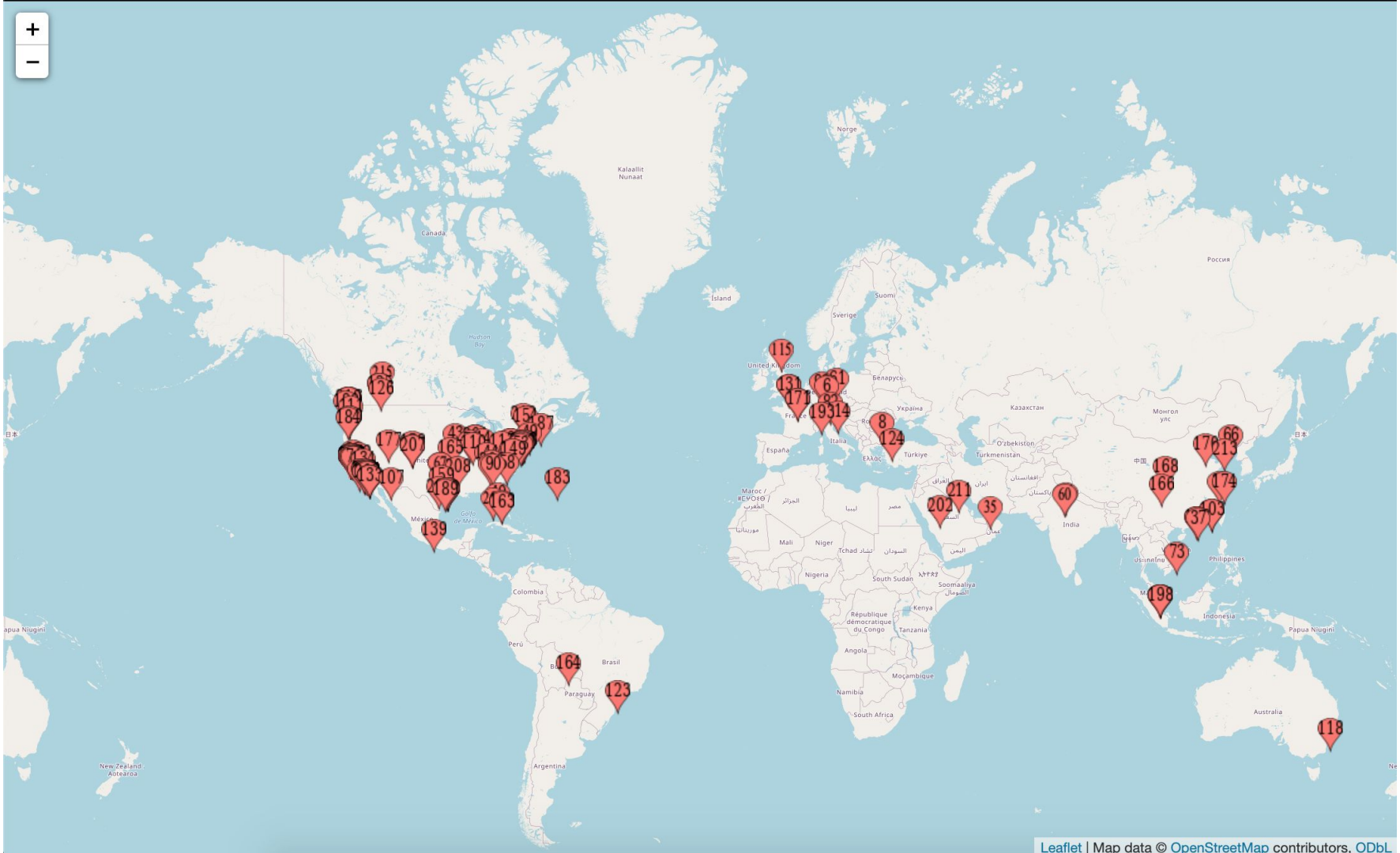
Course Recap

[everything!]

Who were you?

Who were you?

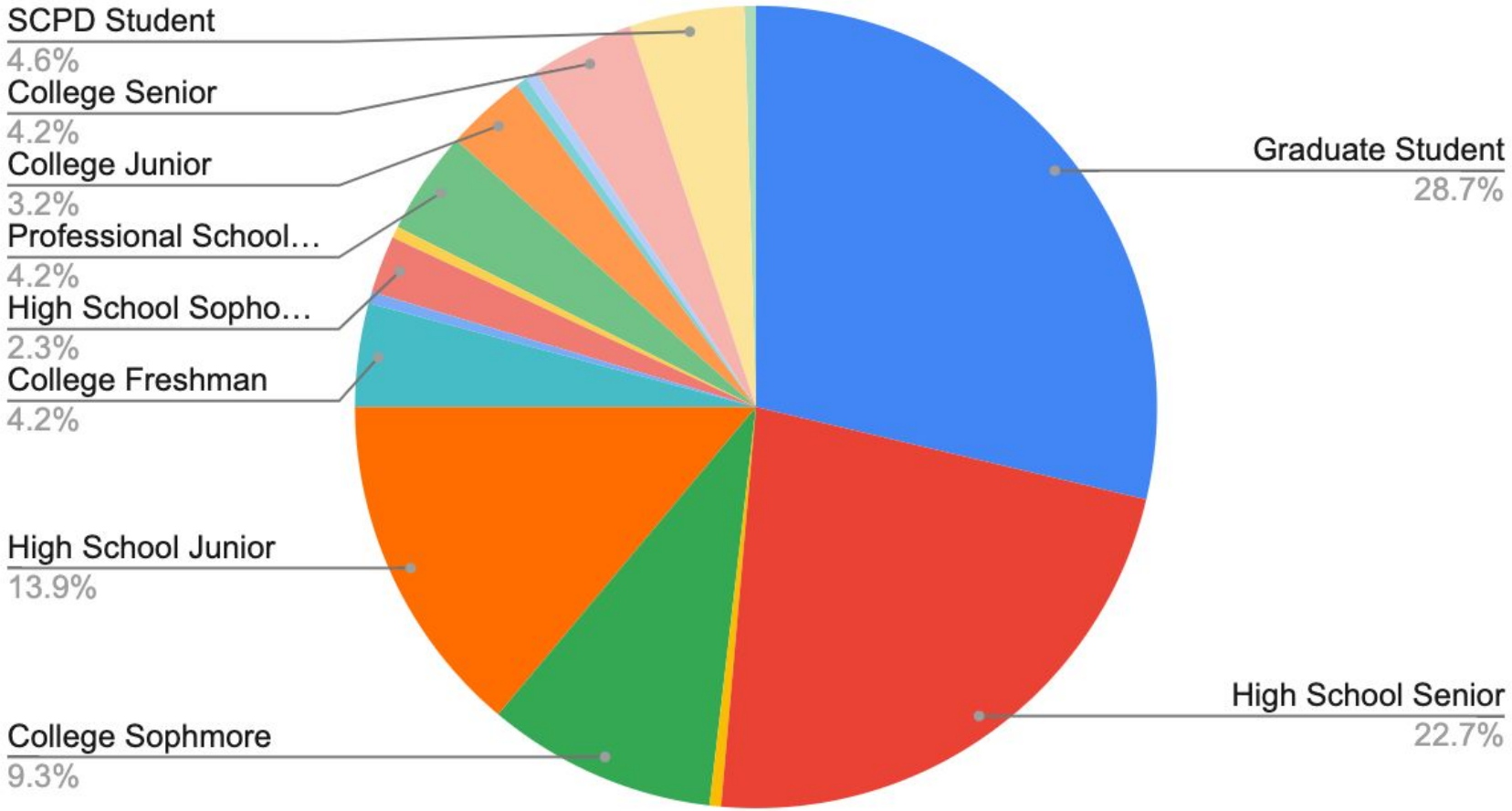
- You spanned 16 different time zones!


Leaflet | Map data © OpenStreetMap contributors, ODbL

Who were you?

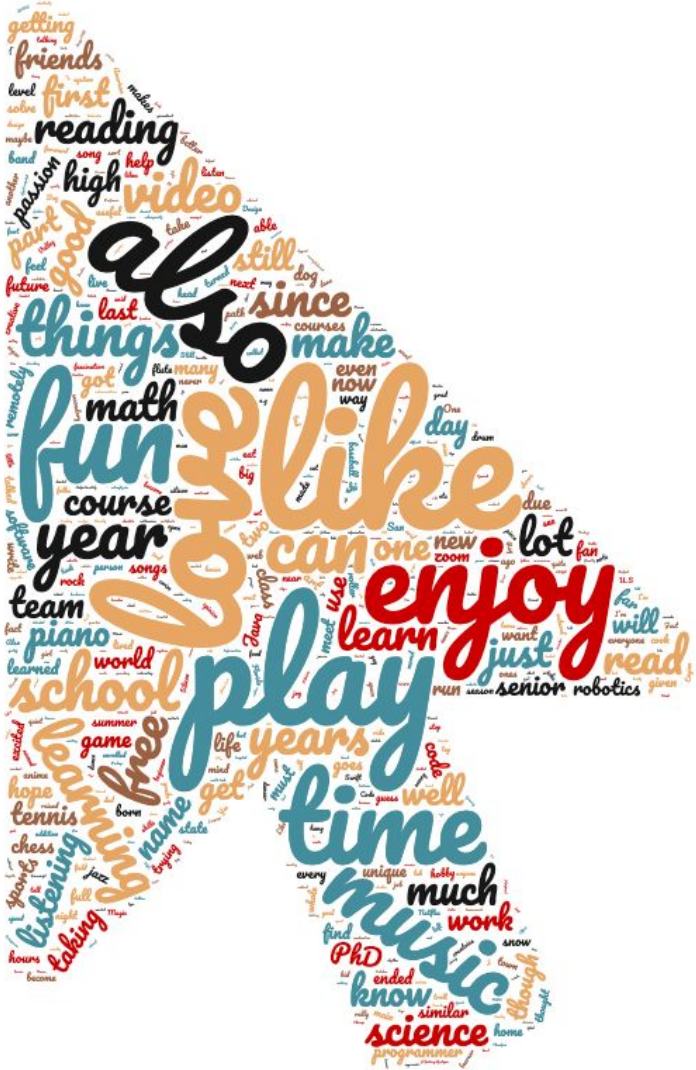
- You spanned 16 different time zones!
- You ranged from high school students to career professionals!

Where are you at in life?



Who were you?

- You spanned 16 different time zones!
- You ranged from high school students to career professionals!
- You had really interesting interests and backgrounds!


also
like
love
fun
time
play
enjoy
music
things
year
school
reading
friends
video
since
make
math
course
can
learn
lot
team
piano
world
free
years
learning
science
know
much
work
PhD
tennis
listening
taking
chess
name
get
well
just
read
will
senior
robotics
day
one
new
use
code
game
hope
first
high
good
part
still
last
even
now
way
got
many
songs
rock
Java
life
summer
learned
excited
run
unique
every
find
ended
similar
though
programmer
home
hours
become
Code
guess
goes
must
mind
born
state
buying
jazz
full
Maybe
Flute
Piano
Guitar
Drums
Violin
Soccer
Basketball
Football
Baseball
Swimming
Dancing
Singing
Painting
Drawing
Cooking
Gaming
Coding
Hiking
Camping
Fishing
Biking
Running
Walking
Sleeping
Eating
Drinking
Watching
Movies
Books
Art
Science
Math
History
Geography
Physics
Chemistry
Biology
Computer
Engineering
Medicine
Law
Business
Economics
Psychology
Sociology
Philosophy
Literature
Languages
Spanish
French
German
Chinese
Japanese
Korean
Arabic
Russian
Italian
Portuguese
Hindi
Bengali
Urdu
Swahili
Dutch
Greek
Turkish
Polish
Swedish
Norwegian
Danish
Finnish
Hungarian
Czech
Slovak
Romanian
Bulgarian
Croatian
Serbian
Slovenian
Estonian
Latvian
Lithuanian
Icelandic
Irish
Welsh
Scottish
Gaelic
Basque
Catalan
Galician
Maltese
Albanian
Armenian
Georgian
Azerbaijani
Kazakh
Uzbek
Turkmen
Kyrgyz
Tajik
Mongolian
Tibetan
Nepali
Sinhala
Tamil
Telugu
Kannada
Malayalam
Marathi
Gujarati
Punjabi
Odia
Assamese
Burmese
Khmer
Lao
Thai
Vietnamese
Indonesian
Malay
Filipino
Tagalog
Cebuano
Ilocano
Hiligaynon
Waray
Pampanga
Bikol
Pangasinan
Maranao
Maguindanao
Tausug
Yakan
Sama
Badjao
Kalagan
Kalibugan
Tiruray
Teduray
Lambangian
Ata Manobo
Dibabawon
Mansaka
Mandaya
Kalinga
Bontoc
Ifugao
Kankanaey
Isneg
Tingguian
Ibaloi
Kalanguya
Karao
Applai
Itneg
Gaddang
Ilocano
Pangasinan
Kapampangan
Sambal
Ayta
Abelling
Agta
Dumagat
Remontado
Casiguran
Dumagat
Kabuloan
Dumagat
Kabuloan
Dumagat
Kabuloan

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
- Abstractions!

What you’ve made this quarter

Assignment 1: C++ legs

NATIONAL ARCHIVES

Blogs · Bookmark/Share · Contact Us

Search Archives.gov Search

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

Research Our Records

Home > Research Our Records > Census Records > Soundex System

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

Soundex System

The Soundex Indexing System

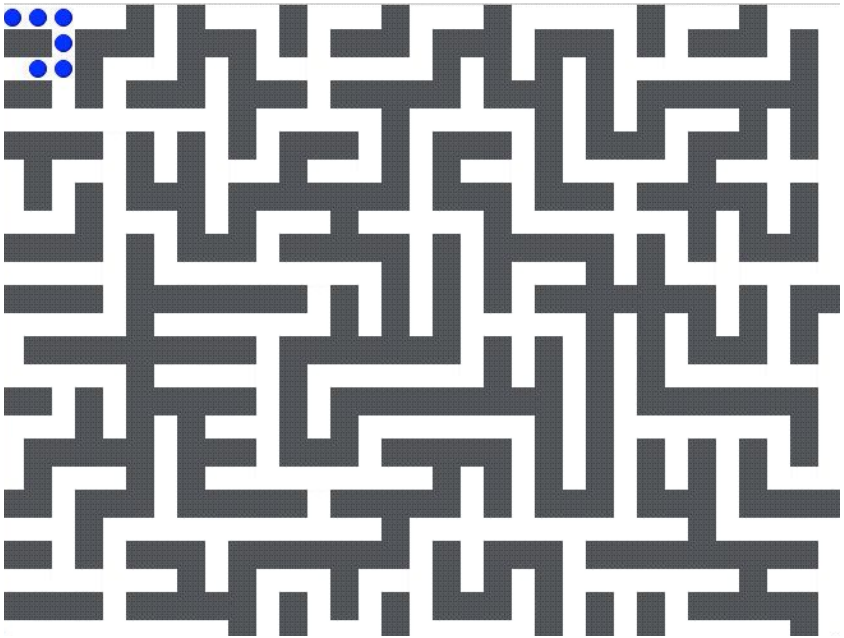
Updated May 30, 2007

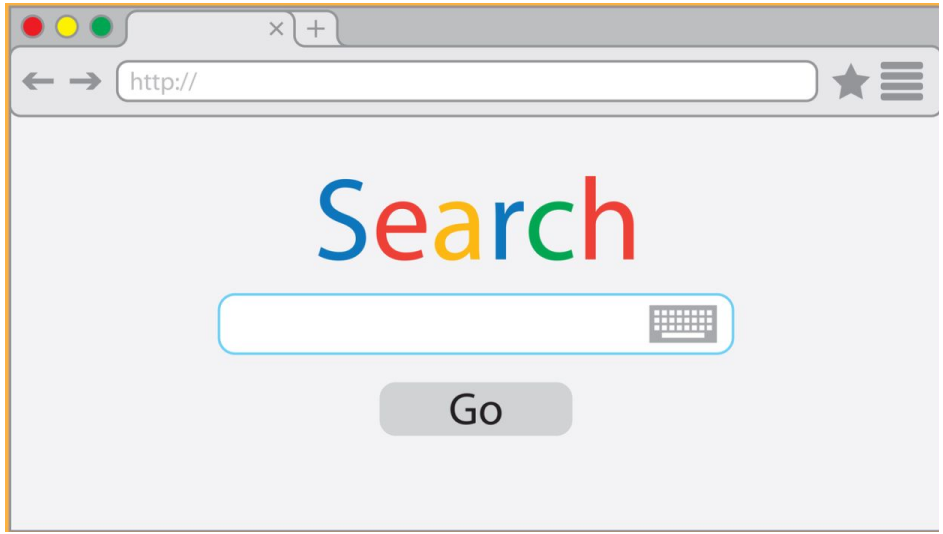
To use the census soundex to locate information about a person, you must know his or her full name and the state or territory in which he or she lived at the time of the census. It is also helpful to know the full name of the *head of the household* in which the person lived because census takers recorded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

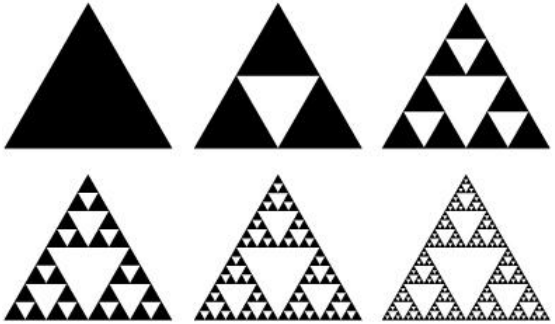
Assignment 2: Fun with Collections



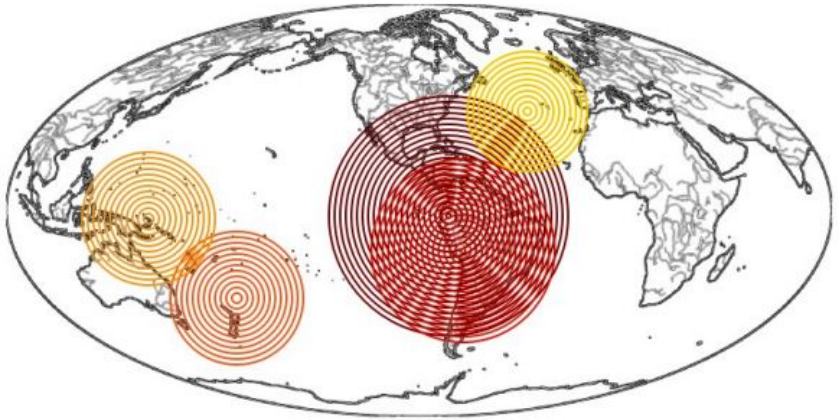


Assignment 3: Recursive Problem-Solving

{ }



Assignment 4: Priority Queue

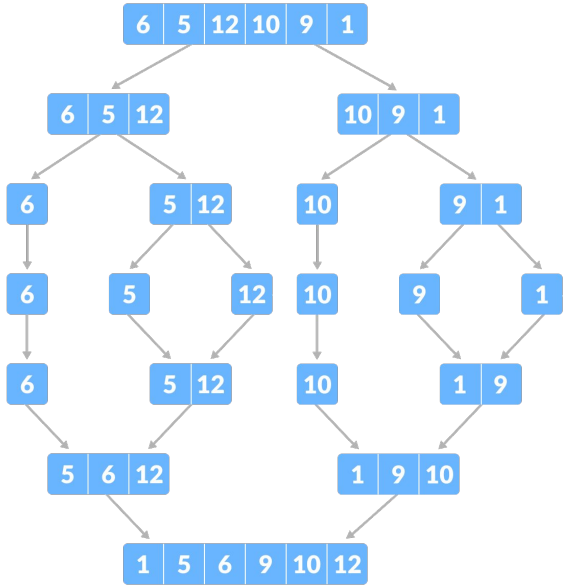


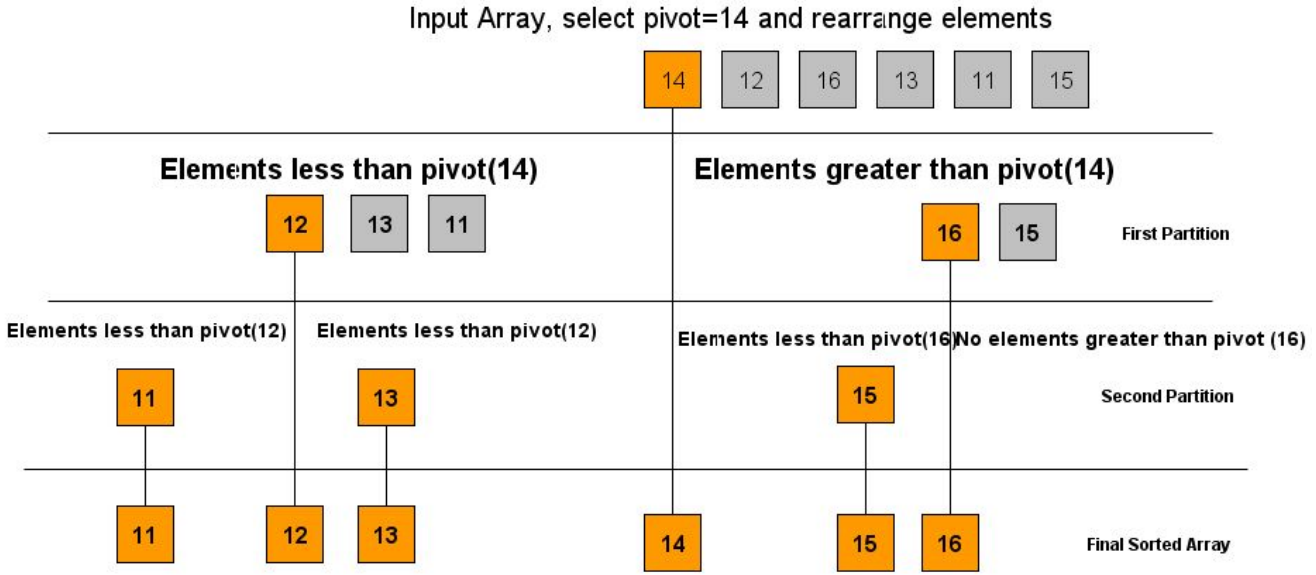
This tool displays the strongest recent earthquakes reported by the US Geological Survey
You can use the controls on the side of the window to select the time interval you're interested in. This visualizer will show the 5 strongest earthquakes within that interval.

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.

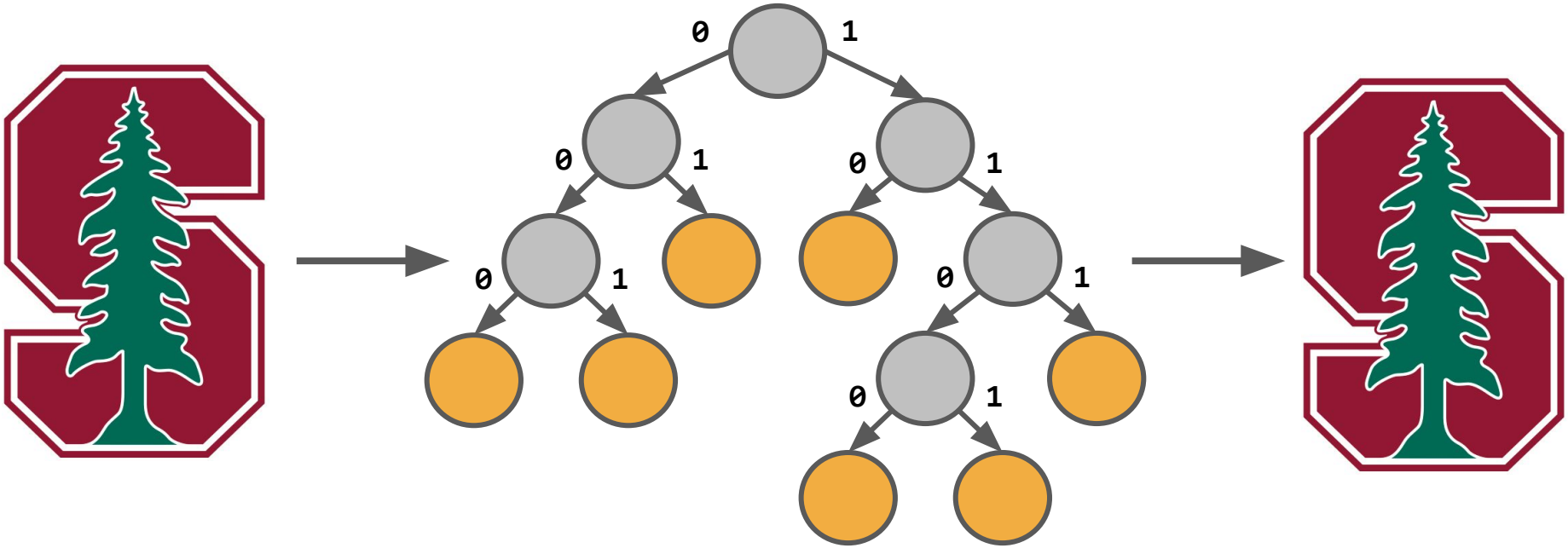
- Magnitude 7.5 115km ESE of Palora, Ecuador at 02:17:22 AM on Feb 22, 2019
- Magnitude 7 27km NNE of Azangaro, Peru at 12:50:41 AM on Mar 01, 2019
- Magnitude 6.4 116km SE of L'Esperance Rock, New Zealand at 07:46:14 AM on Mar 06, 2019
- Magnitude 6.4 49km NW of Namatanai, Papua New Guinea at 06:35:55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





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?

What does life look like after CS106B?

Computer science is more than just programming!

Computer science is more than just programming!

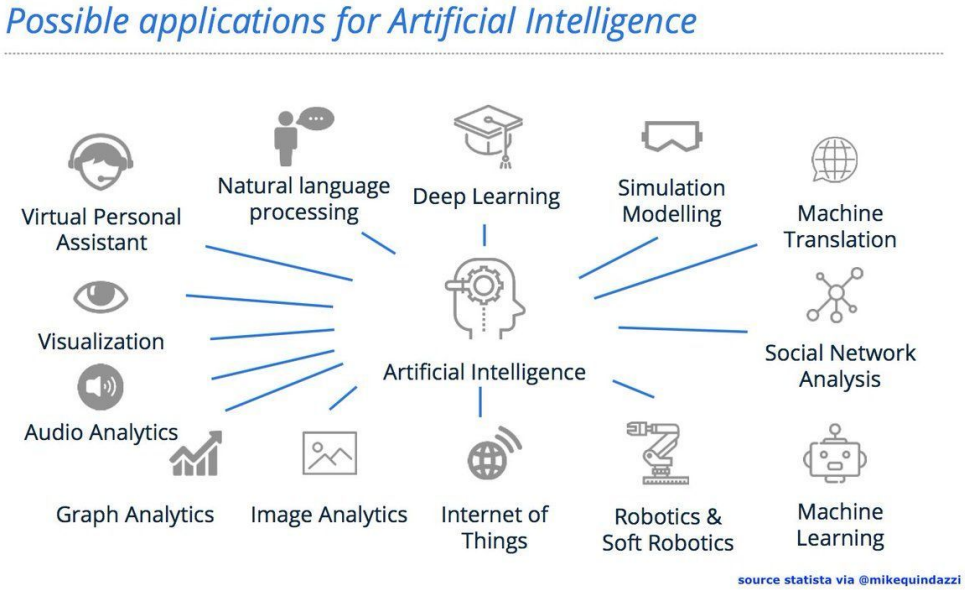
These skills will make you better at whatever you choose to do in life!

What can I study within computer science?

(concentrations within CS)

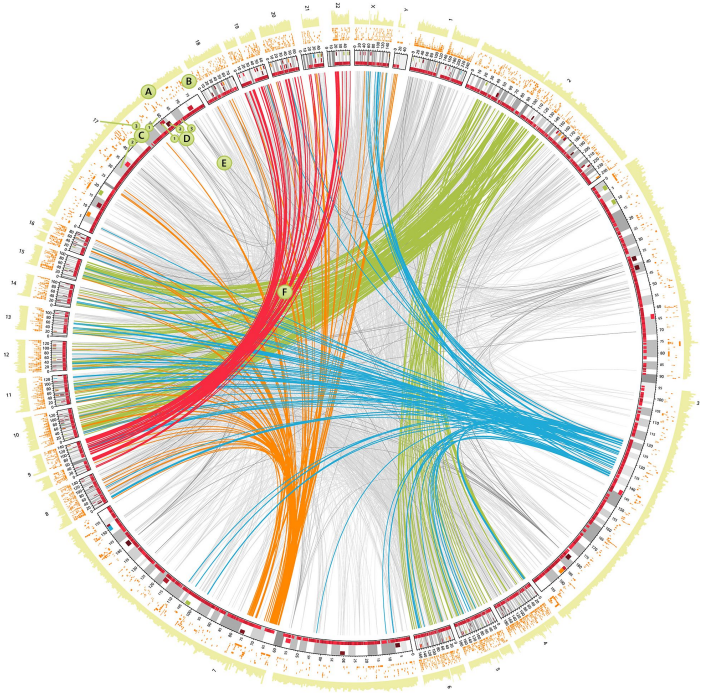
Concentrations in computer science (at Stanford)

- Artificial Intelligence



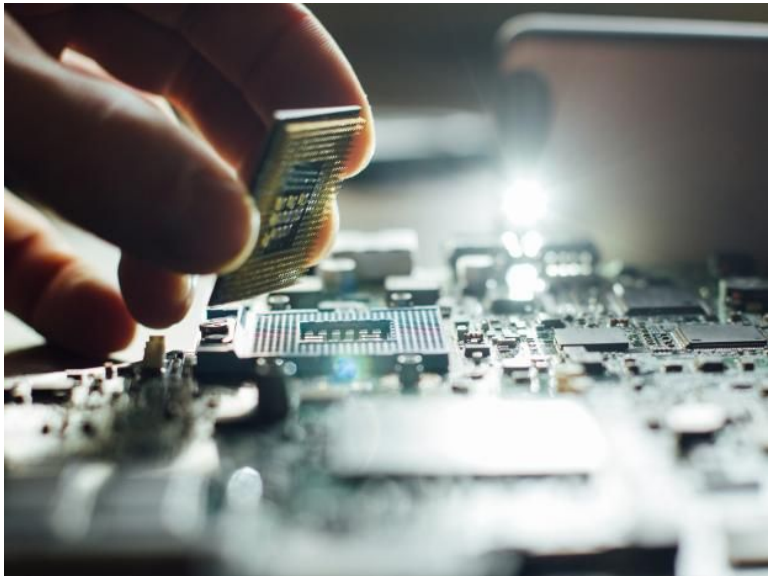
Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation



Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering

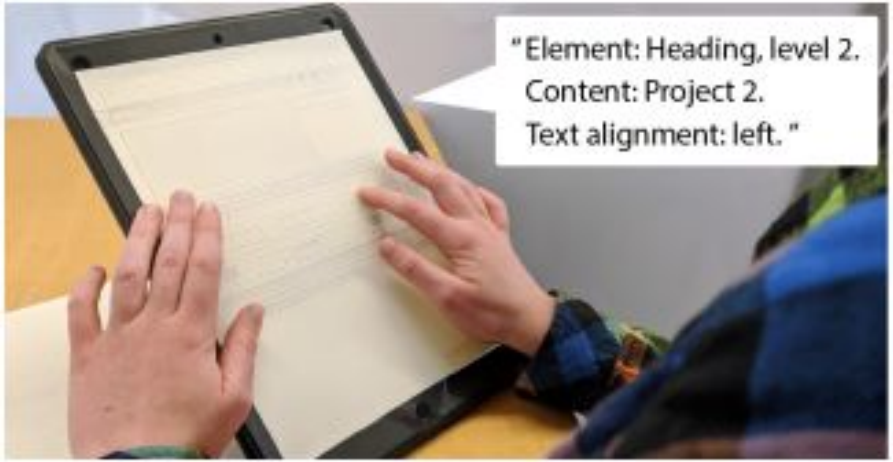


Concentrations in computer science (at Stanford)

- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics

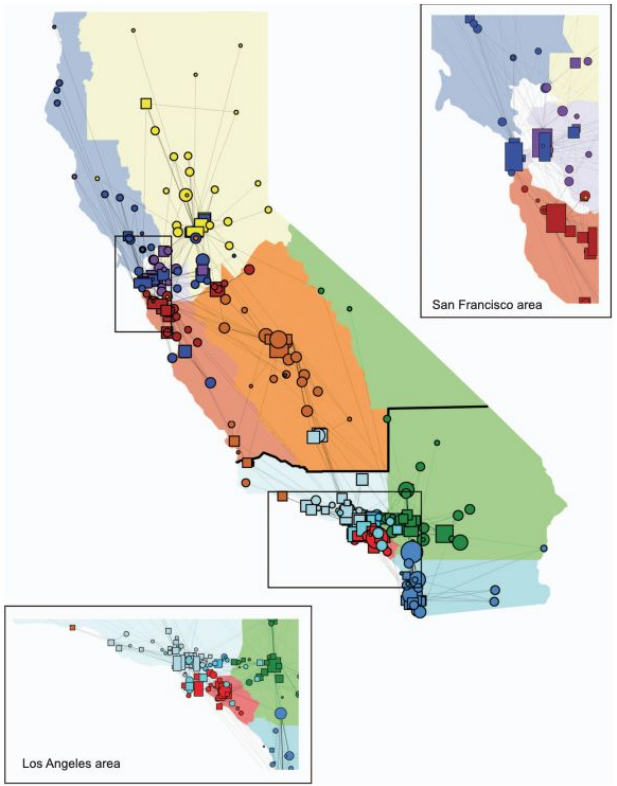
Concentrations in computer science (at Stanford)

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



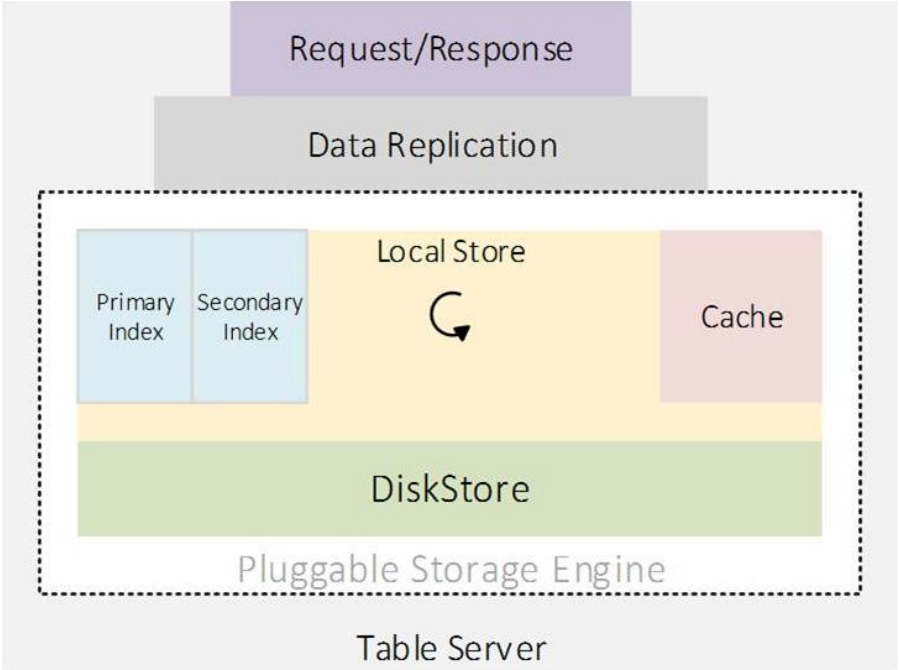
Concentrations in computer science (at Stanford)

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



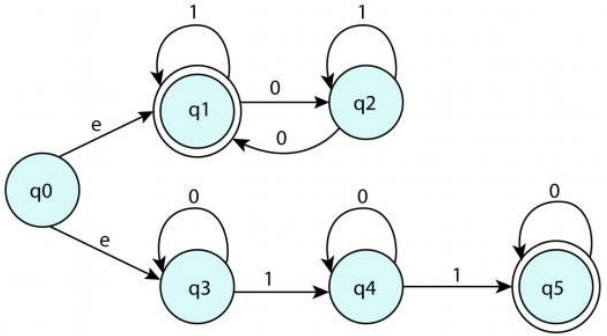
Concentrations in computer science (at Stanford)

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



Concentrations in computer science (at Stanford)

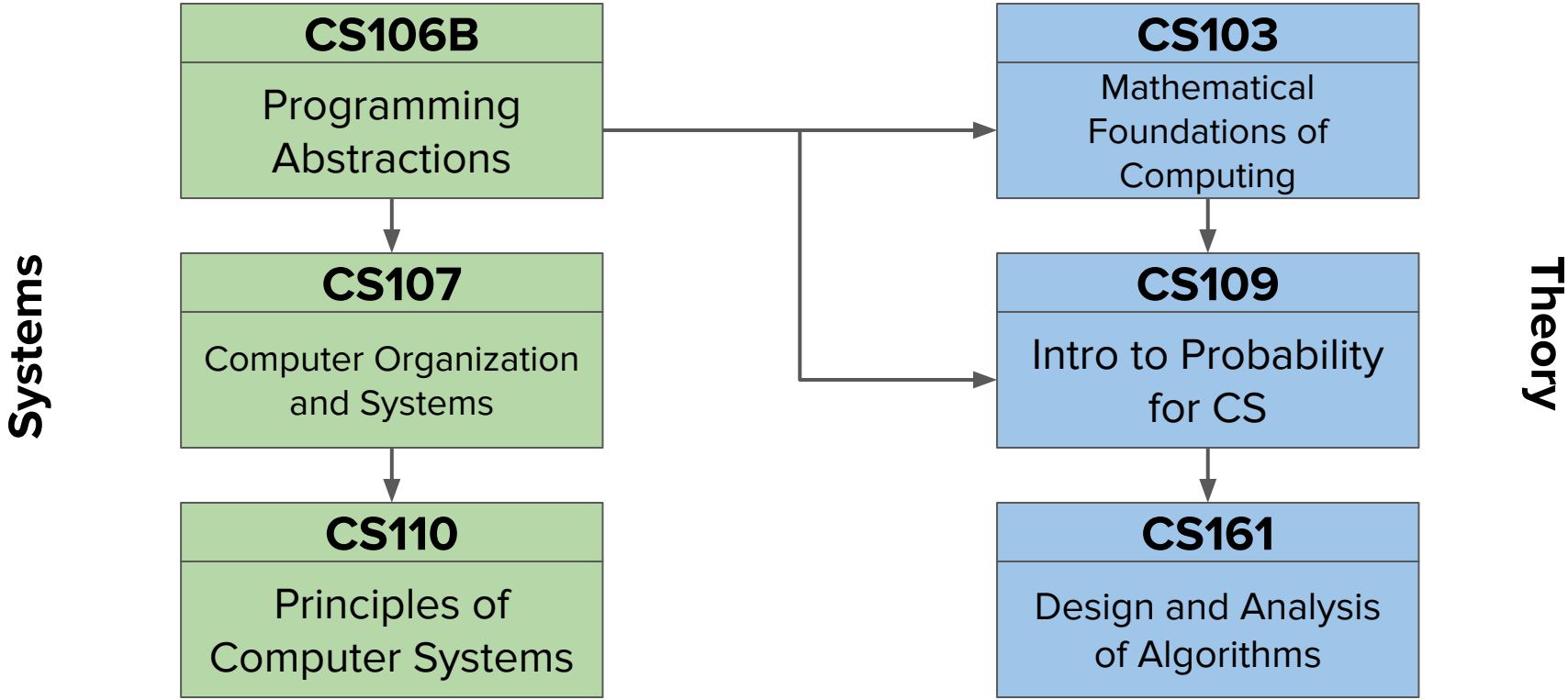
- Artificial Intelligence
- Biocomputation
- Computer Engineering
- Graphics
- Human-Computer Interaction
- Information
- Systems
- Theory



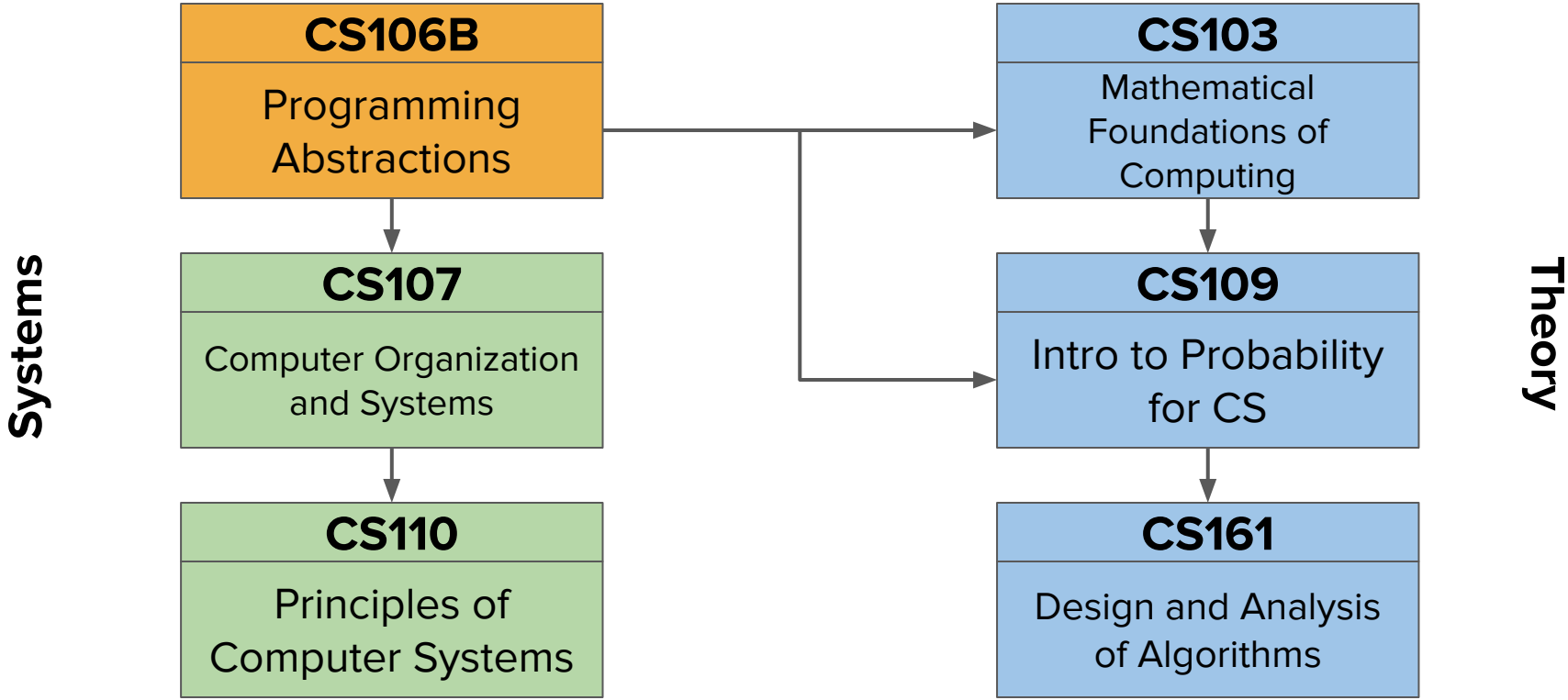
What should I study next?

(based on areas you expressed interested in!)

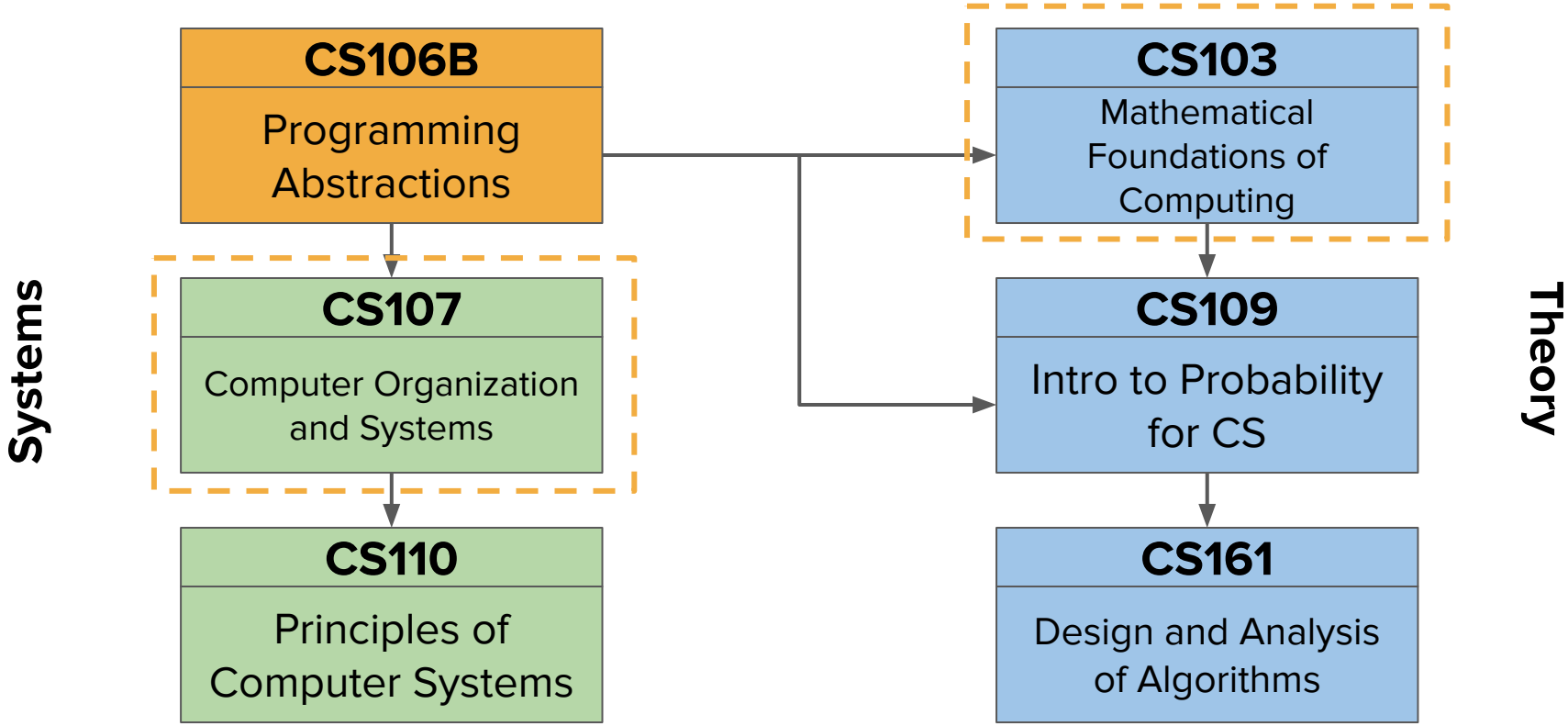
The CS Core (at Stanford)

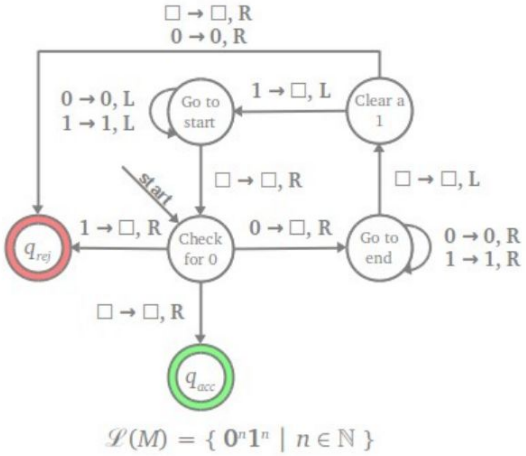


The CS Core (at Stanford)



The CS Core (at Stanford)





CS103

Mathematical Foundations of Computing

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.

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
  - **CS193I:** iOS Programming
  - **CS193P:** iPhone and iPad Programming
  - **CS193Q:** Accelerated Intro to Python

- Great for learning particular technologies

CS106L: Stanford C++ Programming Lab

- Explore what C++ programming looks like outside of 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.

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

- 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

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.

- Nick: 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...

- Trip: Recursion and pointer-related things
  - Even now there are recursive problems I look at and say “nope, never would have figured that one out”

What's your favorite thing about CS and programming?

What's your favorite thing about CS and programming?

- Kylie: The computational thinking skills I've gained to better understand the benefits, limitations, and possibilities of the technology around me.

- Nick: Every 3 months I change my mind about what is interesting and what I want to do with my life, and yet computer science remains the consistent thread that lets me pursue so many different ways to have impact. Also I LOVE coding. The process of writing code is truly so much fun.

- Trip: I love that computer science is a platform for problem-solving. It's incredibly dynamic and applicable, so I feel like there are always interesting things to do.

What made you interested in CS?

What made you interested in CS?

- Kylie: My after-school high school robotics team - I was very fortunate to have a female mentor who encouraged me to go into engineering and has continued to give me advice throughout college.

- Nick: I had an incredibly supportive teacher in high school – he would put so much of his personal time after school and on weekends to encourage me and my classmates to pursue our interest in CS and show us all the cool things that we could accomplish. The powerful sense of original creation that you get when you create a new program or project is also really awesome.

- Trip: I'm not really sure -- I failed the APCS exam in high school, so coming into college I was set on NOT doing CS. I took 106A and loved it and the idea of section leading so much that I never looked back. At times I definitely felt like I wasn't cut for the major, but I slowly convinced myself that I could do it!

What kind of person should pursue a career in CS?

What kind of person should pursue a career in CS?

- Someone who is interested in designing, making, and/or understanding technologies for the benefit of others.
- Someone who enjoys theoretical and/or practical problem-solving using computation.
- A “career in CS” is so much broader than people often think it is!
- *Anyone can pursue a career in CS* – you don’t need to be a particular “type” of person (and the idea that some people are just born “CS-people” is a myth!).

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?

Like we said, 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 can I translate my CS106B skills into Python?

How can I translate my CS106B skills into Python?

- While some of the ADTs we learned in C++ also exist in Python (dicts as maps, lists as vectors, sets as sets), many would require you to implement them yourself (queues, stacks, etc.)!

- Python doesn't enable the low-level memory management and system access that C++ does – **there are no pointers**, and you don't get the choice between allocating memory on the stack vs. the heap.
    - As a result, creating linked data structures like trees or linked lists would require creating an entirely new classes for the Nodes class and the linked data structure itself (turning it into more of an ADT in Python).

- But algorithms like searching/sorting and recursive problem-solving still translate!

Which Stanford courses did you enjoy a lot? Are there any that changed your lives?

Which Stanford courses did you enjoy a lot? Are there any that changed your lives?

- Kylie: Educ 236, Educ 208B, English 12A, CS 206, CS 124, CS247
- Nick: History 50B, Africaam 47, CS398
- Trip: CS182W, CS110, History 155F, Music 24 series

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?

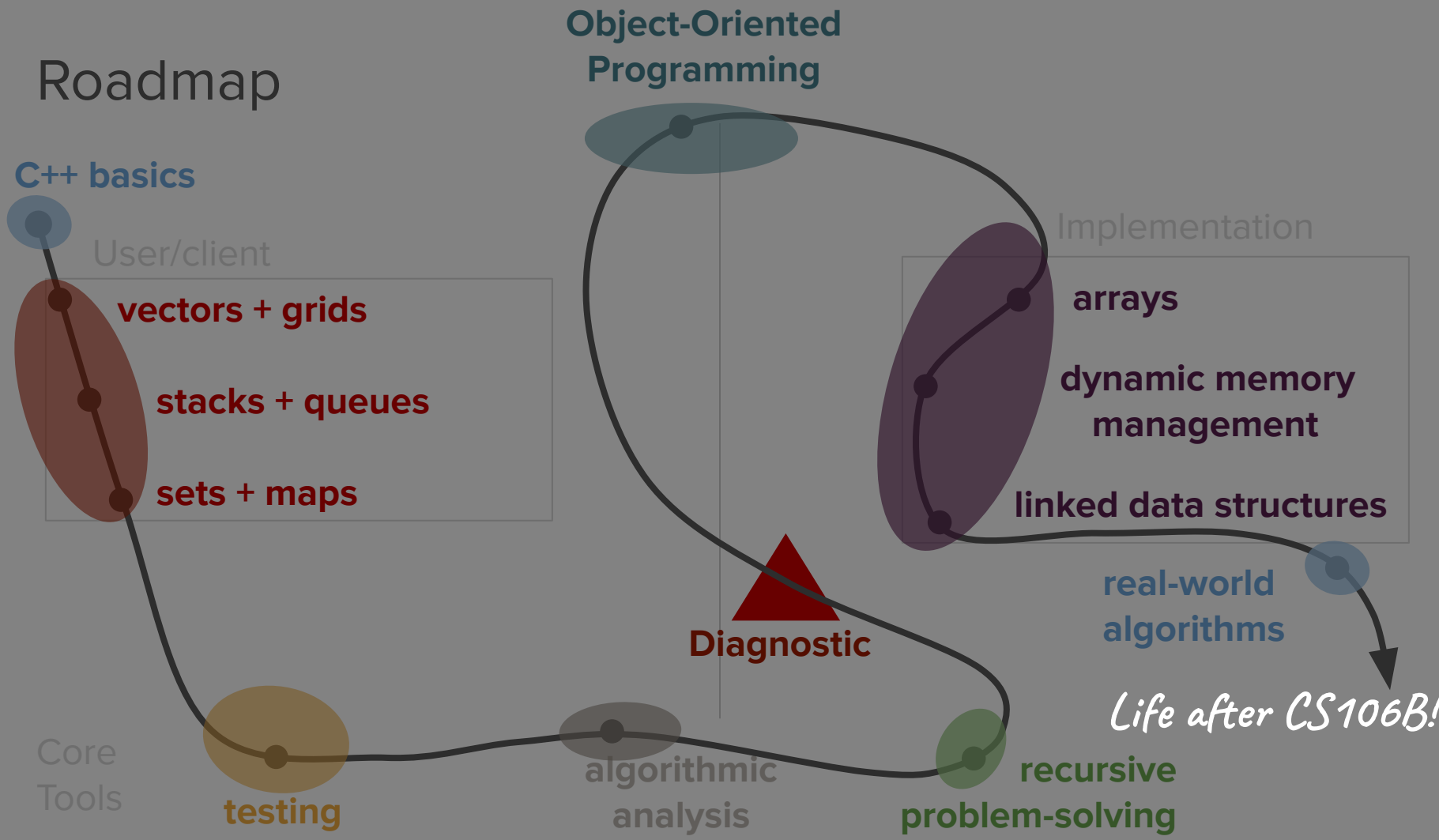
Closing thoughts

- 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


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Diagnostic
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving

Thank you!!!