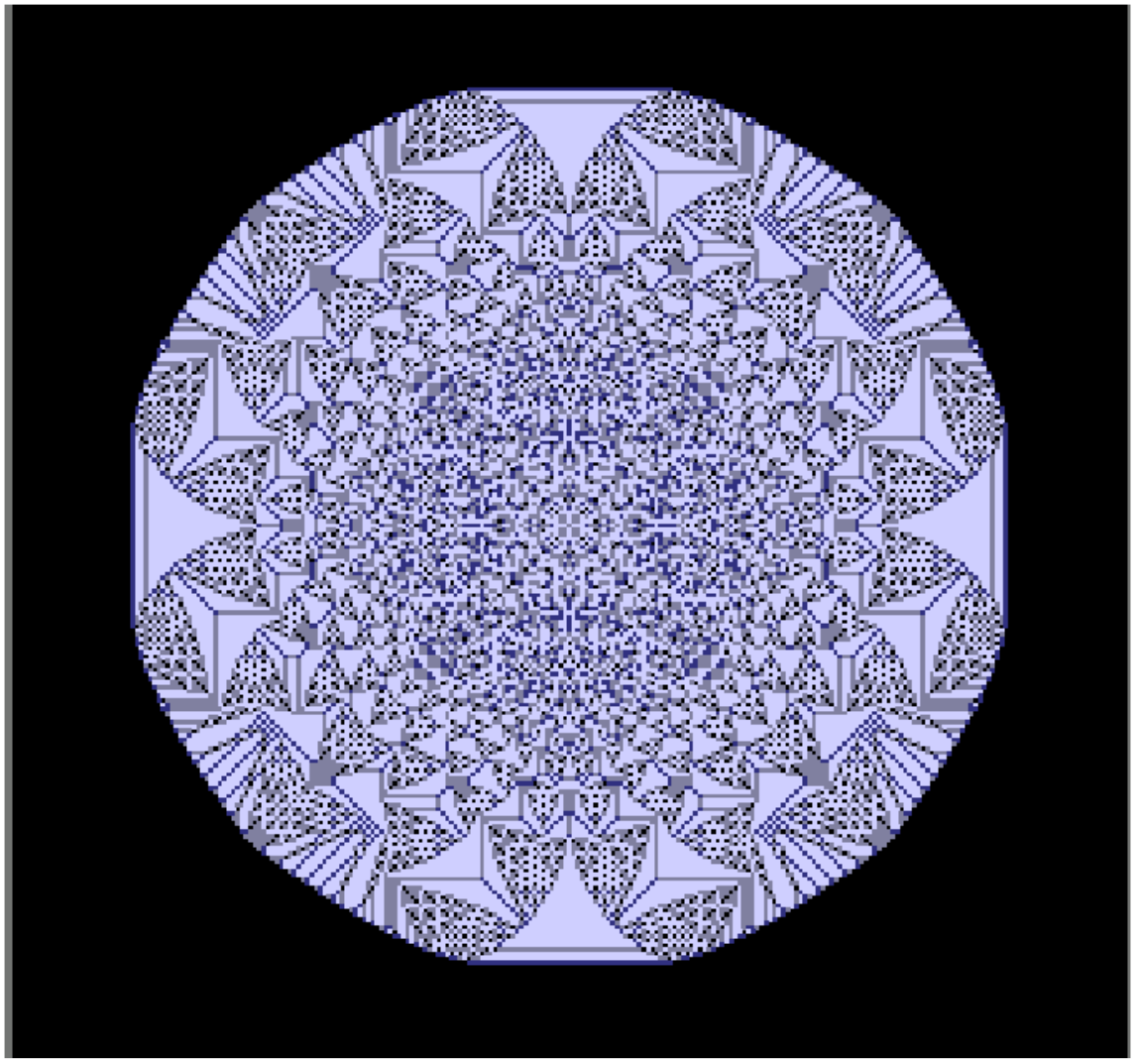
Where to Go from Here

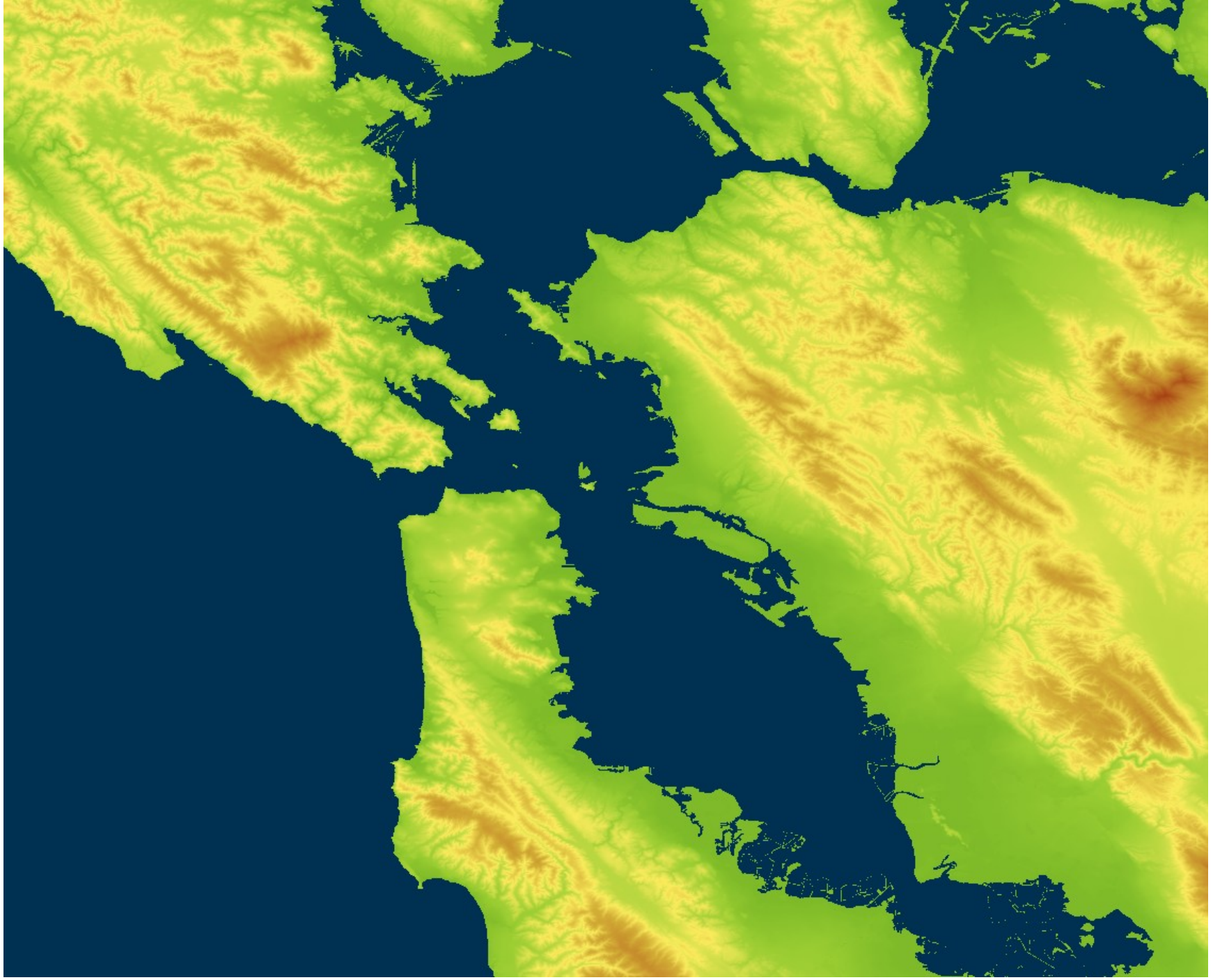
Taking Stock: Where Are We?

Goals for this Course

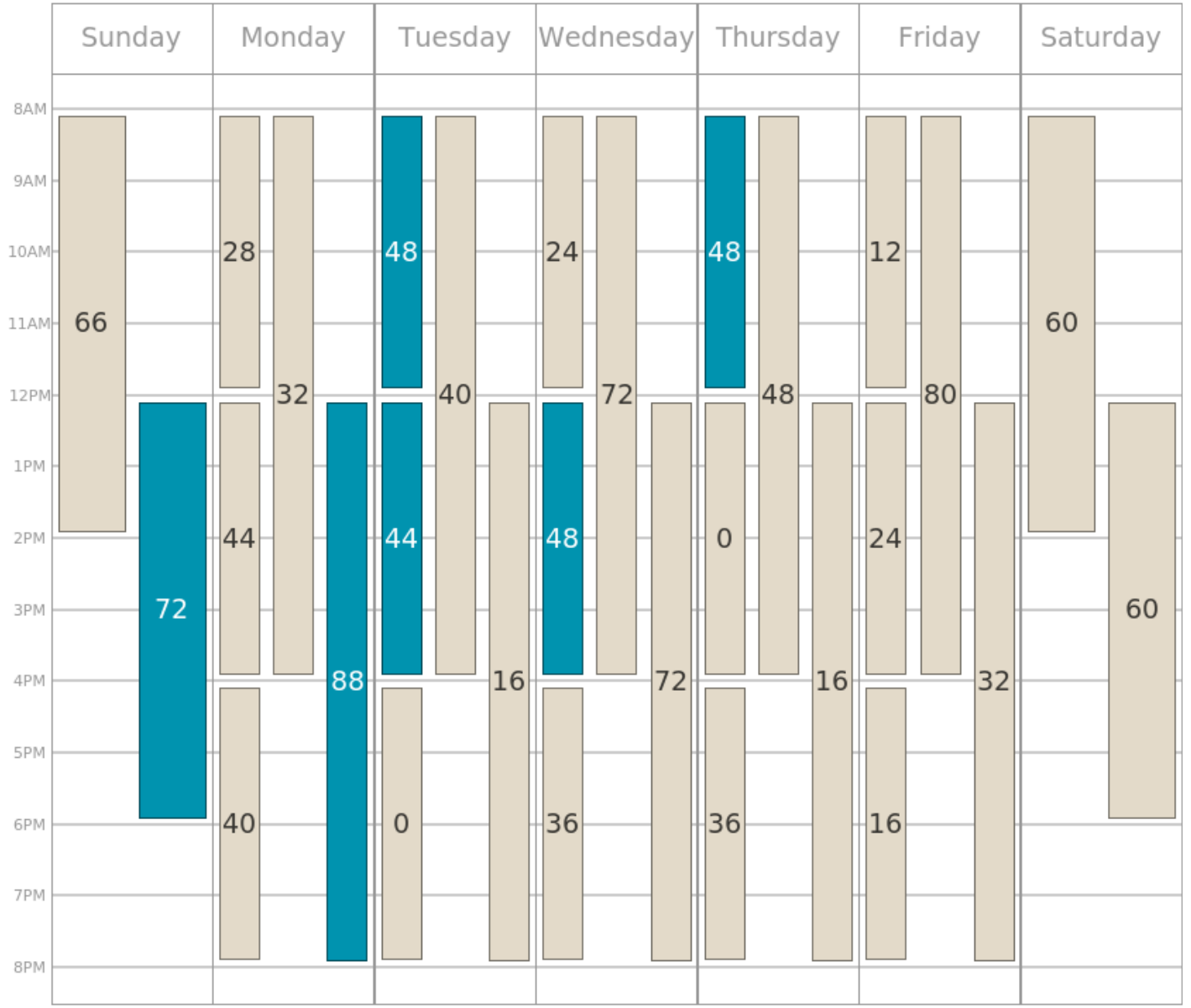
- ***Learn how to model and solve complex problems with computers.***
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.



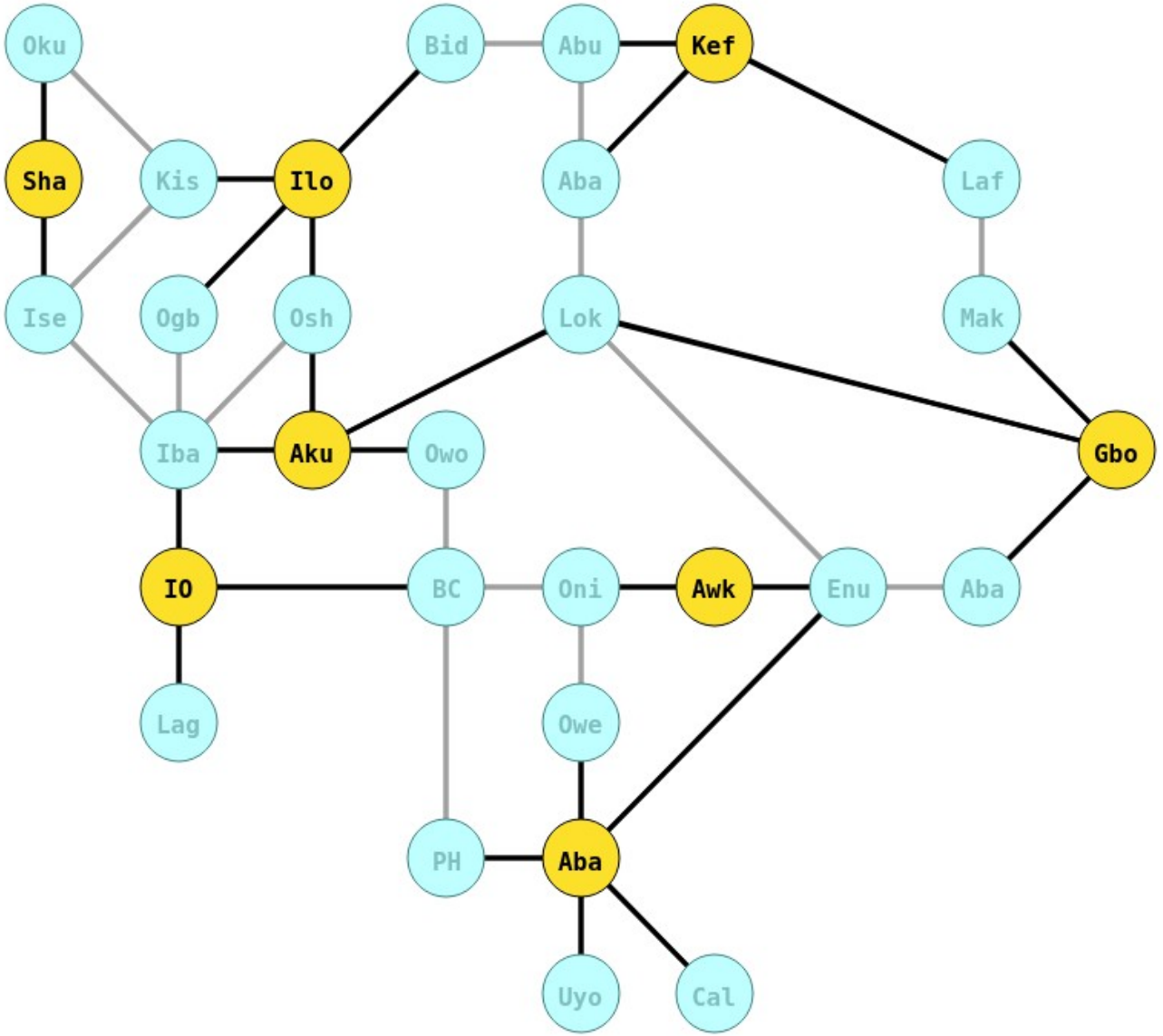
Assignment 1: Strings, Streams, and Recursion



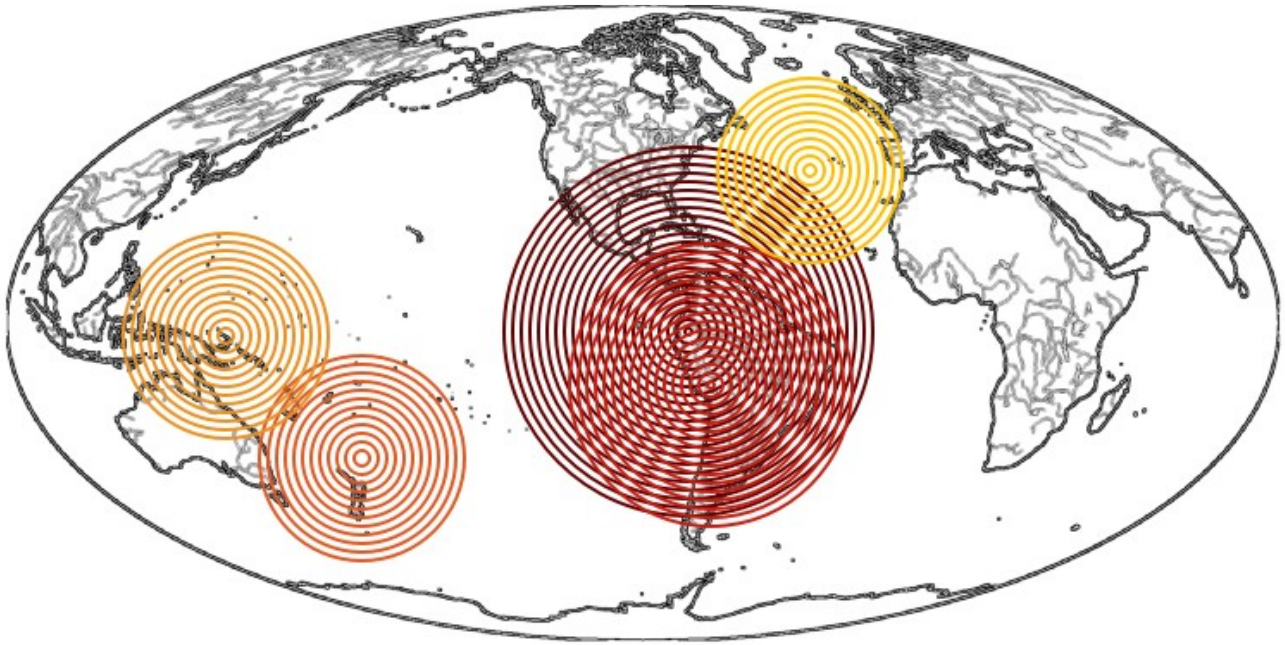
Assignment 2: Container Types



Assignment 3: Memoization, Recursive Optimization



Assignment 4: Recursive Backtracking



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.

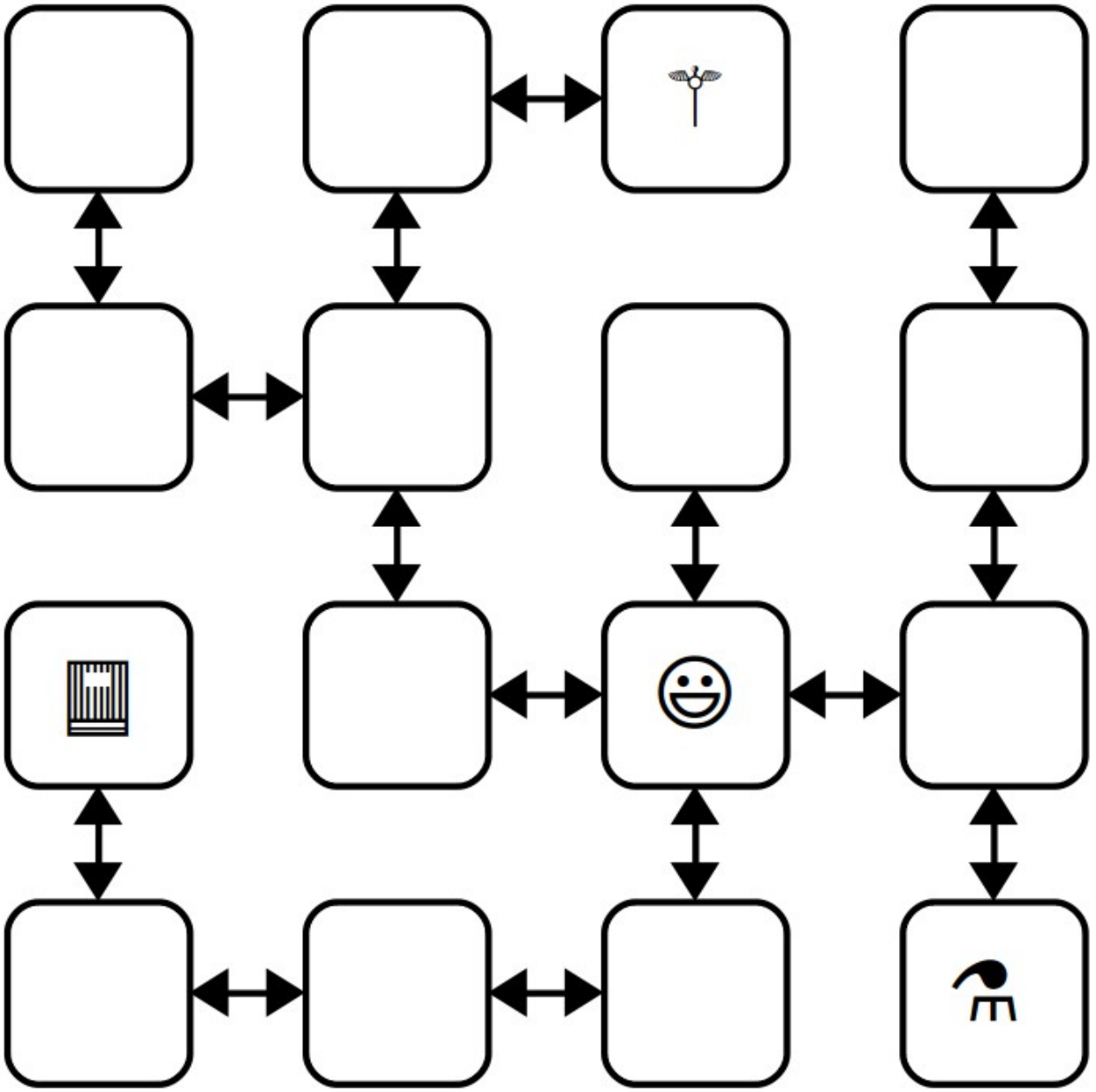
Remember that the earthquake magnitude scale is logarithmic. An earthquake that is on magnitude in strength higher than another releases around 32 times as much energy.

- 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

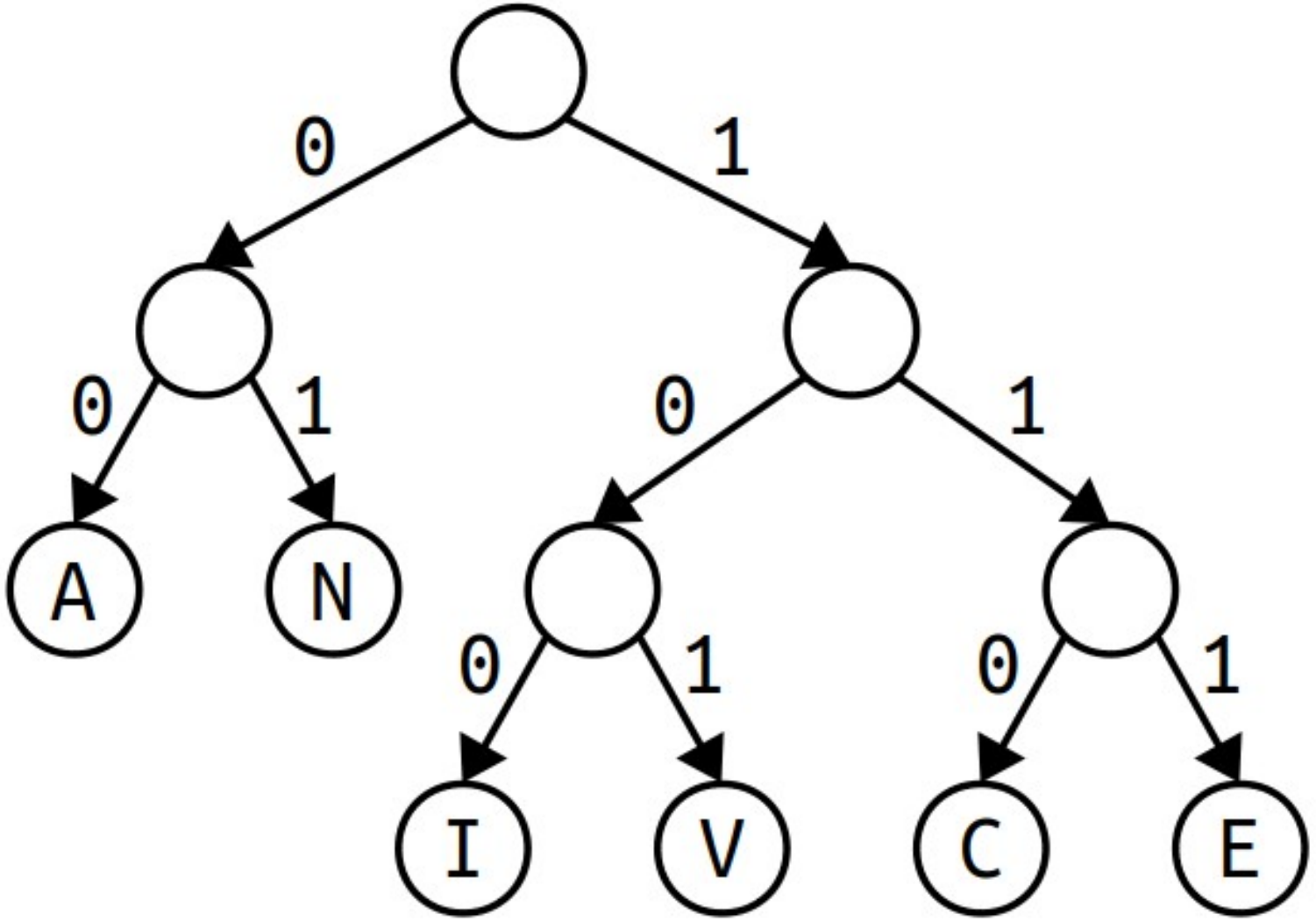
Assignment 5: Big-O, Sorting, Dynamic Arrays

		Chained Hashing	Linear Probing	Robin Hood Hashing
α = 0.5	Insert (success)	758.11ns	388.44ns	406.33ns
	Insert (failure)	424.51ns	247.08ns	262.46ns
	Lookup (success)	411.30ns	244.01ns	265.86ns
	Lookup (failure)	346.17ns	250.69ns	237.27ns
	Remove (success)	451.11ns	242.85ns	447.46ns
	Remove (failure)	285.53ns	251.65ns	240.45ns
α = 0.6	Insert (success)	745.39ns	390.01ns	410.35ns
	Insert (failure)	413.00ns	249.98ns	265.34ns
	Lookup (success)	412.50ns	245.00ns	261.22ns
	Lookup (failure)	349.92ns	255.58ns	236.88ns
	Remove (success)	448.89ns	243.58ns	441.84ns
	Remove (failure)	291.13ns	257.51ns	240.83ns
α = 0.7	Insert (success)	750.09ns	393.45ns	416.94ns
	Insert (failure)	415.35ns	251.90ns	271.68ns
	Lookup (success)	413.80ns	249.08ns	266.31ns
	Lookup (failure)	359.01ns	279.67ns	241.74ns
	Remove (success)	447.78ns	247.36ns	456.06ns
	Remove (failure)	296.00ns	280.64ns	245.12ns

Assignment 6: Hash Functions, Class Design



Assignment 7: Linked Structures



Assignment 8: Trees and Tree Searches

What We've Covered

Strings

Streams

Recursive Problem-Solving

Stacks

Queues

Vectors

Maps

Sets

Lexicons

What We've Covered

Recursive Graphics

Recursive Enumeration

Recursive Backtracking

Big-O Notation

Sorting Algorithms

Designing Abstractions

Constructors and Destructors

What We've Covered

Dynamic Arrays

Chained Hashing

Linear Probing

Robin Hood Hashing

Linked Lists

Binary Search Trees

Huffman Coding

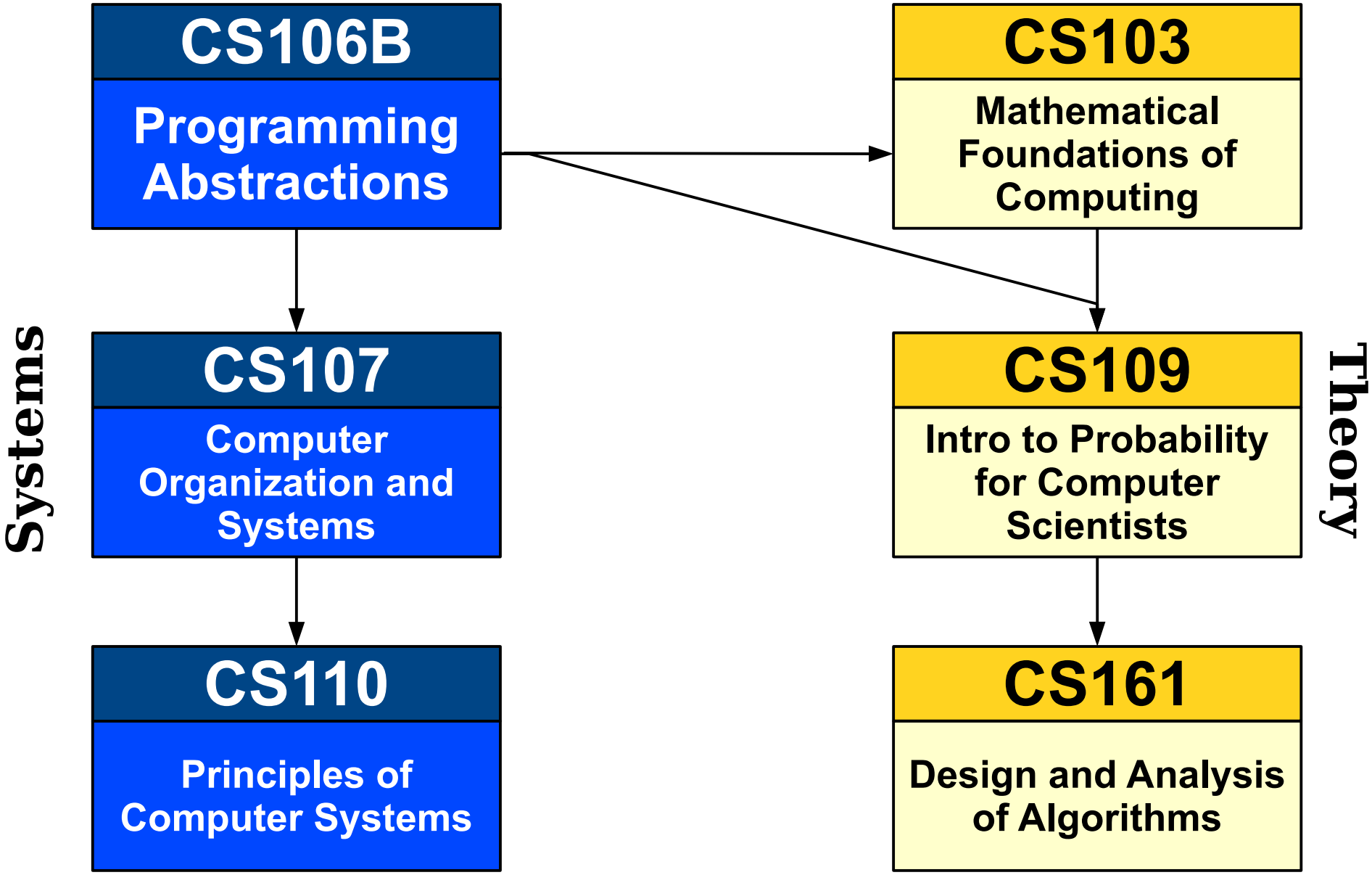
Computer science is more than just programming.

These skills will make you better at whatever you choose to do.

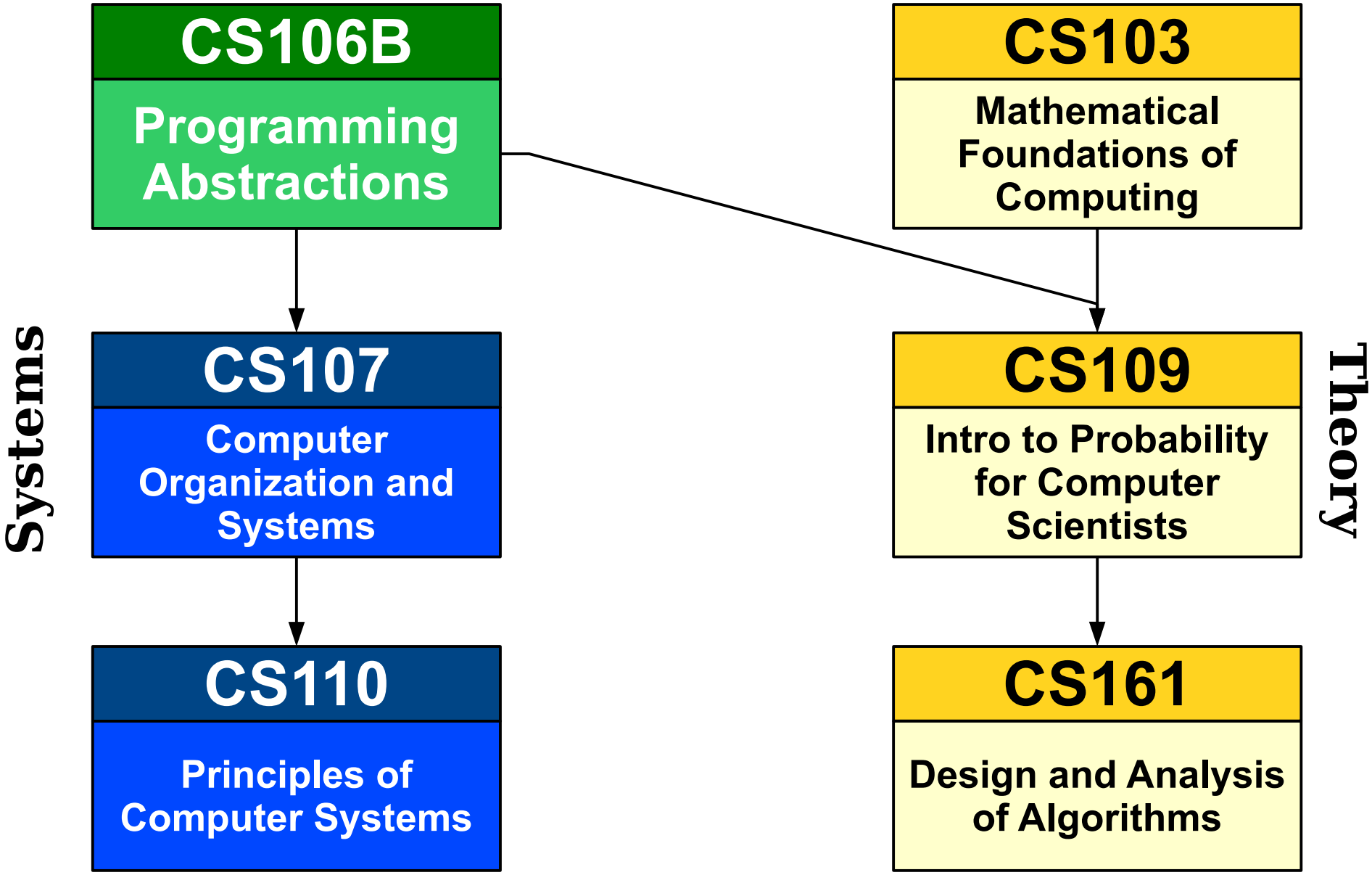
So what comes next?

Courses to Take

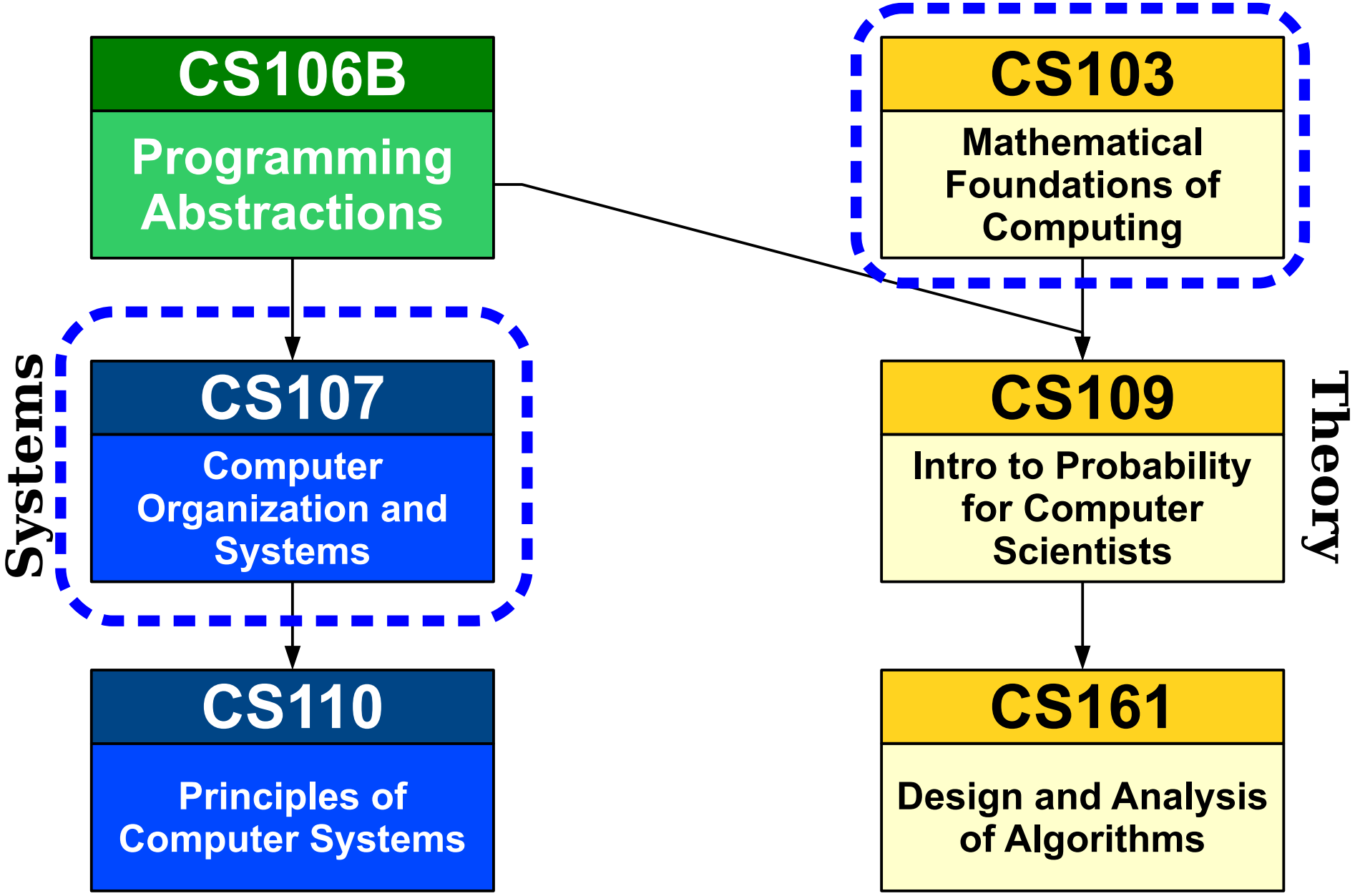
The CS Core

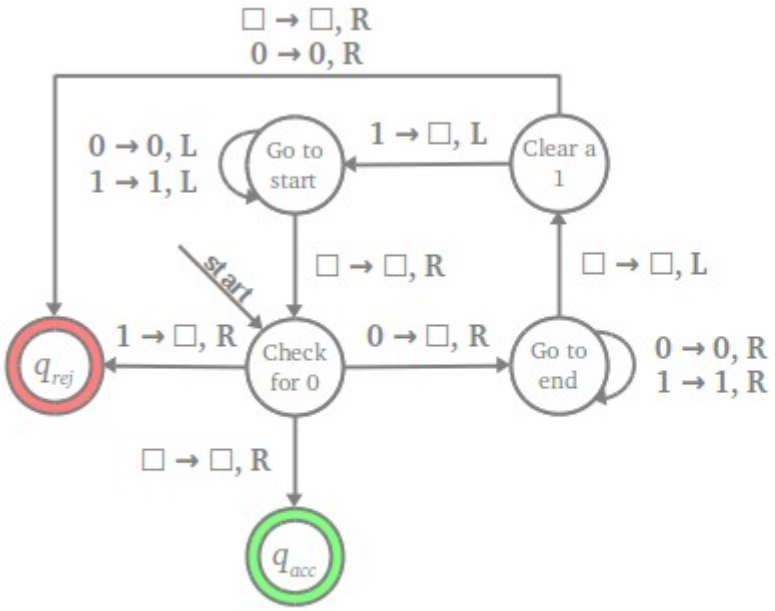

CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms
Systems
Theory

The CS Core


CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms
Systems
Theory

The CS Core


CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
Systems
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
Theory
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms



CS103

Mathematical Foundations of Computing

What are the fundamental limits of computing power?

How can we be certain about this?

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?

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.

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.
- CS107 is ***not*** soul-crushingly impossibly hard.
  - It's tricky. It does not eat kittens.
- *Don't be afraid to try CS107!*

Other CS Courses

CS193

Programming Language Particulars

- 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

Standard 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.

CS106E

Practical Exploration of Computing

- Broad survey of computing topics, including
  - how the internet works,
  - computer security,
  - how operating systems work,
  - bits and bytes, and
  - web programming.
- Great course if you're interested in working in the software industry in a non-technical capacity.

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! √

CS109

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?

The CS Major

Thinking about 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

- A common timetable:
  - Aim to complete *most* of the core by the end of your sophomore year (probably CS106B, CS103, CS107, CS109, and one of CS110 and CS161).
  - 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!

The CS Coterm

The CS Coterm

- The CS coterm is open to students of *all majors*, not just computer science.
  - ***This is intentional***. We want the doors to be open to all comers.
- Thinking about applying?
  - ***Take enough CS classes to establish a track record***.
  - ***Maintain a solid CS GPA***. Aim high!
- TA and RA positions are available to offset the cost.
  - Some of my best TAs did their undergrad in comparative literature, anthropology, and, physics.

The CS Minor

Outside Stanford

Learning More

- Some cool directions to explore:
  - ***Specific technologies***. You already know how to program. You just need to learn new technologies, frameworks, etc.
  - ***Algorithms***. Learn more about what problems we know how to solve.
  - ***Software engineering***. Crafting big software systems is an art.
  - ***Machine learning***. If no new ML discoveries were made in the next ten years, we'd still see huge improvements.

How to Explore Them

- MOOCs are a great way to get an introduction to more conceptual topics.
  - Andrew Ng's machine learning course, Fei Fei Li's computer vision course, Tim Roughgarden's algorithms course, and Jennifer Widom's databases courses are legendary.
- Learning by doing is the best way to pick up new languages and frameworks.
  - Find a good tutorial (ask around), plan to make a bunch of mistakes, and have fun!
- Know where to ask for help.
  - Stack Overflow is an excellent resource.

Some Words of Thanks

Who's Here Today?

- Aero/Astro
- Anthropology
- Art Practice
- Bioengineering
- Biology
- Business
- Chemical Engineering
- Chemistry
- Civil/Environmental Engineering
- Creative Writing
- Data Science
- East Asian Studies
- Economics
- Education
- Electrical Engineering
- Energy Resources Engineering
- English
- Environmental Systems Engineering
- FemGen
- Genetics
- History
- Human Biology
- Immunology
- International Relations
- Law
- Materials Science
- Mechanical Engineering
- Microbiology and Immunology
- Middle Eastern Languages / Culture
- MS&E
- Physics
- Political Science
- Product Design
- Psychology
- Public Policy
- Spanish
- Statistics
- STS
- Symbolic Systems
- SymSys
- ***Undeclared!***

My Email Address

htiek@cs.stanford.edu

You now have a wide array of tools you can use to solve a huge number of problems.

You have the skills to compare and contrast those solutions.

You have expressive mental models for teasing apart those problems.

My Questions to You:

What problems will you choose to solve?
Why do those problems matter to you?
And how are you going to solve them?