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.

What We’ve Covered

Strings

Recursion

Stacks

Queues

Vectors

Maps

Sets

Lexicons

What We’ve Covered

Recursive Graphics

Recursive Enumeration

Recursive Backtracking

Big-O Notation

Sorting Algorithms

Class Design

Pointers and Memory

Constructors and Destructors

What We’ve Covered

Dynamic Arrays

Chained Hashing

Linear Probing

Robin Hood Hashing

Linked Lists

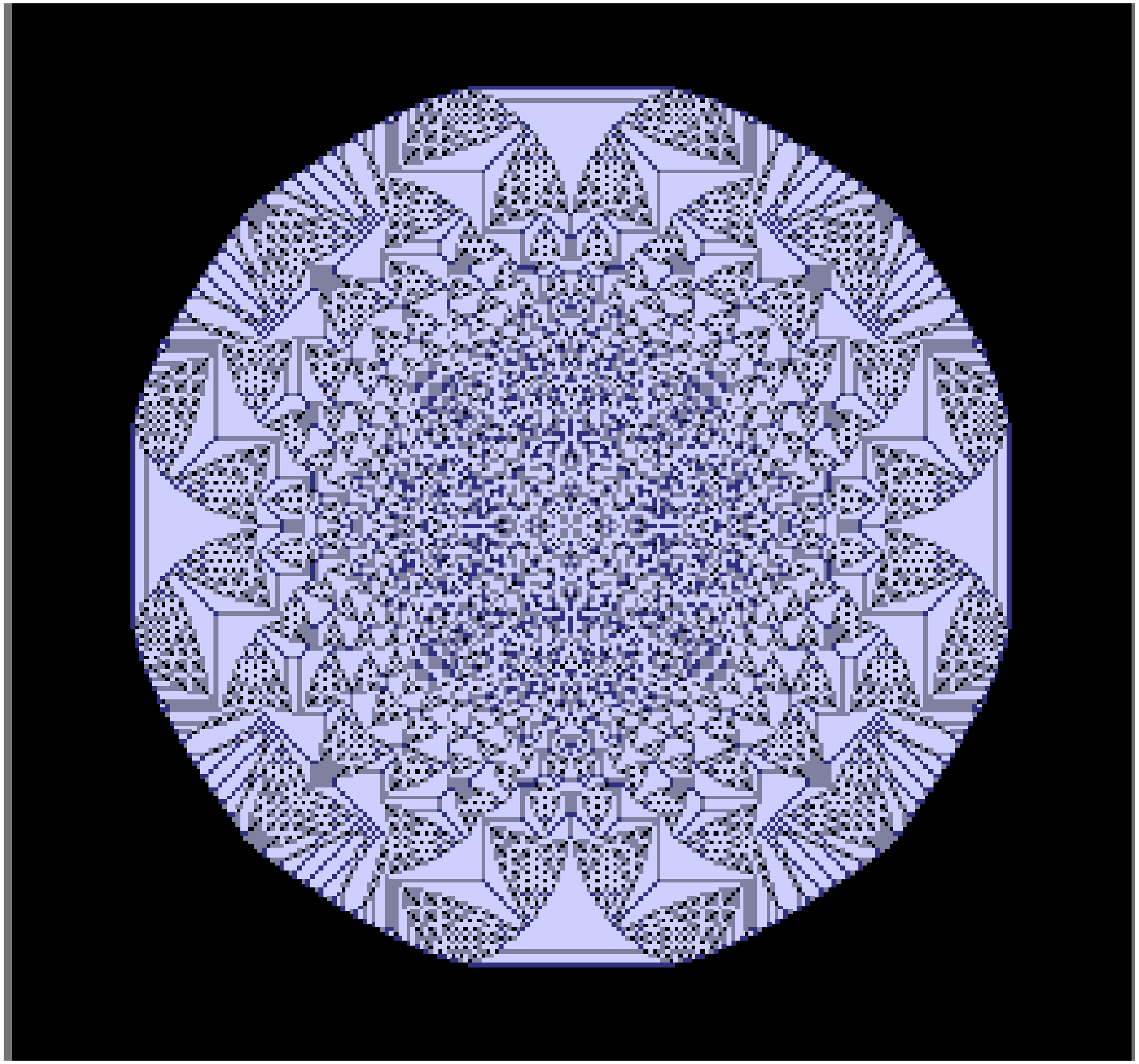
Binary Search Trees

Huffman Coding

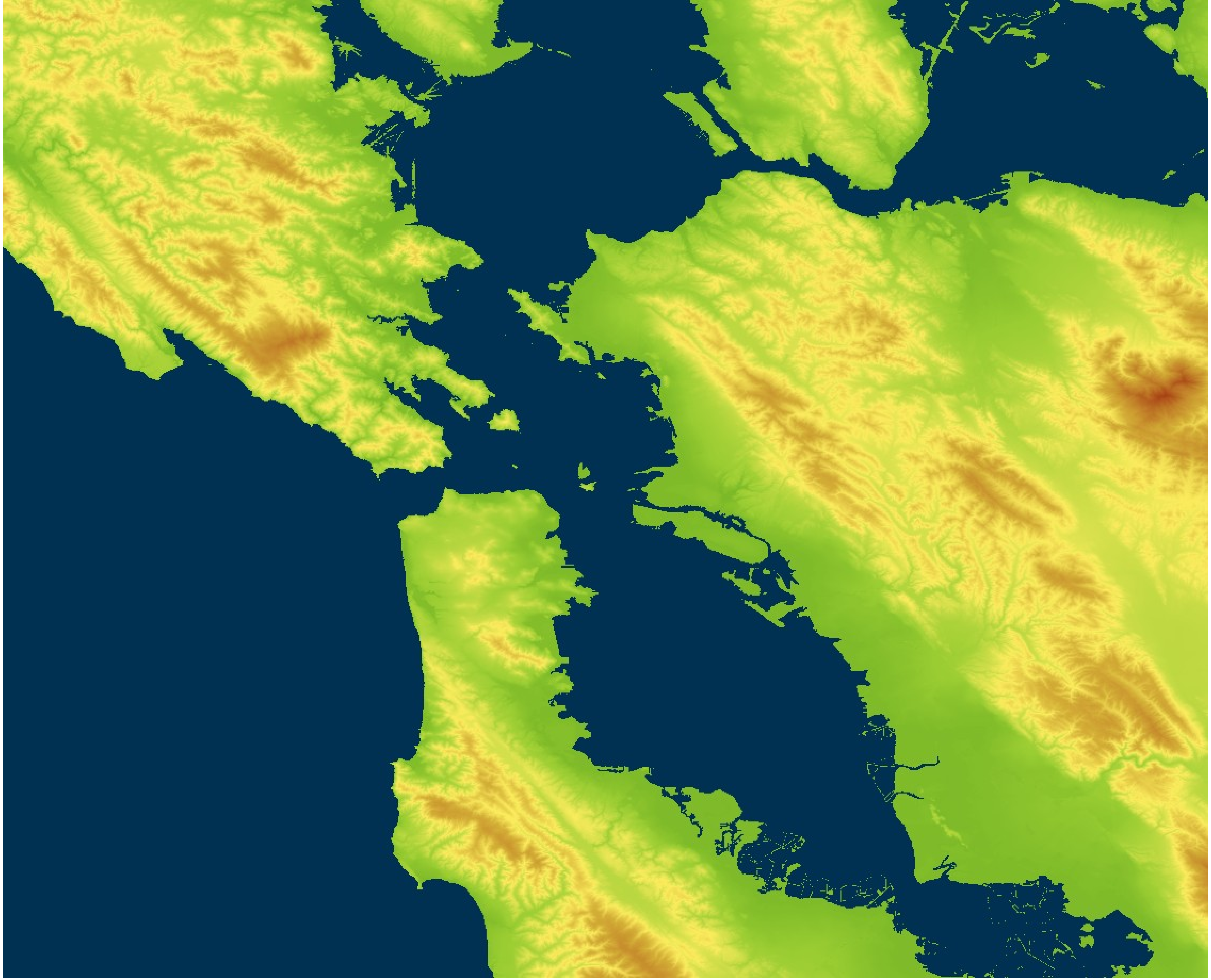
Graphs

You didn’t just learn a list of concepts.

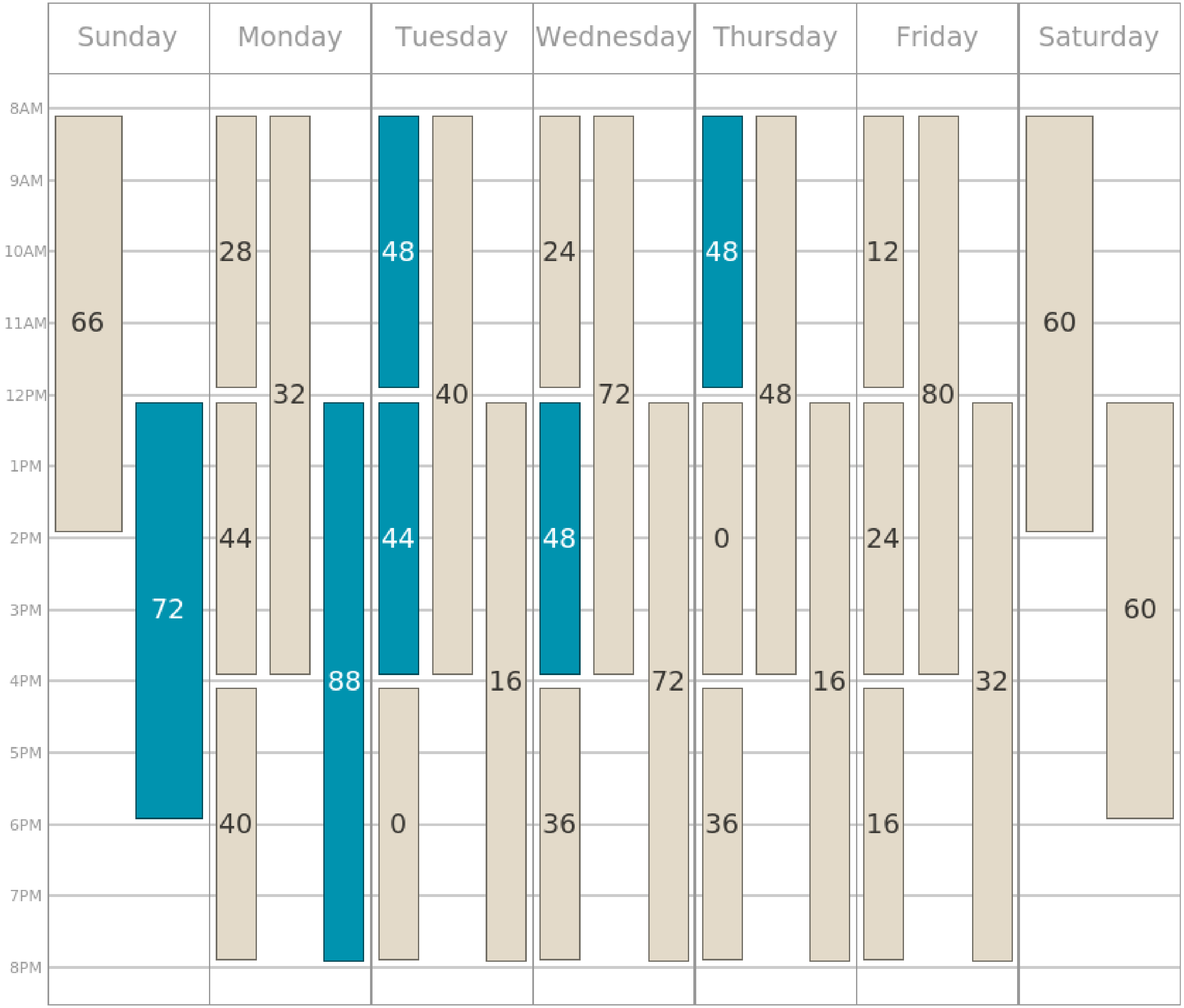
You learned to make those concepts ***shine***.



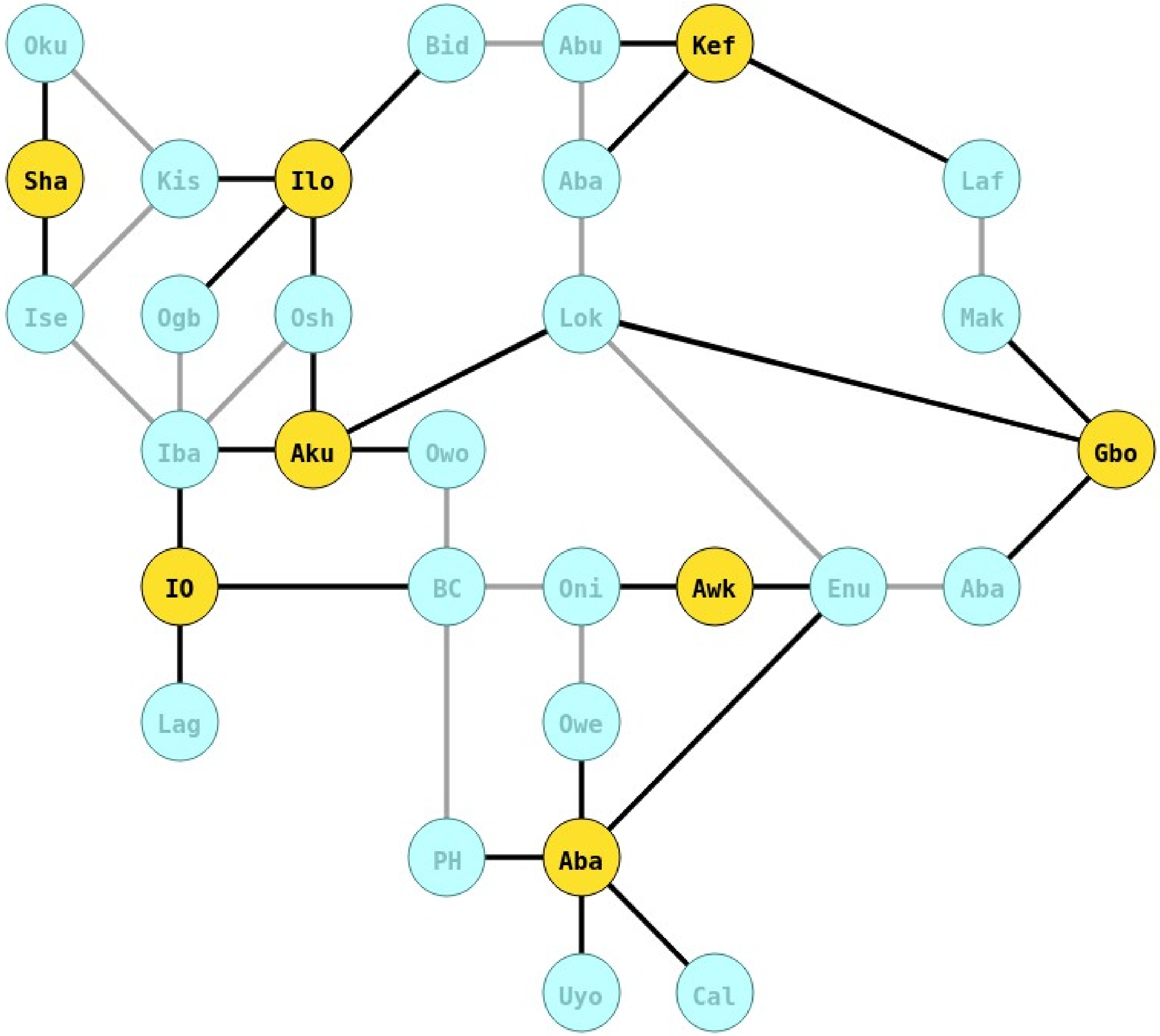
Assignment 1: Strings, Streams, and Recursion



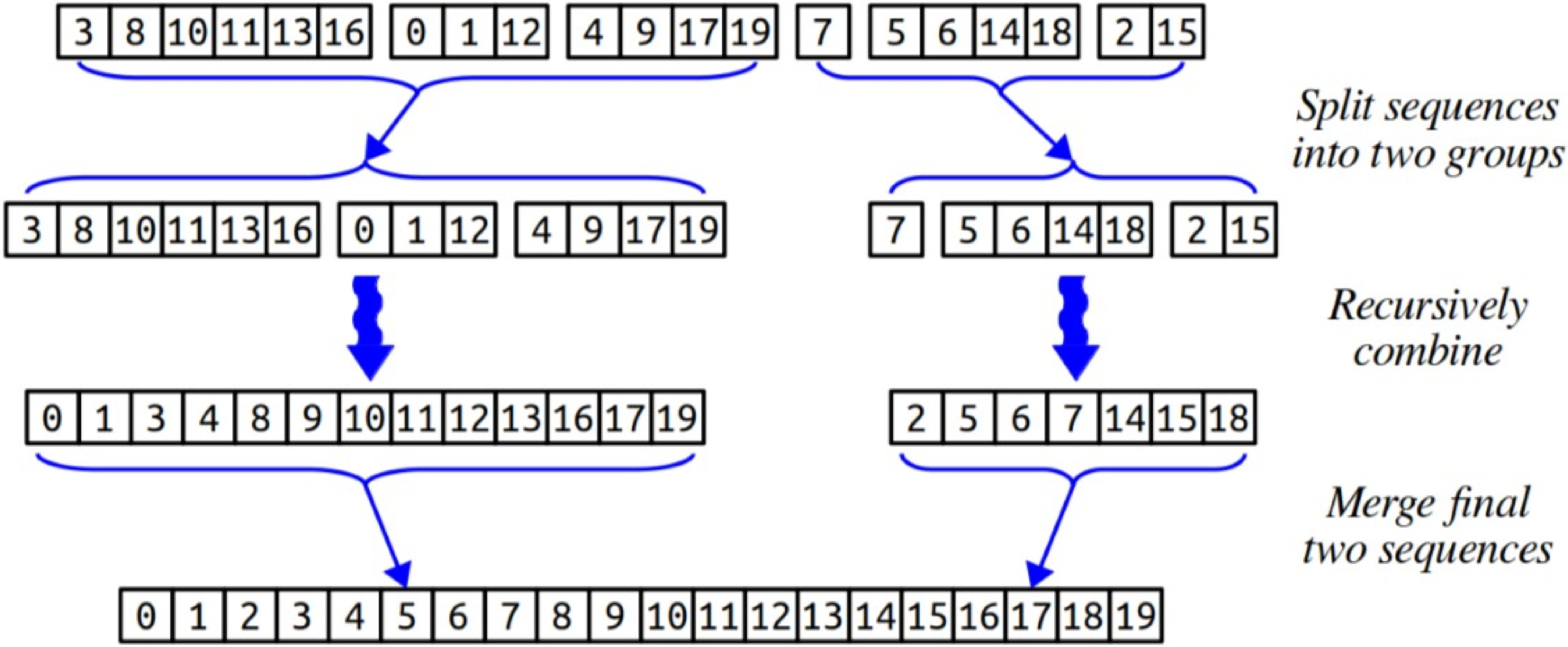
Assignment 2: Container Types



Assignment 3: Recursive Problem-Solving



Assignment 4: Recursive Backtracking



3 8 10 11 13 16 | 0 1 12 | 4 9 17 19 | 7 | 5 6 14 18 | 2 15

Split sequences into two groups

3 8 10 11 13 16 | 0 1 12 | 4 9 17 19

7 | 5 6 14 18 | 2 15

Recursively combine

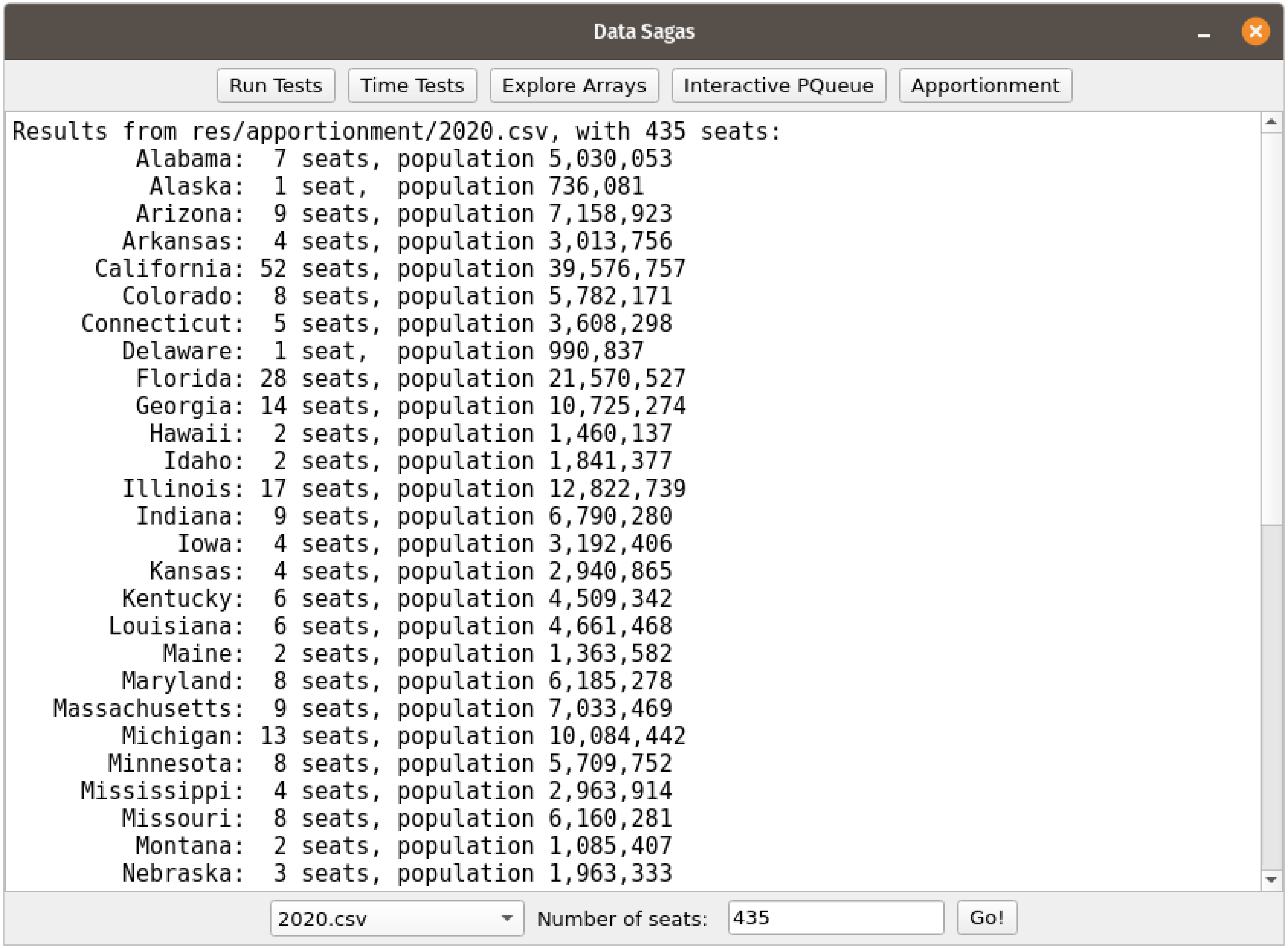
0 1 3 4 8 9 10 11 12 13 16 17 19

2 5 6 7 14 15 18

Merge final two sequences

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Assignment 5: Big-O, Sorting

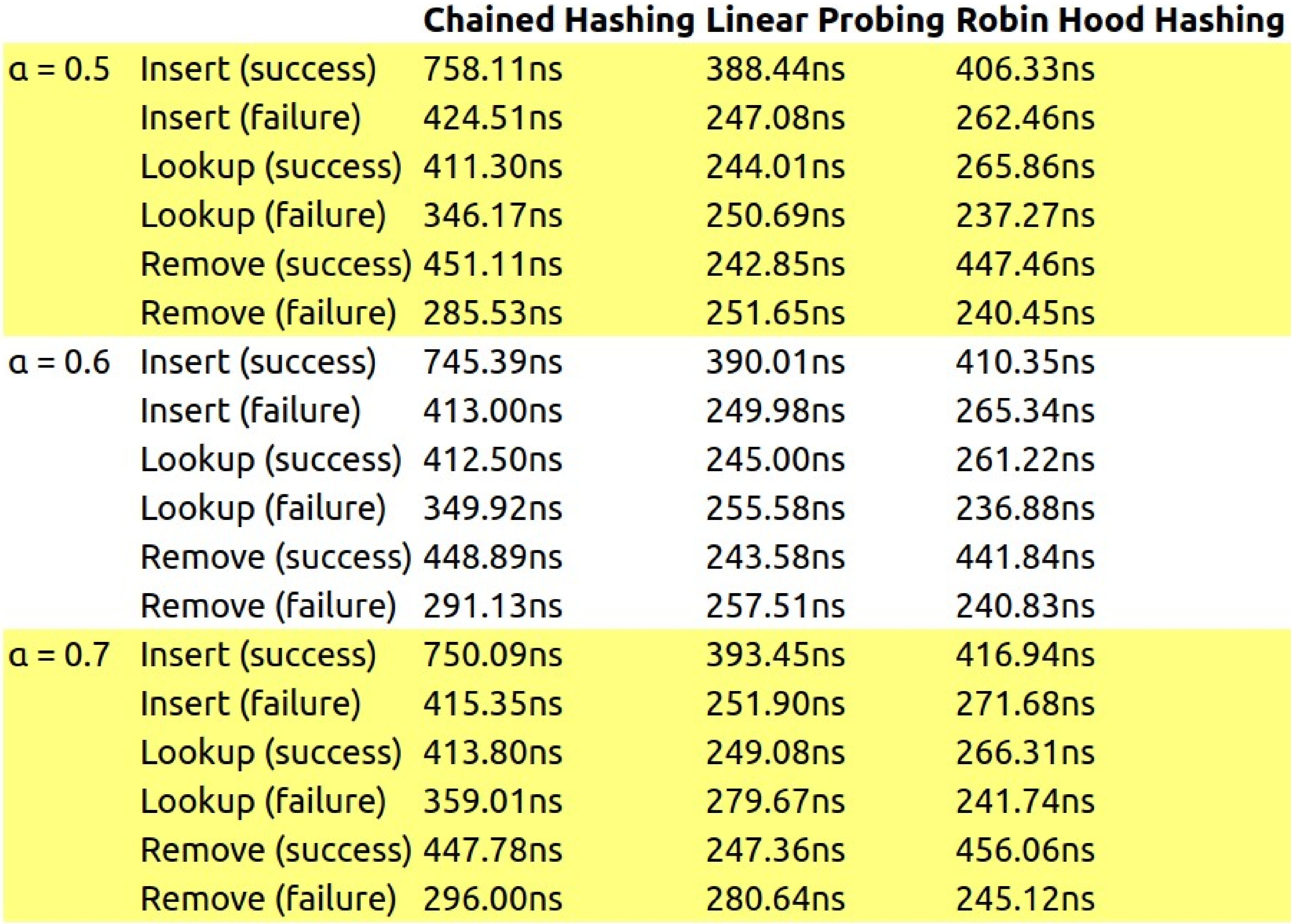


```
Results from res/apportionment/2020.csv, with 435 seats:
         Alabama:  7 seats, population 5,030,053
          Alaska:  1 seat,  population 736,081
         Arizona:  9 seats, population 7,158,923
        Arkansas:  4 seats, population 3,013,756
      California: 52 seats, population 39,576,757
        Colorado:  8 seats, population 5,782,171
     Connecticut:  5 seats, population 3,608,298
        Delaware:  1 seat,  population 990,837
         Florida: 28 seats, population 21,570,527
         Georgia: 14 seats, population 10,725,274
          Hawaii:  2 seats, population 1,460,137
           Idaho:  2 seats, population 1,841,377
        Illinois: 17 seats, population 12,822,739
         Indiana:  9 seats, population 6,790,280
            Iowa:  4 seats, population 3,192,406
          Kansas:  4 seats, population 2,940,865
        Kentucky:  6 seats, population 4,509,342
       Louisiana:  6 seats, population 4,661,468
           Maine:  2 seats, population 1,363,582
        Maryland:  8 seats, population 6,185,278
   Massachusetts:  9 seats, population 7,033,469
        Michigan: 13 seats, population 10,084,442
       Minnesota:  8 seats, population 5,709,752
     Mississippi:  4 seats, population 2,963,914
        Missouri:  8 seats, population 6,160,281
         Montana:  2 seats, population 1,085,407
        Nebraska:  3 seats, population 1,963,333
```

2020.csv

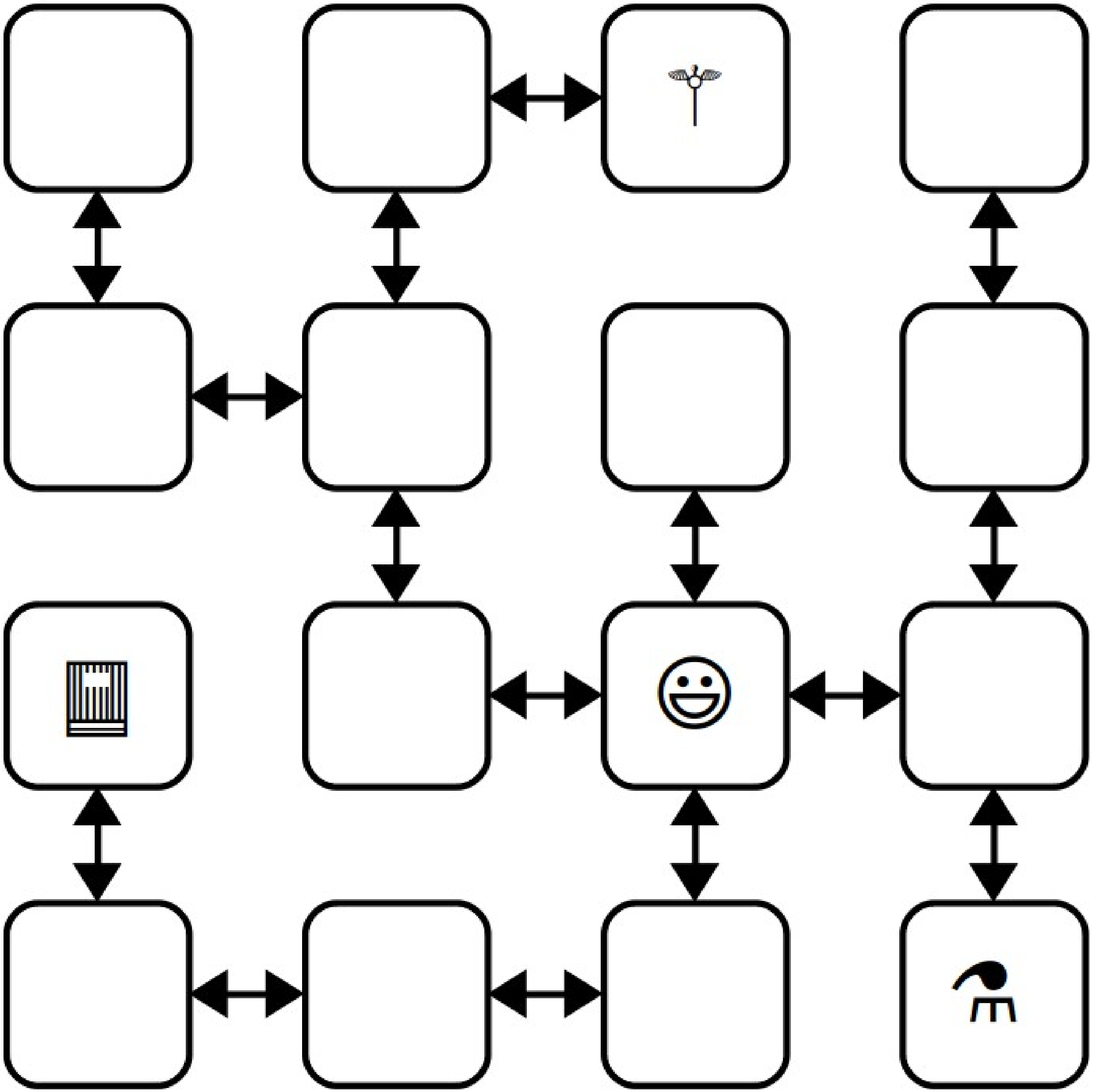
Number of seats: 435

Assignment 6: Classes, Dynamic Arrays

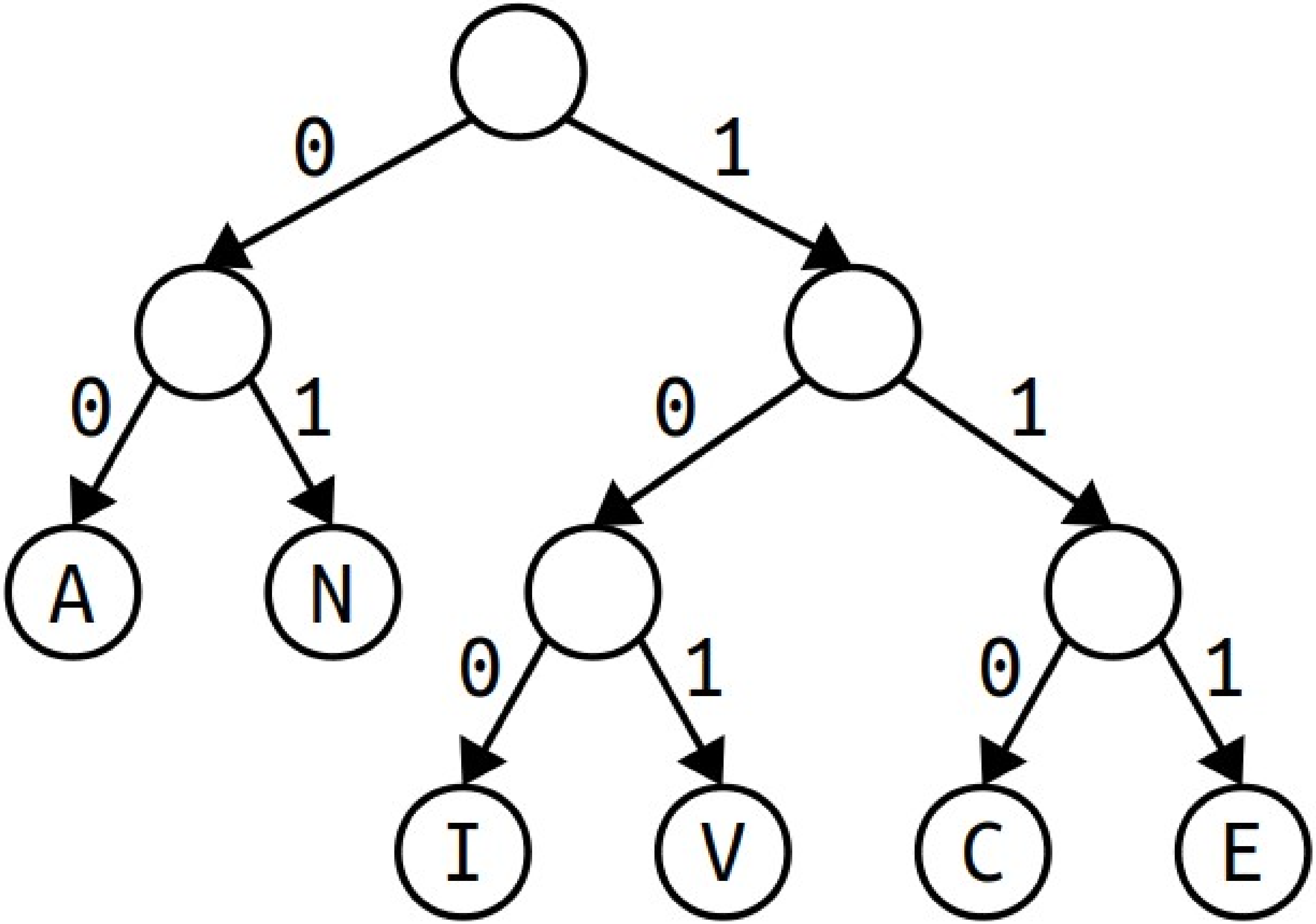


		Chained Hashing	Linear Probing	Robin Hood Hashing
α = 0.5	Insert (success)	758.11ns	388.44ns	406.33ns
	Insert (failure)	424.51ns	247.08ns	262.46ns
	Lookup (success)	411.30ns	244.01ns	265.86ns
	Lookup (failure)	346.17ns	250.69ns	237.27ns
	Remove (success)	451.11ns	242.85ns	447.46ns
	Remove (failure)	285.53ns	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 7: Hash Functions, Class Design



Assignment 8: Linked Structures



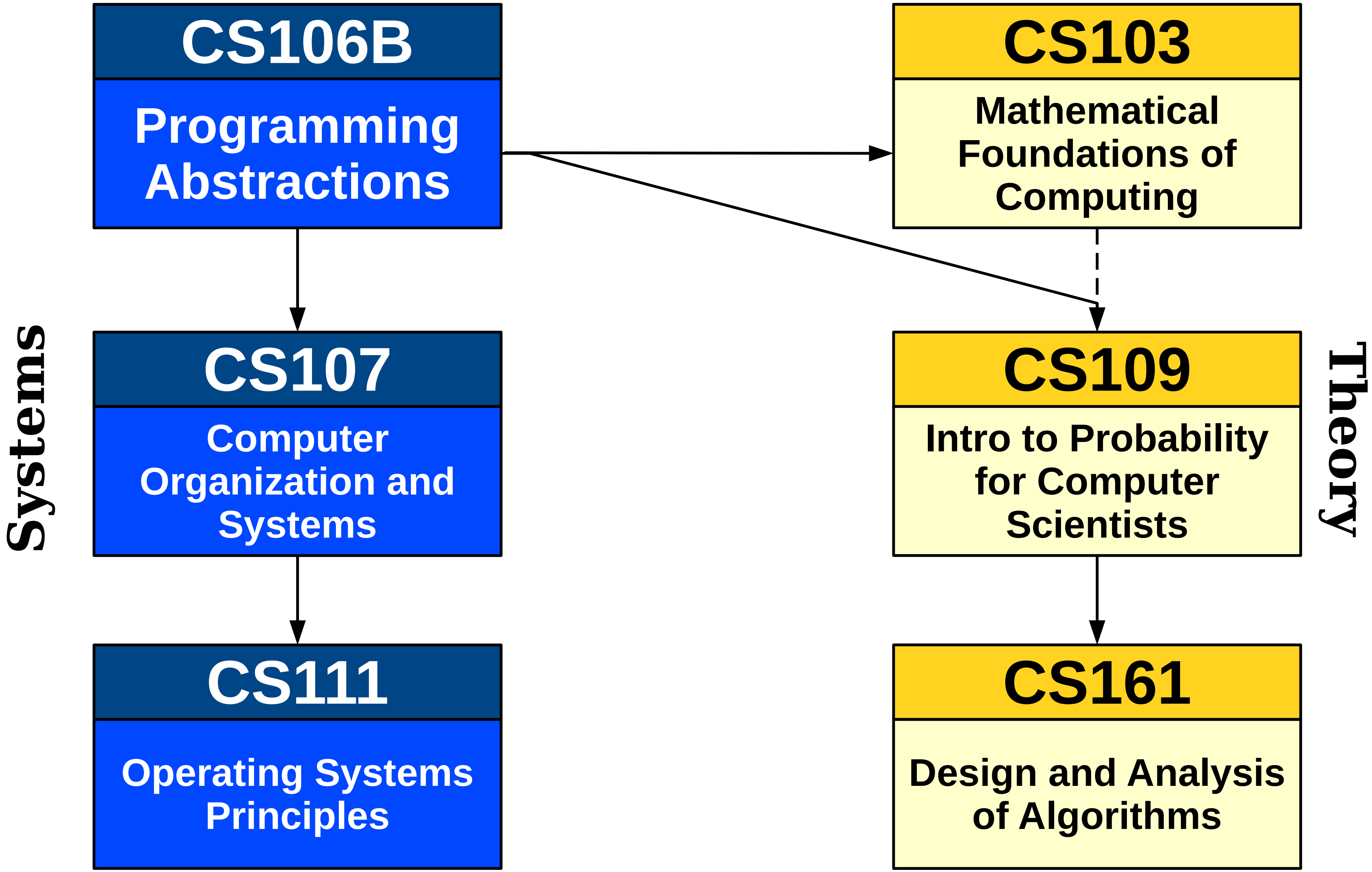
Assignment 9: Trees and Tree Searches

Computer science is more than just programming.

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

So what comes next?

The CS Core



Systems

CS106B
Programming Abstractions

CS107
Computer Organization and Systems

CS111
Operating Systems Principles

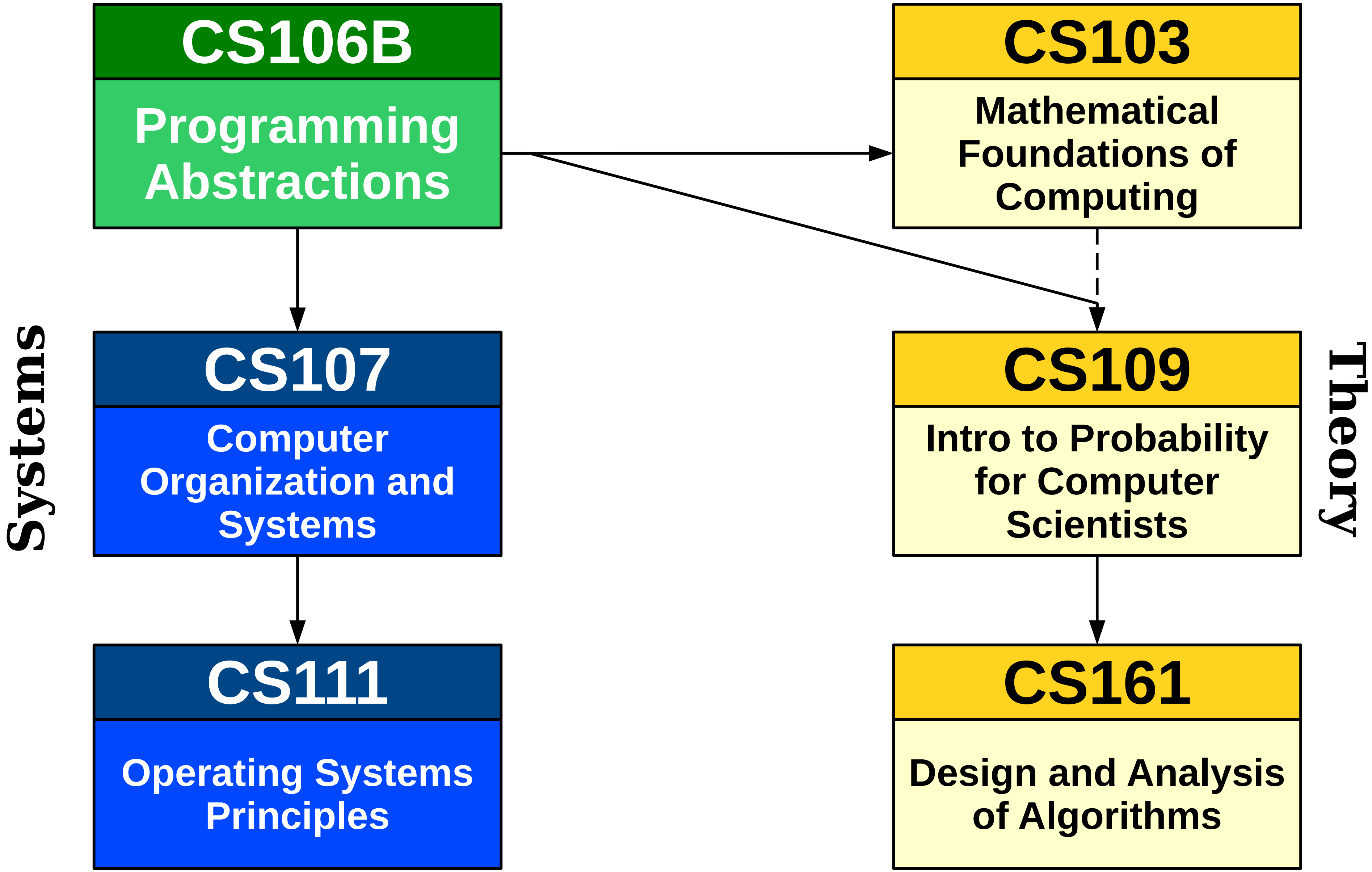
Theory

CS103
Mathematical Foundations of Computing

CS109
Intro to Probability for Computer Scientists

CS161
Design and Analysis of Algorithms

The CS Core



CS106B
Programming Abstractions

CS103
Mathematical Foundations of Computing

Systems

CS107
Computer Organization and Systems

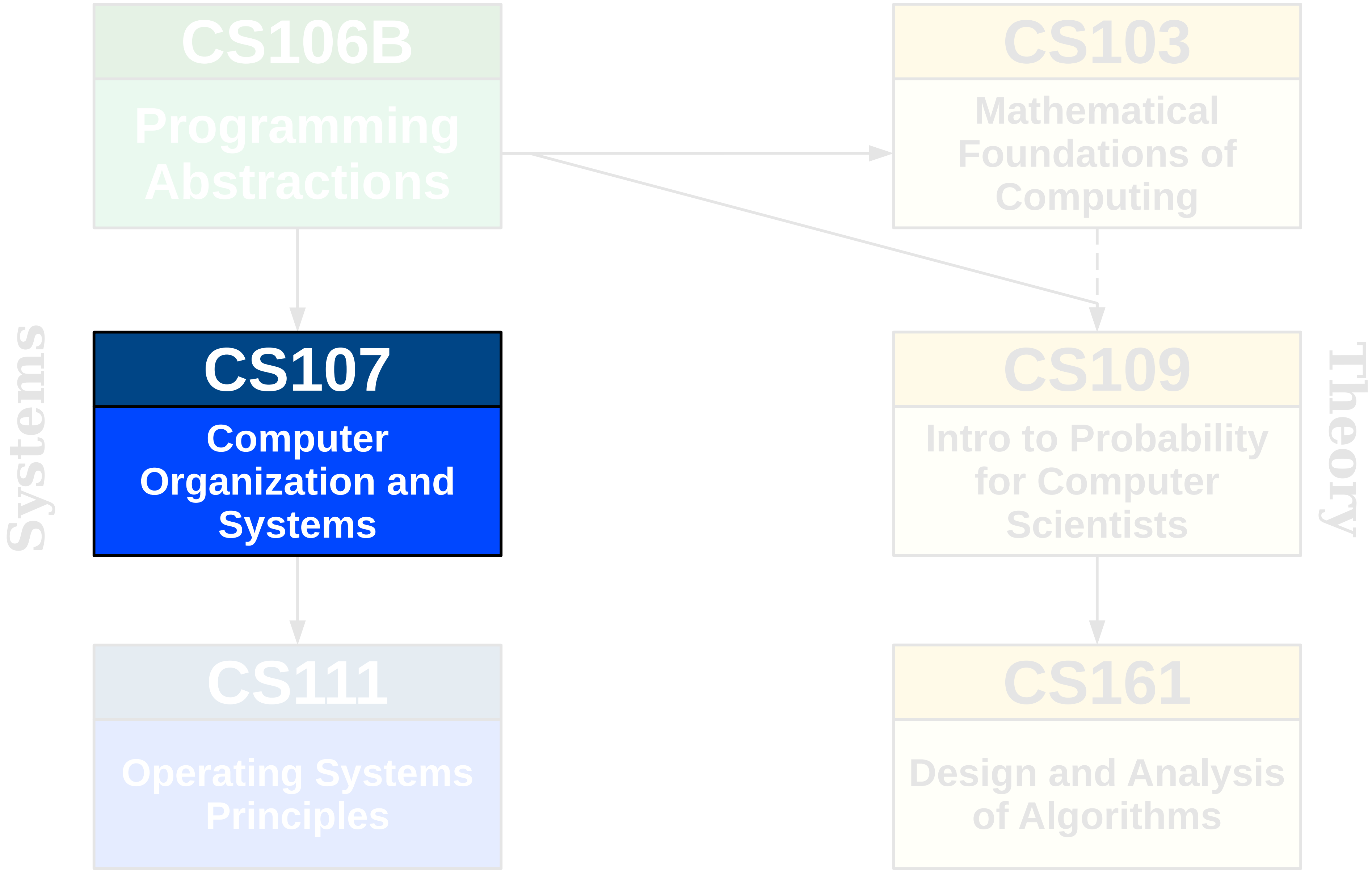
CS109
Intro to Probability for Computer Scientists

Theory

CS111
Operating Systems Principles

CS161
Design and Analysis of Algorithms

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
CS111
Operating Systems Principles
CS161
Design and Analysis of Algorithms

CS107

Computer Organization and Systems

Prerequisite: CS106B

How does the computer work, at its most basic levels?

How do those low-level details lead to larger-scale phenomena?

What levels of abstraction lie beneath basic C++ concepts?

CS107E

Computer Systems from the Ground Up

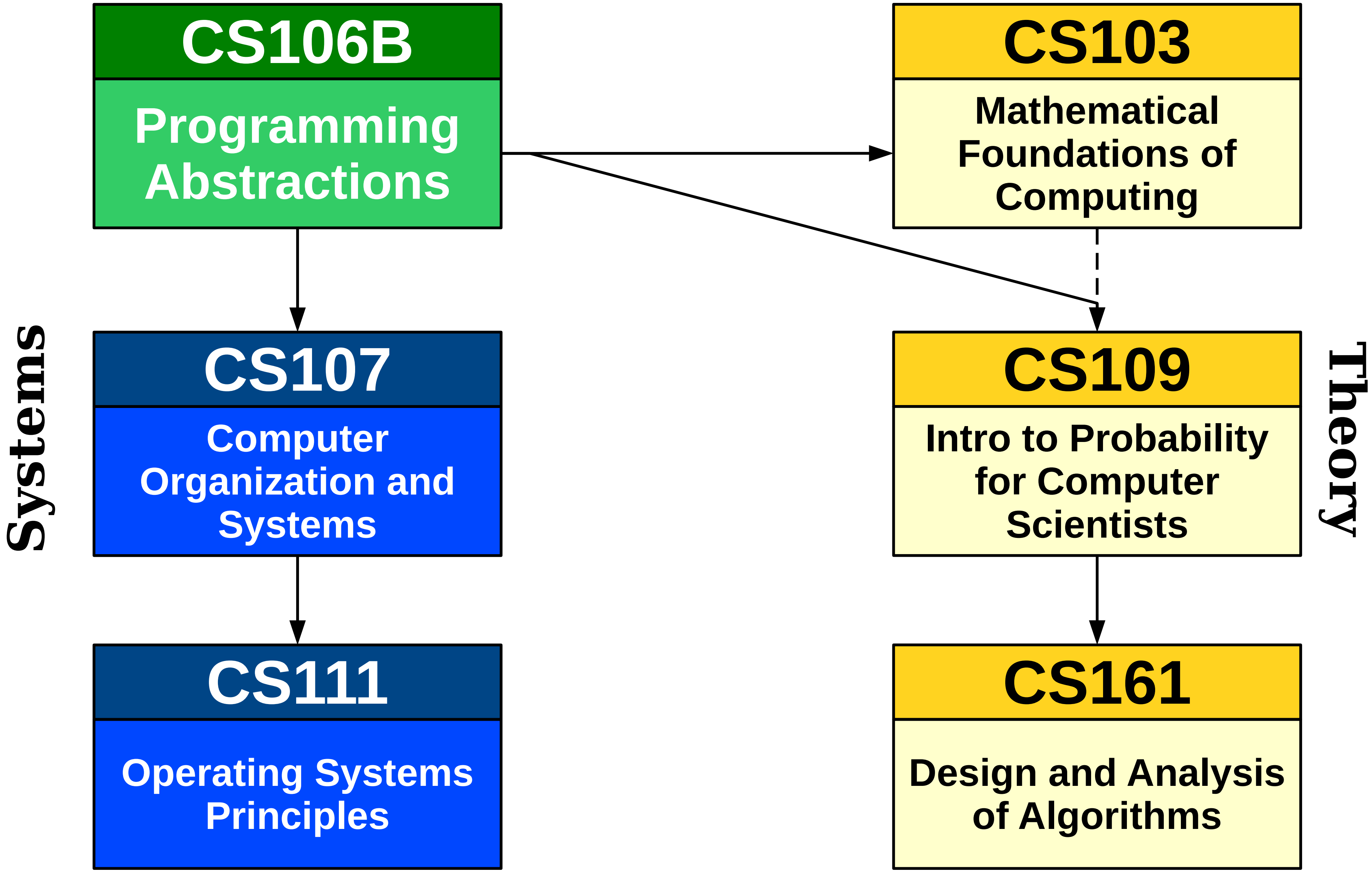
Prerequisite: CS106B

How can we use software to control hardware devices?

How do displays, keyboards, etc. get data into or out of the computer?

What’s it like to build a computer system from scratch?

The CS Core



Systems

CS106B
Programming Abstractions

CS107
Computer Organization and Systems

CS111
Operating Systems Principles

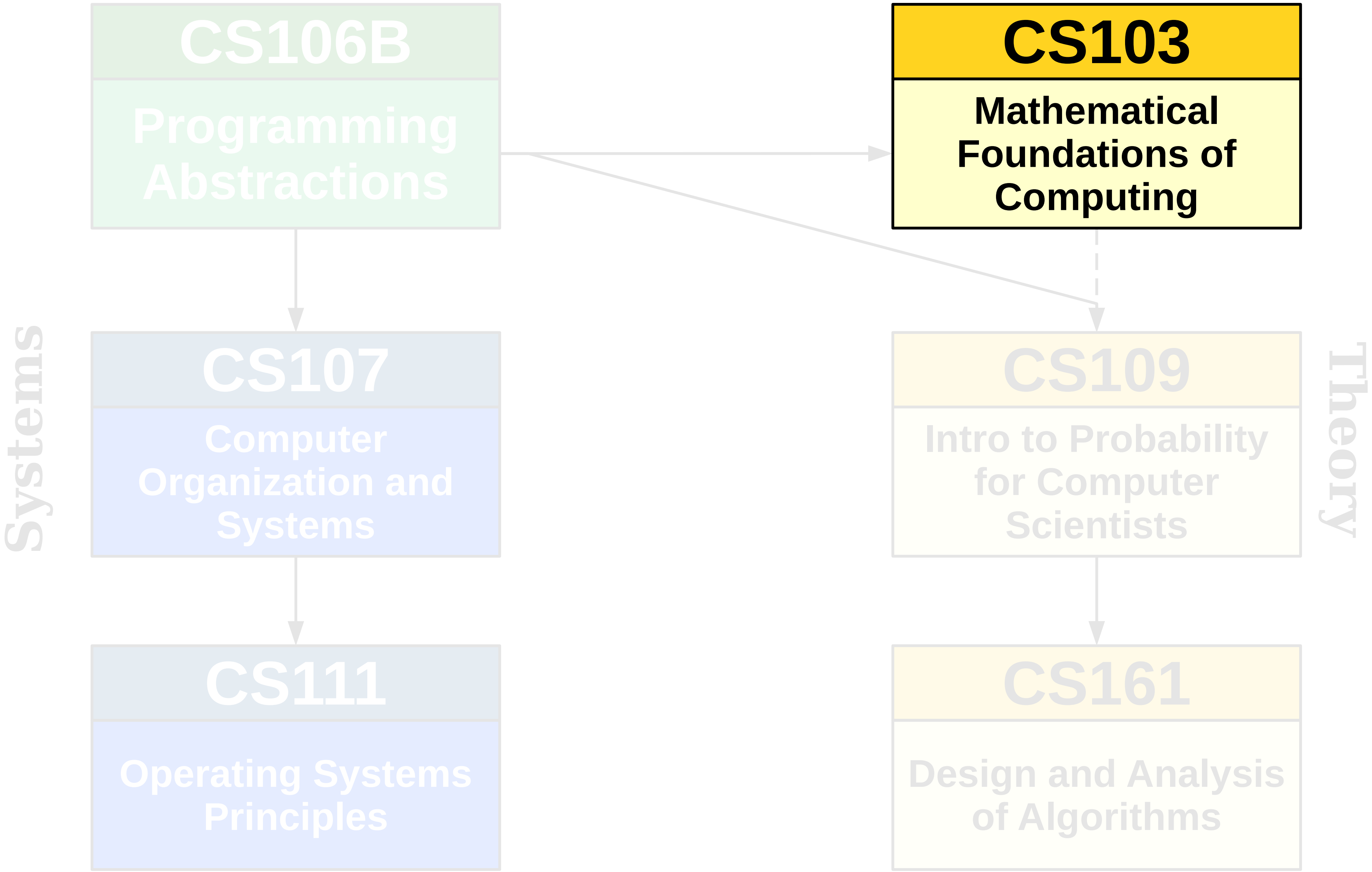
Theory

CS103
Mathematical Foundations of Computing

CS109
Intro to Probability for Computer Scientists

CS161
Design and Analysis of Algorithms

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
CS111
Operating Systems Principles
CS161
Design and Analysis of Algorithms

CS103

Mathematical Foundations of Computing

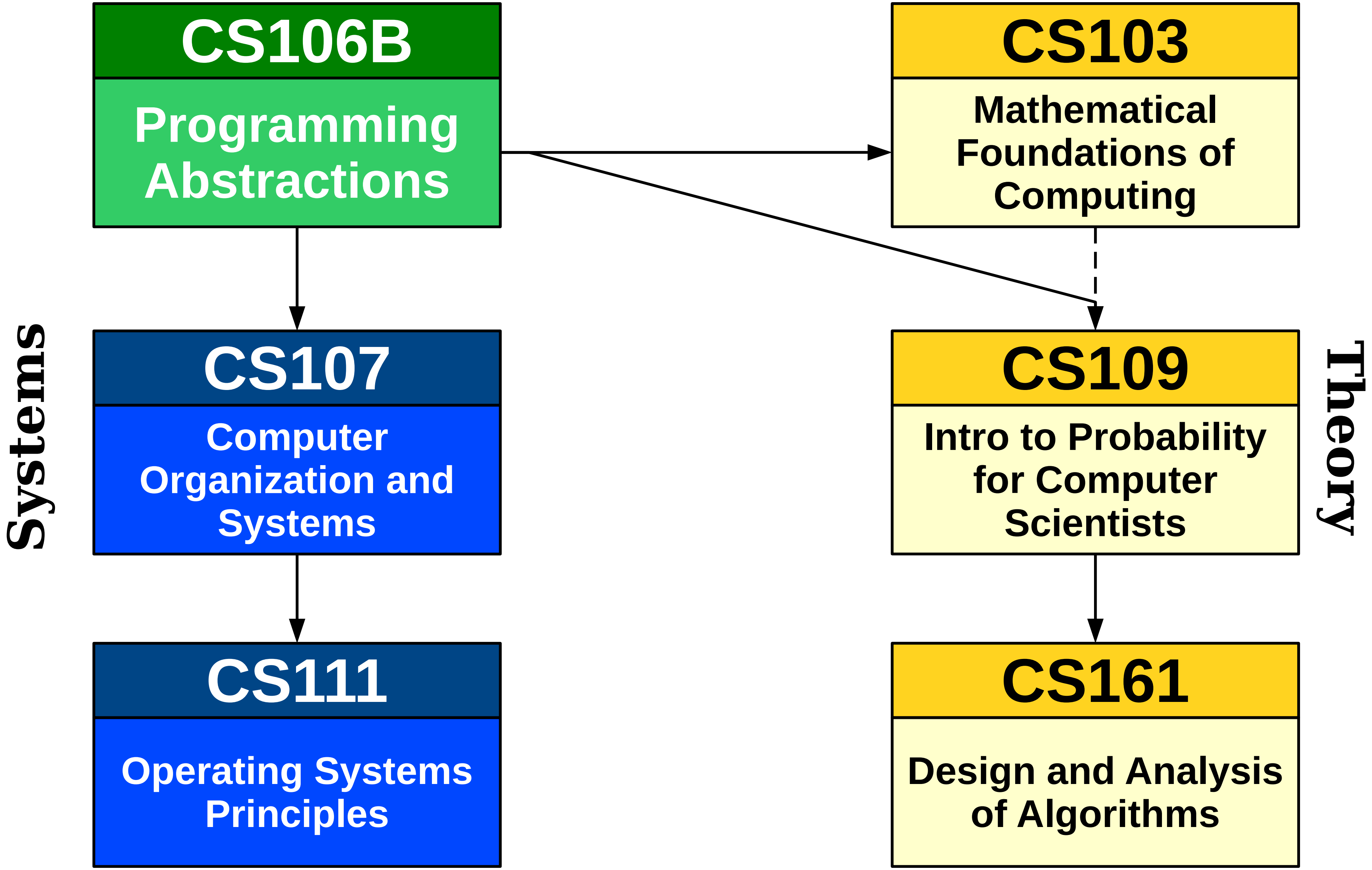
Corequisite: CS106B

What mathematical tools can we use to analyze programs, processes, and graphs?

Why are some problems harder to solve than others?

Are there problems that cannot be solved by computers, and how would we know?

The CS Core



Systems

CS106B
Programming Abstractions

CS107
Computer Organization and Systems

CS111
Operating Systems Principles

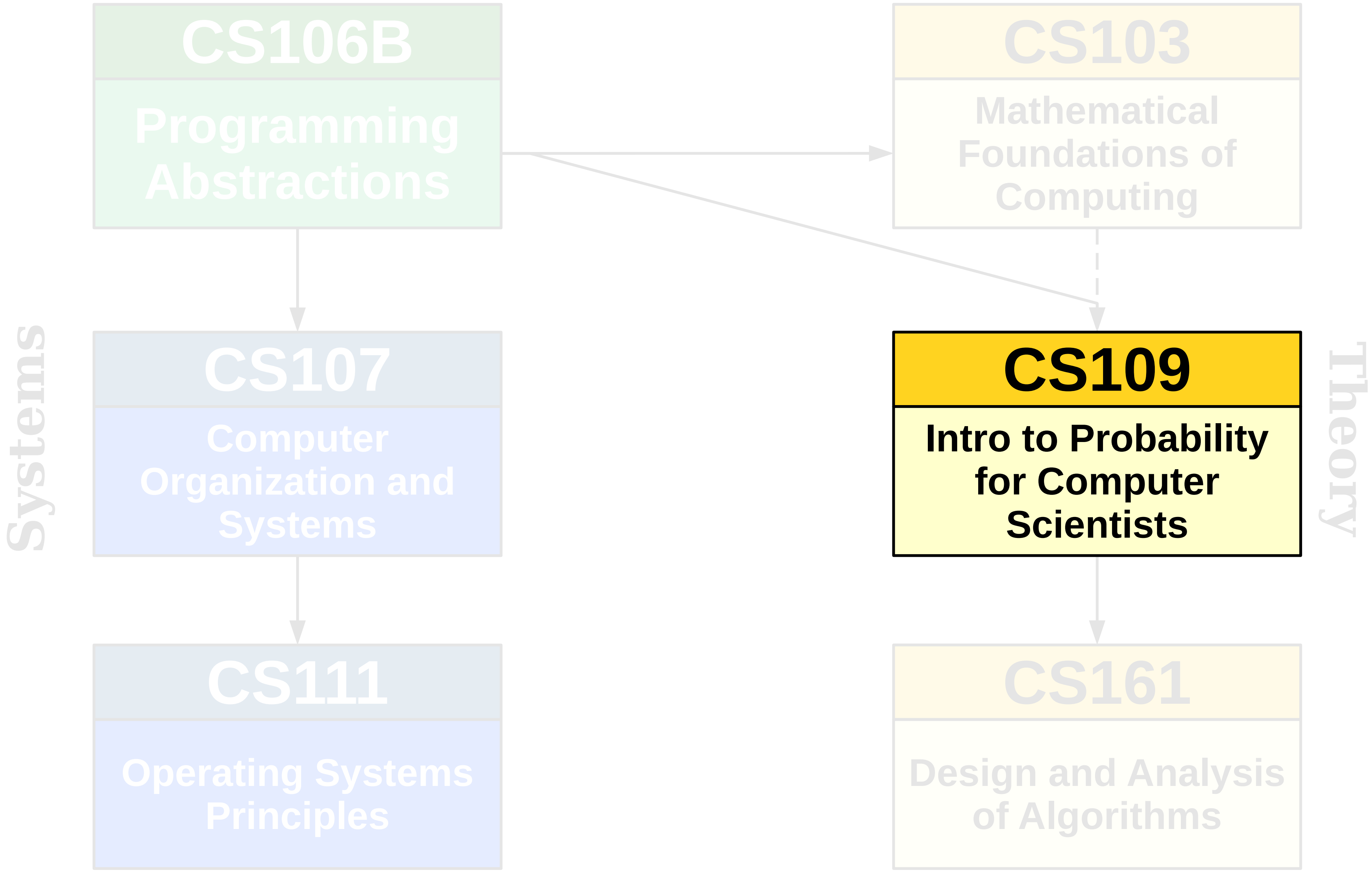
Theory

CS103
Mathematical Foundations of Computing

CS109
Intro to Probability for Computer Scientists

CS161
Design and Analysis of Algorithms

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
CS111
Operating Systems Principles
CS161
Design and Analysis of Algorithms

CS109

Probability for Computer Scientists

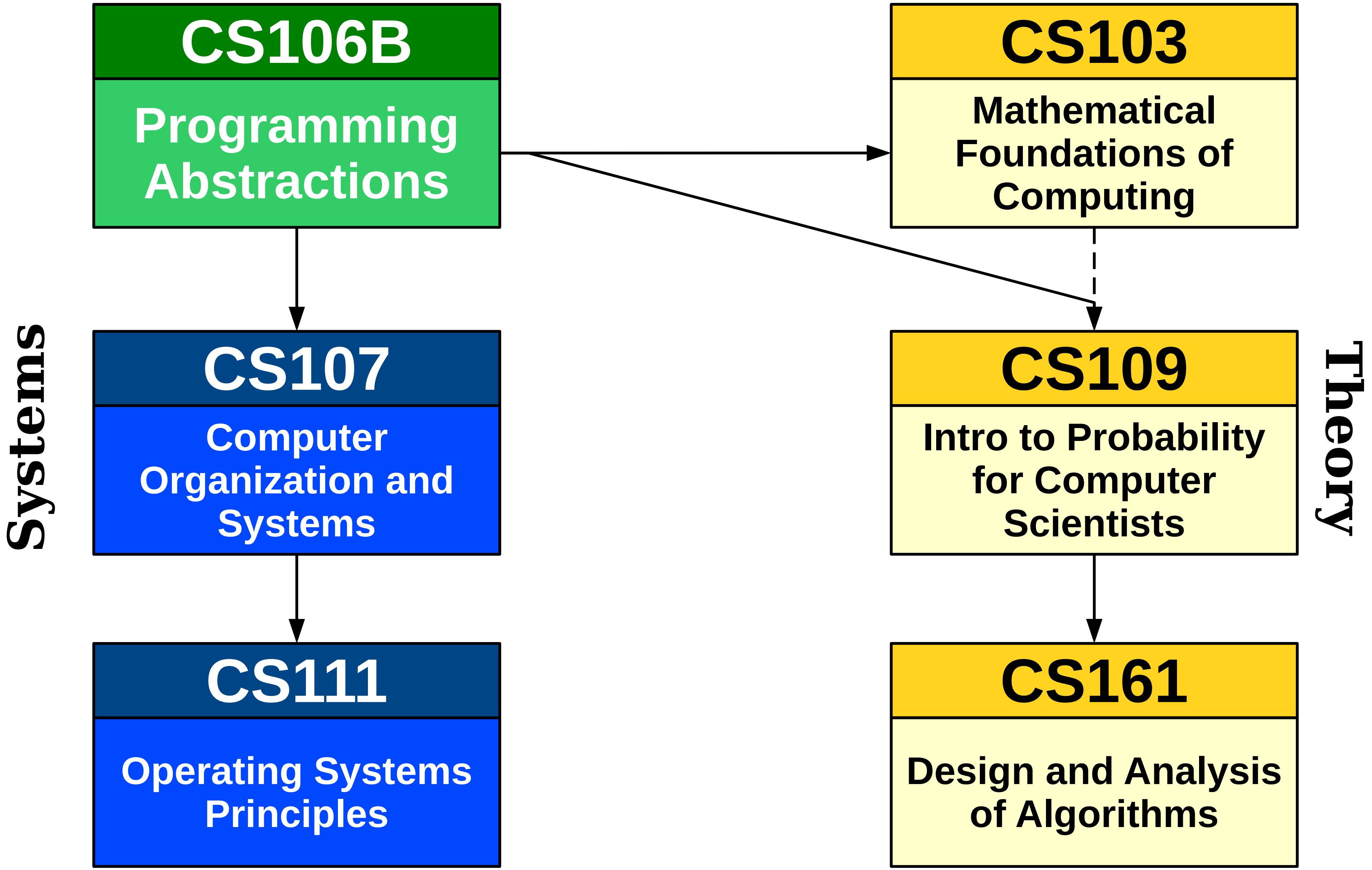
Prerequisites: CS106B, Math 51, “CS103”

Why is a randomly-built binary search tree probably balanced?

How do we use computers to make sense of large data sets?

What is machine learning, and how do machines learn?

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

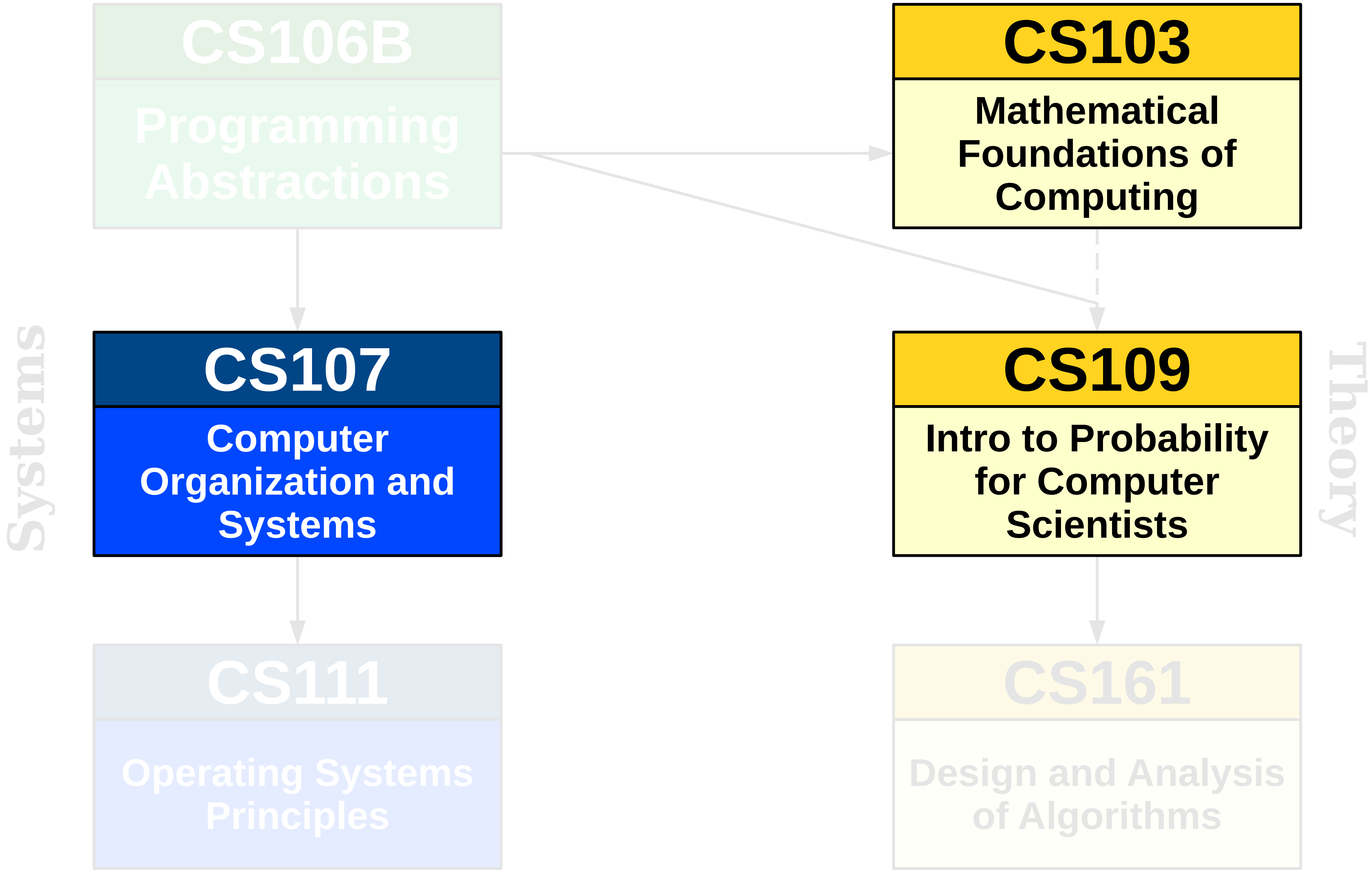
CS111

Operating Systems Principles

CS161

Design and Analysis of Algorithms

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

CS111

Operating Systems Principles

CS161

Design and Analysis of Algorithms

Next Steps in CS

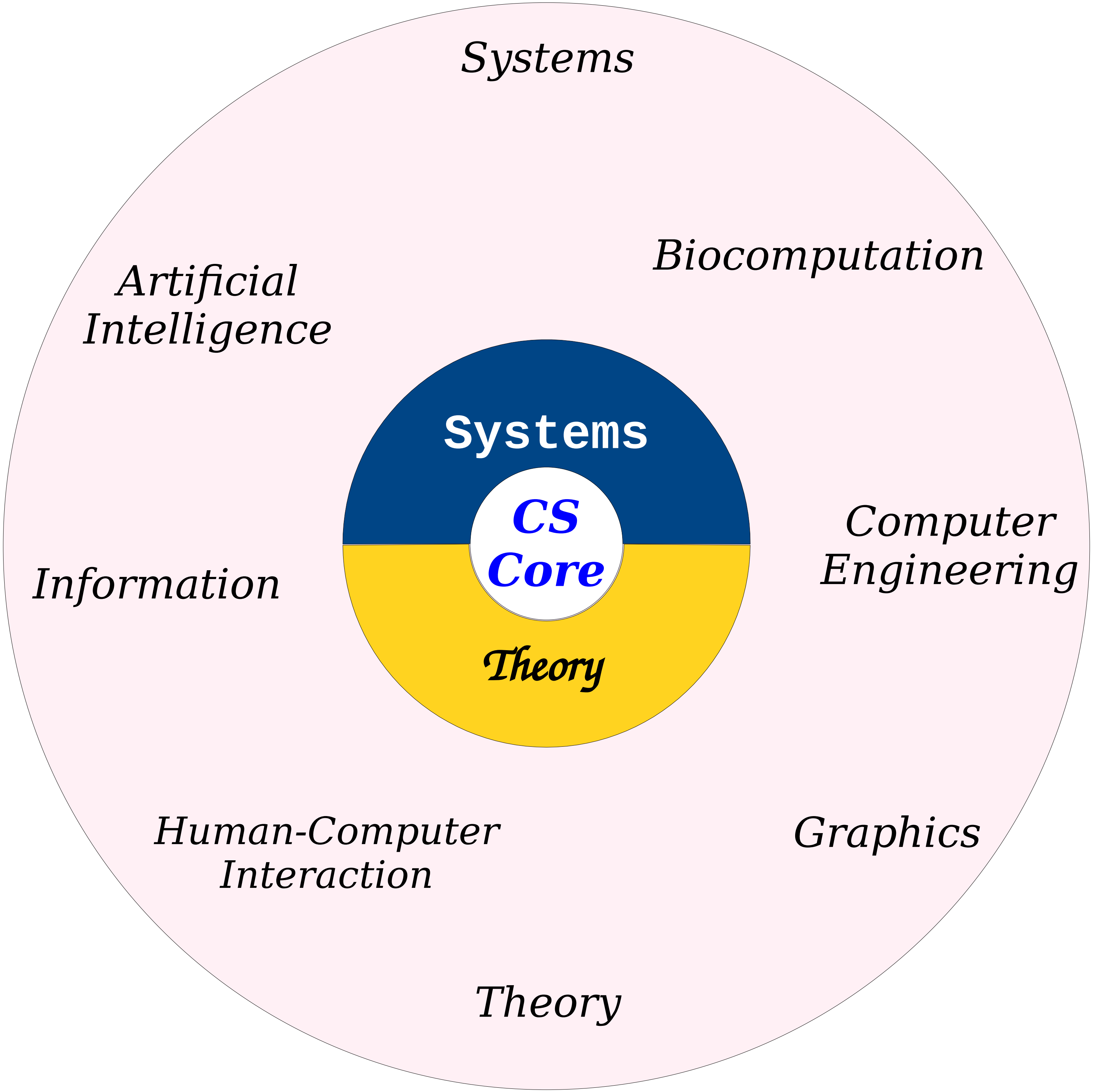
- It’s reasonable to take one of CS107, CS103, or CS109 as a next CS class. You’ll put in a good amount of work and learn a ton in the process.
- ***Do not feel pressured to do everything at once***. Taking two of these classes concurrently is a significant amount of work, and it isn’t expected of you.
- Want some more guidance? Come talk to me after class!

Other CS Classes to Consider

- You now have the prerequisites for each of these courses:
  - CS41: Python Programming
  - CS106E: Survey of Computer Science
  - CS106L: C++ Programming
  - CS147: Human-Computer Interaction
  - CS151: Logic Programming
  - CS182: Ethics, Pub. Policy, and Tech. Change
  - CS193X: Web Programming
  - CS274: Computational Biology
  - CS300: Survey of CS Research
  - CS522: AI in Healthcare
  - CS547: Survey of Human-Computer Interaction
- Come talk to me after class if you'd like to learn more!

The CS Major

https://cs.stanford.edu/degrees/undergrad/


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

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 – or I want to steer how technology is developed and used in the world.
- Bad reasons to think about not doing CS:
  - I really enjoy this, but other people are better coders than me.
  - I’m learning a lot, but other people have been doing this longer than me and there’s no way for me to catch up.
  - I like the classes I’m taking, but the field is so big and I have no idea which area to focus in.
  - I don’t know what I’m going to be doing many years down the line, and I don’t want to be pigeonholed into just a tech person.

The CS Coterm

https://csmajor.stanford.edu/academicz/masters/coterm-faq

What’s the Coterm?

- It’s a ***coterminal master’s degree***.
- Work concurrently on your BS (in any subject) and your MS (in computer science).
- Designed with two populations in mind:
  - Give existing CS majors access to more depth and breadth of knowledge.
  - Give non-CS majors a chance to explore CS and emerge with a thorough command of the material.
- All Stanford undergrads are welcome to apply. This is intentional, and the door is open to all comers!

The CS Minor

https://cs.stanford.edu/degrees/ug/Minor.shtml

What’s the CS Minor?

- Five classes in CS: take CS103, CS107, CS109, plus two other depth classes.
- Nice option if you want to keep exploring CS while pursuing another major.
- For more information, visit

https://cs.stanford.edu/degrees/ug/Minor.shtml

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 a huge impact.

How to Explore Them

- Online courses through Coursera, Udacity, edX, etc. are fantastic ways to learn new concepts.
    - 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.
    - Online resources like Stack Overflow can provide help (if you know how to ask questions well; that can take some practice!)

Some Words of Thanks

Who's Here Today?

- Aero/Astro
- Afro-American Studies
- Anthropology
- Art History
- Biochemistry
- Bioengineering
- Biology
- Biomedical Informatics
- Business
- Chemistry
- Civil/Env. Engr
- Classics
- Creative Writing
- Comparative Lit
- CSRE
- Computer Science
- CME
- Earth Systems
- Economics
- Education
- Electrical Engineering
- Energy Resources
- Epidemiology
- Human Biology
- Immunology
- International Policy
- Intl. Relations
- Latin Amer. Studies
- Law
- Mech. Engineering
- MS&E
- Neuroscience
- Physics
- Psychology
- Public Policy
- Statistics
- TAPS
- ***Undeclared!***
- Urban Studies

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?