Where to Go from Here

Announcements

- Assignment 6 due right now.
  - Due Friday at 2:15PM with one late day.
  - Due Monday at 2:15PM with two late days.
- Assignment 7 out, due next Wednesday at **12:15PM** (note that this is not the usual time.)
  - **No late days may be used.**
  - **No late submissions accepted.**
- The LaIR will be staffed this week and next Sunday through Tuesday from 7PM – 11PM.
- Dawson and I will hold our normal office hours next Monday and Tuesday.

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.

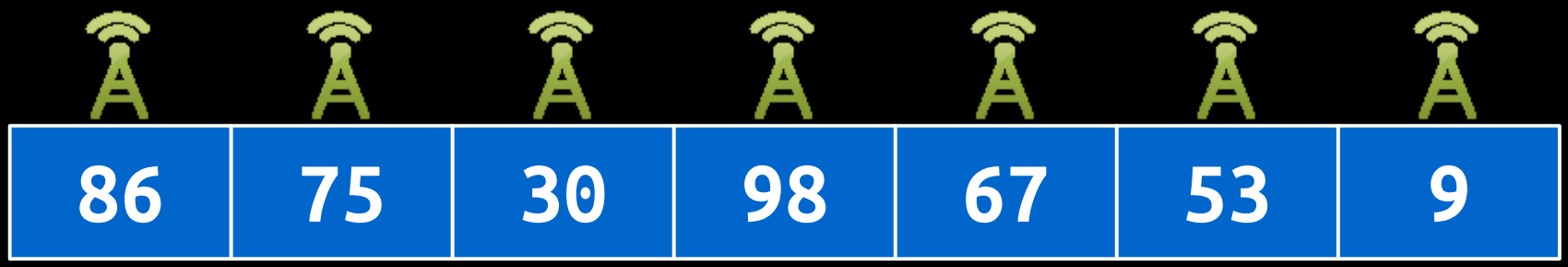
How do we model real-world problems in software?

How do we model real-world problems in software?

We're going to need a way to model **sequences of data**.

So let's study stacks, queues, and vectors.

15 - 3 + 4 * 5 / 6


86
75
30
98
67
53
9

```
username: htiek
password: **********
```

What about associative data?
Or unordered data?

This is where maps, sets,
and lexicons come in.

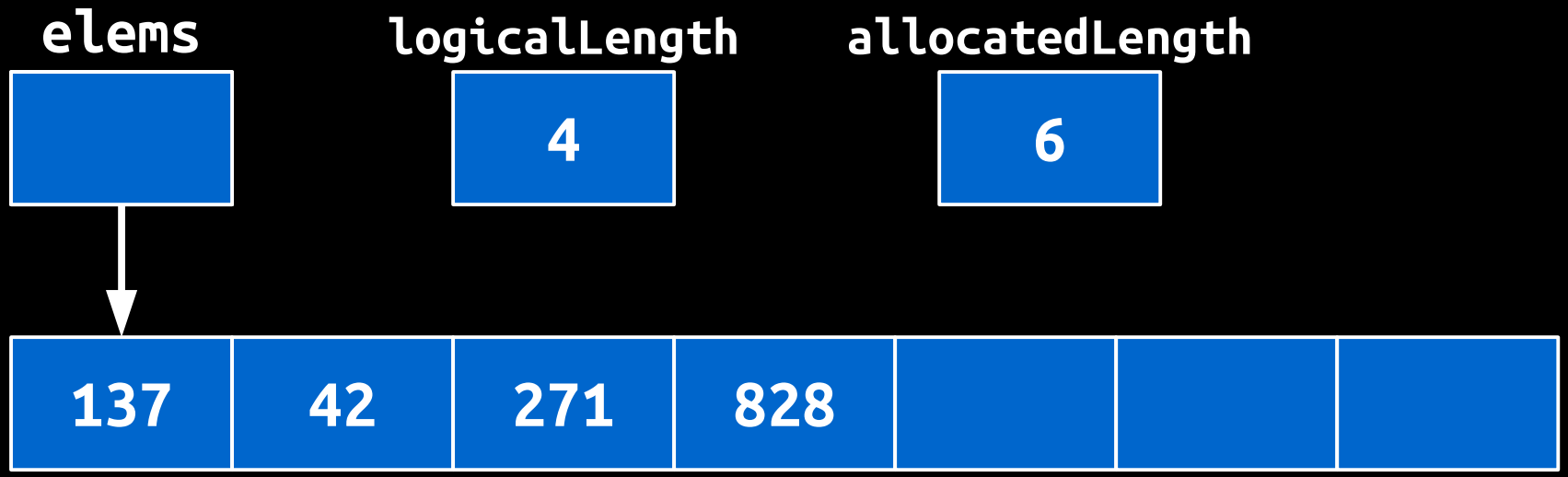
dik·dik

opts
post
pots
spot
stop
tops

How do we model **networks**?

We can use graphs, which we can build out of those tools we just saw!

But how does all of this work?

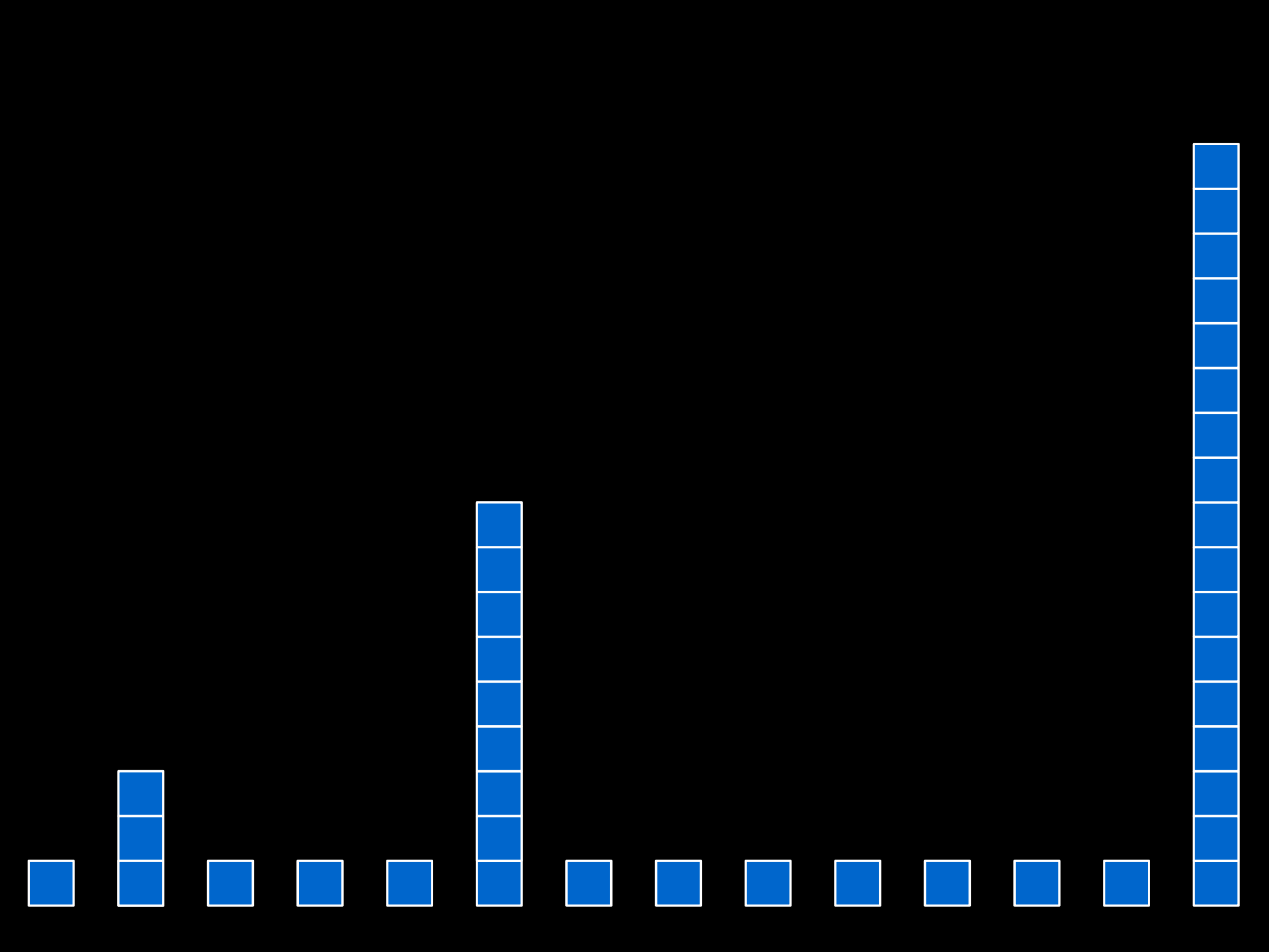

elems
logicalLength
allocatedLength
4
6
137
42
271
828


Dikdik
Quokka
Nubian Ibex

We need a way to compare solutions.

So let's invent big-O notation.

So... how do they compare?

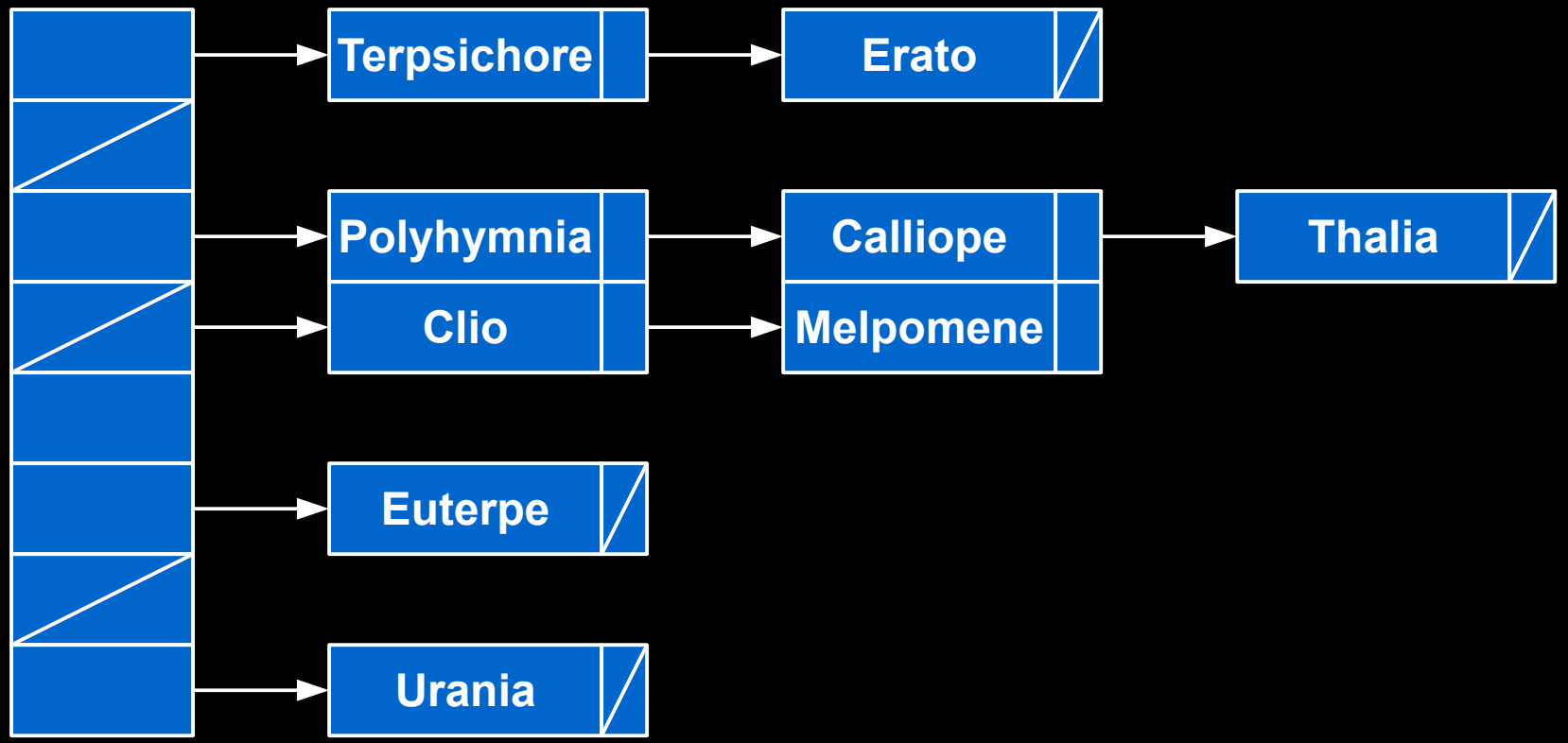


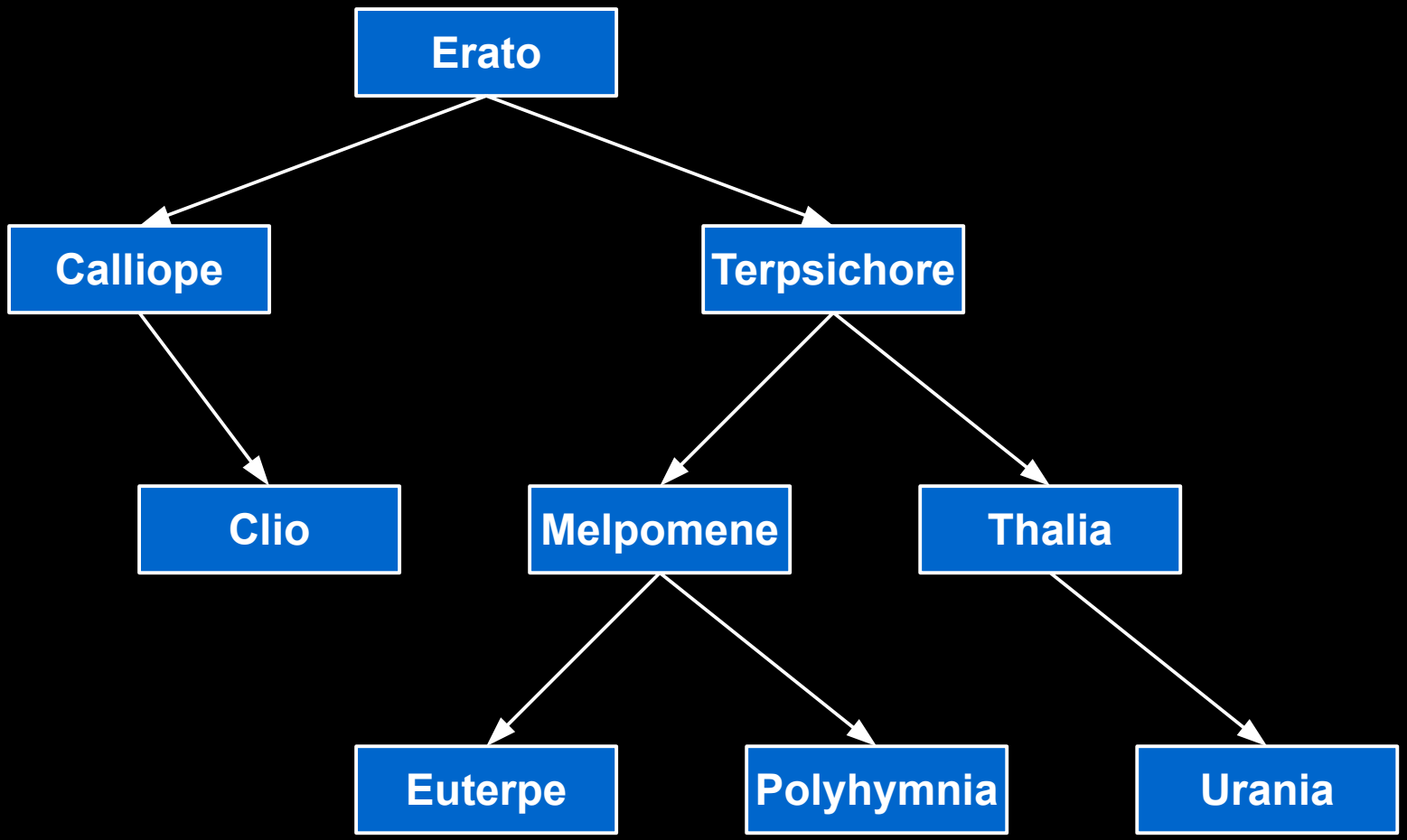

Dikdik
Quokka


Dikdik
Quokka
Nubian Ibex

Interesting! Neither of the implementations is strictly better than the other!

How might we make maps and sets?


Terpsichore
Erato
Polyhymnia
Calliope
Thalia
Clio
Melpomene
Euterpe
Urania


Erato
Calliope
Terpsichore
Clio
Melpomene
Thalia
Euterpe
Polyhymnia
Urania

As before, neither of these structures is clearly better than the other!

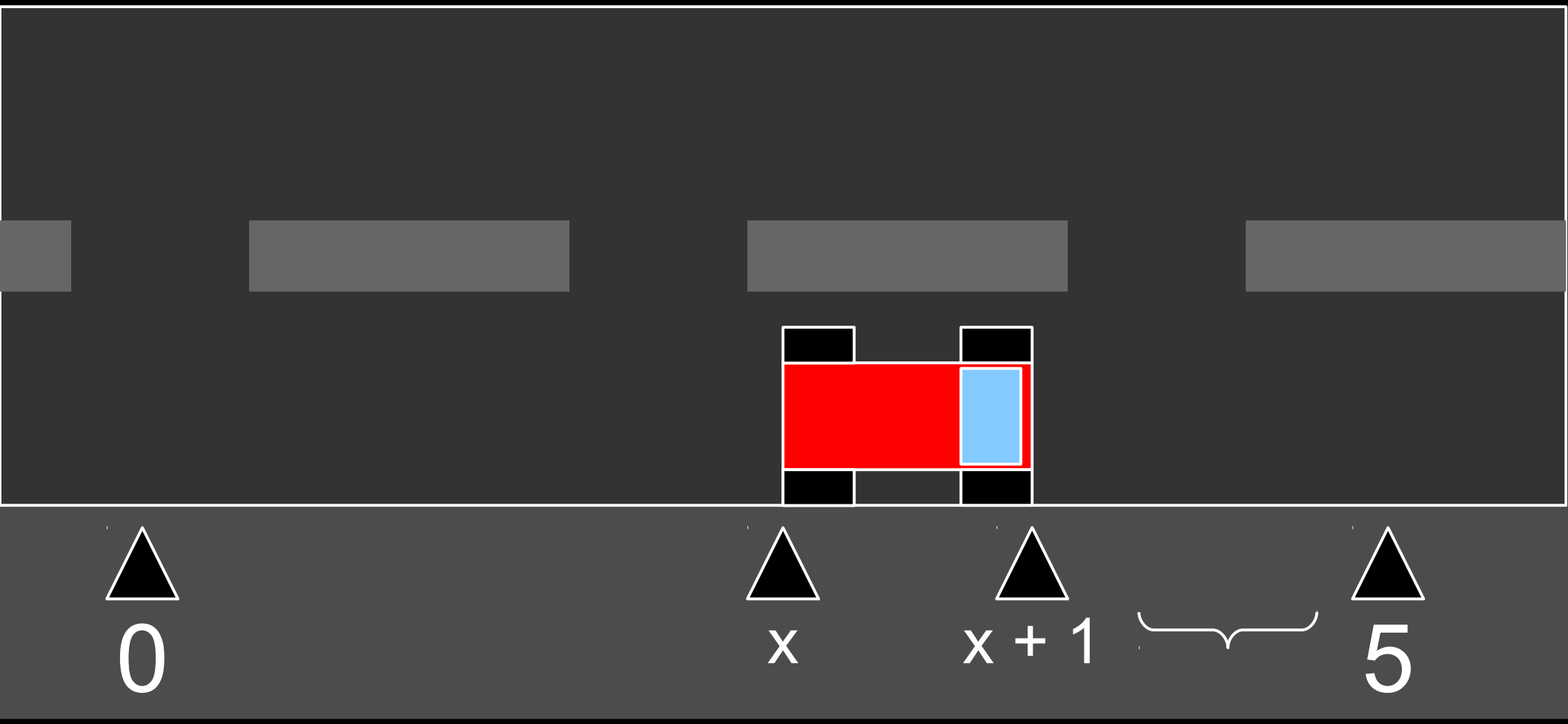
This hits on a key point:
Engineering is all about trade-offs.

Cool! Now we have a bunch of tools for modeling problems!

So how do we go about solving them?


1
2
5
8


1
2
5
8


0
x
x + 1
5

Nifty! Some problems are self-similar!

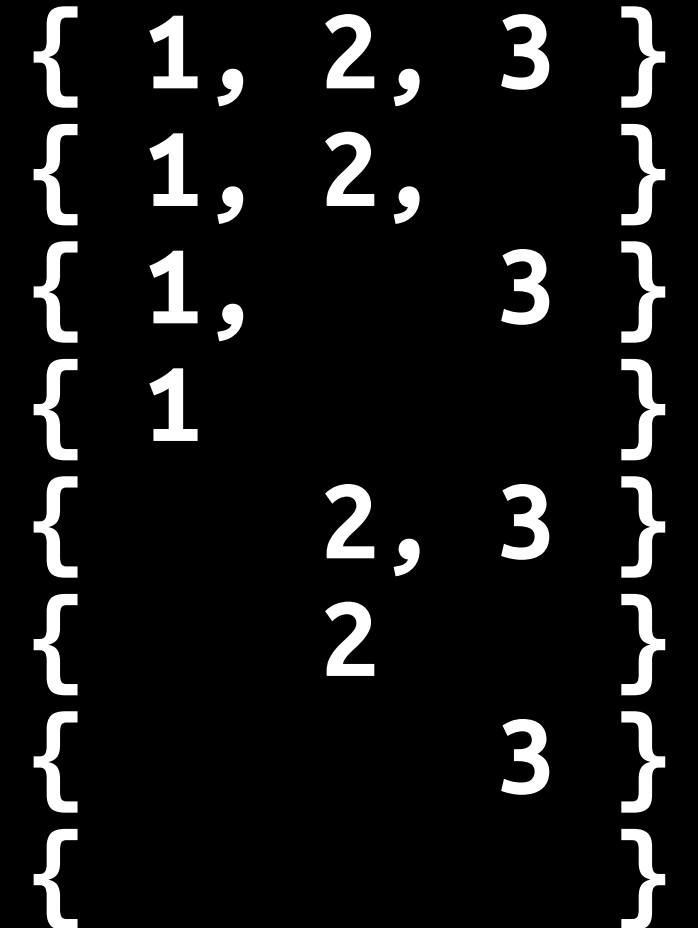
So how do we solve them?

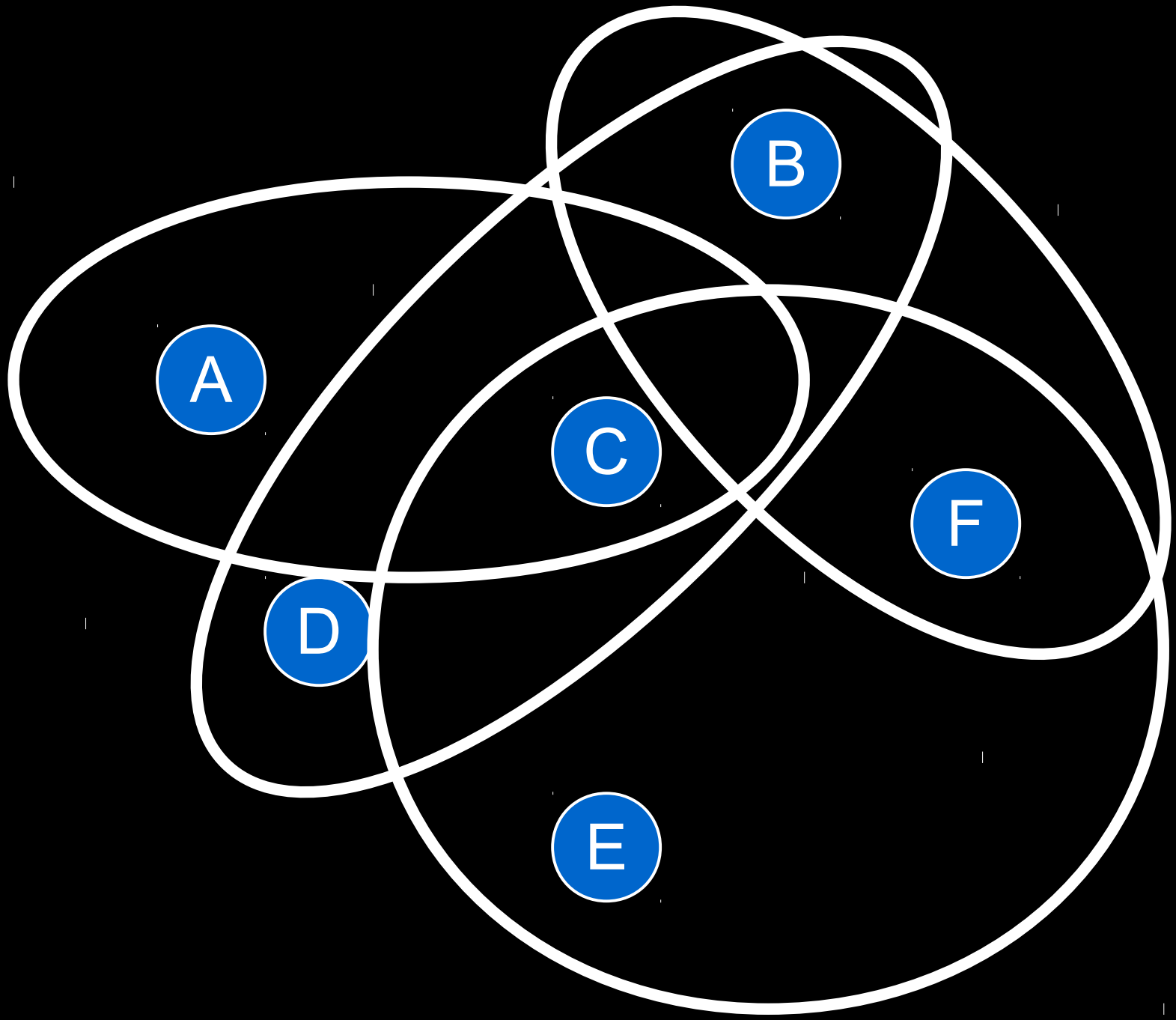
```
int sumOfDigits(int n) {
  if (n < 10)
     return n;
  else
     return (n % 10) + sumOfDigits(n / 10);
}
```

```
void moveTower(int n, char from,
               char temp, char to) {
    if (n > 0) {
        moveTower(n - 1, from, to, temp);
        moveDisk(from, to);
        moveTower(n - 1, temp, to, from);
    }
}
```

Amazing! All we need to do is figure out a base case and a recursive step.

What else can we do with recursion?


{ 1, 2, 3 }
{ 1, 2, }
{ 1, 3 }
{ 1 }
{ 2, 3 }
{ 2 }
{ 3 }
{ }


B
A
C
F
D
E

aahed
abaci
haole
eclat
diets

Self-similarity is **everywhere**!

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

The skills you have just learned will follow you through the rest of your programming career.

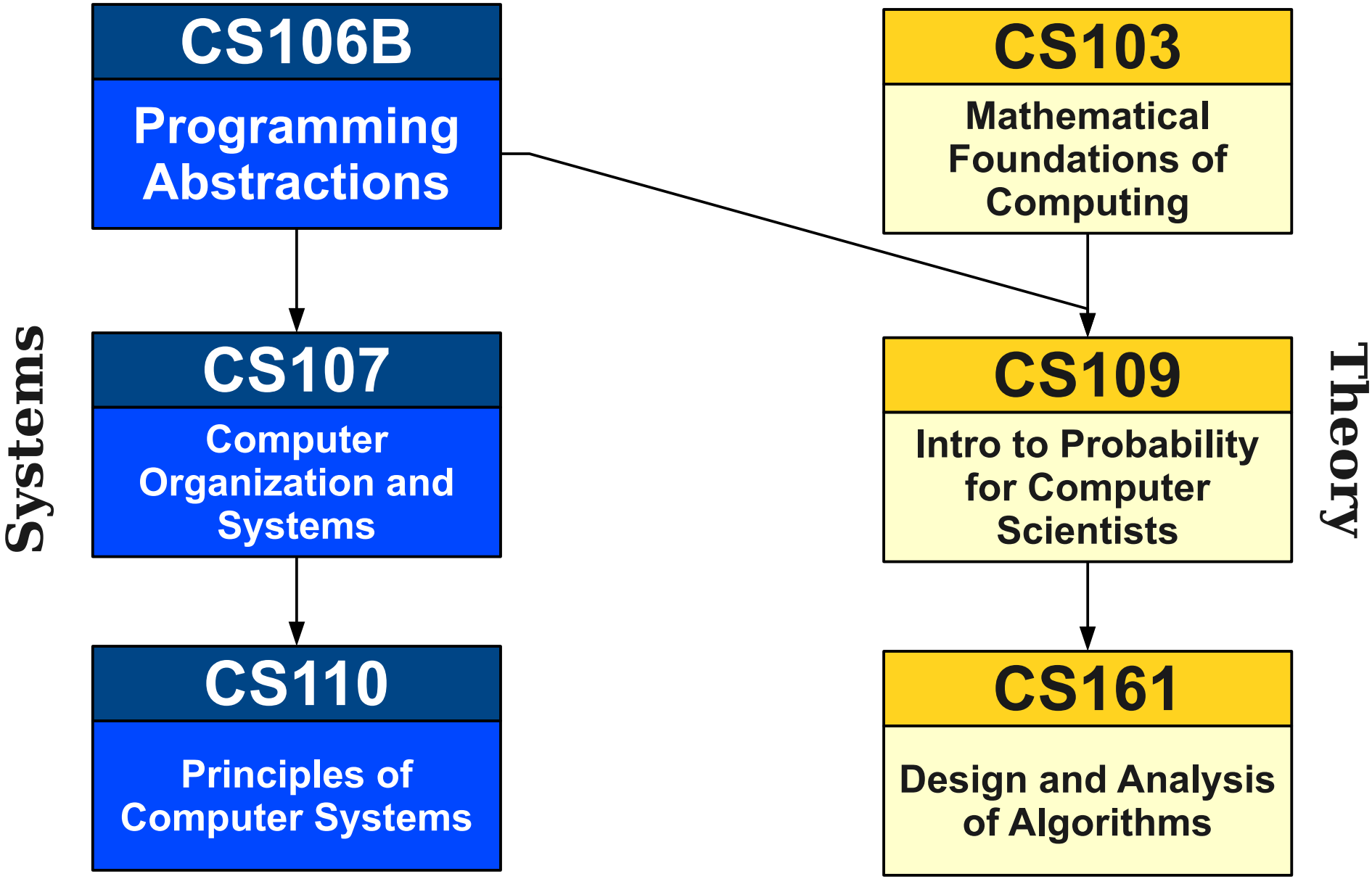
Programming is all about exploring new ways to model and solve problems.

The skills you have just learned will follow you through the rest of your programming career.

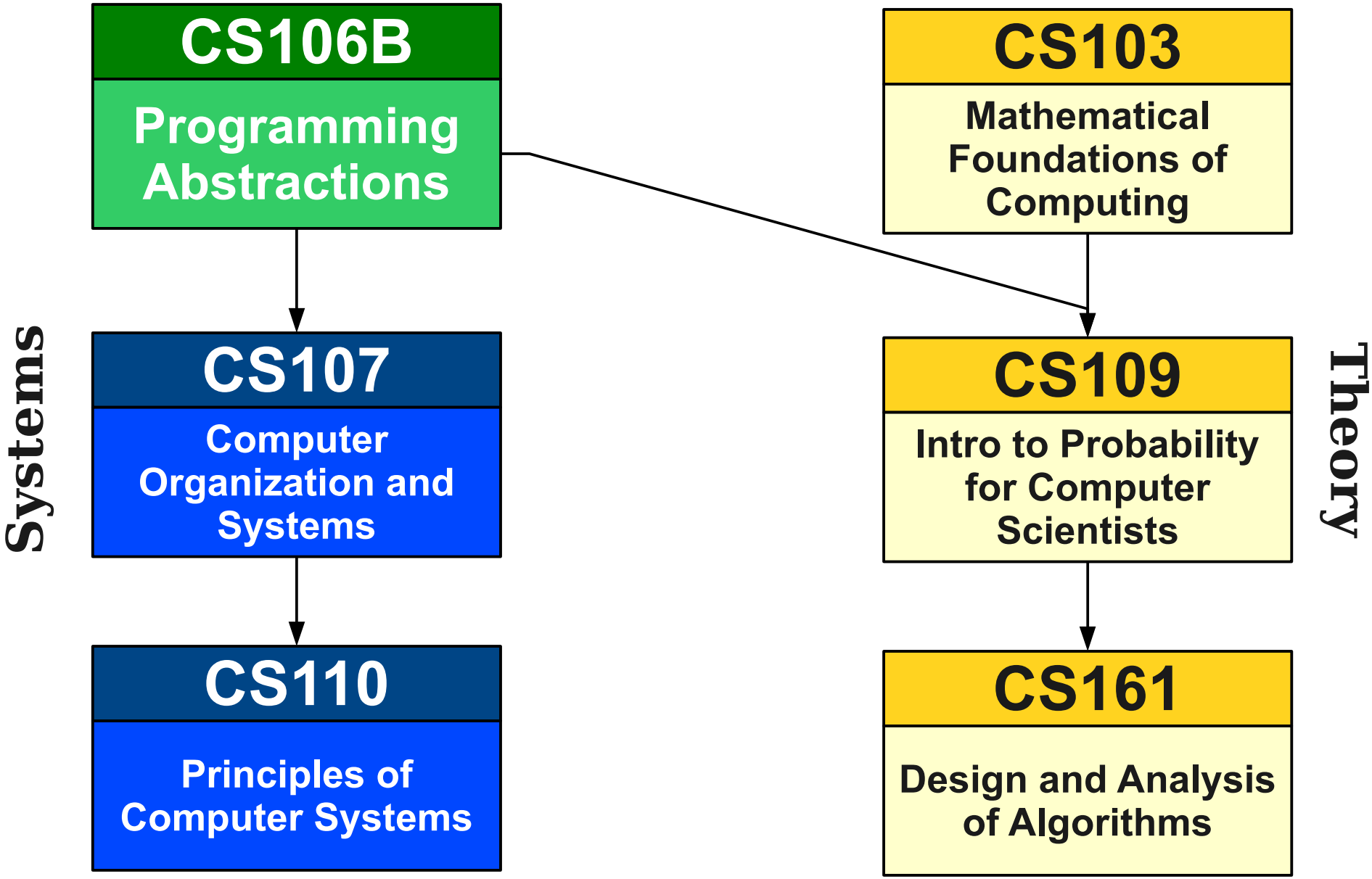
So what comes next?

Courses to Take

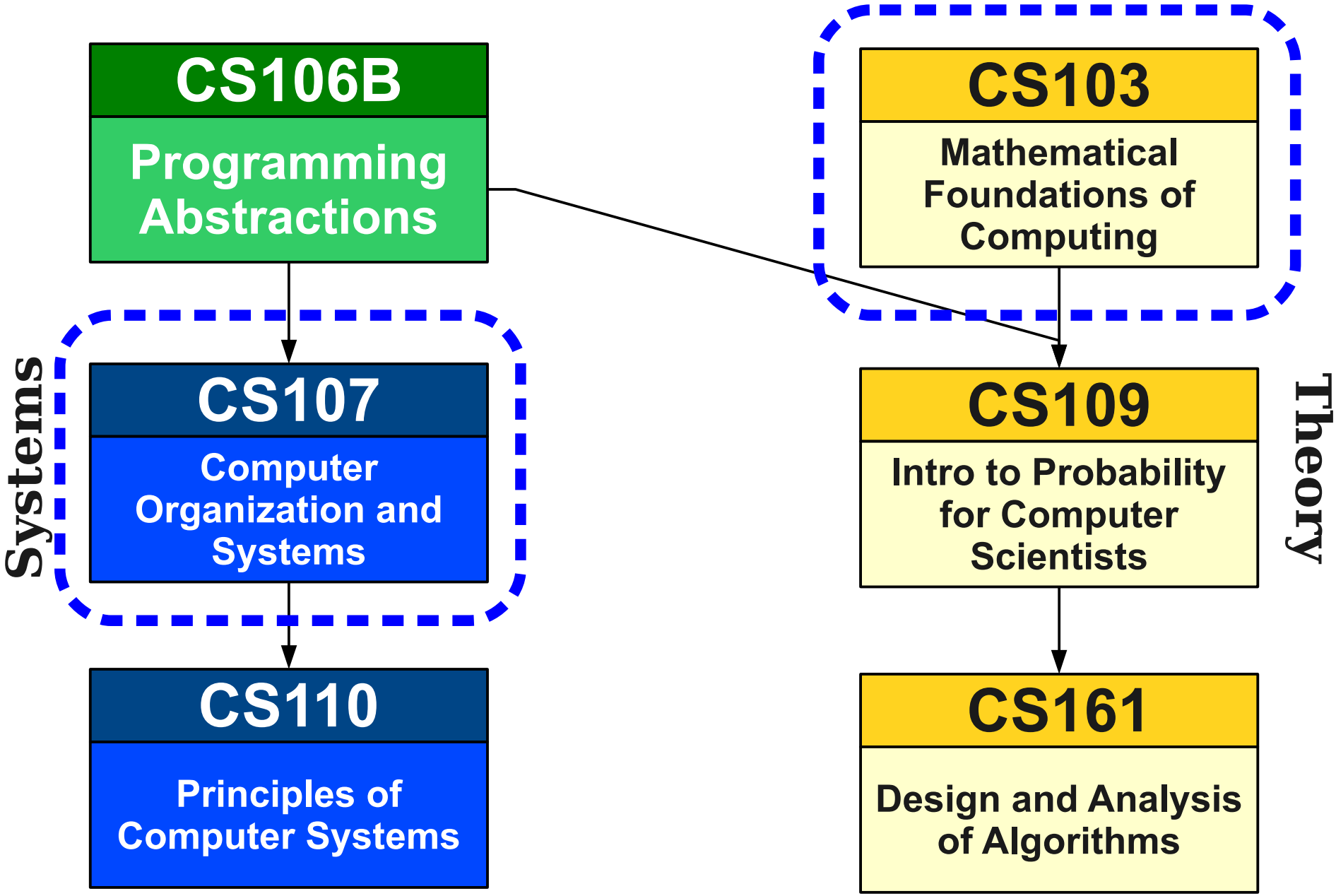
The CS Core

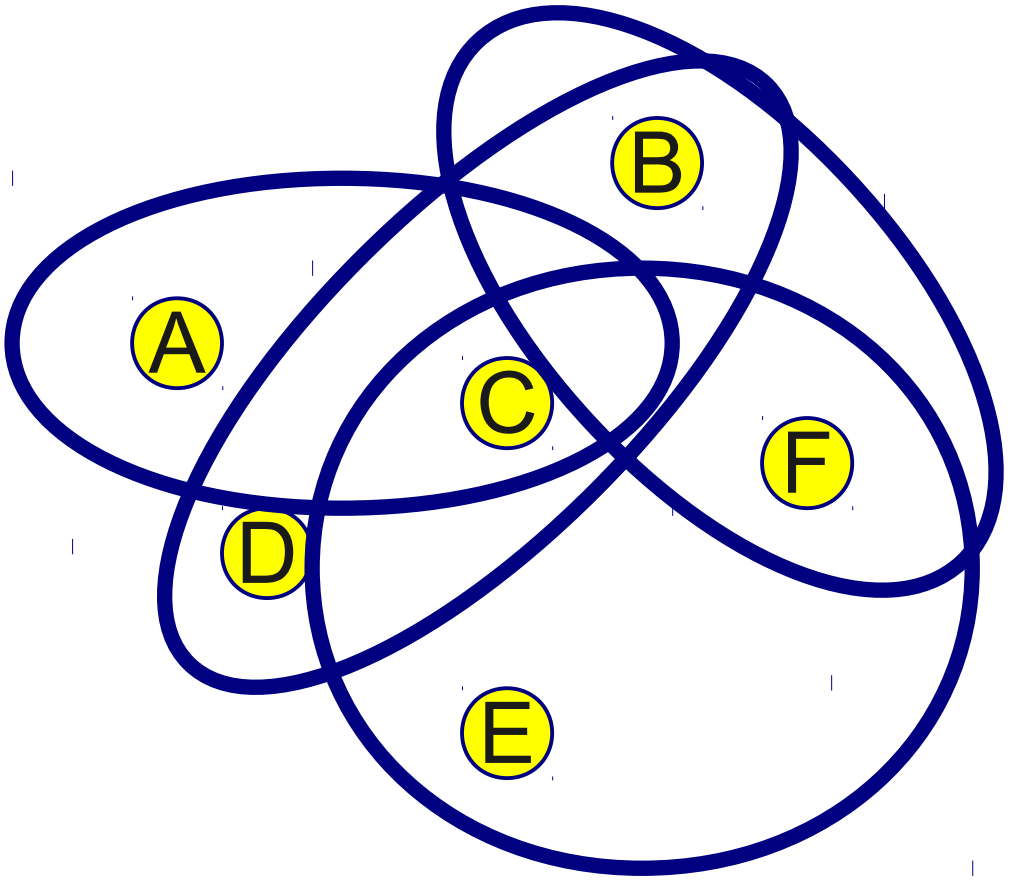

CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms
Systems
Theory

The CS Core

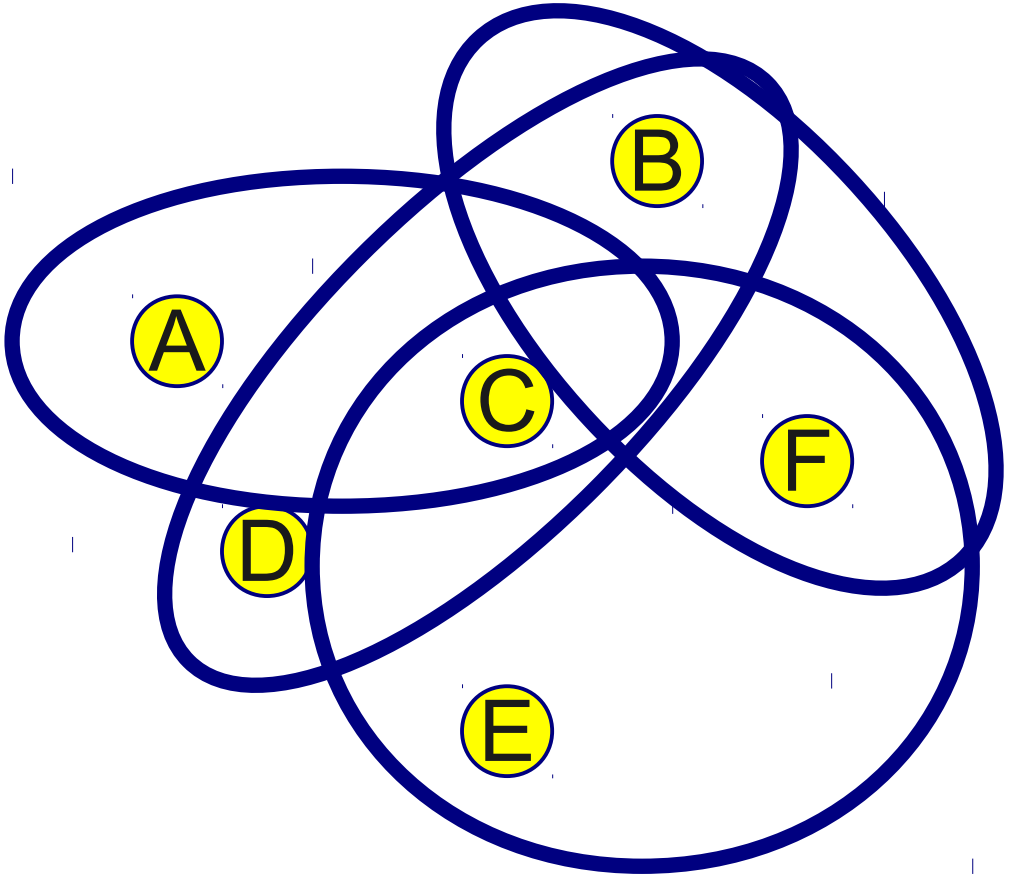

CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms
Systems
Theory

The CS Core

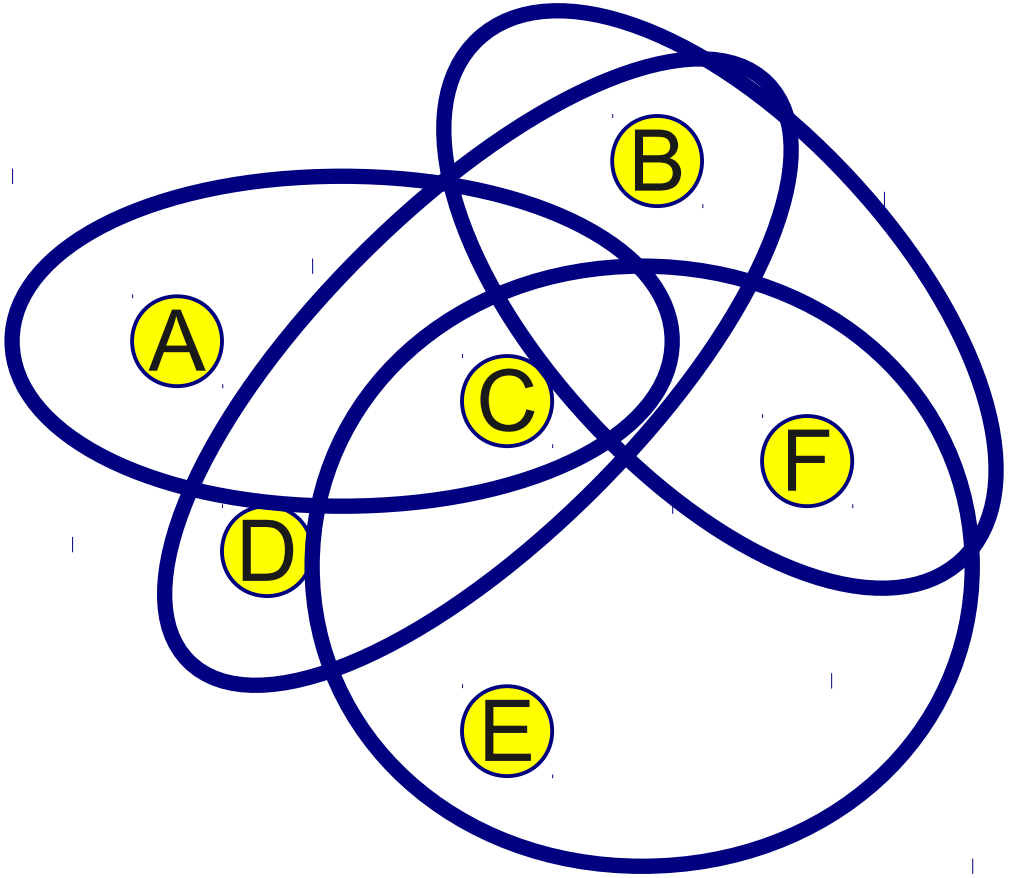

CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
Systems
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
Theory
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms


A
B
C
D
E
F

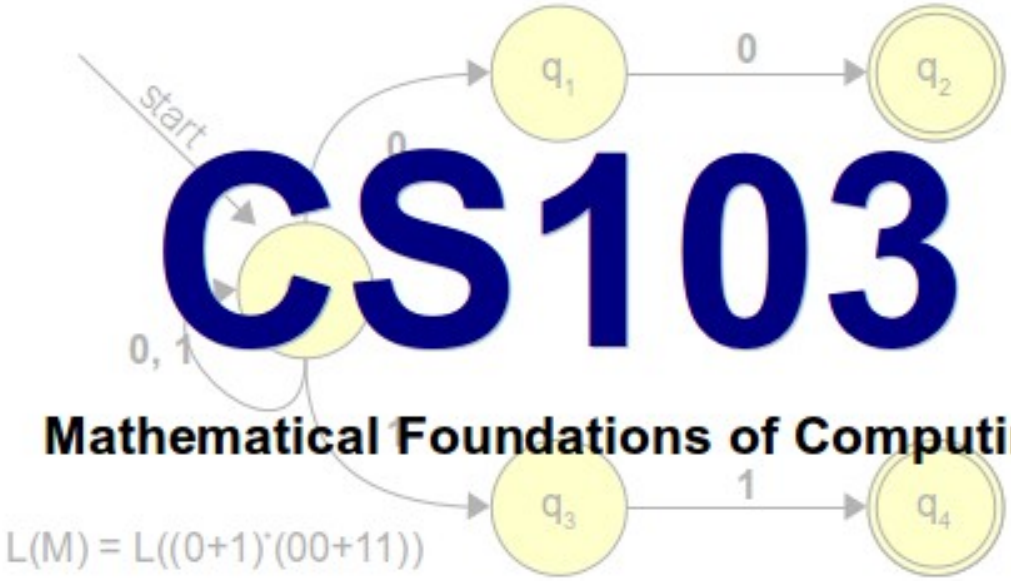
What is the most efficient algorithm to solve this problem?



What is the most efficient algorithm to solve this problem?



No one knows!



CS103

Mathematical Foundations of Computing

Can computers solve all problems?

Why are some problems harder than others?

How can we be certain about this?

It's all Bits and Bytes!

K	01001011
I	01001001
R	01010010
K	01001011
'	00100111
S	01010011
	00100000
D	01000100
I	01001001
K	01001011
D	01000100
I	01001001
K	01001011

It's all Bits and Bytes!

K	01001011
I	01001001
R	01010010
K	01001011
'	00100111
S	01010011
	00100000
D	01000100
I	01001001
K	01001011
D	01000100
I	01001001
K	01001011

```
01001011010010010101001001001011
00100111010100110010000001000100
01001001010010110100010001001001
01001011
```

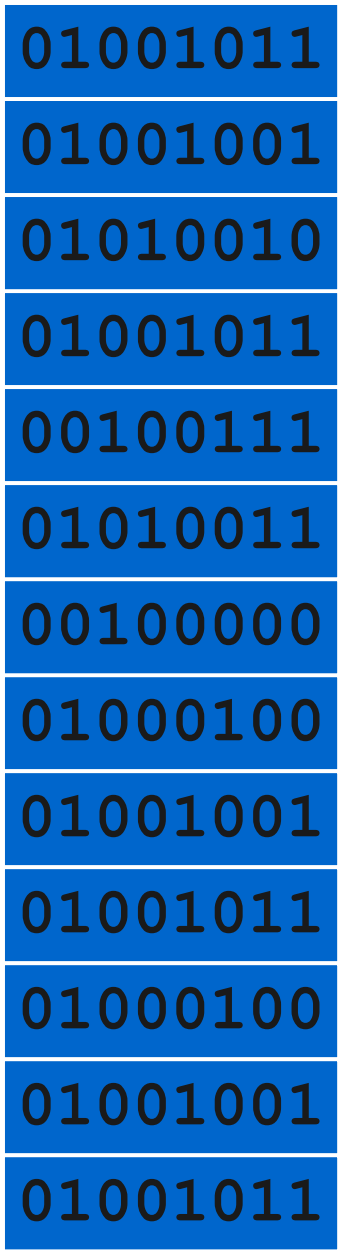
It's all Bits and Bytes!

```
01001011010010010101001001001011
00100111010100110010000001000100
01001001010010110100010001001001
01001011
```

It's all Bits and Bytes!

01001011	01001001	01010010	01001011
00100111	01010011	00100000	01000100
01001001	01001011	01000100	01001001
01001011			

It's all Bits and Bytes!



It's all Bits and Bytes!

K	01001011
I	01001001
R	01010010
K	01001011
'	00100111
S	01010011
	00100000
D	01000100
I	01001001
K	01001011
D	01000100
I	01001001
K	01001011

CS107
Computer Organization and Systems

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?

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 hard. It does not eat kittens.
- *Don't be afraid to try CS107!*

Other CS Courses

CS181

Computers, Ethics, and Public Policy

- Some sample news headlines from the New York Times this year:
  - Google Concedes That Drive-By Prying Violated Privacy.
  - Cyberattacks Against U.S. Corporations Are on the Rise.
- Today's Daily: “MOOCs Face Challenge in Humanities.”
- ***We have the technology, but how should we use it?***

CS108

Object-Oriented Systems Design

- ***How do you build large software systems in a team?***
- Introduction to
  - Unit-testing frameworks
  - Object-oriented design.
  - Multithreaded applications.
  - Databases and web applications.
  - Source control.
- Excellent if you're interested in learning industrial programming techniques.

CS193

- Many 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
- Great for learning particular technologies.

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?***

The CS Minor

The CS Coterm

Outside Stanford

Learning More

- MOOCs: Coursera, Udacity, and edX all offer CS courses.
  - e.g. Check out Udacity's Web Development course at https://www.udacity.com/course/cs253.
- Summer programs: Hacker School, Hackbright Academy, etc. offer intensive workshops.
- Explore and play around! There are great resources for getting unstuck on the internet.
  - e.g. Stack Overflow (http://www.stackoverflow.com) is a place to ask for help on programming-related questions.
- Read a book! There are great books that introduce most programming languages and frameworks.

Taking Classes

- Many community colleges have great courses in software engineering.
- Want to come back to the farm? Take classes through SCPD!
  - (They did not pay me to say that. I promise.)

Some Words of Thanks

Who's Here Today?

- African Studies
- Applied Physics
- Bioengineering
- Biology
- Business Administration
- Chemical Engineering
- Chemistry
- Classics
- Civil and Environmental Engineering
- Computational and Mathematical Engineering
- Computer Science
- Creative Writing
- East Asian Studies
- Economics
- Education
- Electrical Engineering
- Energy Resource Engineering
- English
- Financial Mathematics
- Film and Media Studies
- French
- History
- International Relations
- Japanese
- Law
- Materials Science and Engineering
- Mathematical and Computational Sciences
- Mathematics
- Mechanical Engineering
- Medicine
- Management Science and Engineering
- Modern Language
- Music
- Neuroscience
- Physics
- Political Science
- Psychology
- Science, Technology, and Society
- Statistics
- Symbolic Systems
- Undeclared!

My Email Address

htiek@cs.stanford.edu

You're ready to take on big, important challenges with the skills you've just learned.

Best of luck wherever they take you!