Welcome to CS106B!

- Four Handouts
- Today:
  - Course Overview
  - Where are We Going?
  - Introduction to C++

Who's Here Today?

- Biochemistry
- Bioengineering
- Biology
- Business
- Chemical Engineering
- Chemistry
- Classics
- Civil and Environmental Engineering
- CME
- Computer Science
- Earth Systems
- Economics
- Electrical Engineering
- Energy Resources Engineering
- Environmental Engineering
- Ethics in Society
- Geological and Environmental Sciences
- History
- Linguistics
- Materials Science
- Mathematical and Computational Sciences
- Mathematics
- Mechanical Engineering
- MS&E
- Musics
- Physics
- Political Science
- Psychology
- Science, Technology, and Society
- Statistics
- Structural Biology
- Symbolic Systems
- Undeclared!

Course Staff

Instructor: Keith Schwarz
(htiek@cs.stanford.edu)

Head TA: Zach Galant
(galant@cs.stanford.edu)

The CS106B Section Leaders
The CS106B Course Helpers

Course Website

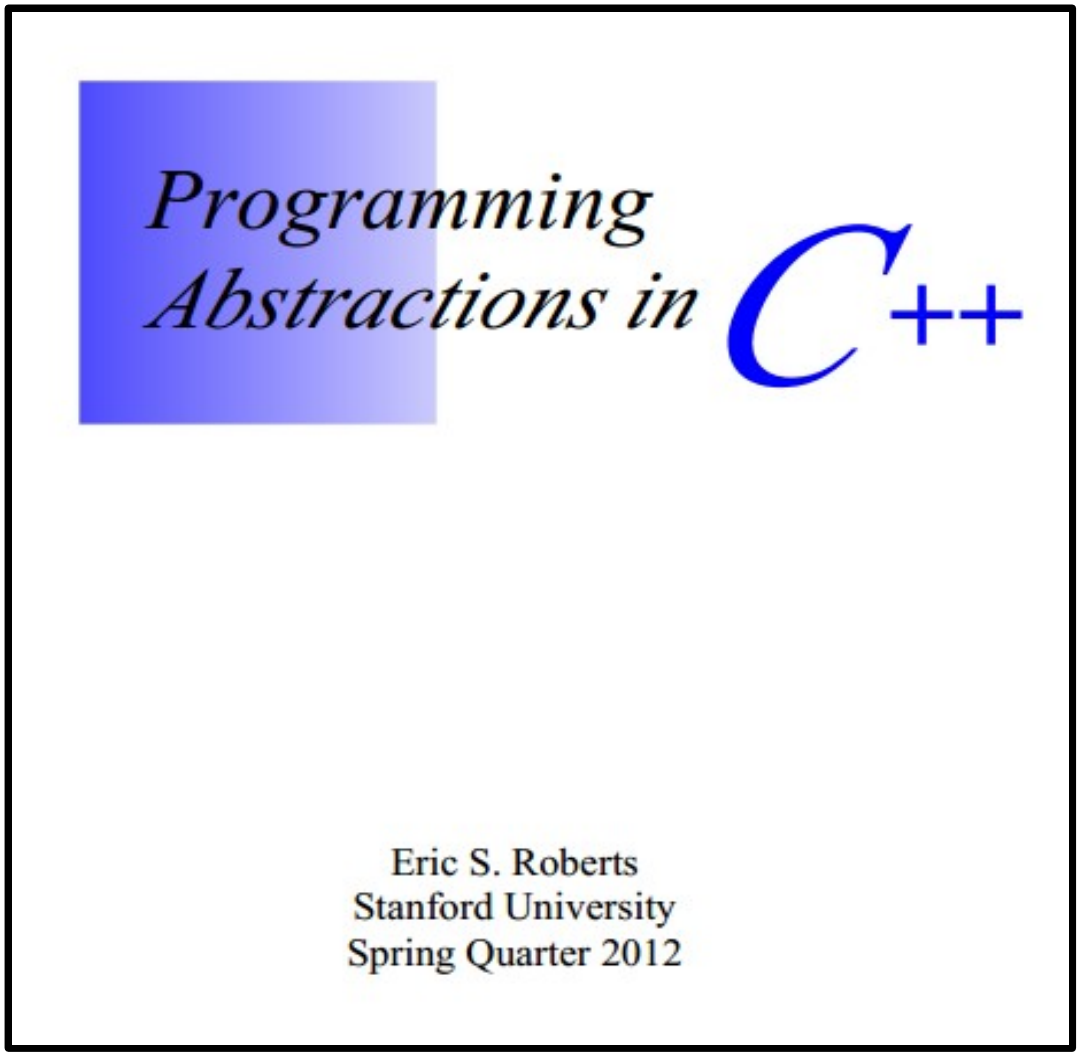
http://cs106b.stanford.edu

Prerequisites

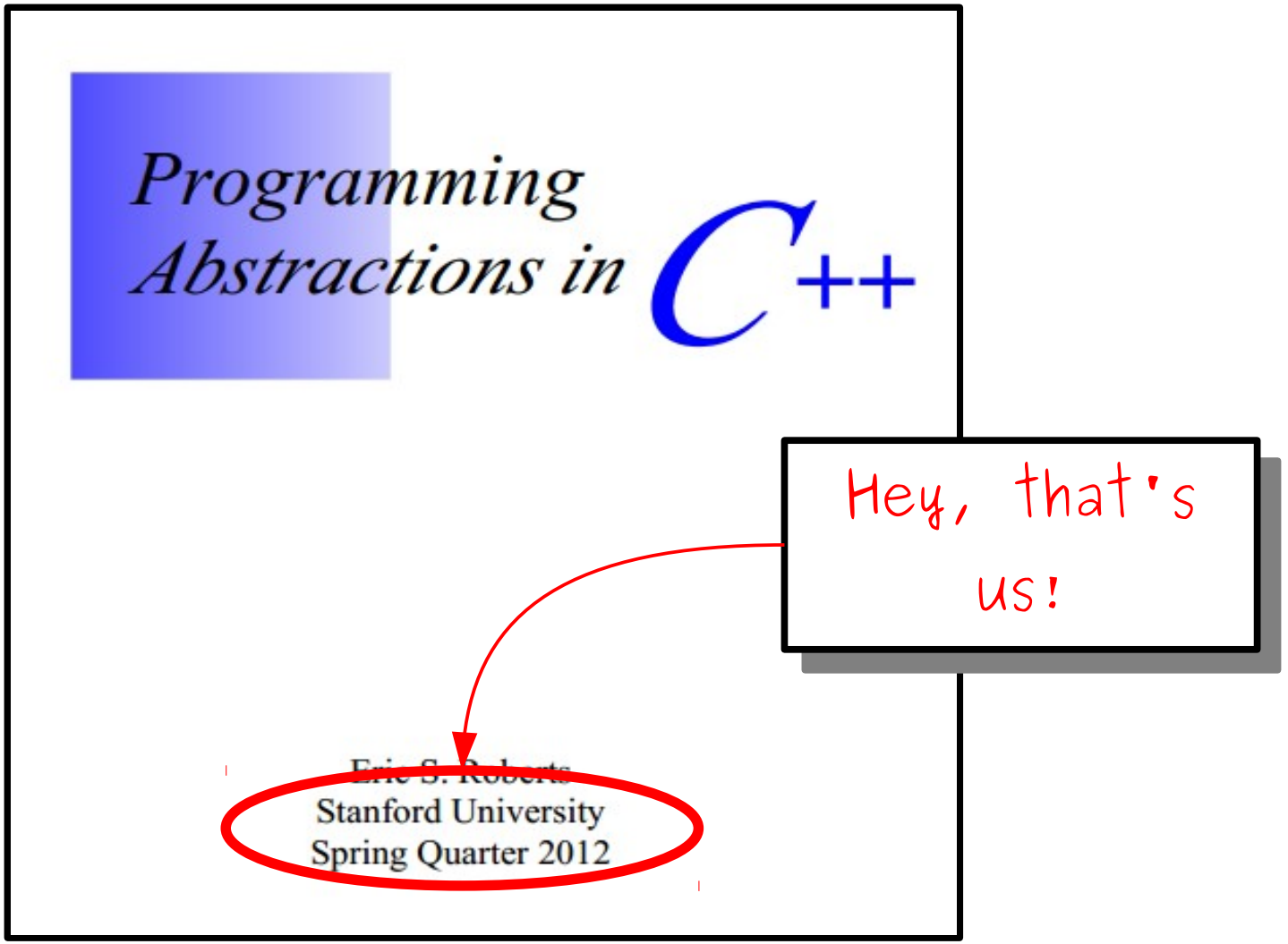
CS106A

(or equivalent)

Required Reading

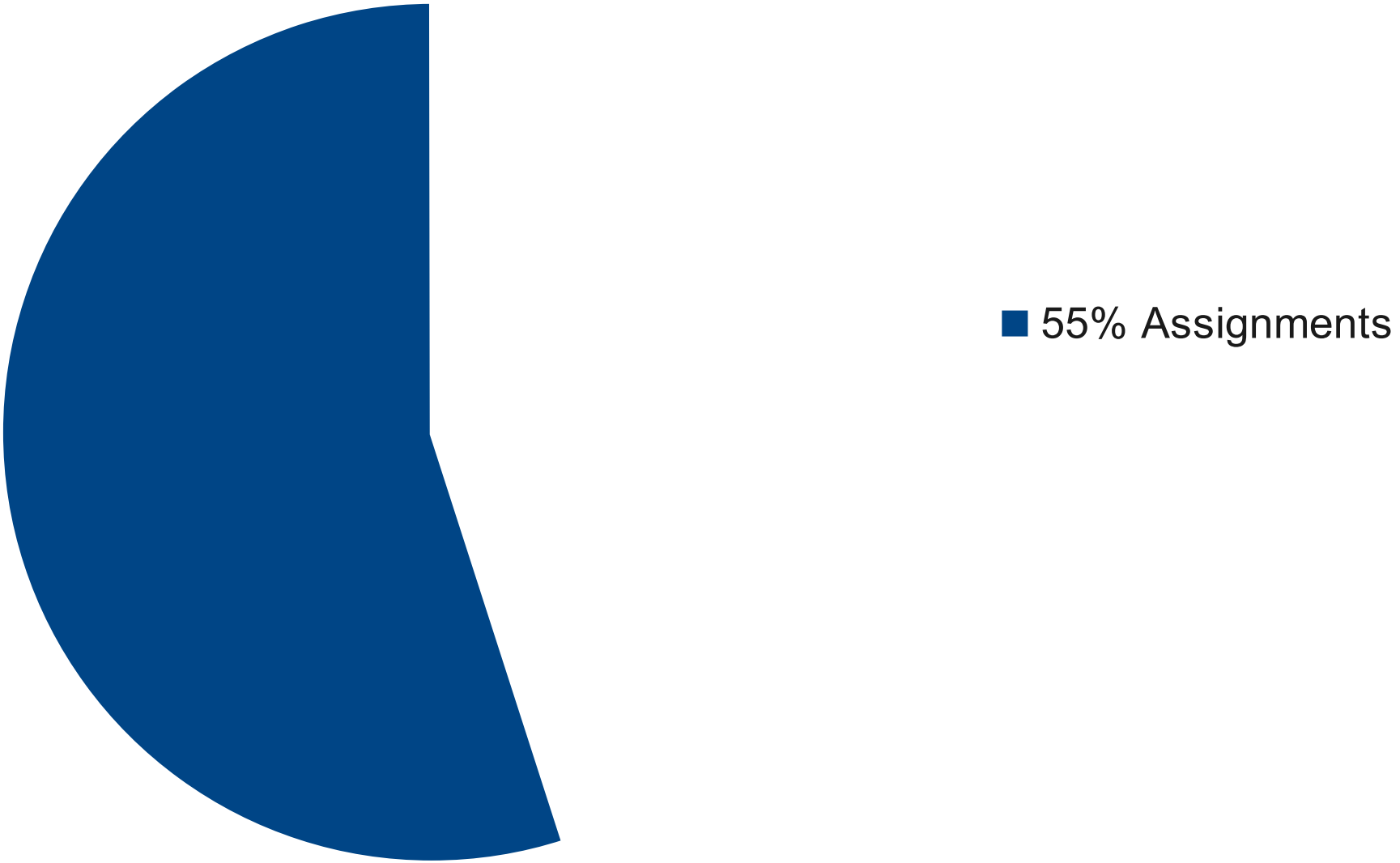


Required Reading



Grading Policies

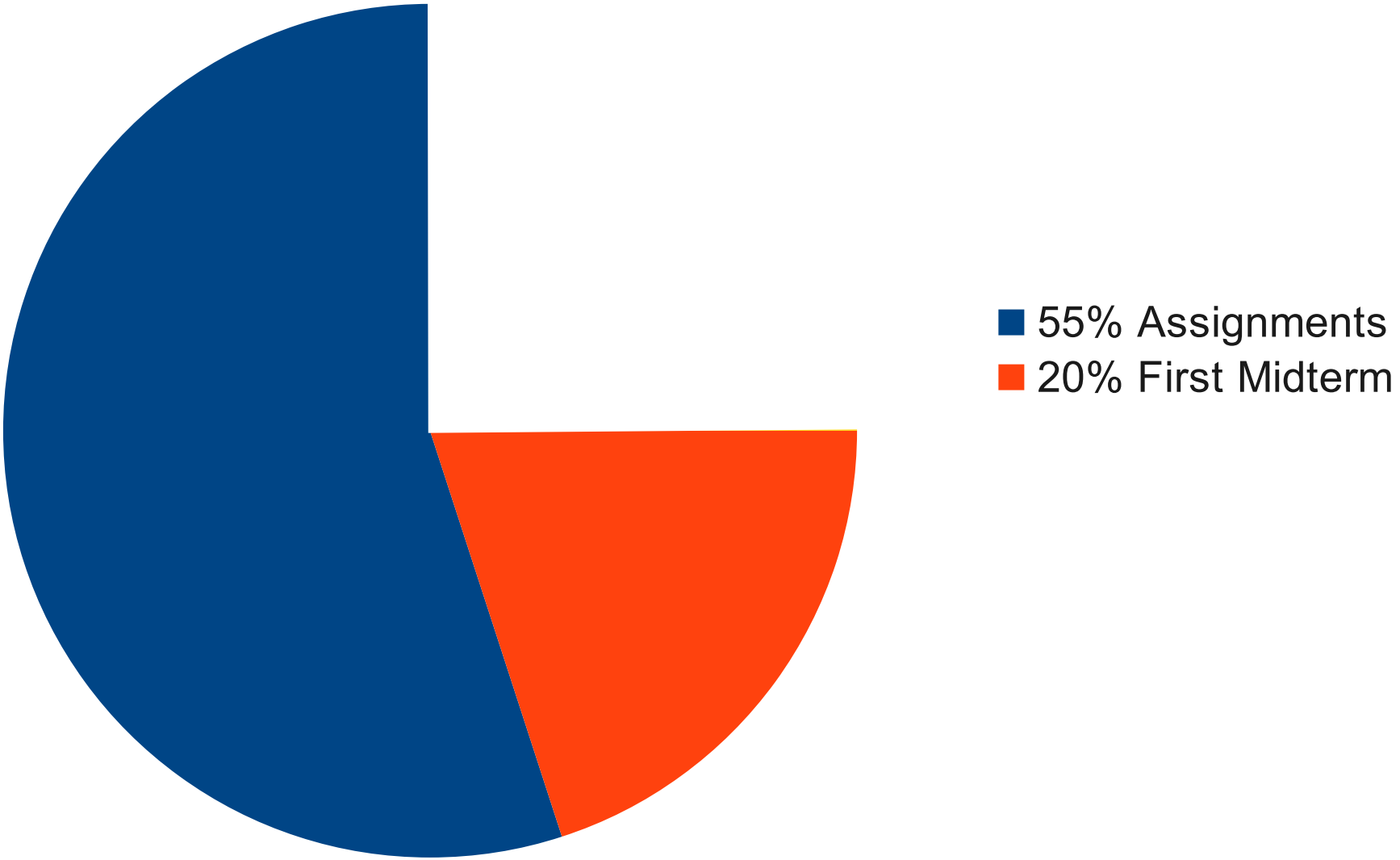
Grading Policies



Grading Policies

Seven Programming Assignments

Grading Policies

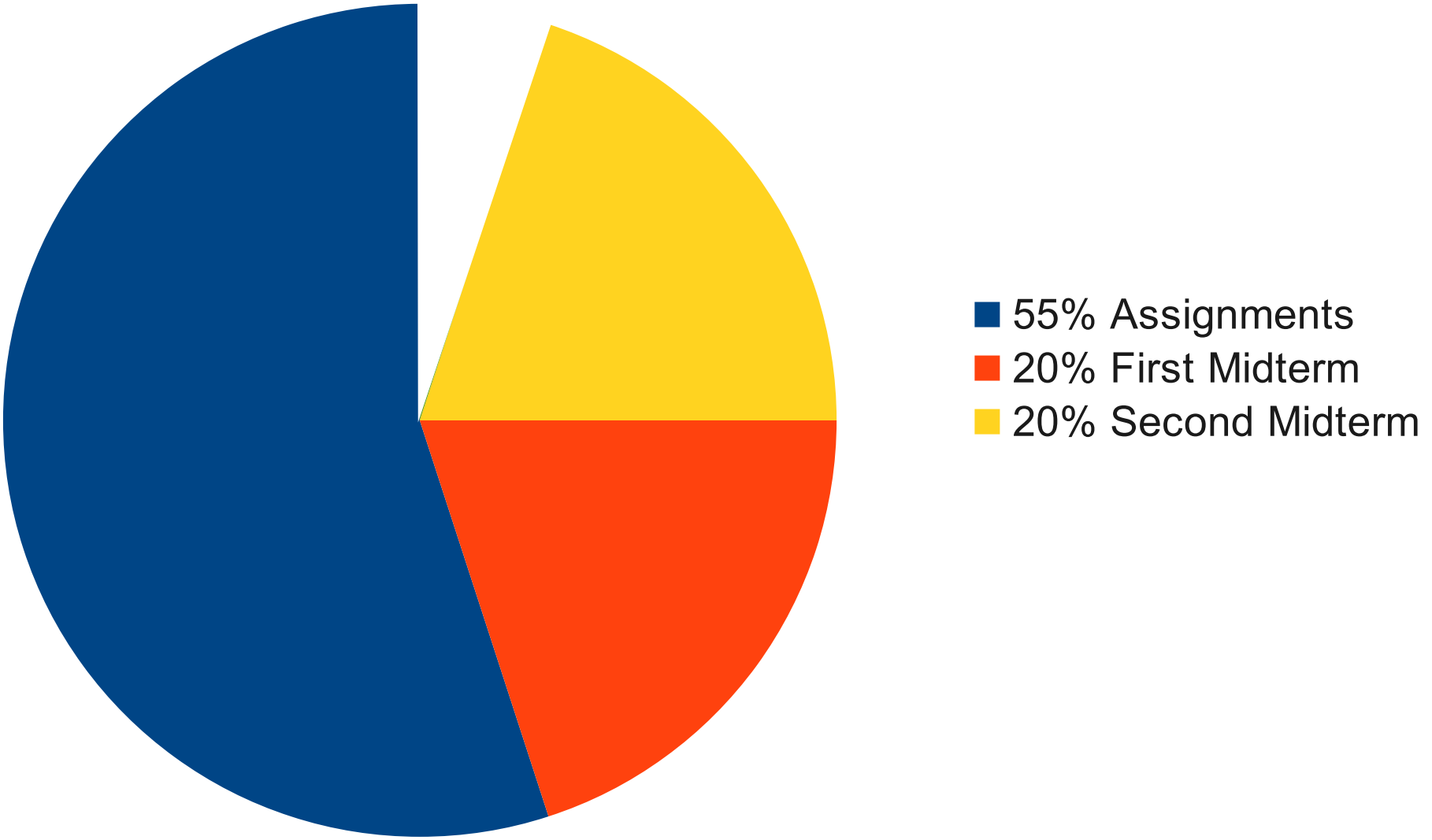


Grading Policies

First Midterm Exam

Thursday, May 3
7PM – 10PM

Grading Policies



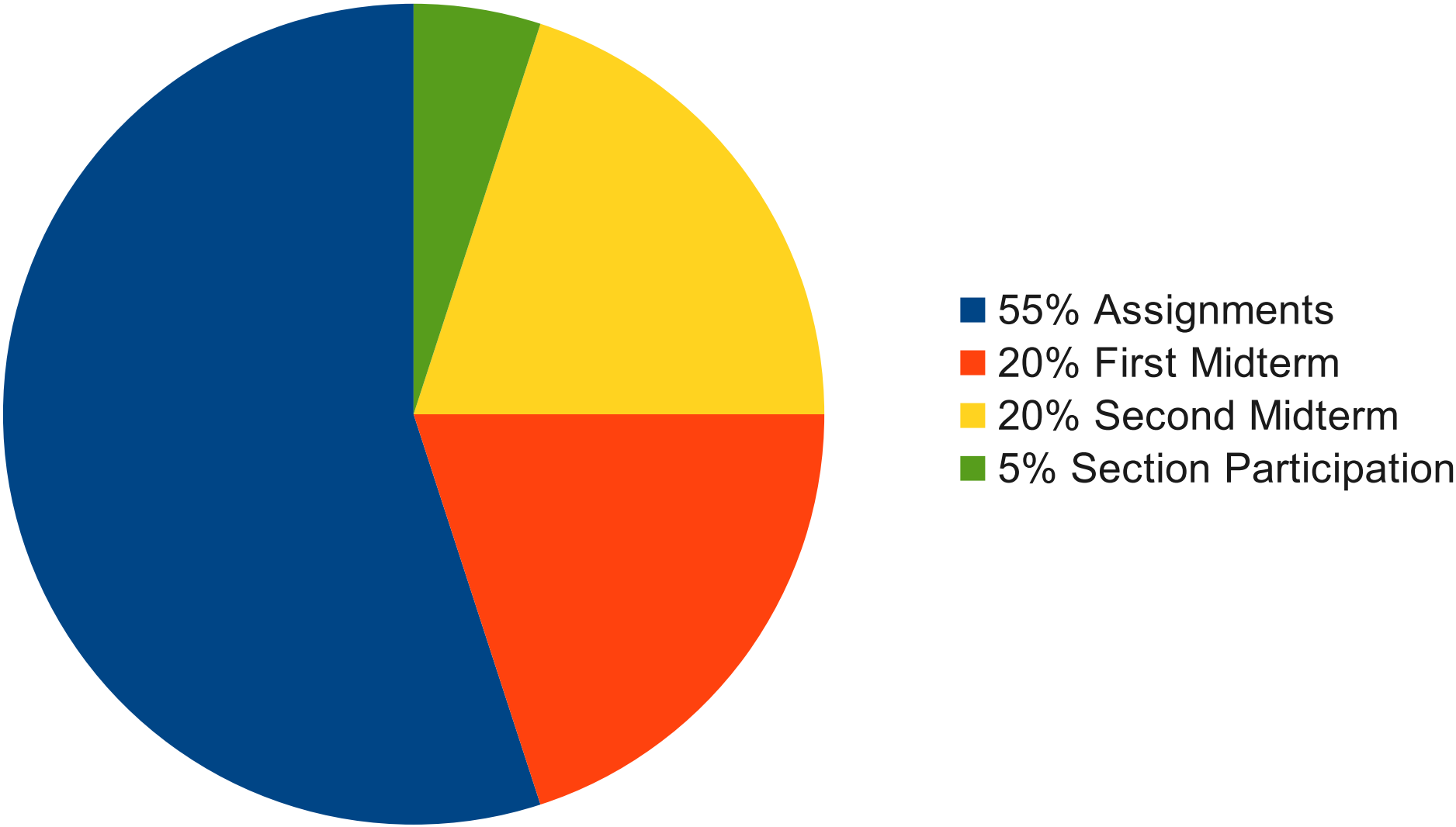
Grading Policies

- 55% Assignments
- 20% First Midterm
- 20% Second Midterm

Second Midterm Exam

Thursday, May 31
7PM – 10PM

Grading Policies



Discussion Sections

- Weekly discussion sections.
- Section attendance is **required** in CS106B.
- Sign up between Thursday, April 5 at 5:00PM and Sunday, April 8 15 at 5:00PM at

http://cs198.stanford.edu/section

- You don't need to (and shouldn't!) sign up for a section on Axess; everything is handled through the above link.

How Many Units?

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {




}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;









}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;



}
```

How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;
    if (reallyBusy()) {
       return 3;
    }


}
```

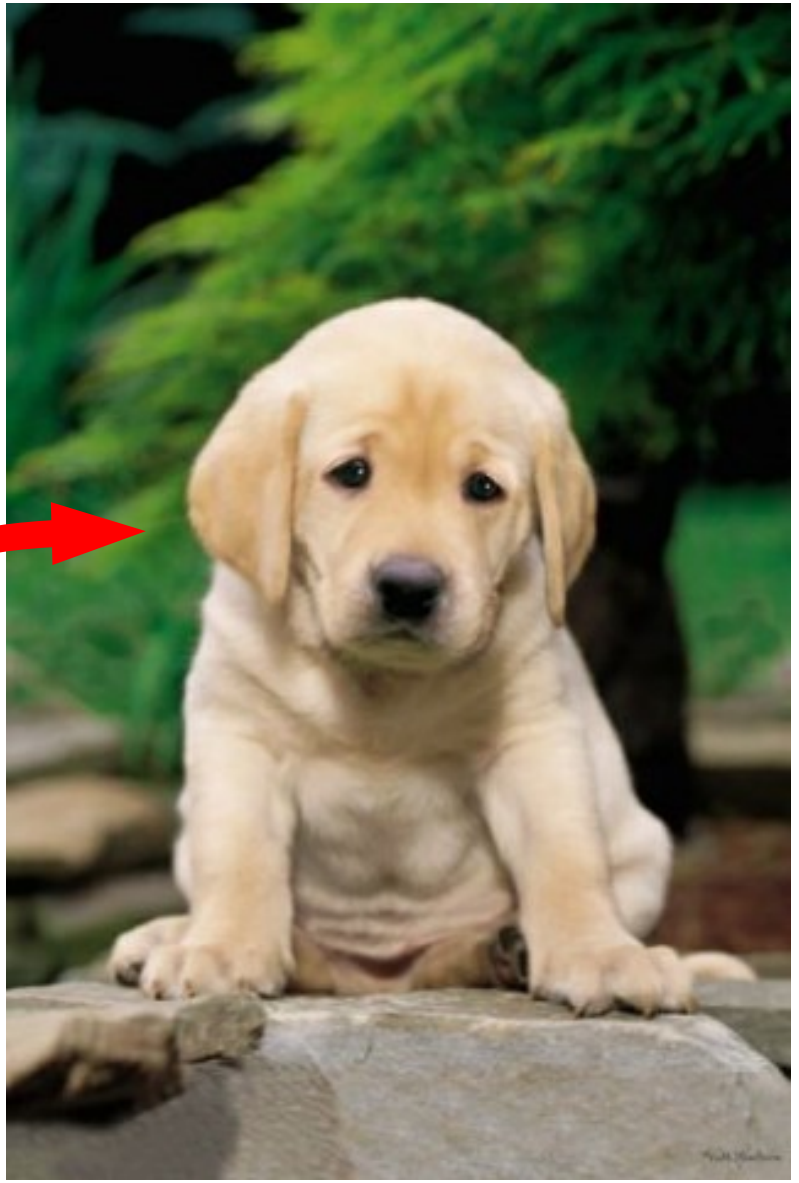
How Many Units?

```
int numUnits(bool isGrad, bool wantsFewerUnits) {
    if (!isGrad) return 5;
    if (!wantsFewerUnits) return 5;

    if (reallyBusy()) {
       return 3;
    } else {
       return 4;
    }
}
```

A Word on the Honor Code...

YOU MAKE PUPPY CRY



A Word on the Honor Code...

- Feel free to discuss general ideas with other students, but **do not** share any programs or code (text of the programs).
- **Cite all sources** you use and everyone you collaborated with.
- This is **not** an exhaustive list; please see Handout #03 for a full discussion of the Honor Code.

But, on the plus side...

…there's The LaIR!

Getting Help

- LaIR Hours!
  - Sunday – Thursday, 6PM – Midnight
  - Starts next week.
- Zach's Office Hours in Gates 160
  - Monday/Wednesday, 11AM – Noon
  - Thursday, 2PM – 4PM
- Keith's Office Hours in Gates 178
  - Tuesday, 2 – 4PM

What's Next in Computer Science?

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.

Goals for this Course

Learn how to model and solve complex problems with computers.

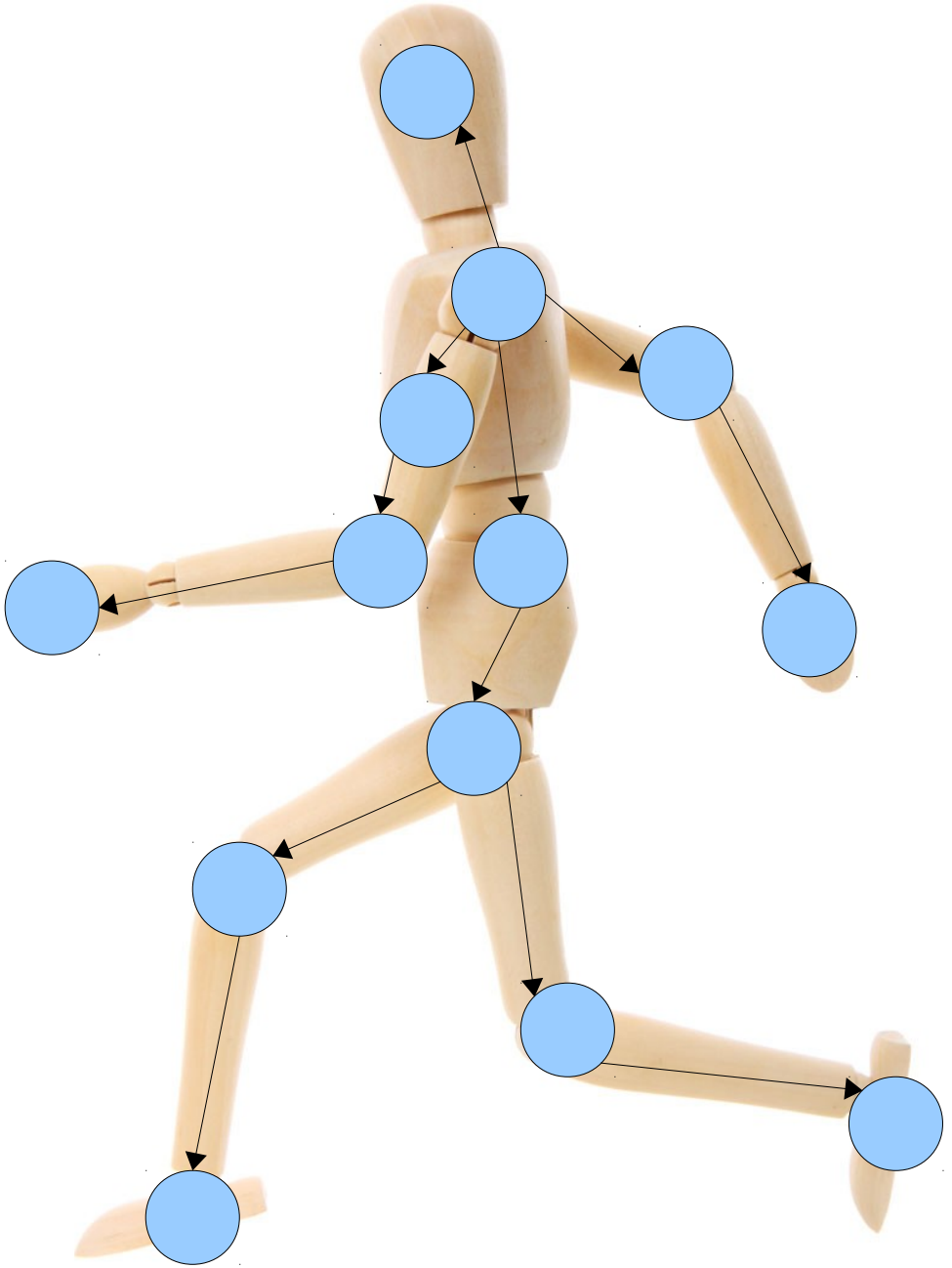
To that end:

- Explore common abstractions for representing problems.

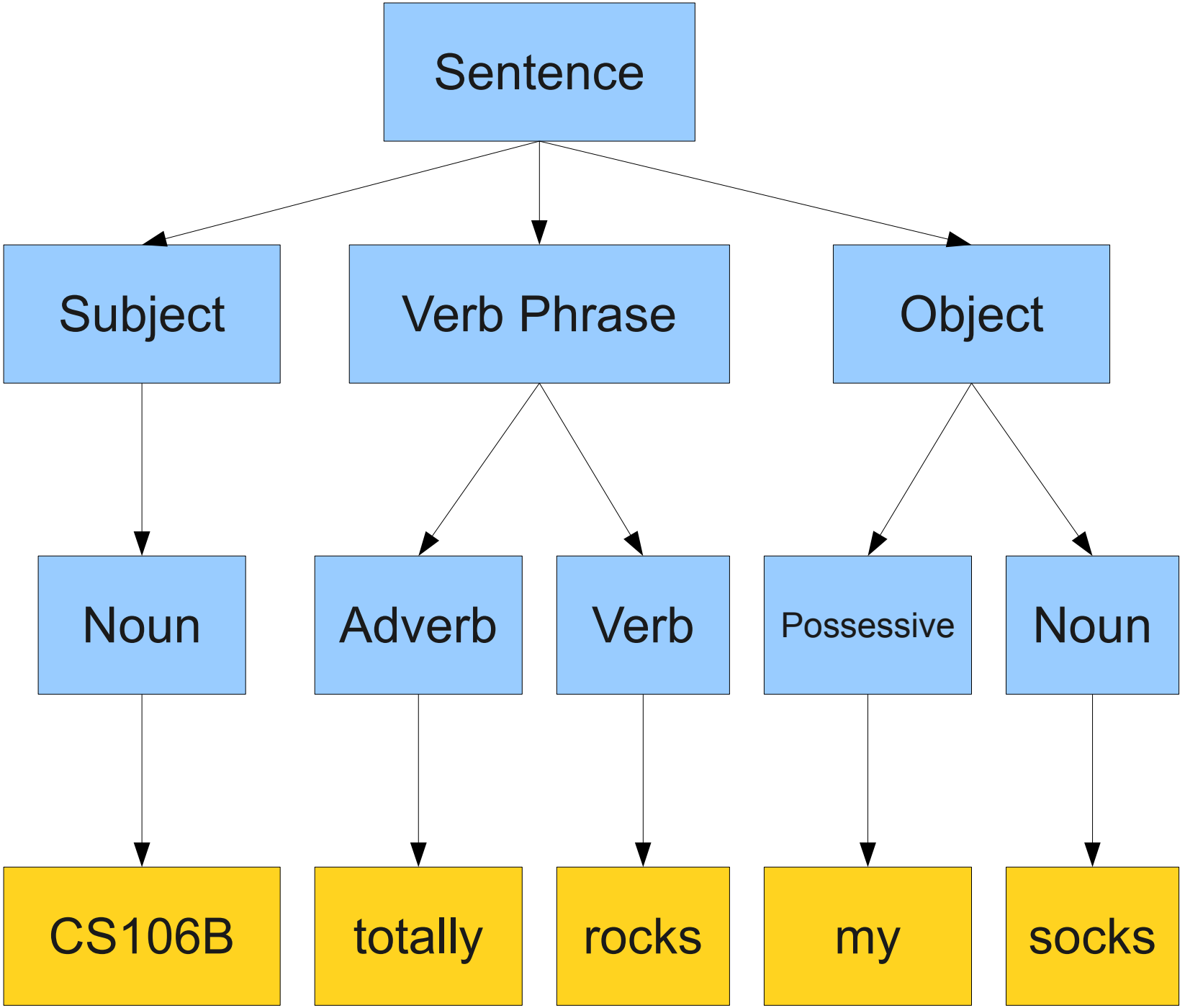
  Harness recursion and understand how to think about problems recursively.

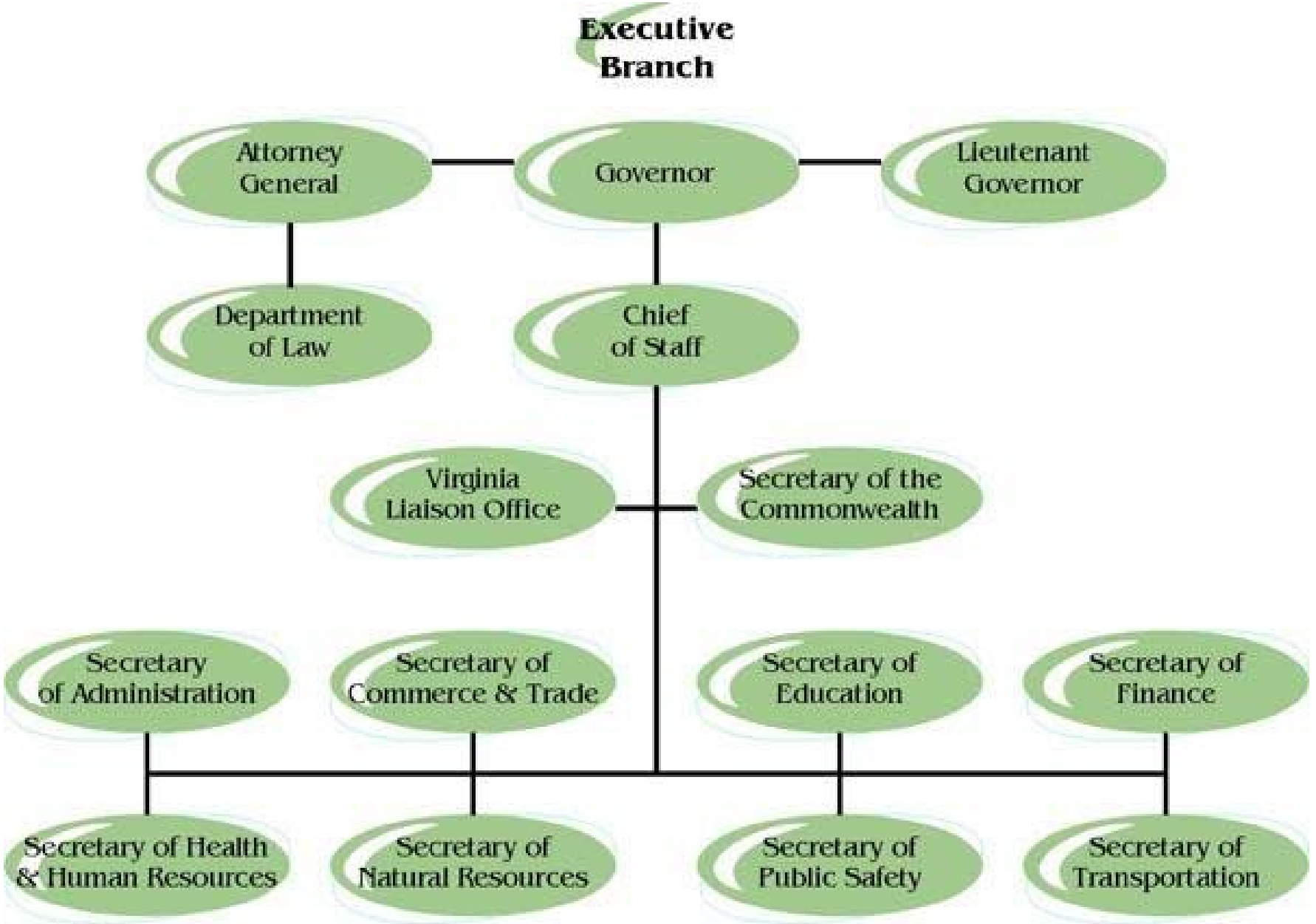
  Quantitatively analyze different approaches for solving problems.

http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg

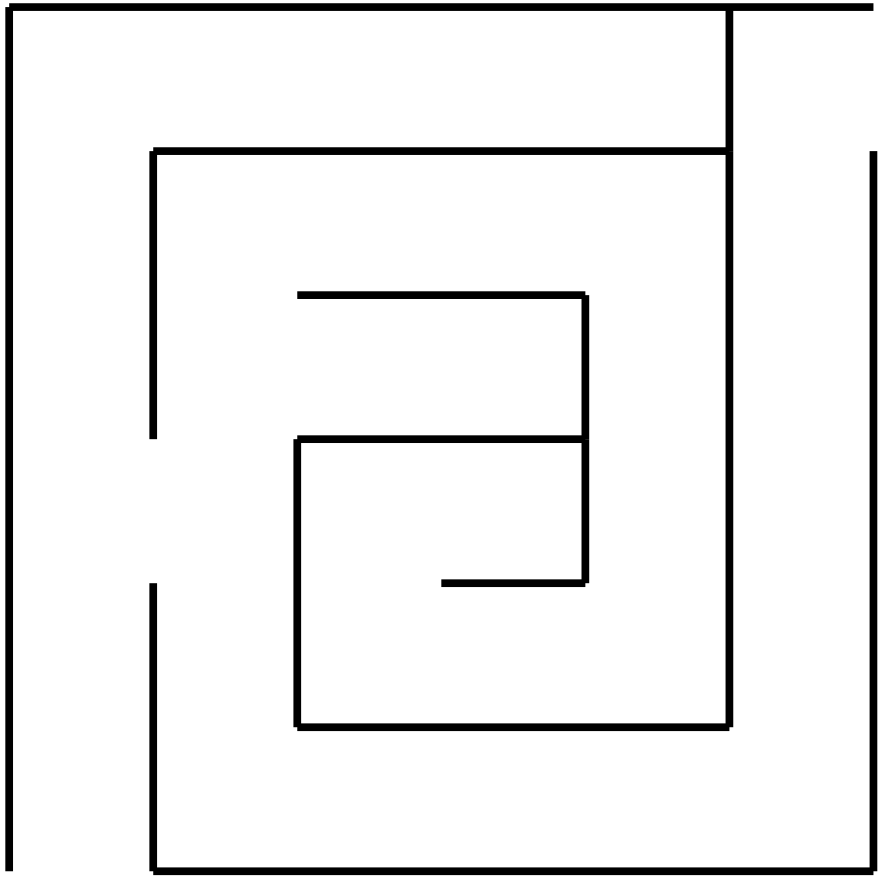


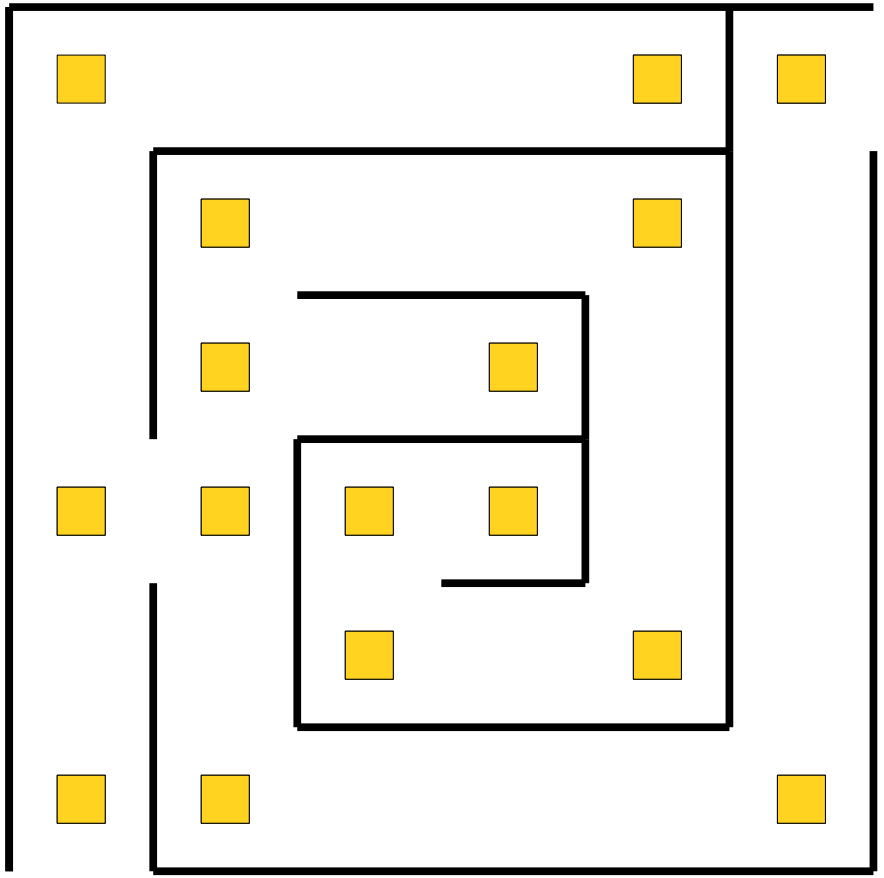
http://www.publicdomainpictures.net/pictures/10000/velka/1-1265899974oKJ9.jpg

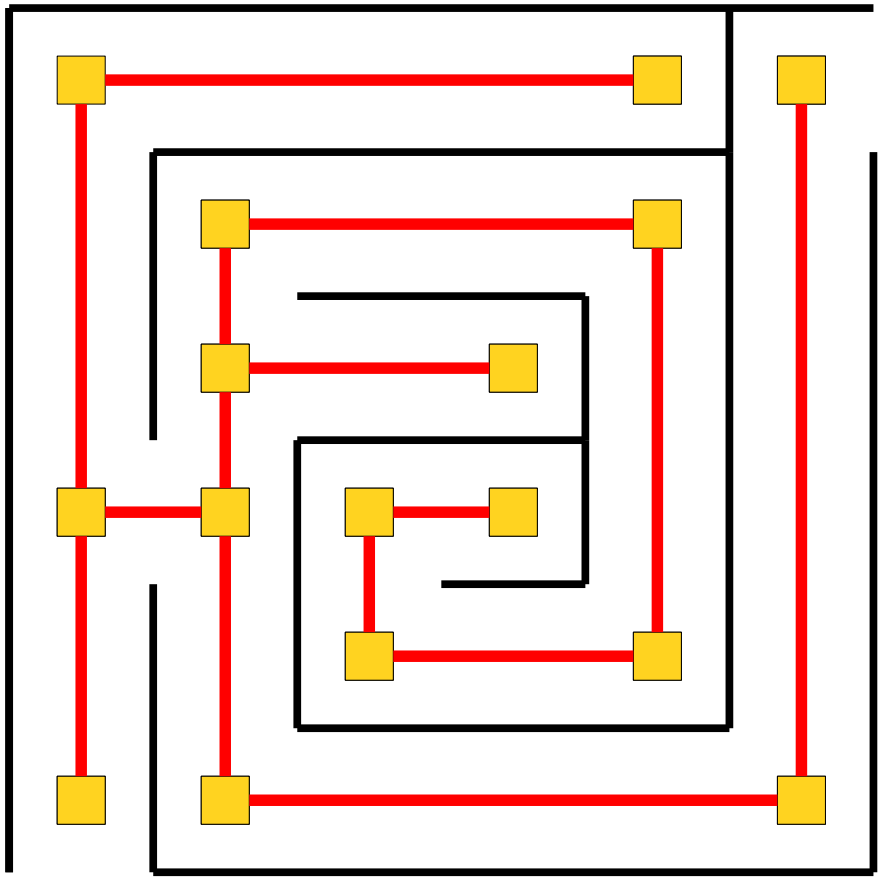

Sentence
Subject
Verb Phrase
Object
Noun
Adverb
Verb
Possessive
Noun
CS106B
totally
rocks
my
socks

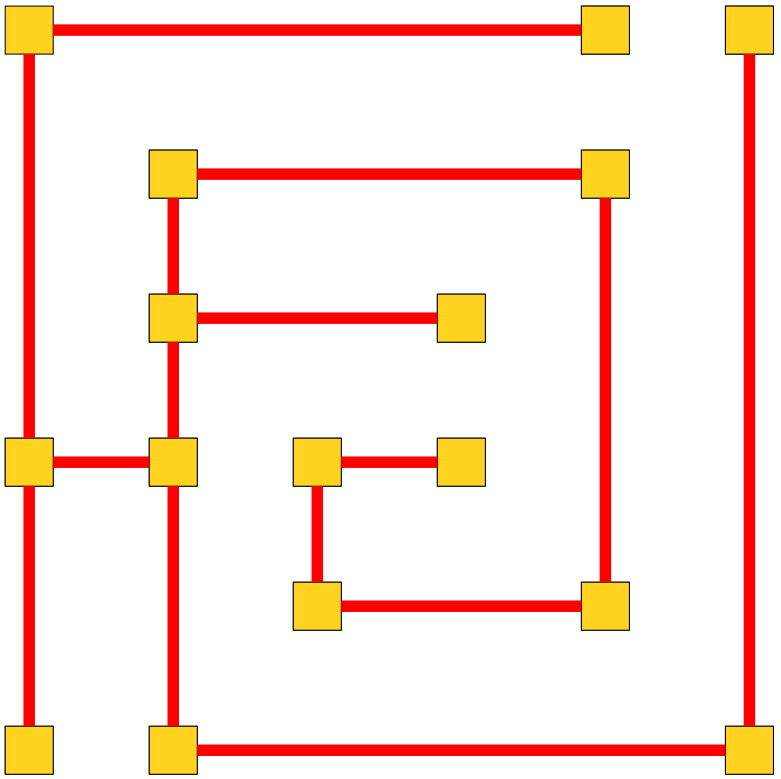


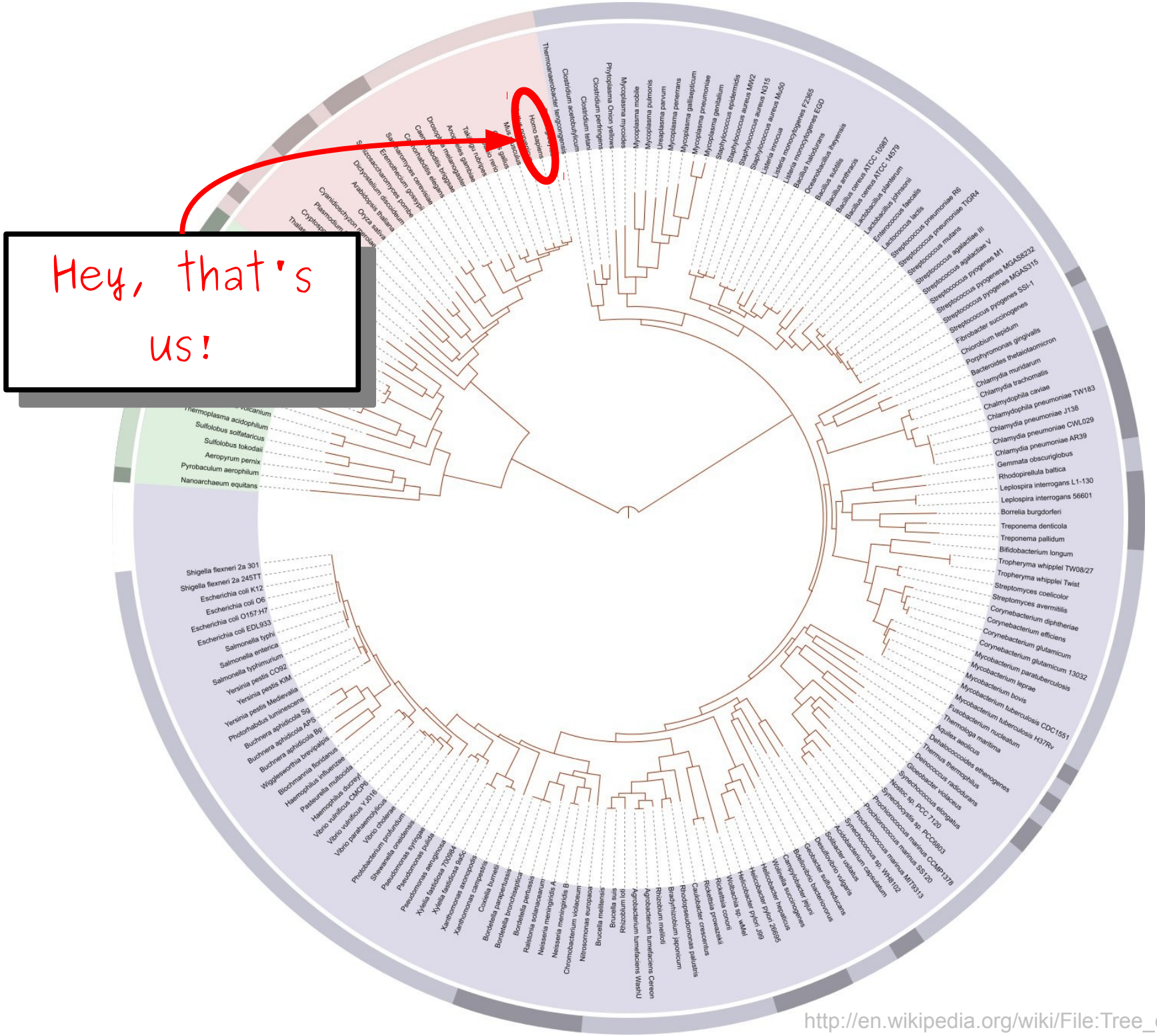
http://www.virginia.gov/images/wholeexecbranch2.jpg



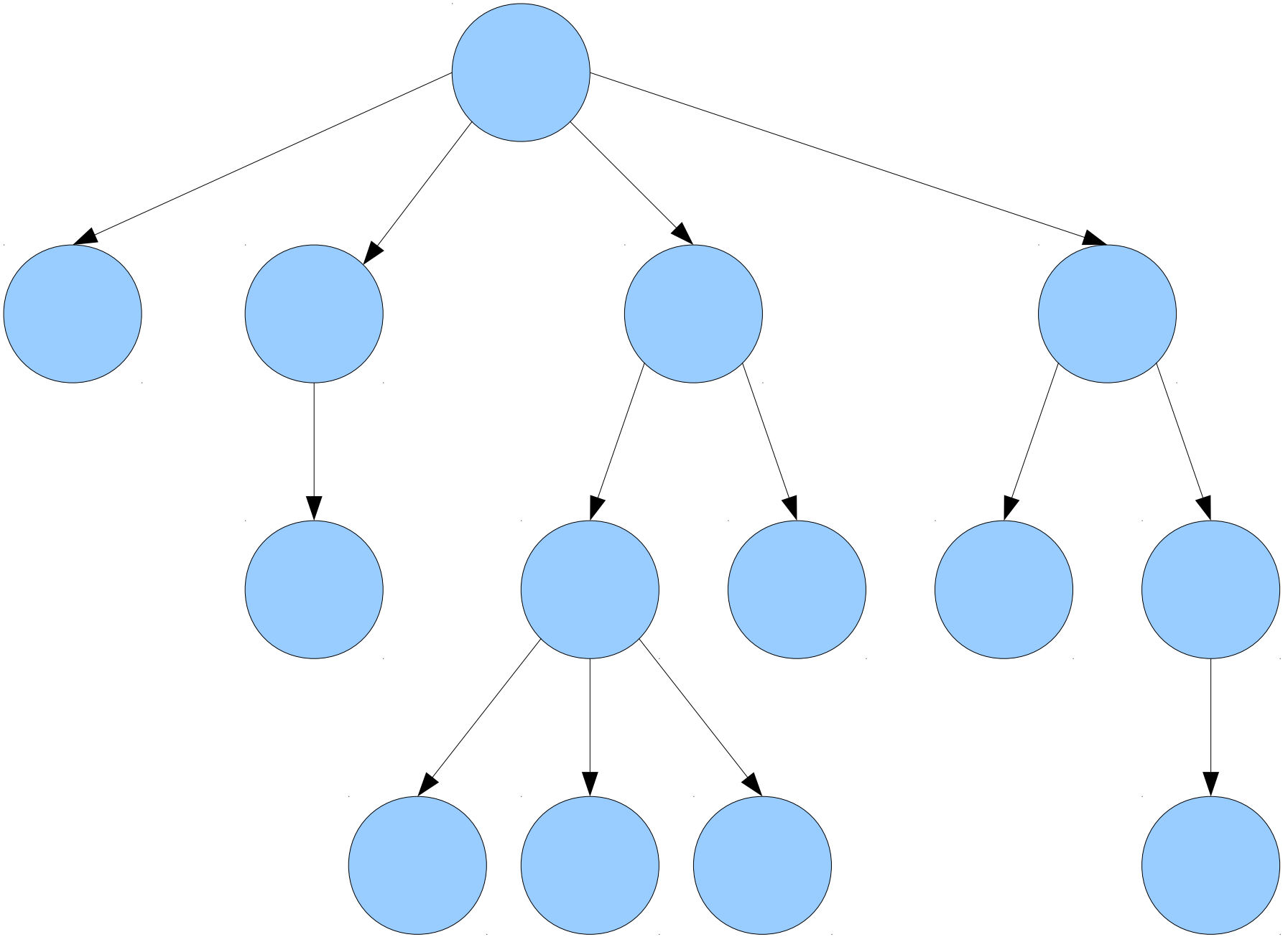








http://en.wikipedia.org/wiki/File:Tree_of_life_SVG.svg



Building a vocabulary of **abstractions**
makes it possible to describe
larger classes of problems.

Building a vocabulary of **abstractions**
makes it possible to **solve**
larger classes of problems.

Goals for this Course

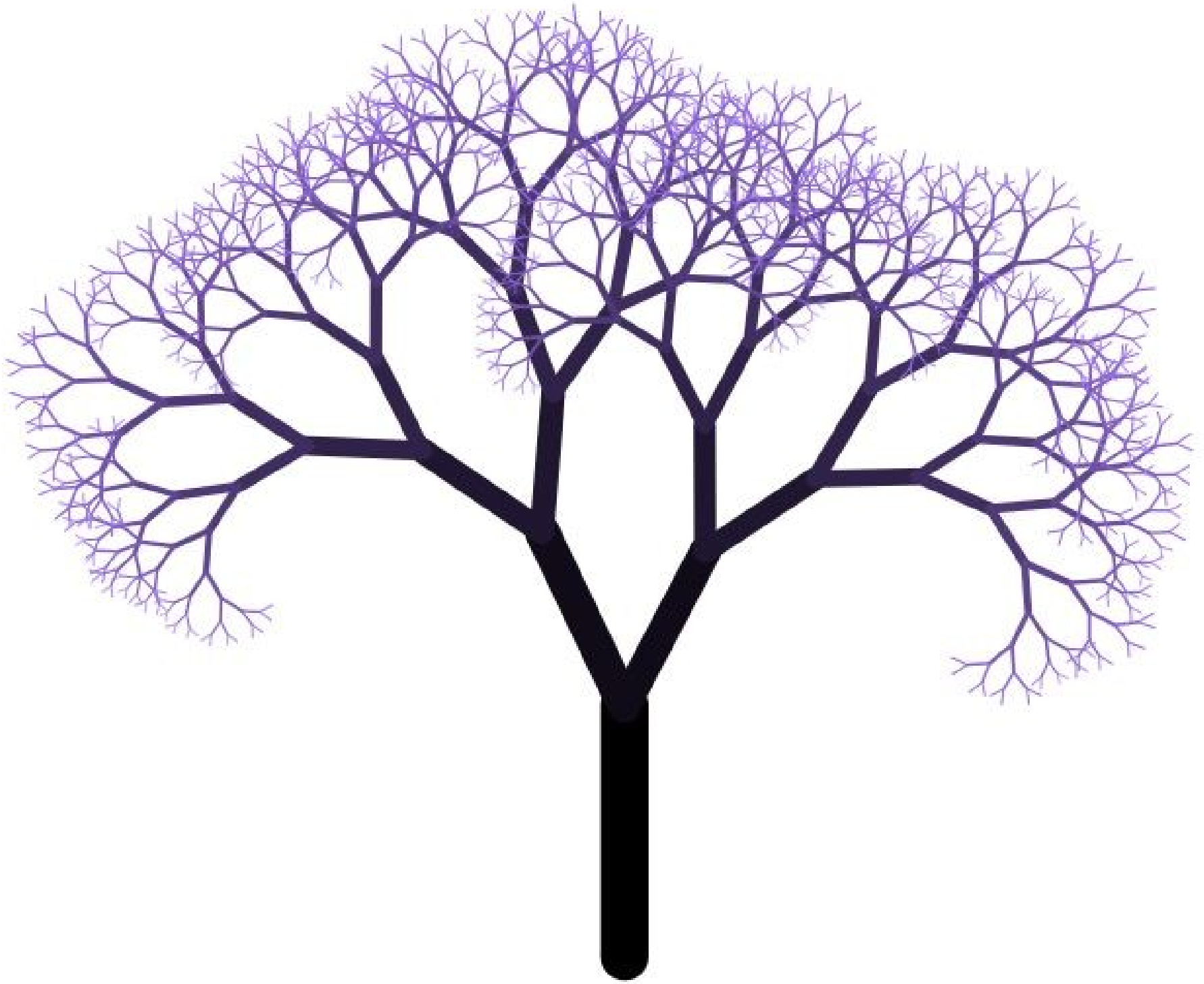
- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Goals for this Course

Learn how to model and solve complex problems with computers.

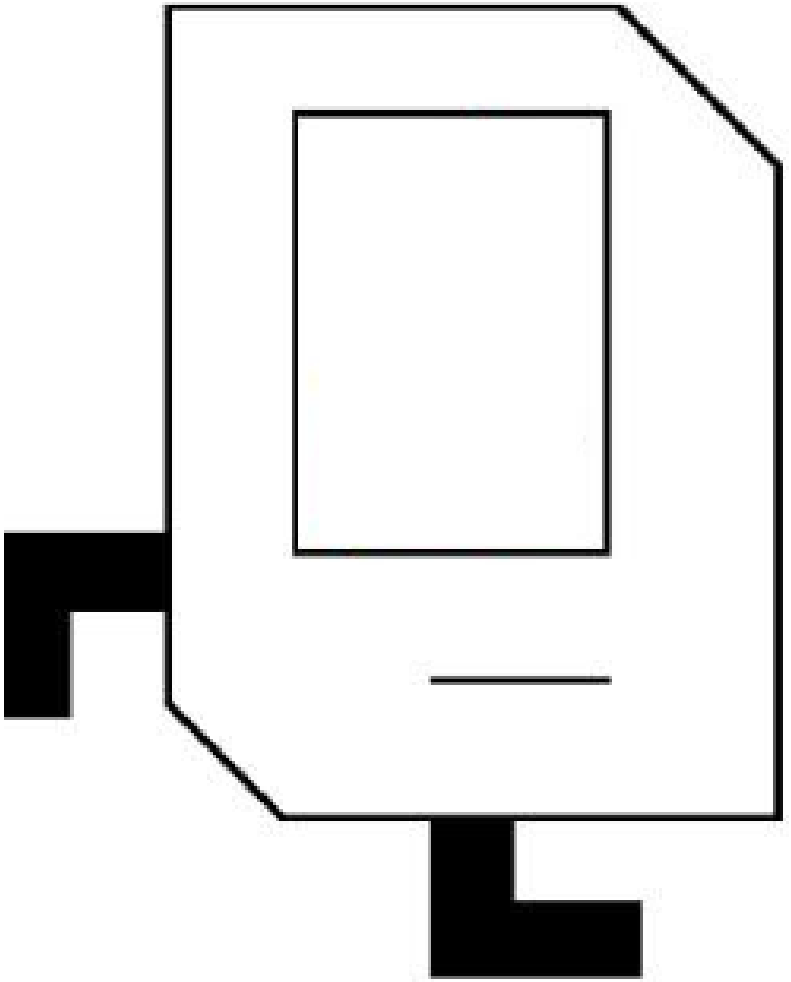
To that end:

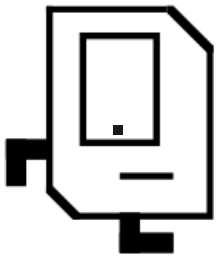
- Explore common abstractions for representing problems.
- Harness recursion and understand how to think about problems recursively.
- Quantitatively analyze different approaches for solving problems.

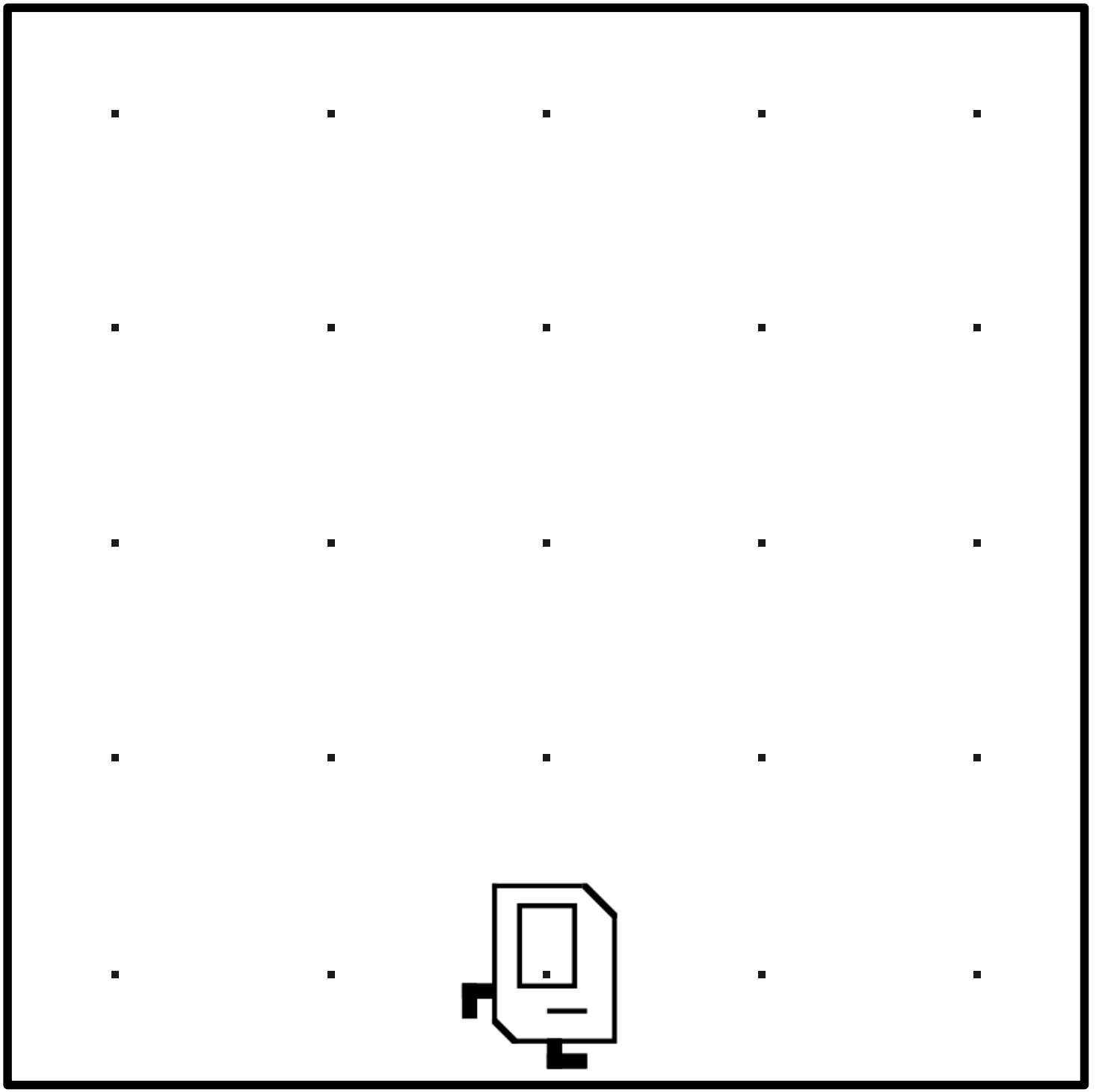


http://www.marketoracle.co.uk/images/2010/Oct/fractal-tree2.jpg

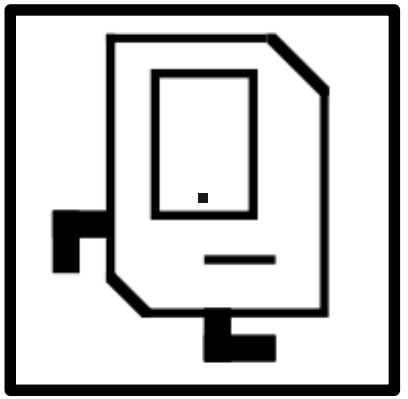
http://1.bp.blogspot.com/-Hwf-U9hRwEI/TX480ONoo6I/AAAAAAAAAcM/fyBi0I5HsUI/s1600/the_great_wave_off_kanagawa.jpg



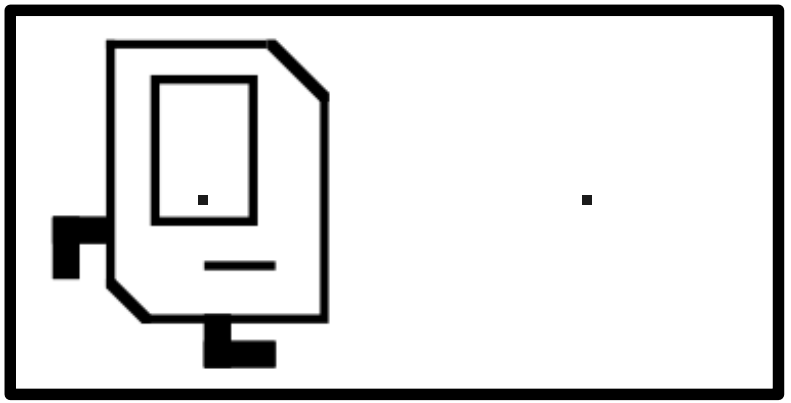


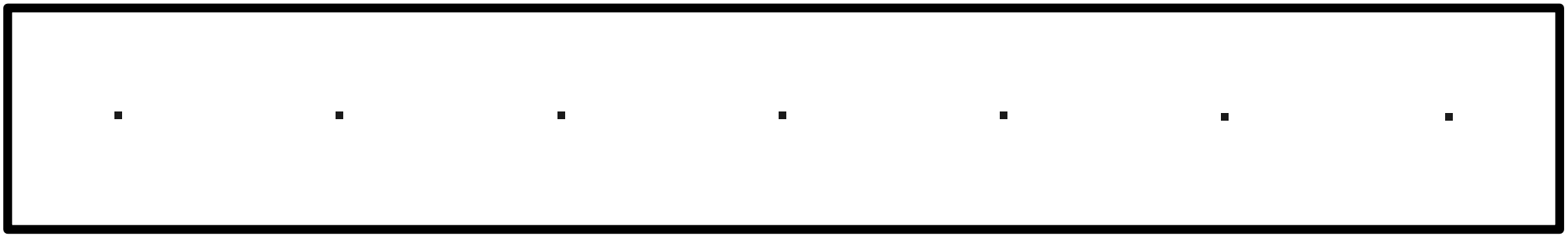


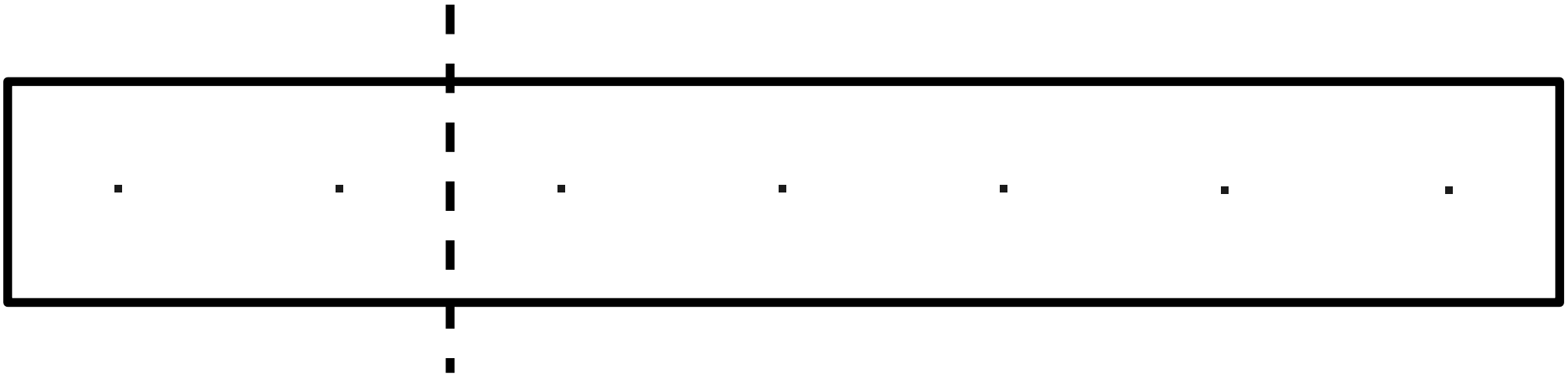
Width 1

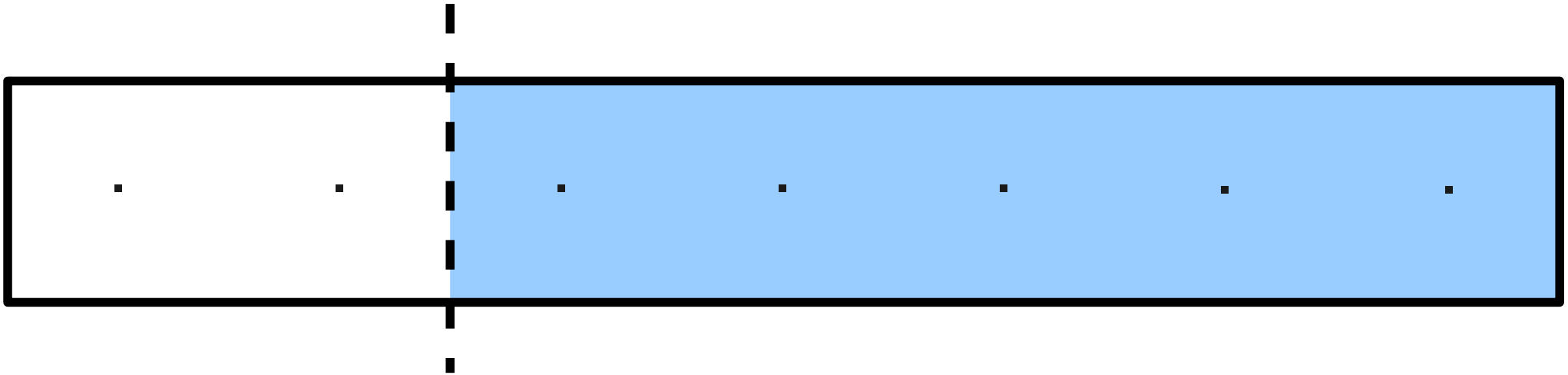


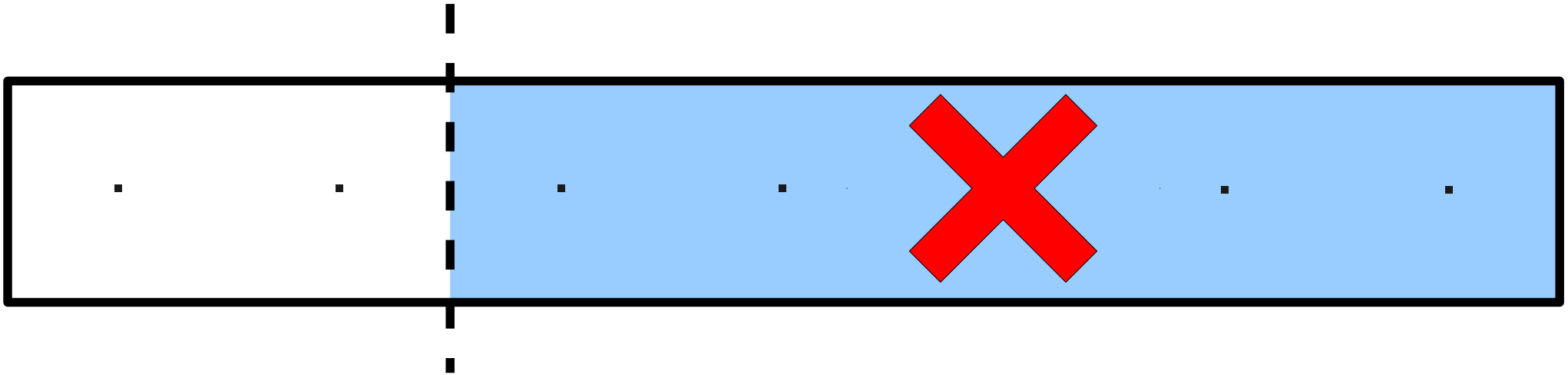
Width 2

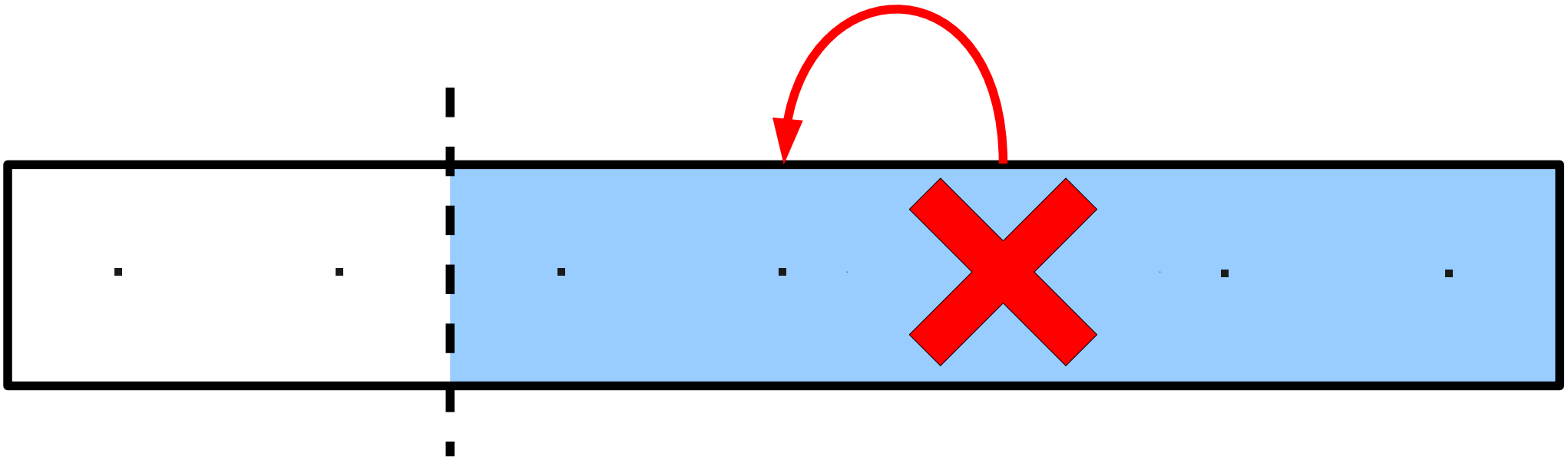


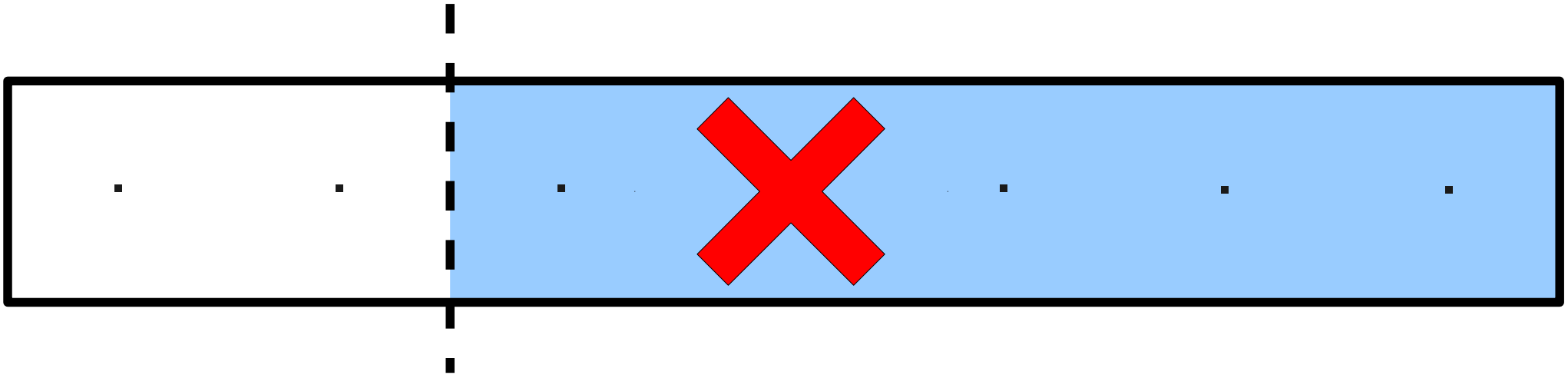


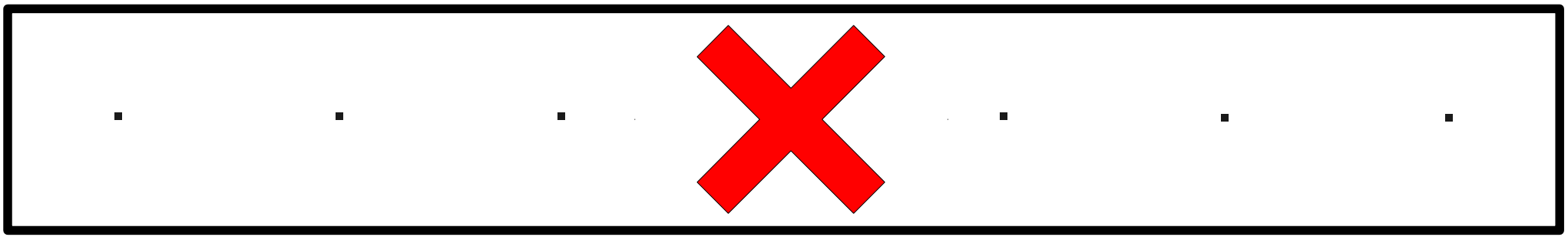












Finding the Midpoint

- If the width is 1, Karel is standing on the midpoint.
- If the width is 2, either position is the midpoint.
- Otherwise:
  - Take two steps forward.
  - Find the midpoint of the rest of the world.
  - Take one step backward.

A Surprisingly Short Solution

A **recursive solution** is a solution that is defined in terms of itself.

Thinking recursively allows you to solve an enormous class of problems cleanly and concisely.

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.

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.


Rhapsody™


Spotify™


P
PANDORA®

Adobe® Photoshop® CS2

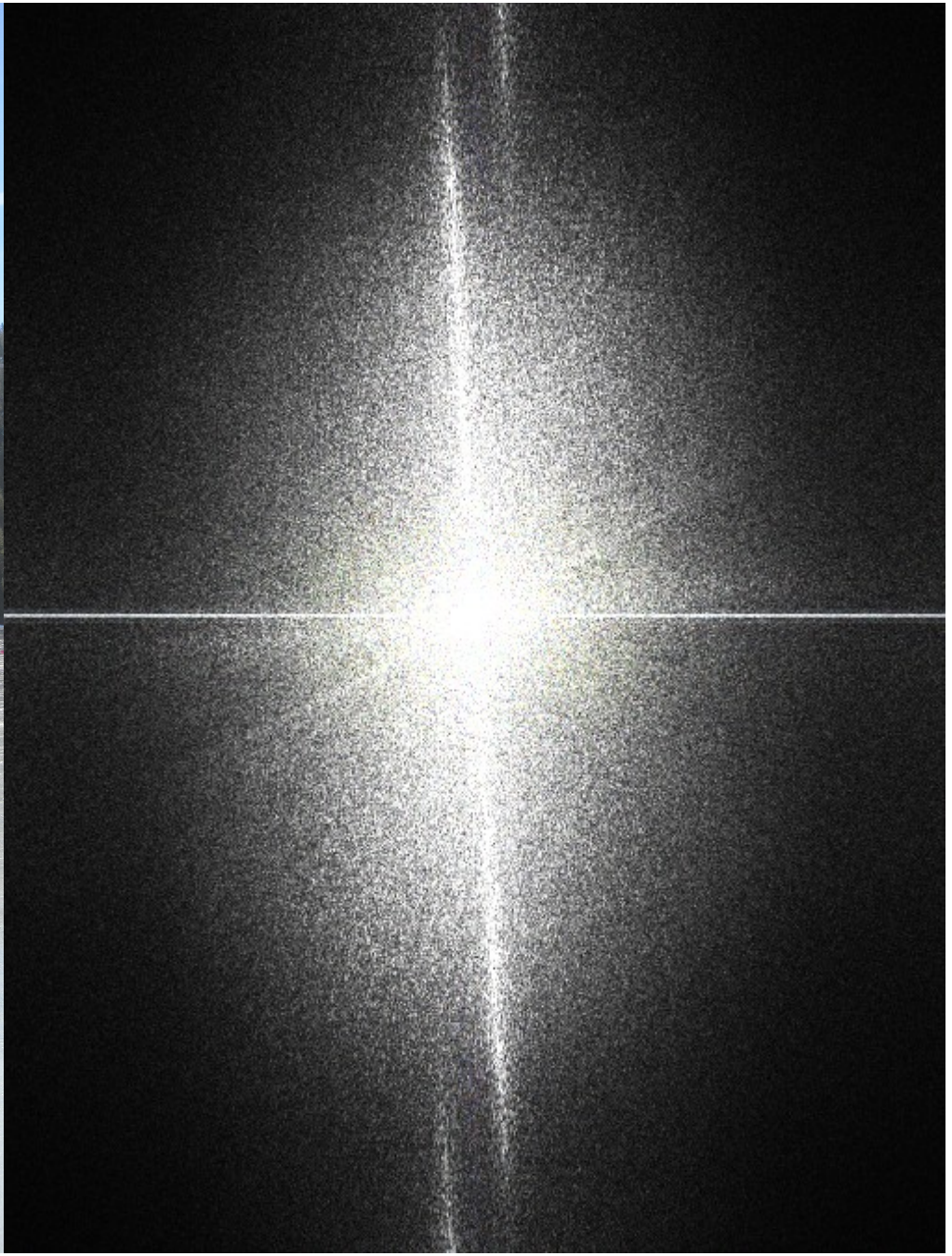

2:30
Google
Gmail
Maps
YouTube
Market


Phone
People
Messaging
Outlook
XBOX LIVE
Office


Verizon 3G
3:26 PM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Notes
Utilities
iTunes
App Store
Game Center
Settings
Phone
Mail
Safari
iPod

The Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi i \frac{k\,n}{N}}$$



Naive Algorithm:
Approximately N^2 operations.

Naive Algorithm:
Approximately N^2 operations.

Fast Fourier Transform:
Approximately $N \log N$ operations.

N	N^2	$N \log N$
10	10^2	10
10^2	10^4	2×10^2
10^3	10^6	3×10^3
10^4	10^8	4×10^4
10^5	10^{10}	5×10^5
10^6	10^{12}	6×10^6

N	N^2	$N \log N$
10	10^2	10
10^2	10^4	2×10^2
10^3	10^6	
10^4	10^8	
10^5	10^{10}	5×10^5
10^6	10^{12}	6×10^6

This is about 160,000 times faster!

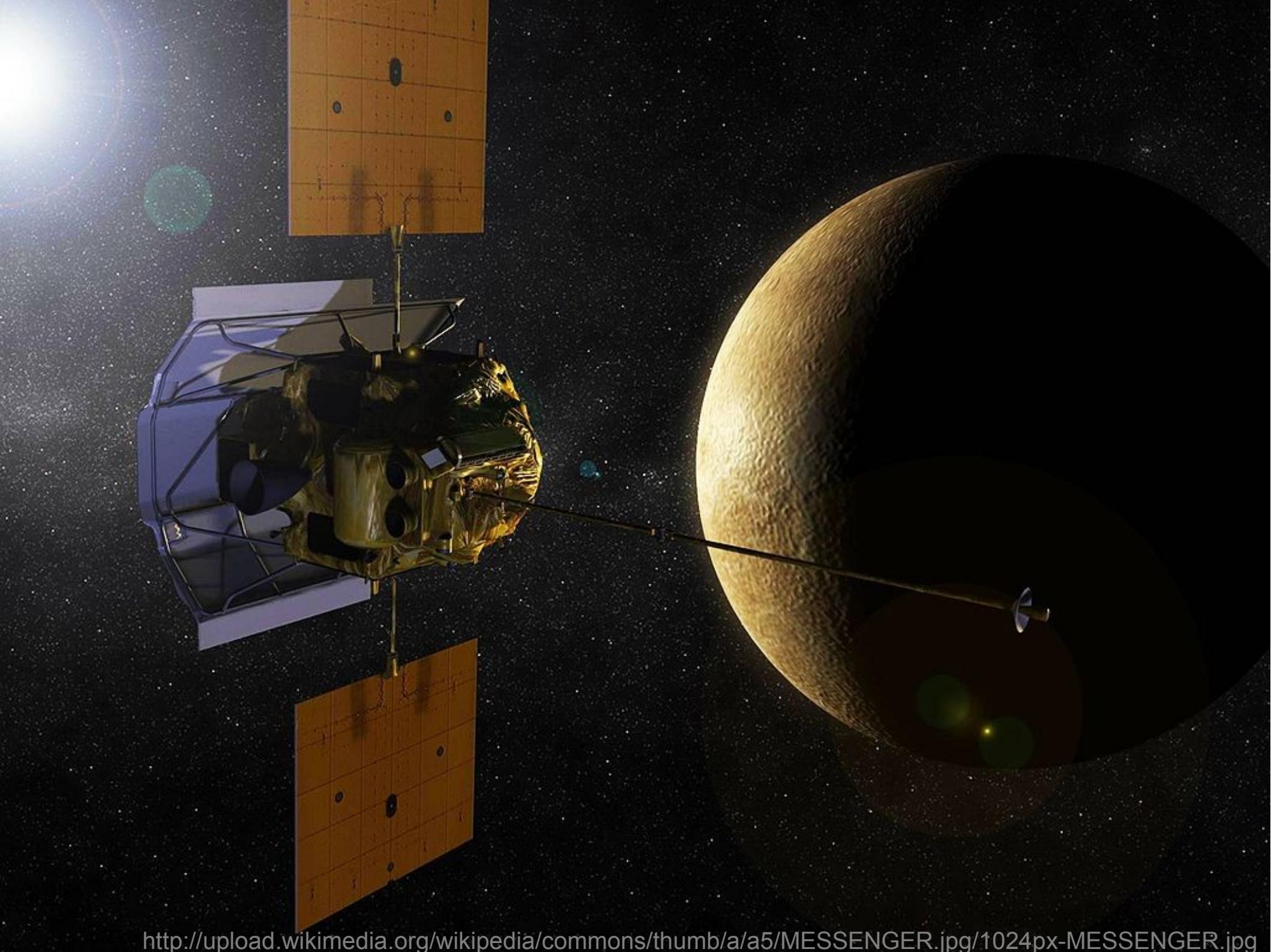
At one operation per nanosecond, what's the largest N for which we can compute an answer in one second?

At one operation per nanosecond, what's the largest N for which we can compute an answer in one second?

N^2: About 30,000.

At one operation per nanosecond, what's the largest N for which we can compute an answer in one second?

N^2: About 30,000.
$N \log N$: About 150,000,000.



http://upload.wikimedia.org/wikipedia/commons/thumb/a/a5/MESSENGER.jpg/1024px-MESSENGER.jpg

Assume there are N objects in space.

Assume there are N objects in space.

Naive Algorithm:
Approximately N^2 calculations / frame.

Assume there are N objects in space.

Naive Algorithm:
Approximately N^2 calculations / frame.

Fast Monopole Method:
Approximately N calculations / frame.

At one operation per nanosecond, what's the largest N for which we can compute sixty frames per second?

At one operation per nanosecond, what's the largest N for which we can compute sixty frames per second?

N^2: About 6,500.

At one operation per nanosecond, what's the largest N for which we can compute sixty frames per second?

N^2: About 6,500.
N: About 42,000,000.

Quantitatively **analyzing algorithms** lets us compare different processes and reason about their performance.

Goals for this Course

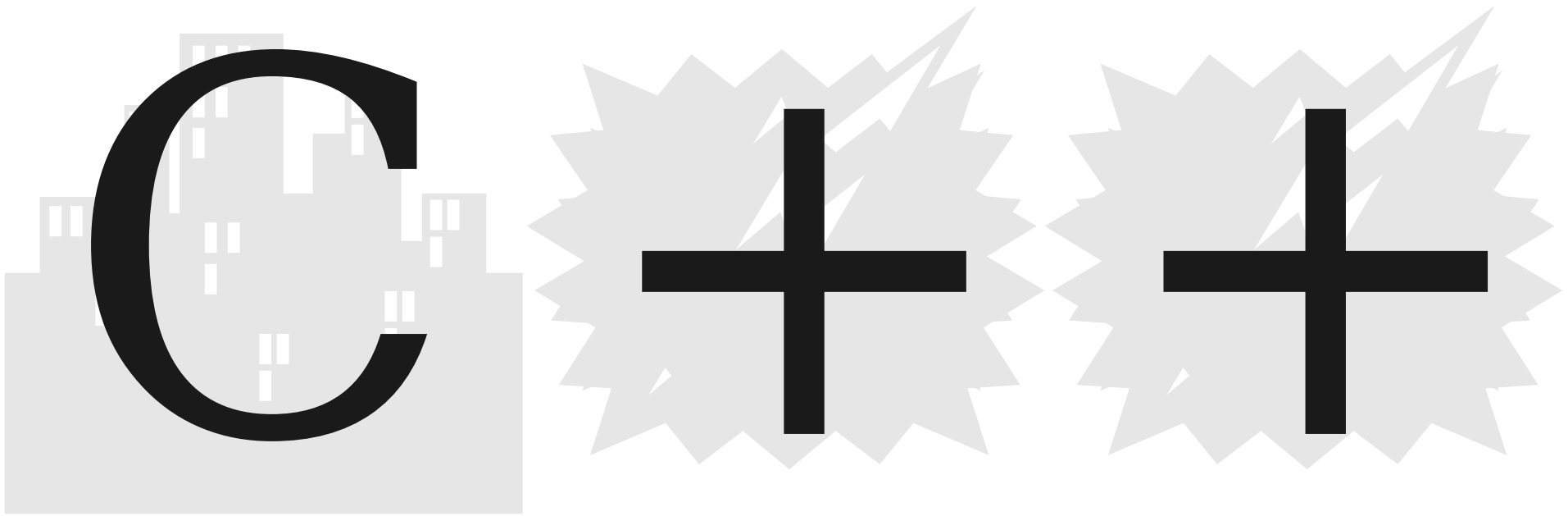
- **Learn how to model and solve complex problems with computers.**
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

Who's Here Today?

- Biochemistry
- Bioengineering
- Biology
- Business
- Chemical Engineering
- Chemistry
- Classics
- Civil and Environmental Engineering
- CME
- Computer Science
- Earth Systems
- Economics
- Electrical Engineering
- Energy Resources Engineering
- Environmental Engineering
- Ethics in Society
- Geological and Environmental Sciences
- History
- Linguistics
- Materials Science
- Mathematical and Computational Sciences
- Mathematics
- Mechanical Engineering
- MS&E
- Musics
- Physics
- Political Science
- Psychology
- Science, Technology, and Society
- Statistics
- Structural Biology
- Symbolic Systems
- Undeclared!

One more detail...

C++


C++

What is C++?

- Programming language developed in 1983 by Bjarne Stroustrup.
- Widely used for general programming when performance is important.
- Supports a variety of programming styles.

```
/* File: hello-world.cpp
 *
 * A canonical Hello, world! program
 * in C++.
 */

#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
}
```

```cpp
/* File: retain-evens.cpp
 *
 * A program to filter out odd numbers from a list.
 */
#include <iostream>
#include "vector.h"
using namespace std;

Vector<int> retainEvens(Vector<int> values) {
    Vector<int> result;
    for (int i = 0; i < values.size(); i++) {
        if (values[i] % 2 == 0)
            result += values[i];
    }
    return result;
}

int main() {
    Vector<int> values;
    values += 1, 2, 3, 4, 5;

    Vector<int> processed = retainEvens(values);

    for (int i = 0; i < processed.size(); i++) {
        cout << processed[i] << endl;
    }
}
```

CS106L

- Optional, one-unit companion course to CS106B.
- In-depth treatment of C++'s libraries and language features.
- Excellent complement to the material from CS106B; highly recommended!
- Not a replacement for section; it's purely an add-on.

Next Time

- Welcome to C++!
  - Defining functions.
  - Reference parameters.
  - Introduction to recursion.