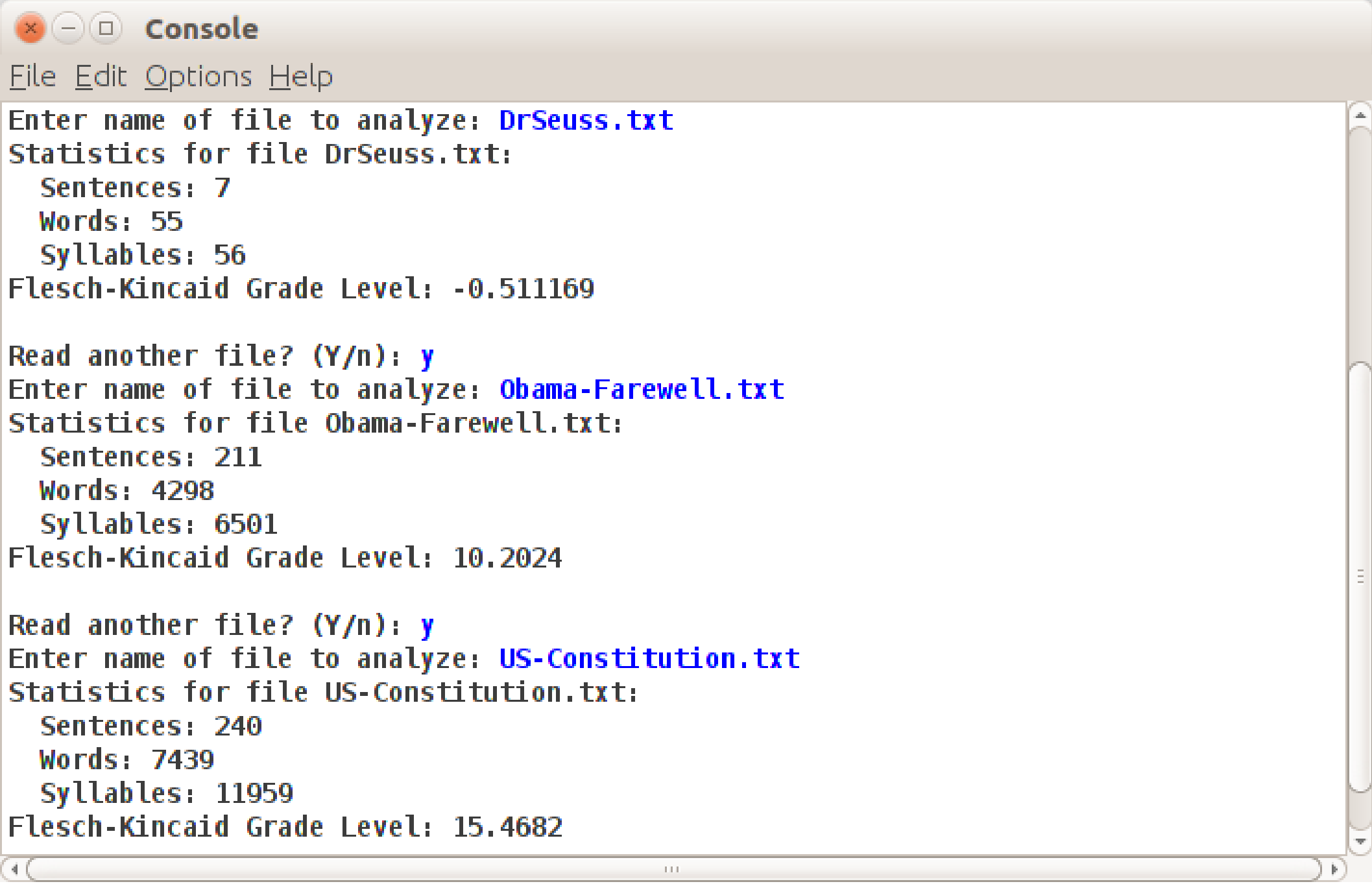
# Where to Go from Here

# Announcements

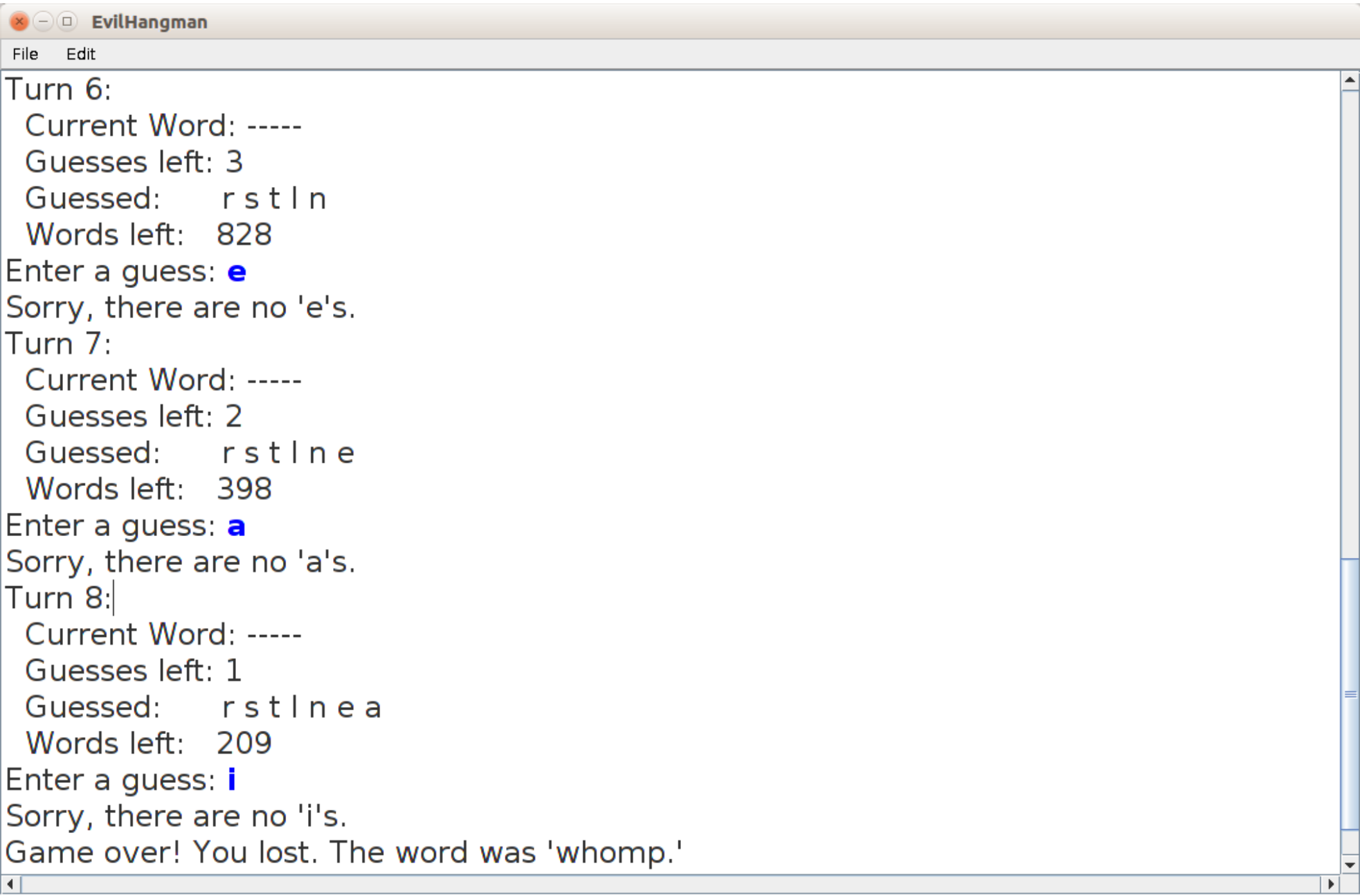
- Assignment 7 is due right now.
  - ***Congratulations! You've completed the last assignment of the quarter!***
- Final exam is this upcoming Monday from 8:30AM – 11:30AM. We'll email out exam locations later today.
  - Closed-book, closed-computer. You get one double-sided sheet of 8.5" × 11" notes with you.
  - We'll include two reference sheets with the exam: the library reference from the midterm and an algorithms reference that's now up online.

# Goals for this Course

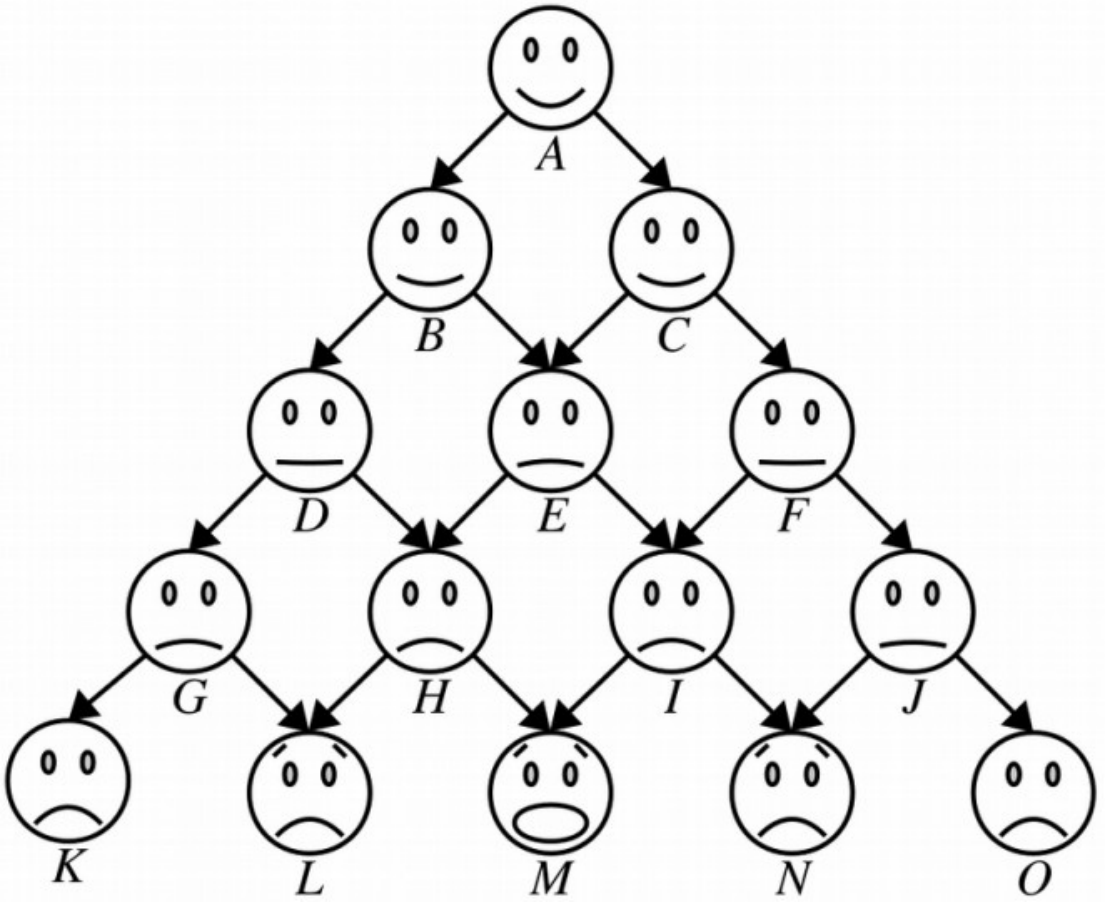
- ***Learn how to model and solve complex problems with computers.***
- To that end:
  - Explore common abstractions for representing problems.
  - Harness recursion and understand how to think about problems recursively.
  - Quantitatively analyze different approaches for solving problems.

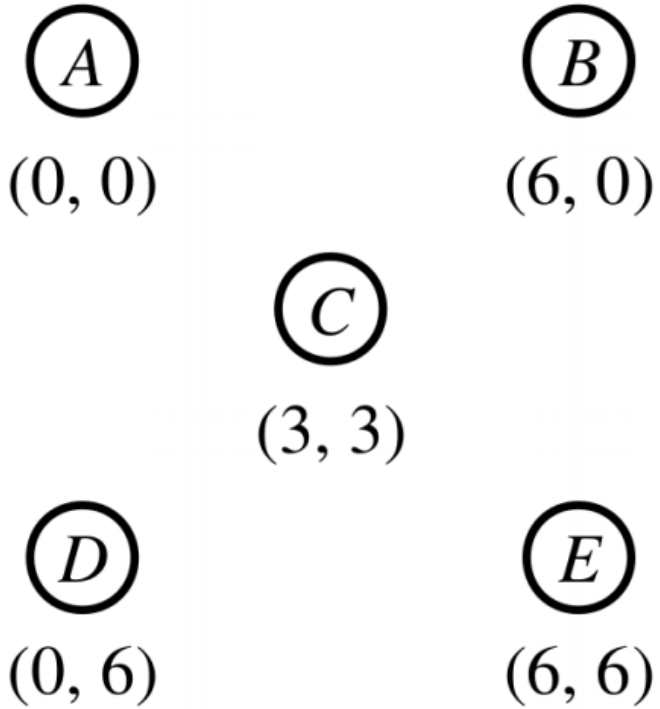



***Assignment 1***: Strings, Streams, and Recursion

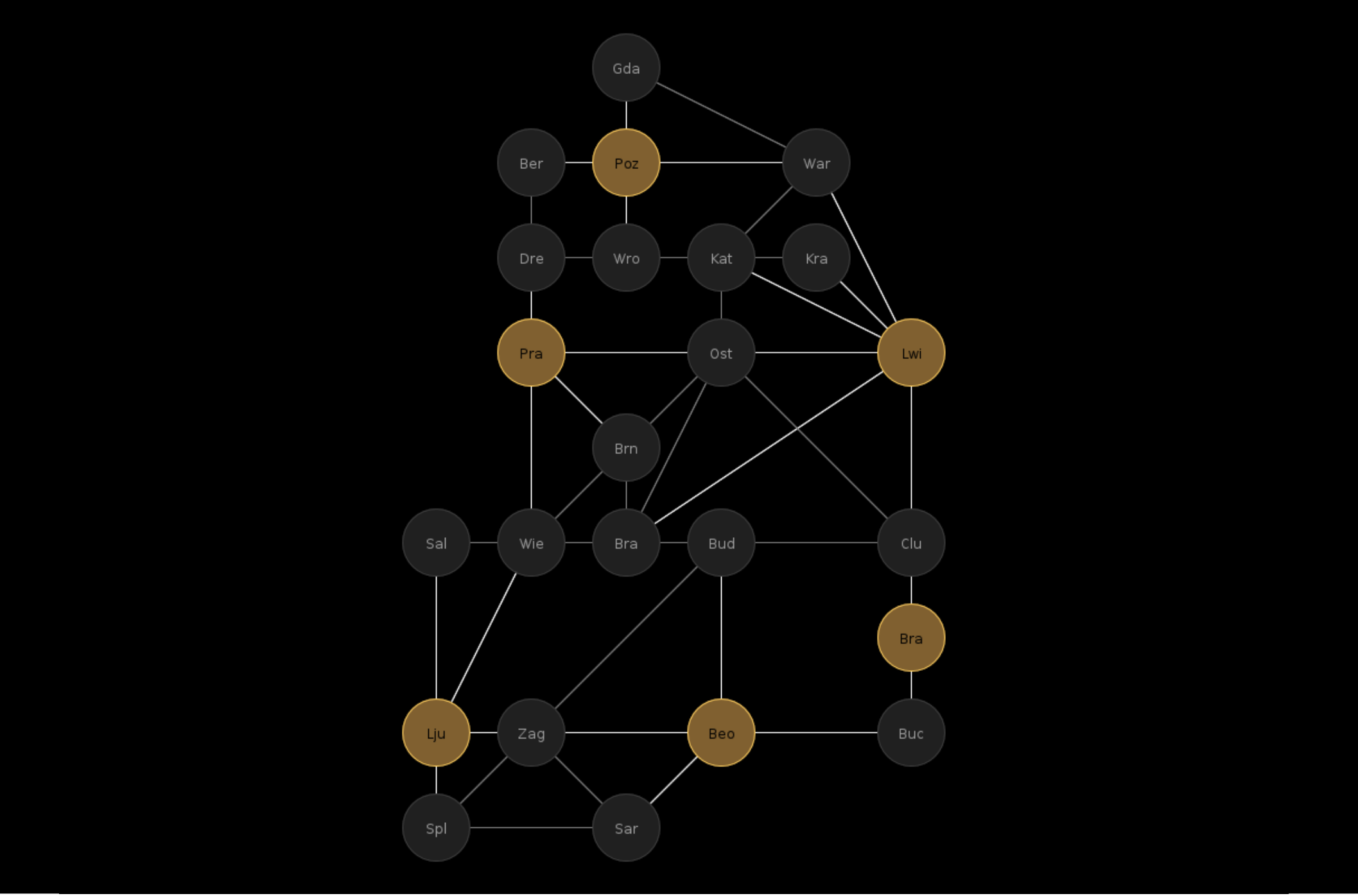



# ***Assignment 2***: Container Types

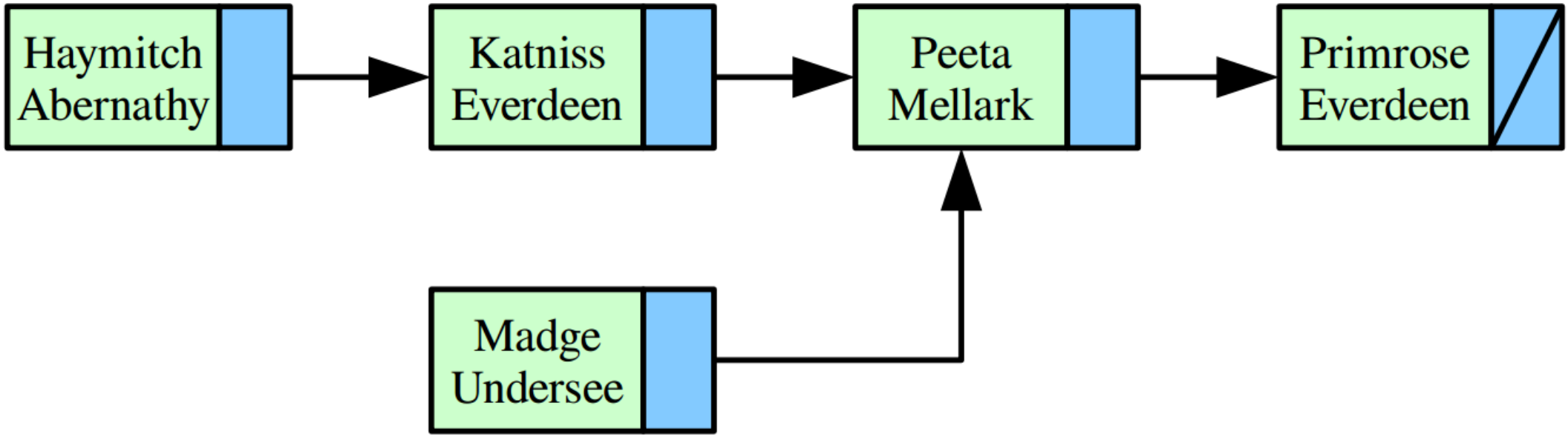


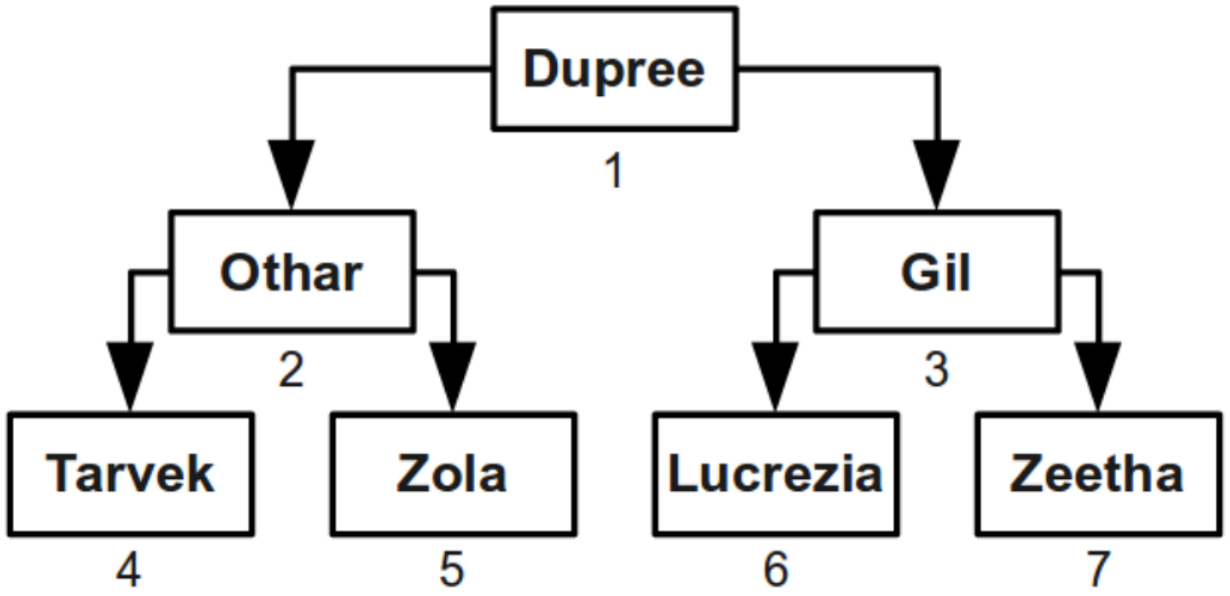




***Assignment 3***: Memoization, Recursive Optimization

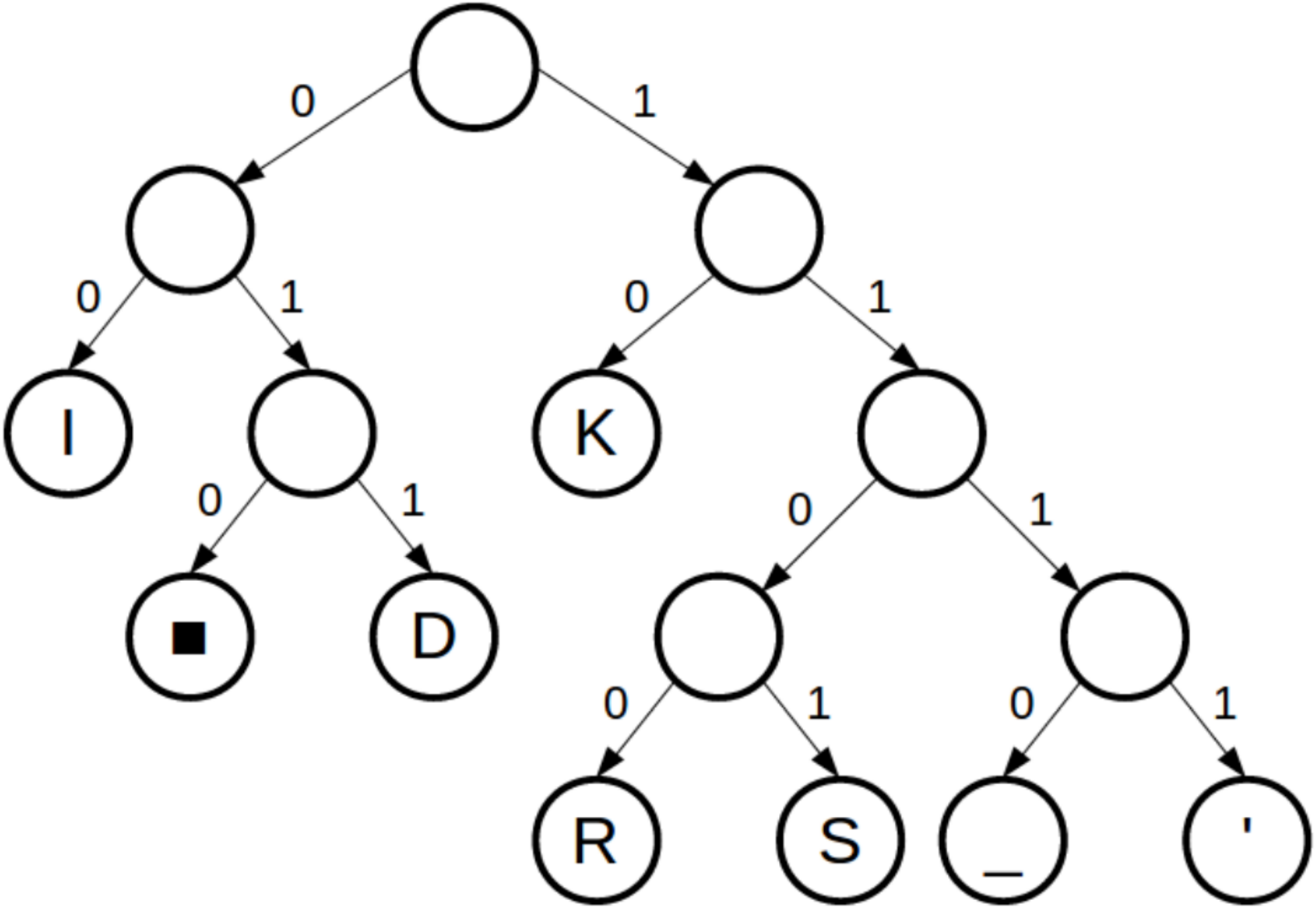



# ***Assignment 4***: Recursive Backtracking

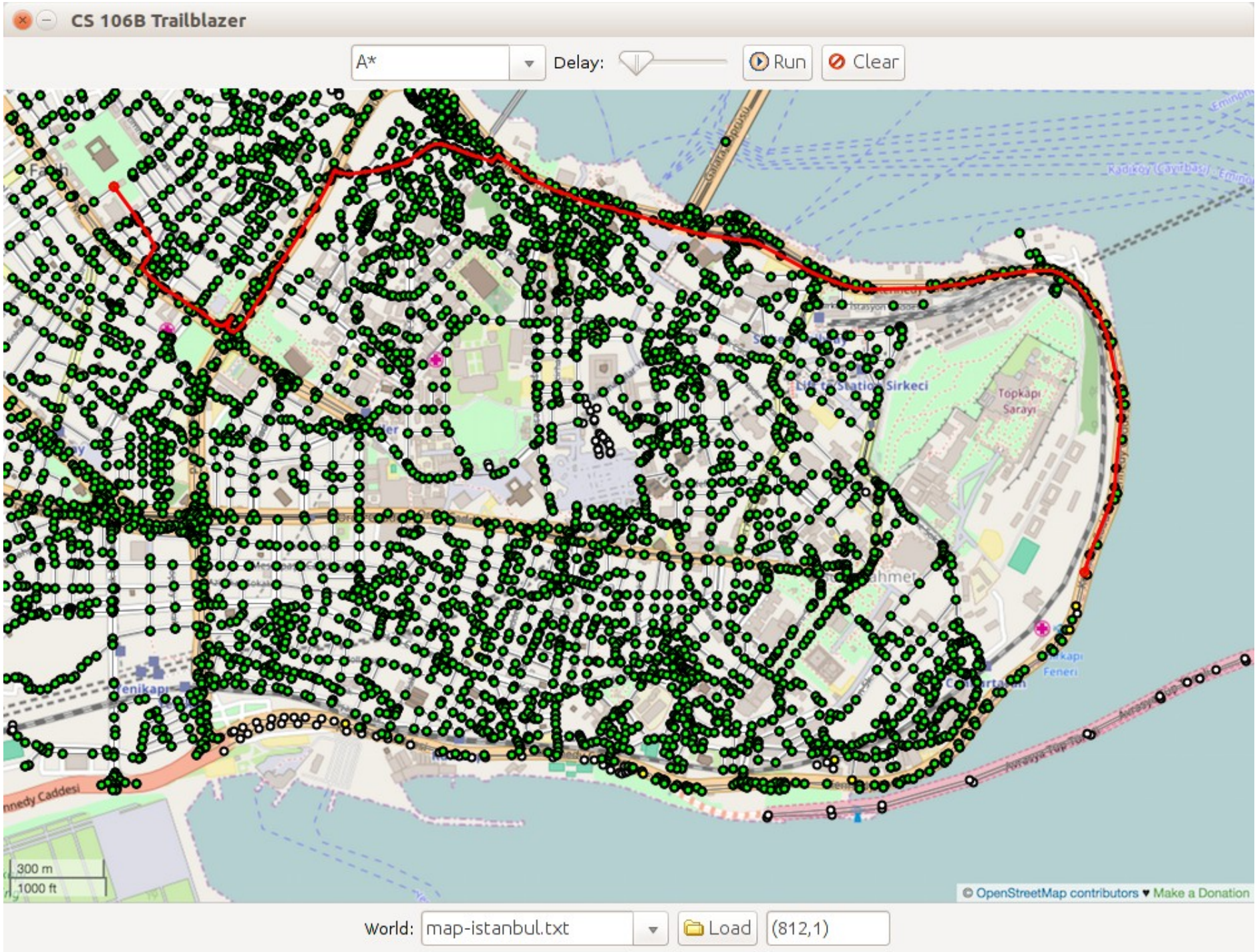



***Assignment 5***: Linked Lists, Dynamic Arrays




***Assignment 6***: Binary Trees, Priority Queues




# ***Assignment 7***: Graphs, Pathfinding Algorithms

# What We've Covered

Strings

Streams

Recursive Problem-Solving

Stacks

Queues

Vectors

Maps

Sets

Lexicons

# What We've Covered

Recursive Graphics

Recursive Enumeration

Recursive Optimization

Recursive Backtracking

Big-O Notation

Selection Sort

Insertion Sort

Mergesort

# What We've Covered

Designing Abstractions

Constructors

Destructors

Dynamic Allocation

Dynamic Arrays

Linked Lists

Binary Search Trees

Hash Tables

# What We've Covered

Graph Representations

Breadth-First Search

Depth-First Search

Dijkstra's Algorithm

A* Search

Minimum Spanning Trees

Kruskal's Algorithm

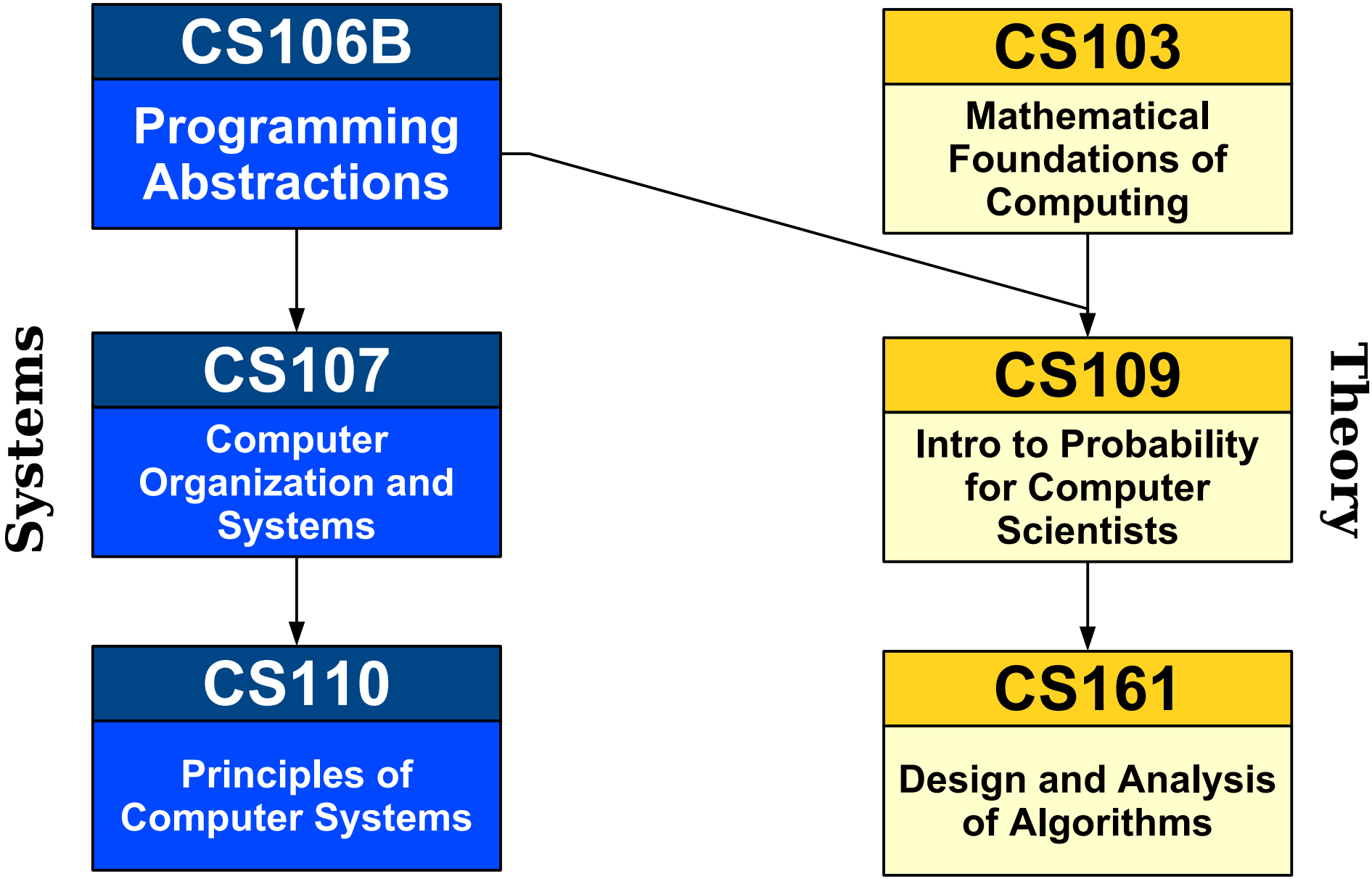
Programming is all about exploring new ways to model and solve problems.

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

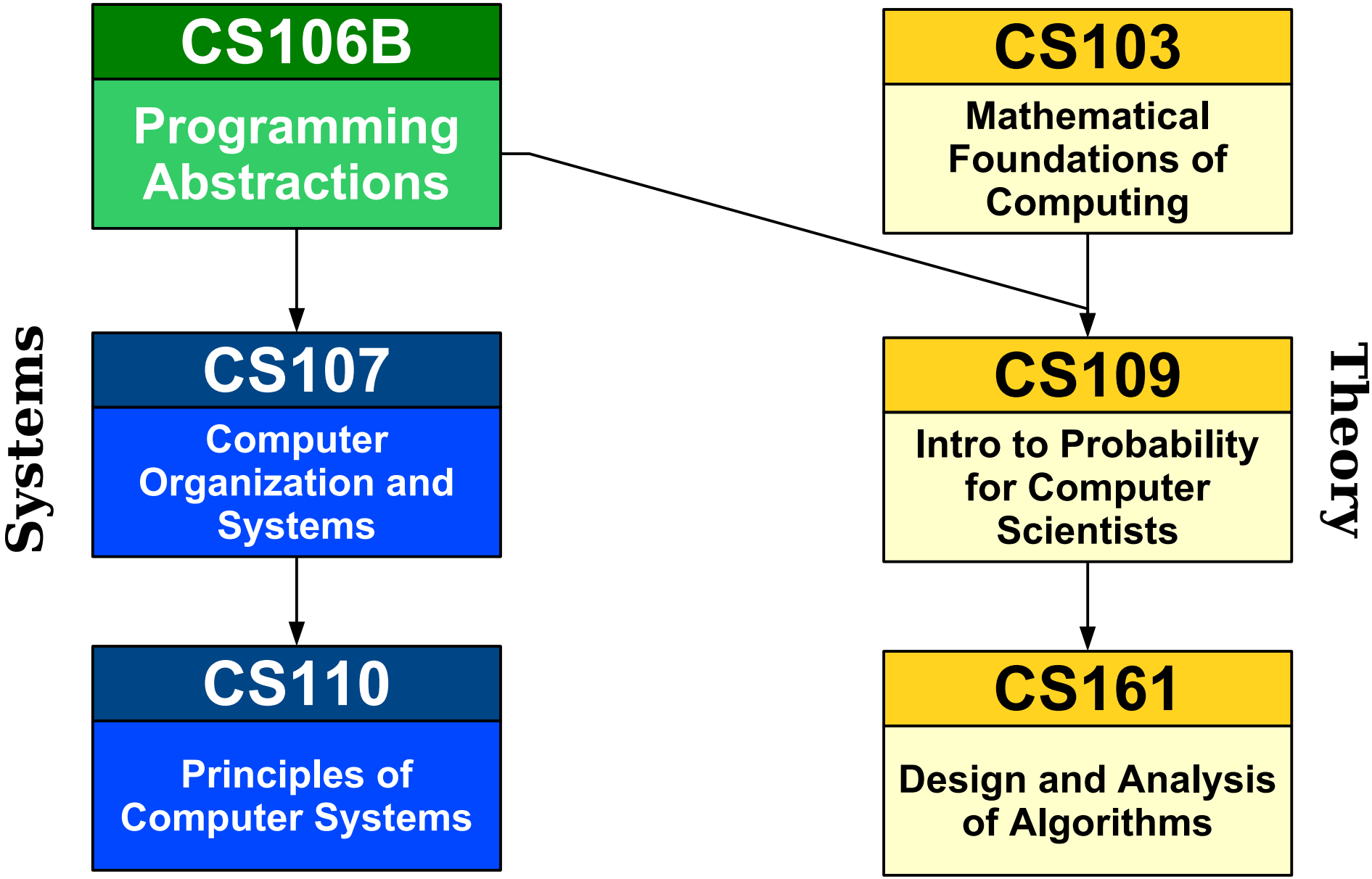
***So what comes next?***

# Courses to Take

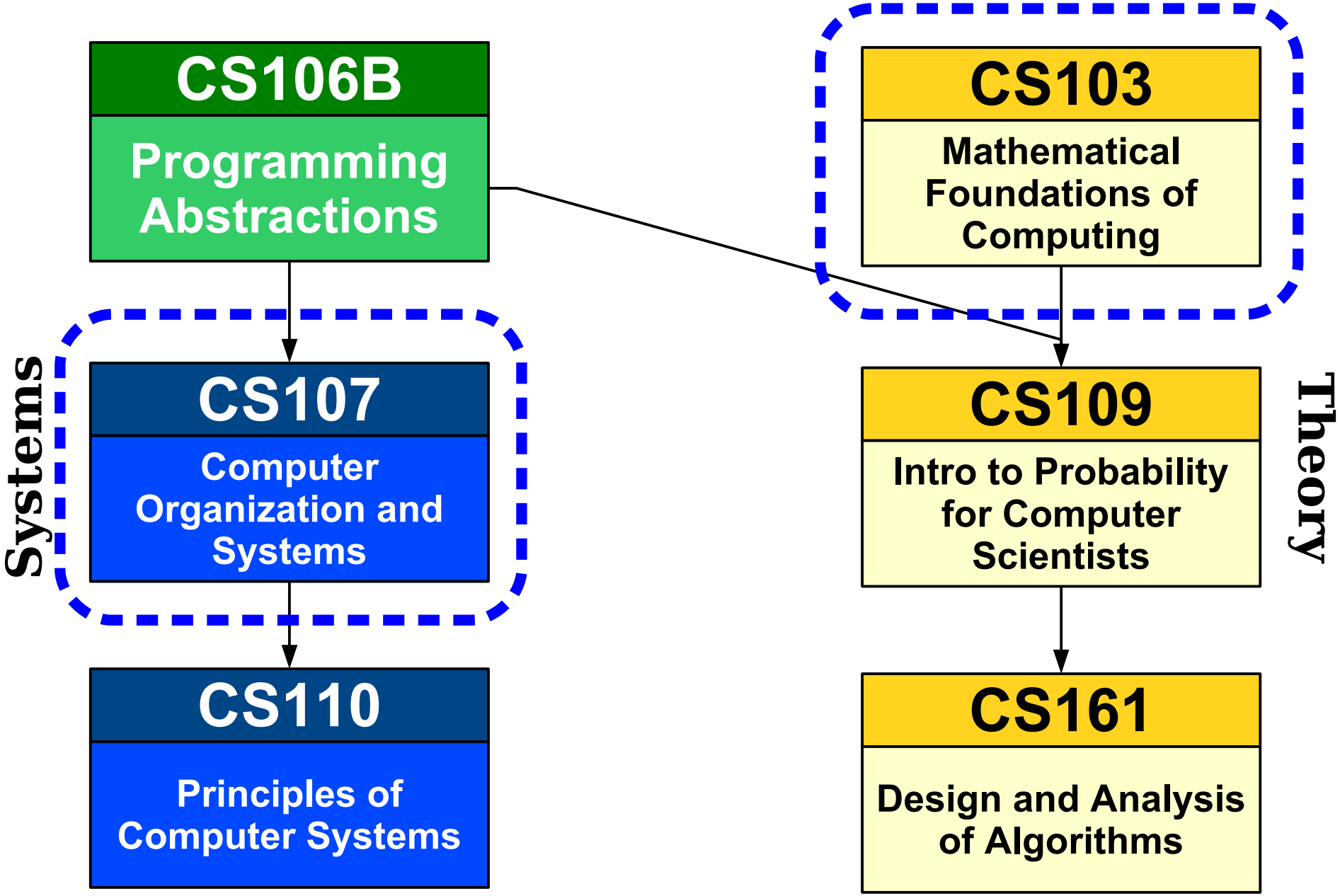
# The CS Core

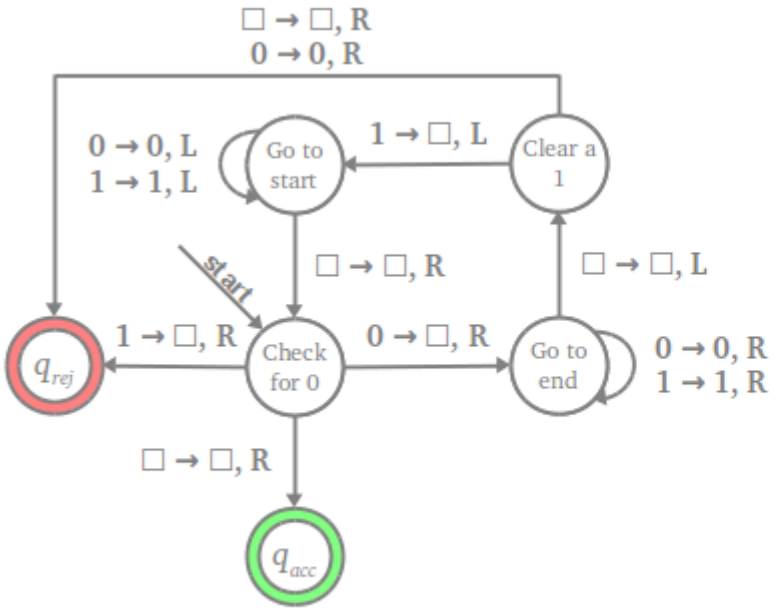

CS106B
Programming Abstractions
CS103
Mathematical Foundations of Computing
CS107
Computer Organization and Systems
CS109
Intro to Probability for Computer Scientists
CS110
Principles of Computer Systems
CS161
Design and Analysis of Algorithms
Systems
Theory

# 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?***

The mathematical nature of infinity has practical, real-world 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 hard. It does not eat kittens.
- *Don't be afraid to try CS107!*

# Other CS Courses

# CS193

- 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
  - CS193X: Web Programming Fundamentals
- 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.
- Taught by one of our SLs, who’s crazily good at this!

# 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! ✓

# 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.

# 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 financial security.
- 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, 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?

- Aeronautics and Astronautics
- Biochemistry
- Bioengineering
- Biology
- Biomedical Informatics
- Business Administration
- Chemical Engineering
- Chemistry
- Chinese
- Civil and Environmental Engineering
- Computational and Mathematical Engineering
- Computer Science
- Creative Writing
- East Asian Studies
- Economics
- Electrical Engineering
- Energy Resources Engineering
- Engineering
- Environment and Resources
- Feminism, Gender, and Sexuality Studies
- Film and Media Studies
- German Studies
- Human Biology
- Immunology
- International Policy Studies
- Law
- Management Science and Engineering
- Materials Science and Engineering
- Mathematical and Computational Sciences
- Mechanical Engineering
- Medicine
- Music
- Petroleum Engineering
- Physics
- Political Science
- Psychology
- Public Policy
- Science, Technology, and Society
- Statistics
- Stem Cell Biology and Regenerative Medicine
- Symbolic Systems
- Theater and Performing Studies
- ***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?