Welcome to
CS 3000: Algorithms & Data!

Section 1

Instructor Tim LaRock (he/him/his)

larock.t@northeastern.edu

bit.ly/cs3000syllabus

Zoom Notes

I will be recording our Zoom lectures.

Keep both your video and audio muted **at all times** unless you are speaking.

- Multiple video streams increases the bandwidth required for a smooth video.
- As I understand it:
  - If you are muted, you are not part of the recording.
  - If you unmute your video or audio, you will be recorded.

If you have a question, **use the chat box** to either (a) write your question directly or (b) indicate you would like to ask a question out loud.

- I prefer the chat to the “raise hand” feature because it is persistent.

The Zoom chat is always archived. I will probably delete it very soon after recording.

Today

Brief instructor introduction

Some presentation of the what/why of Algorithms

Course logistics + questions

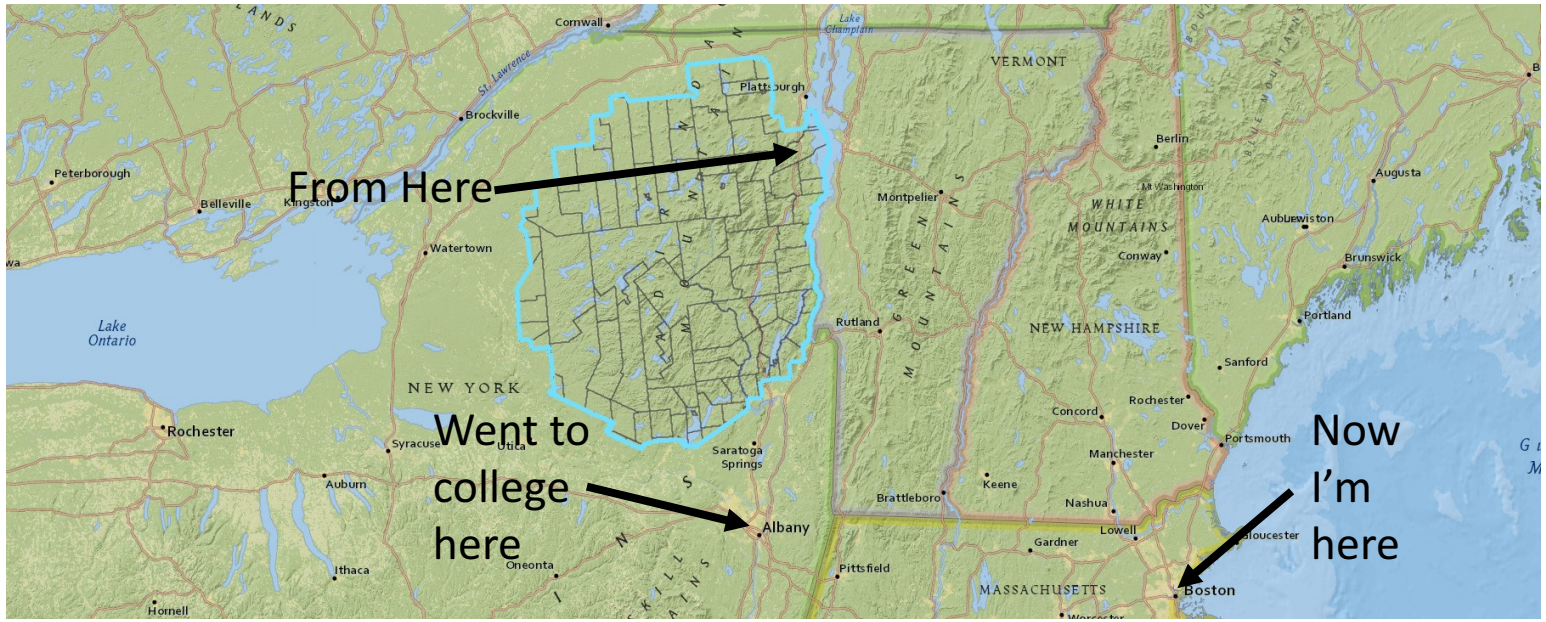
Some content

Me

Tim LaRock (he/him/his)

Just call me Tim!

I grew up in the Adirondack Mountains



Researcher at the Network Science Institute

Usually: Understanding how *things* move through networks, e.g. how a ship moves through a network of ports.

Lately: Analyzing mobility data to understand the impact of mobility restrictions on the spread of COVID-19.

Now: Your instructor!

This Course

We are going to learn about **algorithms**, which are sets of instructions for how to manipulate **data**

> Erickson definition: “An algorithm is an explicit, precise, unambiguous, mechanically-executable sequence of elementary instructions, usually intended to accomplish a specific purpose.”

Specifically, we will cover things like…

- Transforming problems from informal descriptions to formal mathematical descriptions
- Formulating strategies for solving formal problems efficiently
- Understanding, designing, and choosing appropriate data structures for our solutions
- Proving the correctness of a solution mathematically
- Determining the complexity in terms of (i) running time and (ii) memory requirements for a proposed solutions
- Categorize problems and solutions based on classes of complexity
- …and much more!

Why?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?
2. What does a solution to the problem look like?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?
2. What does a solution to the problem look like?
3. How we can go from the input to a solution?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?
2. What does a solution to the problem look like?
3. How we can go from the input to a solution?
4. Can we guarantee that a solution is correct?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?
2. What does a solution to the problem look like?
3. How we can go from the input to a solution?
4. Can we guarantee that a solution is correct?
5. Can we guarantee a solution is found in a reasonable amount of time?

Reason 1: Effective communication is important!

In order to implement anything, we first need to communicate clearly:

1. What is the problem we are trying to solve?
2. What does a solution to the problem look like?
3. How we can go from the input to a solution?
4. Can we guarantee that a solution is correct?
5. Can we guarantee a solution is found in a reasonable amount of time?
6. And more…

In this course, we learn mathematical techniques that allow us to effectively communicate answers to these questions.

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

“Recipe” is a classic example of an algorithm

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Algorithm for constructing 1 PB&J sandwich

Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool

Algorithm:
1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. Cut in half if desired.

Output: PB&J

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Algorithm for constructing 1 PB&J sandwich

Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool, **desire, direction**

Algorithm:
1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. ~~Cut in half if desired.~~
CutInHalf(desire, direction)

Output: PB&J

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Assume it takes 2 minutes to make a sandwich.

Algorithm for constructing 1 PB&J sandwich

Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool

Algorithm:
1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. Cut in half if desired.

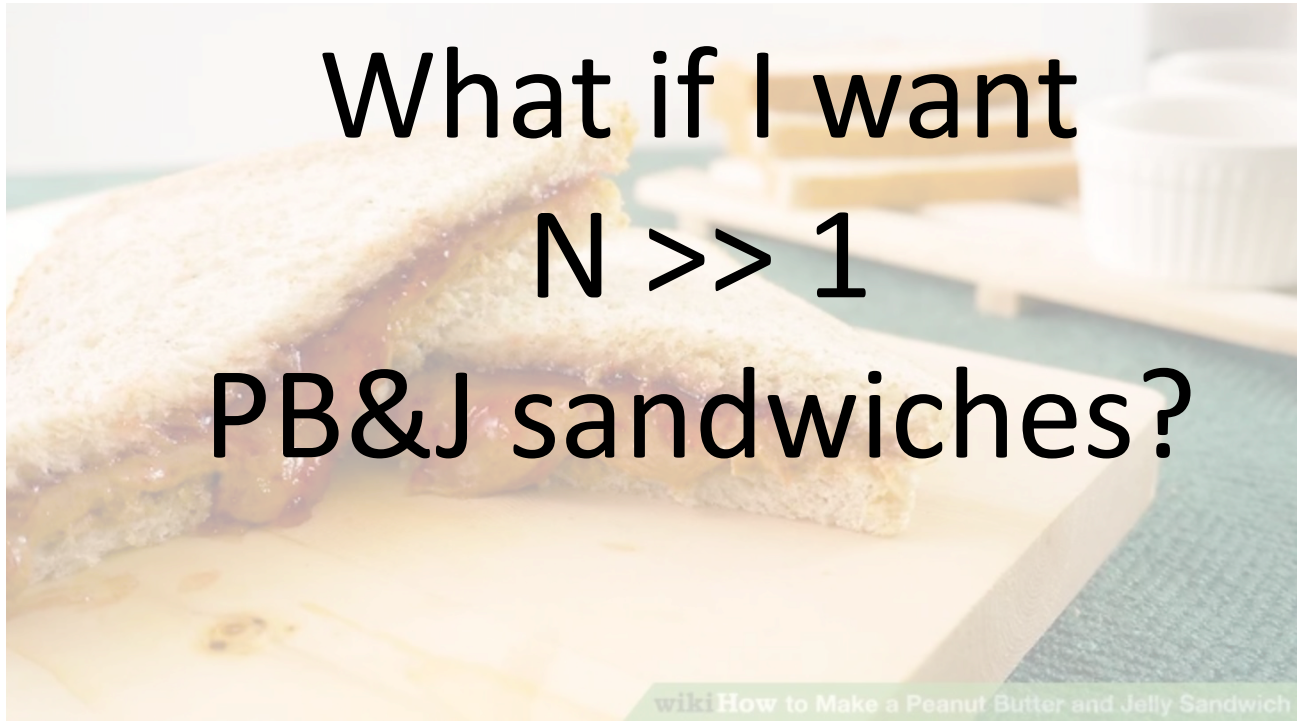
Output: PB&J

Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Assume it takes 2 minutes to make a sandwich.

Algorithm for constructing 1 PB&J sandwich



Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool

Algorithm:
1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. Cut in half if desired.

Output: PB&J

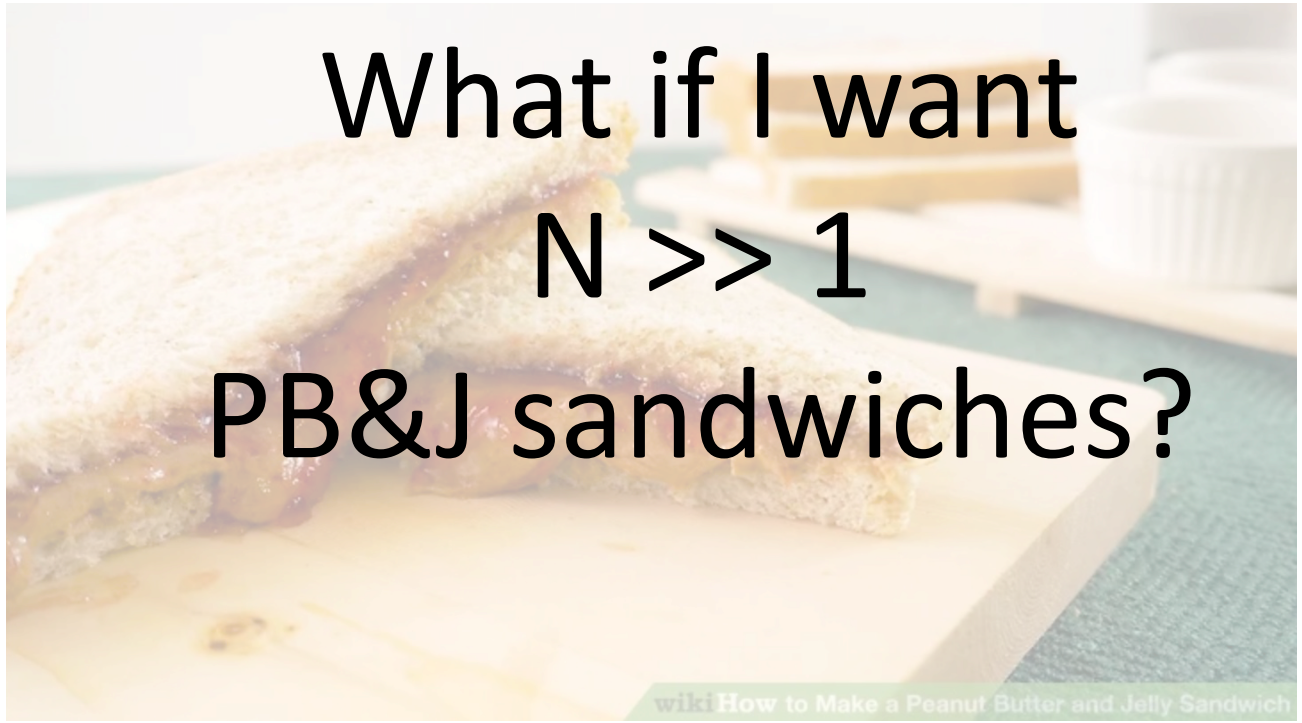
Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Assume it takes 2 minutes to make a sandwich.

Algorithm for constructing N PB&J sandwiches.

Runtime: N*2 minutes



REPEAT N TIMES:

Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool

Algorithm:

1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. Cut in half if desired.

Output: PB&J

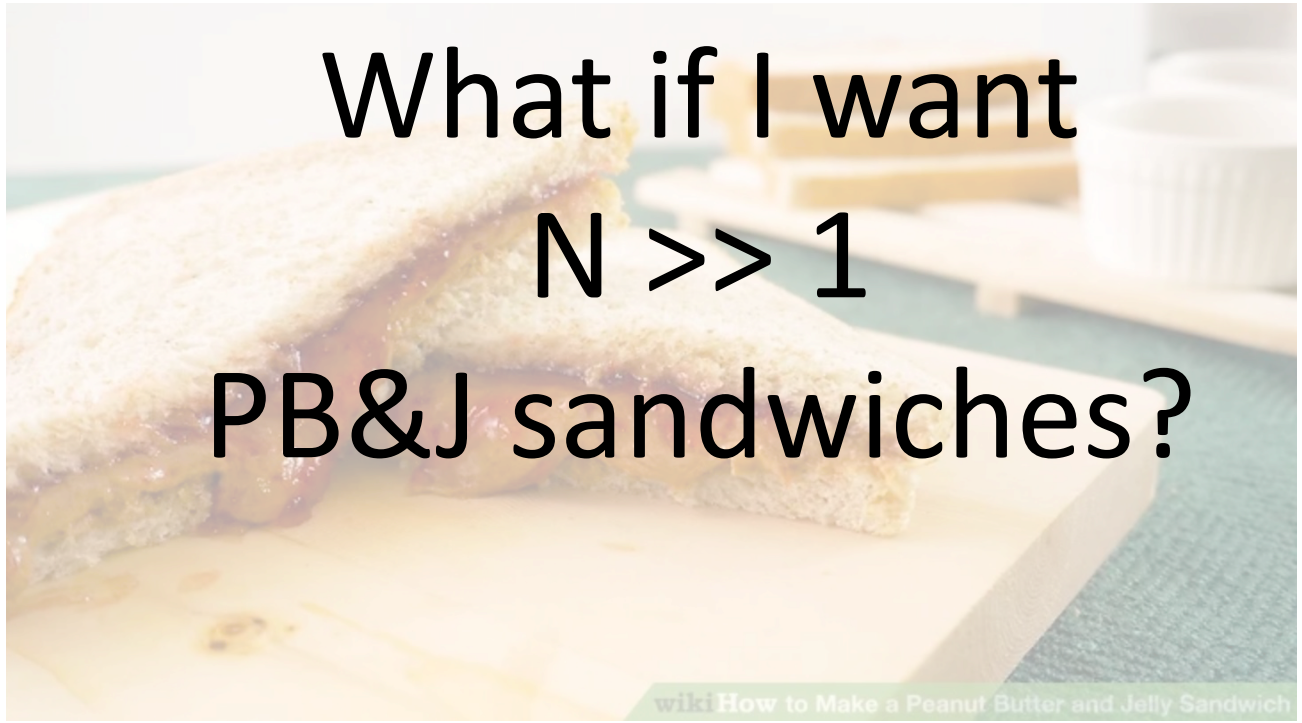
Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Runtime: N*2 minutes

Probably fine if I want less than N=10 sandwiches.

If I want N=1000, I will quickly run out of **resources** and **time**!



REPEAT N TIMES:

Input: 2 slices of bread, jar of PB, jar of jelly, spreading tool

Algorithm:

1. Use the tool to spread PB on one slice of bread
2. Use the tool to spread jelly on the slice of bread without peanut butter
3. Put the two slices of bread together so that the PB and J are facing each other.
4. Cut in half if desired.

Output: PB&J

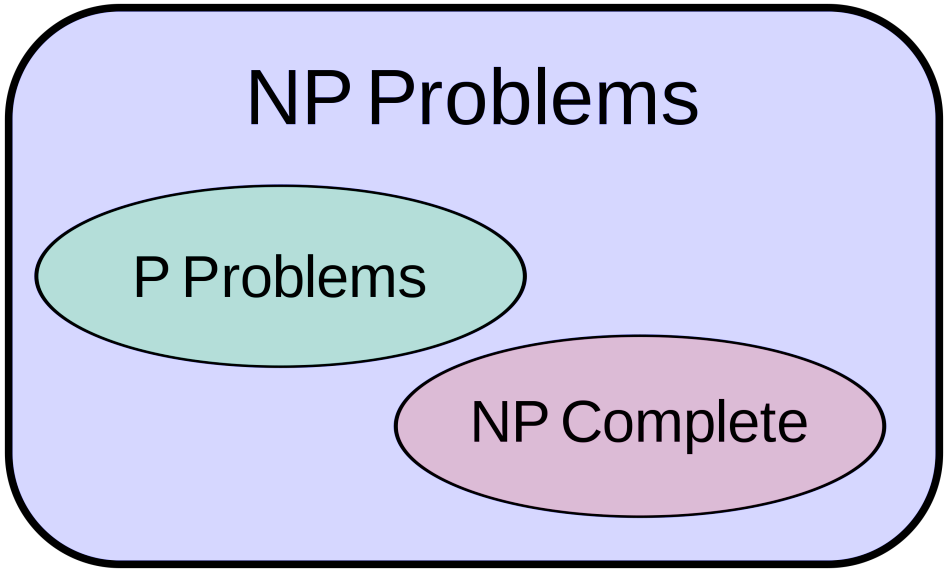
Reason 2: Efficient algorithms are important in practice!

Scalability or efficiency of an algorithm can be the difference between a computation running in 5 minutes or never finishing before the heat death of the universe.

Sometimes we don’t even know if a scalable solution to a problem could possibly exist – the techniques you learn here will give you the tools to answer that question!

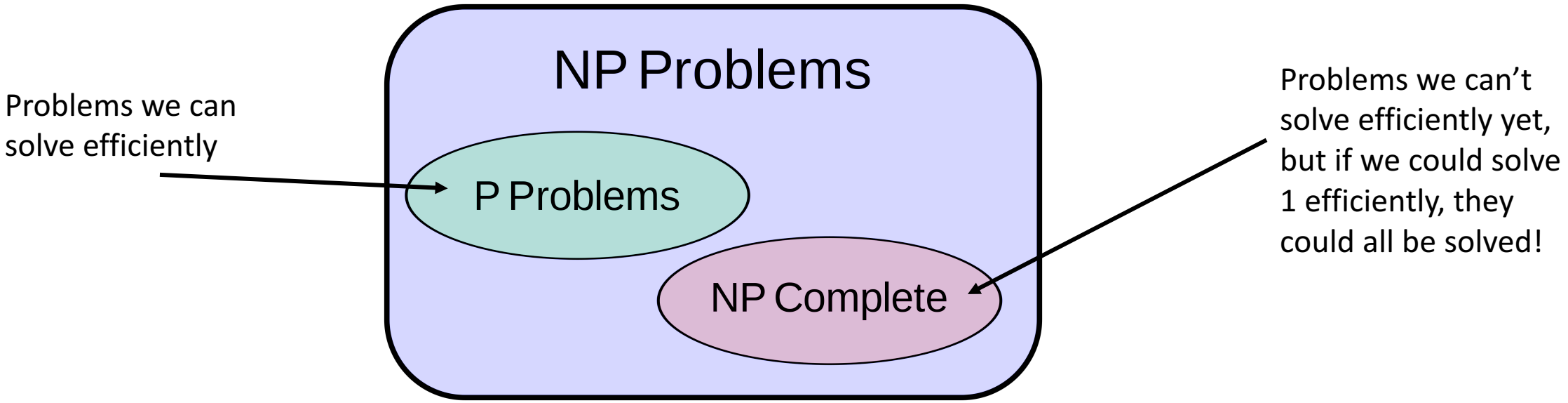
Reason 3: Algorithms/complexity theory is an interesting field of mathematics

Theoretical advances have serious practical implications (P=NP)



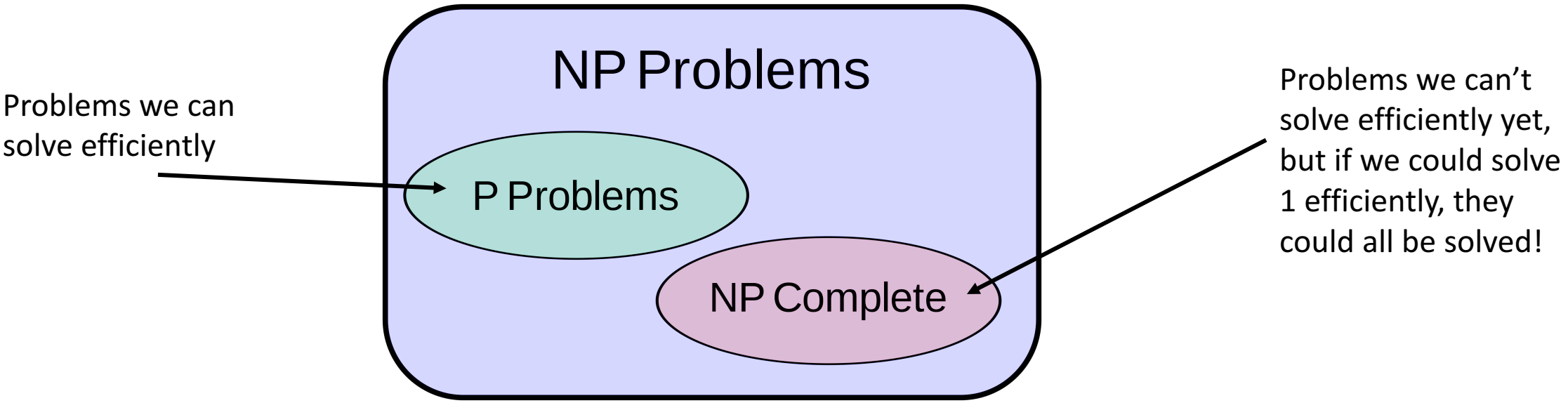
Reason 3: Algorithms/complexity theory is an interesting field of mathematics

Theoretical advances have serious practical implications (P=NP)



Reason 3: Algorithms/complexity theory is an interesting field of mathematics

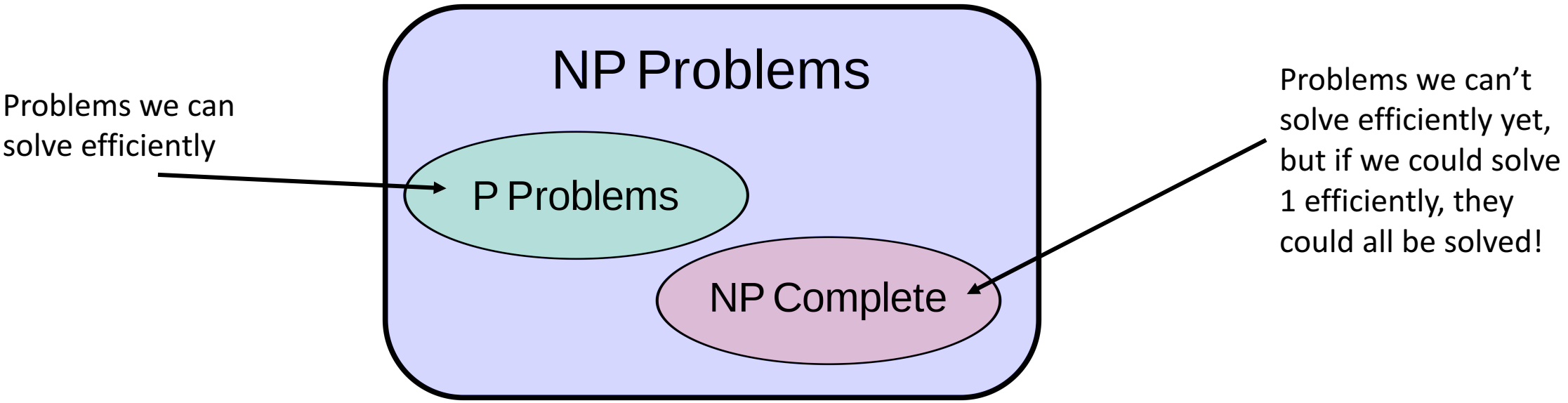
Theoretical advances have serious practical implications (P=NP)



We will likely touch on this formally towards the end of the semester, but take Theory of Computation to learn more!

Reason 3: Algorithms/complexity theory is an interesting field of mathematics

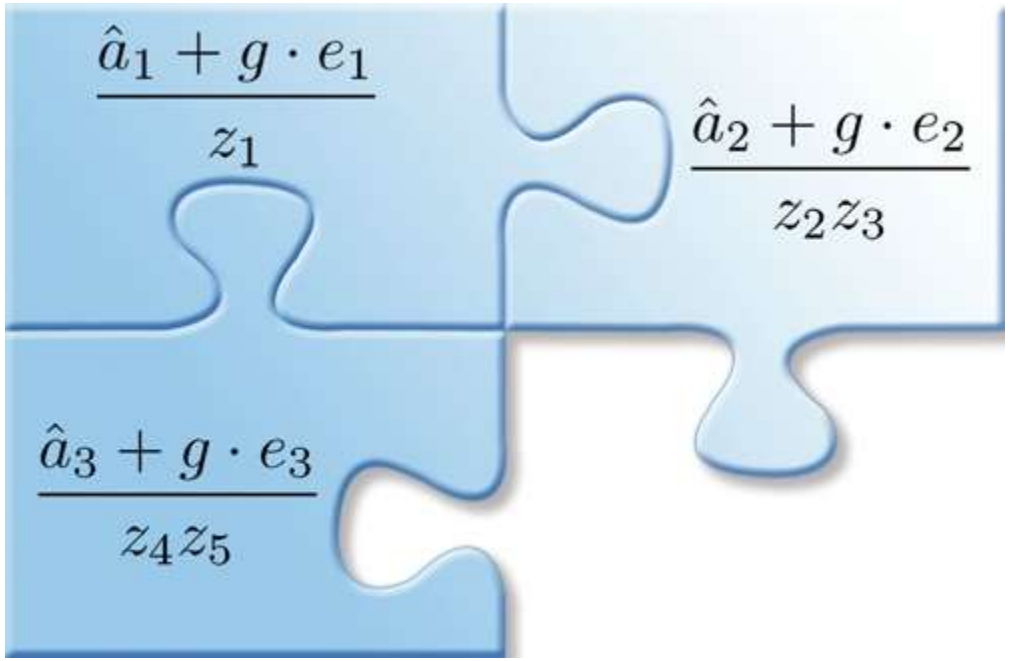
Theoretical advances have serious practical implications (P=NP)



We will likely touch on this formally towards the end of the semester, but take Theory of Computation to learn more!

Bonus Reason

Studying algorithms often feels like solving a puzzle!



Logistics

Logistics - Course Structure

Lectures (like this one) Monday – Thursday, 1:30-3:10PM

Homework – Approximately weekly (45% of grade)

Exams – 2 Midterms (15% each) and a final exam (25%)

Resources:

Course website: bit.ly/cs3000syllabus

Canvas: Contact me ASAP (larock.t@northeastern.edu) if you do not have access!

Logistics – Lecture details

Lectures (like this one) Monday – Thursday, 1:30-3:10PM

Recorded live and uploaded to Canvas

Attendance is encouraged if possible, but not required

Regardless of live attendance, it **is expected** that you have watched the lecture at some point! Anything discussed in lecture is fair game for homework/exams!

Logistics – Homework details

Assigned approximately weekly, with variation depending on timing of exams

Not meant to take you hours upon hours to complete – if you are stuck, ask for help (more to come on various ways to do so)

Collaboration **is** okay!

- Write solutions in your own words and include all collaborators names on everyone's submissions

Copying **is not** okay!!

- We reserve the right to ask you to explain any answer you submitted!

Okay to look up resources online for help, but..

- ALWAYS evaluate your sources carefully!
  - A textbook page is preferable to Wikipedia, Wikipedia is much more reliable than a stack exchange answer with 0 votes, etc. Use your judgement!
- NEVER copy solutions if you find them.
  - If you find an exact answer and can't "unsee it", **do not copy it!** Just send me an email.

Logistics – Exam details

Exams – 2 Midterms and a final

All “take home” format, meaning you will have a set time period to work on them outside of class

Similar to homework assignments, except absolutely NO collaboration is allowed and use of the internet is limited to textbooks ONLY

PLEASE DO NOT CHEAT!

Obviously we are on the honor system, and my default attitude is to trust you! But if you are caught cheating there will be severe penalties, including escalation to the College and/or University level.

Logistics – Instructor Office Hours

I will hold open office hours at the following times:

4-5 PM on Tuesdays
8:30-9:30 AM on Wednesdays
1:30-2:30 PM on Fridays

The specific structure of these hours is not decided and depends somewhat on level of demand.

You can always reach out via email to schedule a 1-1 or small group conversation with me.

Logistics – TAs and Office Hours

We have 8 Teaching Assistants for the course – they are a resource!

Name	email	Office Hour 1	Office Hour 2
Saurabha	jirgi.s@husky.neu.edu	Wednesday, 10AM-11AM	Thursday, 11AM-12PM
Ronn	jacob.r@husky.neu.edu	Wednesday, 12PM-1PM	Thursday, 12PM-1PM
Himanshu	budhia.h@husky.neu.edu	Tuesday, 4PM-5PM	Monday, 12PM-1PM
Dania	abuhijleh.d@husky.neu.edu	Monday, 9AM-10AM	Wednesday, 9AM-10AM
Drew	bodmer.d@husky.neu.edu	Thursday, 10AM-11AM	Monday, 3PM-4PM
Angela	gross.an@husky.neu.edu	Wednesday, 12PM-1PM	Friday,Friday 2PM-3PM
Luke	boyer.l@husky.neu.edu	Monday, 7PM-8PM	Tuesday, 8PM-9PM
Kevin	hui.k@husky.neu.edu	Wednesday, 6PM-7PM	Thursday, 6PM-7PM

Canvas

Northeastern's replacement for Blackboard

New to me, new to 1/3 of you (according to entry form)

Plan to use it for a couple of things:

- Assignment submission
- Grades
- Online discussions

Everything else will be at: bit.ly/cs3000syllabus

Logistics – Online Discussion

Canvas has online discussion boards, I encourage you to post there when you have questions you aren't ready to bring to me/the TAs yet!

Obviously it is not okay for anyone to post solutions on the discussion board, but clarifying and helping guide classmates is okay.

Textbooks

I will assign some reading from two freely available books:

1. Algorithms by Jeff Erickson
2. Introduction to Algorithms by Cormen, Leiserson and Rivest (CLR)

See the syllabus and Canvas for links, or just search for the titles.

Algorithm Design by Tardos and Kleinberg is no longer required

- If you got a copy, it is a great resource that I encourage you to use!

Answers to entry form questions

50 people filled it out – thank you!

Answers to entry form questions

- Will this course require a 0-credit recitation like in the fall, and will we also need to take this?
  - To my knowledge, there is no 0-credit recitation for this course.
- Approximately how many homework assignments will we have in this class?
  - Between 5-7, with lowest grade dropped.
- How many hours should I take per day to compete the homework?
  - Homework assignments are not meant to take over your life. I expect between 1-5 hours over the course of a week (e.g. ~ 1 hr per day) to be enough.
- When are homework assignments typically due?
  - Still working out the full schedule, but most will likely be due either Fridays or Mondays.

Answers to entry form questions

- When will we know the dates/times for midterms/finals?
  - I will try to finalize the schedule and let you know ASAP.
- Is there any coding in this class?
  - No, sorry. I may ask a coding question (in your fav language) as extra credit, but this class is not intended to teach you about programming.
- Can I get some useful websites for learning LaTeX?
  - Yes. I will also try to spend a few minutes in one of these early lectures talking about the very basics.
- A couple of people expressed frustration with assumed knowledge across courses.
  - I will try to explain concepts as I introduce them, but there are prerequisites for the course so some familiarity with the topics covered in those is assumed! Feel free to ask in the chat if you aren't sure what I am talking about.

More questions before we move to content?

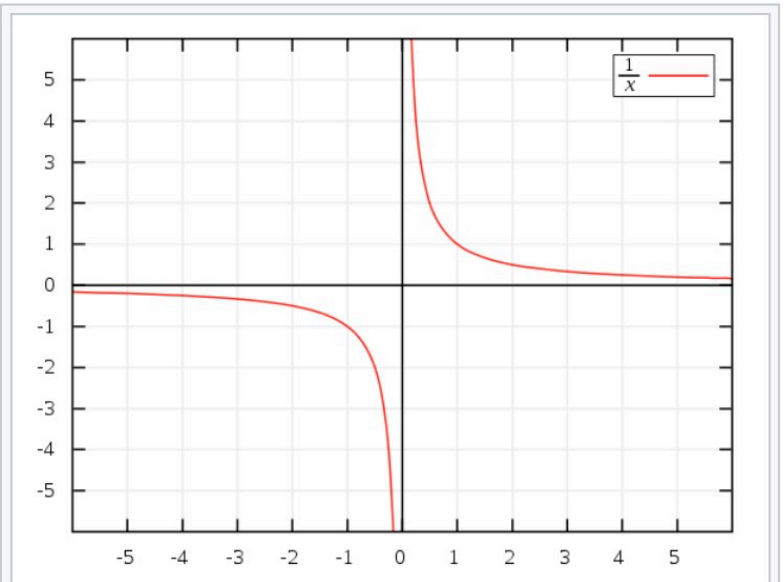
Content

Refresher: What is an asymptote?

"…an **asymptote** (/ˈæsɪmptoʊt/) of a curve is a line such that the distance between the curve and the line approaches zero as one or both of the *x* or *y* coordinates tends to infinity." – Asymptote on Wikipedia

Refresher: What is an asymptote?

"…an **asymptote** (/ˈæsɪmptoʊt/) of a curve is a line such that the distance between the curve and the line approaches zero as one or both of the *x* or *y* coordinates tends to infinity." – Asymptote on Wikipedia



$f(x) = \frac{1}{x}$ graphed on Cartesian coordinates.
The *x* and *y*-axes are the asymptotes.

Asymptotes and Runtimes

What do asymptotes have to do with algorithms?

Sorting

Sorting is extremely important to computer users and scientists!

Sorting

Sorting is extremely important to computer users and scientists!

A simple example: Finding the median of a set of numbers

```
Input: L, an array of N numbers
Output: The median of L
Procedure:
    1. Sort L
    2. If N is odd, return the number at L[⌈N/2⌉]
    3. If N is even, return the mean of the
       numbers at L[⌈N/2⌉] and L[⌈N/2⌉+1]
```

Bubble Sort

Idea: Items “bubble up” to the top as they are sorted pairwise

Bubble Sort

Idea: Items “bubble up” to the top as they are sorted pairwise

```
Input: L, an array of N numbers
Output: L sorted in ascending order
Procedure:
    Let swapped = True
    while swapped = True:
        swapped = False
        for i from 1 to N-1:
            if L[i] > L[i+1]:
                Swap L[i] and L[i+1]
                swapped = True
```

Bubble Sort Example

5 3 6 2 2

Bubble Sort Analysis

```
Input: L, an array of N numbers
Output: L sorted in ascending order
Procedure:
    Let swapped = True
    while swapped = True:
        swapped = False
        for i from 1 to N-1:
            if L[i] > L[i+1]:
                Swap L[i] and L[i+1]
                swapped = True
```

Next Time

A better approach to sorting

Divide and Conquer Algorithms

More asymptotic analysis

Suggested Reading:

- Erickson book: Introduction thru Chapter 1.1
- CLR book: Introduction thru Chapter 2

Homework 1: To be released tomorrow, due next Monday