Record?

CS3000: Algorithms & Data
Drew van der Poel

Lecture 1:
- Course Overview
- Counting Students
- Proof by Induction

May 10, 2021

Me

- Name: Drew van der Poel
  - Feel free to call me Drew
  - Second year at Northeastern
  - Office Hours: Wednesday 330-5pm
    Location: Online (Teams)
  - Fun Fact: Won 2 of my 3 fantasy football leagues in 2020

- Research:
  - Parameterized graph algorithms

The TA Team (all OH on MS Teams)

- **Kristen Colavita**
  - Office Hours: Tuesday 8-10pm
Thursday 8-10pm
  - Fun Fact: likes to spend free time knitting and crocheting Super Mario characters!

- **Zhenxiang (Steven) Guan**
  - Office Hours: Tuesday 9-11am
Saturday 10am-12pm
  - Fun Fact: just took up snowboarding last winter!

The TA Team (all OH on MS Teams)

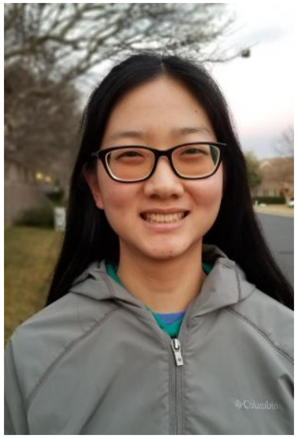
- **Anne Lee**
  - Office Hours: Wednesday 6-9pm
  - Fun Fact: lives ten minutes from the beach in Southern California!

- **Ariana Lozner**
  - Office Hours: Thursday 6-8pm
    Saturday 12-1pm
  - Fun Fact: plays 5 instruments (guitar, bass, violin, piano, and drums)!

The TA Team (all OH on MS Teams)

- **Sameer Marathe**
  - Office Hours: Tuesday 11am-1pm
    Thursday 11am-1pm
  - Fun Fact: knows 5 languages, completely self-taught!

- **Amy Min**
  - Office Hours: Sunday 12-3pm
    Monday 5-7pm
  - Fun Fact: has a 15 year-old pet hermit crab!



The TA Team (all OH on MS Teams)

- **Tingwei Shi**
  - Office Hours: Sunday 9-11am
Friday 9-11am
  - Fun Fact: only plays one video game!

- **Rishabh Dutta**
  - Office Hours: Saturday 2-4pm
Sunday 3-5pm
  - Fun Fact: eats pizza crust first!

- **Prateek Gulati**
  - Office Hours: Monday 12-1pm
Wednesday 10am-1pm
  - Fun Fact: only plays chess on his gaming rig!

Algorithms

- What is an algorithm?

> *An explicit, precise, unambiguous, mechanically-executable sequence of elementary instructions for solving a computational problem.*
>
> *-Jeff Erickson*

- Essentially all computer programs are algorithms for some computational problem.

Algorithms

- What is Algorithms?

The study of how to solve computational problems.

computational problem – can be solved by computer; *precise, well-defined*

Ex.

* determining if a number is prime

* sorting last names in alphabetical order

* finding shortest route by distance between two cities

Algorithms

- What is CS3000: Algorithms?

The study of how to solve computational problems.

Algorithms

- What is CS3000: Algorithms?

  The study of how to solve computational problems.

  - Abstract and formalize **computational problems**
  - Identify broadly useful algorithm **design principles** for solving computational problems
  - Rigorously **analyze properties** of algorithms
    - This Class: correctness, running time, space usage
    - Beyond: extensibility, robustness, simplicity,…

Algorithms

- What is CS3000: Algorithms?

  The study of how to solve computational problems.
  How to rigorously prove properties of algorithms.

  - **Proofs** are about understanding and communication, not about formality or certainty
    - Rigorous, complete explanations (resilient to doubters)
    - *Different* emphasis from courses on logic
    - We'll talk a lot about proof techniques and what makes a correct and convincing proof

- Problems:

- Alg. techniques:

- Analysis:

- Proof techniques:

Algorithms

- That sounds hard. Why would I want to do that?

- Build Intuition/Become a Better Programmer:
  - How to decipher new **problems** in the real world?
  - Which **design techniques** work well in different cases?
  - How to **compare** different solutions?
  - How to **know** if an algorithm is correct?

Algorithms

- That sounds hard. Why would I want to do that?

- Improve Communication:
  - How to convince someone that a solution is **correct**?
  - How to convince someone that a solution is **best**?

Someone may be an interviewer, boss, or co-worker!

Algorithms

- That sounds hard. Why would I want to do that?

- Learn Problem Solving / Ingenuity

    - "When I look at LeBron James, I see his cleverness… What makes the difference for me, when I see him, he's clever." – Thierry Henry

Algorithms

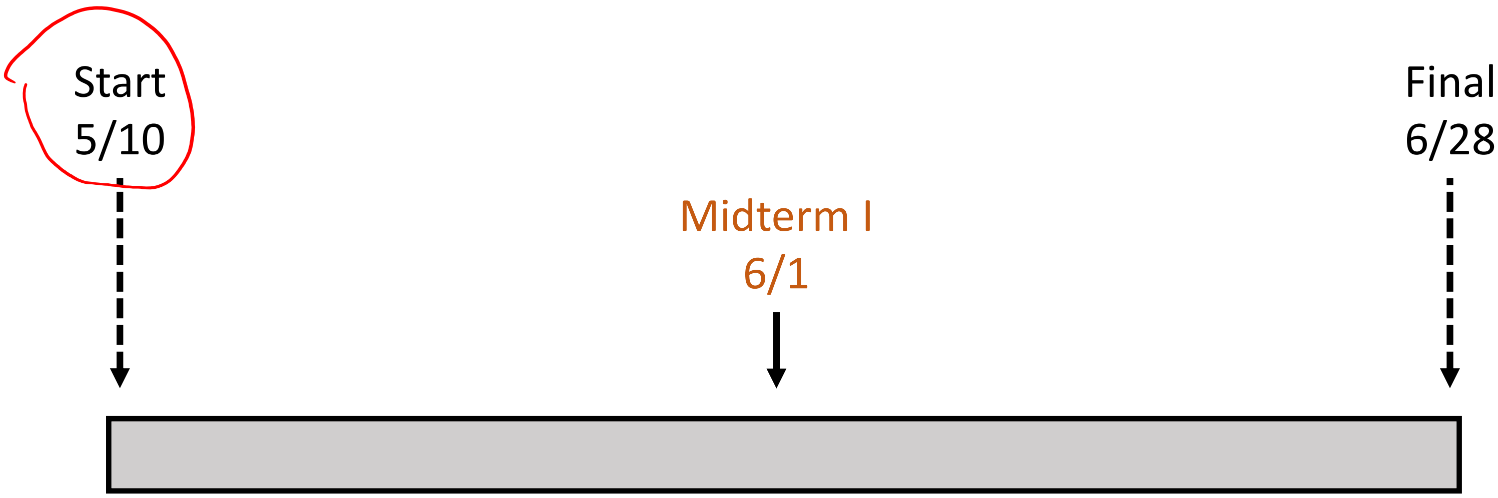
- That sounds hard. Why would I want to do that?

- Get Rich:
  - Many of the world's most successful companies   (e.g. Google) began with algorithms (also **jobs!)**
- Applications to the Real World:
  - Transportation, biology, marketing, etc. often boil down to algorithms
- Fun:
  - For real!

Algorithms

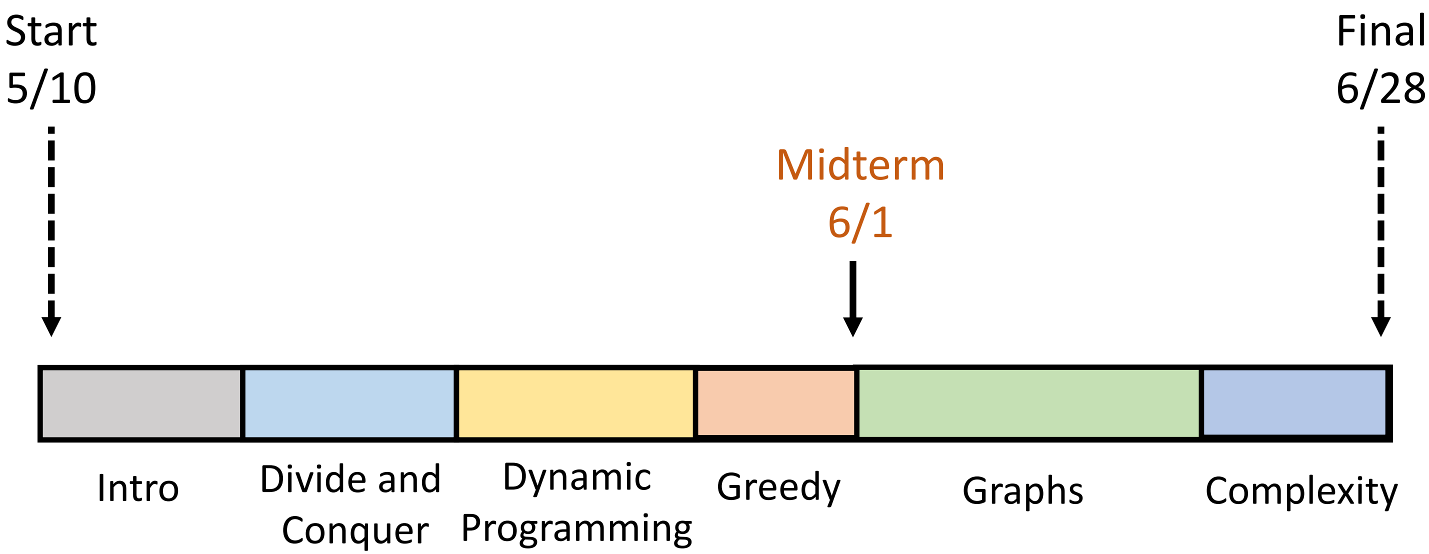
- You can only gain these skills with practice!
  - Office hours, homework, textbook, Piazza, recitations, etc.

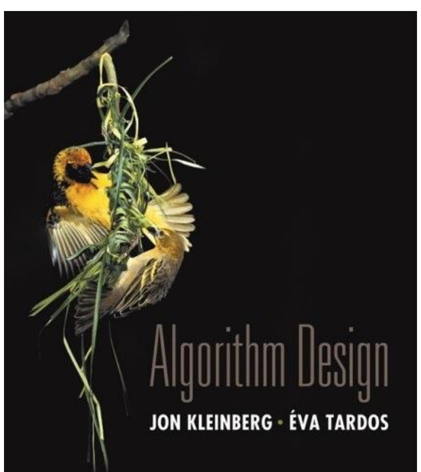
Course Structure



- HW (5) = 45%
- Quizzes (5) = 10%
- Exams = 45%
  - Midterm I = 20%
  - Final = 25%

Course Structure





Textbook:
Algorithm Design by Kleinberg and Tardos

More resources on the syllabus

Homework

- HW Assignments (45% of grade)
  - Due by 11:59pm on Fridays*
  - **HW1 out Friday! Due 5/21**
  - No extensions, no late work (***start early!***)
  - 6 HWs; lowest HW score is dropped

- A mix of proofs, analysis, algorithm design, and computations

Homework Policies

- Homework must be typeset!
  - **Crucial: your math must be easy to read**
  - Many resources & good editors available (Overleaf, TexShop, TexStudio, Word)
  - I will provide HW .tex source on Piazza

The Not So Short Introduction to LaTeX 2_ε

Or LaTeX 2_ε in 157 minutes

by Tobias Oetiker

Hubert Partl, Irene Hyna and Elisabeth Schlegl

Version 5.06, June 20, 2016

Homework Policies

- Homework will be submitted on Gradescope!
  - Entry code: **ZRPXK4**
  - Sign up today, or even right this minute!
  - *Use Gradescope for regrades*
  - **Tag problems to pages!**

Homework Policies

- You are encouraged to work with your classmates on the homework problems.
  - You may not use the internet
  - You may not use students/people outside of the class

- **Collaboration Policy:**
  - You must write all solutions by yourself
  - You may not share any written solutions
  - You must state all of your collaborators
  - We reserve the right to ask you to explain any solution

Quizzes

- Quizzes (10% of grade)
  - Every week there isn't an exam (so 6 in total)
  - Lowest quiz gets dropped
  - Released Fridays on Canvas
  - Due by 11:59am on Monday

- Primary goal: to ensure you understand that week's concepts and aren't falling behind

Exam Policies

- **Exams will be online (likely over Canvas)**
- **Exams are to be done alone, without help from others and without help from the internet/Piazza/the slides/etc.**
- **If you are found to have received or provided aid on an exam, severe penalties will be issued.**

Discussion Forum & Course Websites

- We will use Piazza for discussions
  - Ask questions and help your classmates
  - Please use private messages sparingly
- Canvas will host lecture notes, HW, & other materials
- E-mail me if you have not been added to Piazza

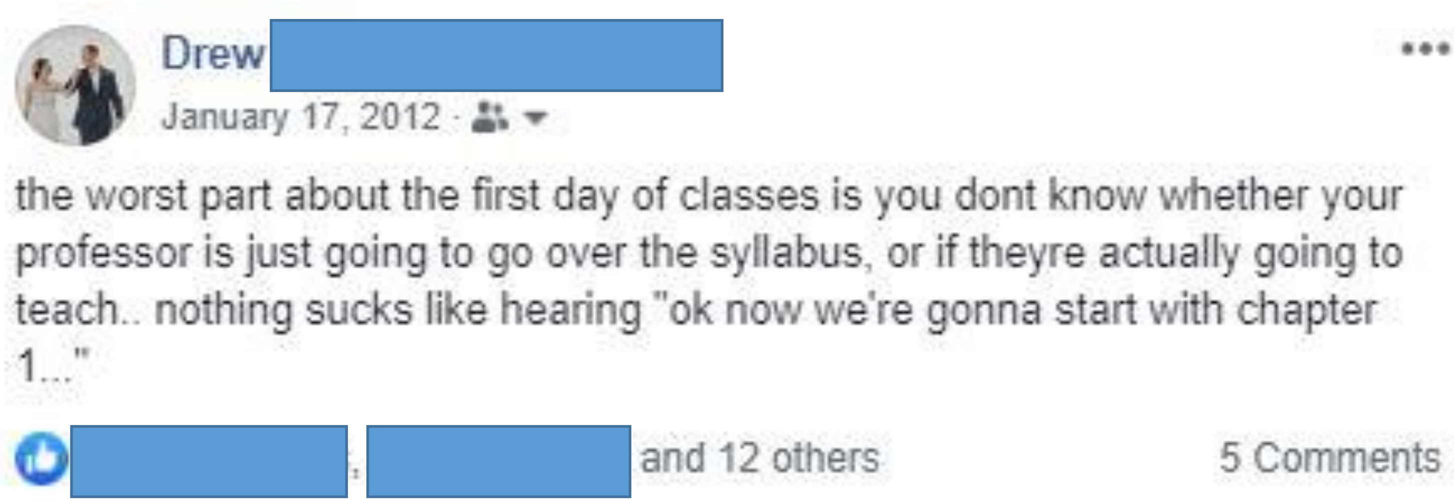
“Recitations”

- Each week I will release recitation problems which can be discussed freely with classmates/TAs/me (on or off Piazza)
- Each Monday from 3:30-5pm, I will hold a recitation/open problem session on Zoom
- These sessions are **optional** and **will be recorded**
- These sessions are to review topics we covered in class, they are **not for specific/individual HW questions**

Questions

Please ask questions when you’re confused/stuck/make a connection/etc.!!!

- In class (orally and/or via Zoom chat)
- On Piazza
- During recitation
- Etc.



Our First Problem: Student Counting

Input: A classroom of standing students

Problem: Determine how many students are in the class.

Output: The number of students (a positive integer)

- Problems: **counting students**

- Alg. techniques:

- Analysis:

- Proof techniques:

Simple Counting

```
SimCount:
  Drew says 0 and passes ball to a student

  Until only 1 student is standing:
      Student w/ ball sets their number to (1 + what
          last  person said)
      Student w/ ball says their number
      Student w/ ball passes to a standing student &
          sits

✓ Student w/ ball sets their number to (1 + what last
    person said)
✓ Student w/ ball says their number
```

Simple Counting

```
SimCount:
  Drew says 0 and passes ball to a student

  Until only 1 student is standing:
      Student w/ ball sets their number to (1 + what
          last  person said)
      Student w/ ball says their number
      Student w/ ball passes to a standing student &
          sits

  Student w/ ball sets their number to (1 + what last
    person said)
  Student w/ ball says their number
```

- Is this correct? Why or why not?
- How many steps does this require with n students?
  - # of steps = # of lines executed that don't begin w/ "Until"

Simple Counting

```
SimCount:
+1   Drew says 0 and passes ball to a student

     Until only 1 student is standing:
3(n-1)  +1 Student w/ ball sets their number to (1 + what
loops           last  person said)
        +1 Student w/ ball says their number
        +1 Student w/ ball passes to a standing student &
                sits

+1   Student w/ ball sets their number to (1 + what last
       person said)
+1   Student w/ ball says their number
```

- Is this correct? Yes – each student contributes exactly 1 to the count
- How many steps does this require with n students?

$\boxed{T(n)} = 3n = 3(n-1) + 3 = 3n - 3 + 3$

Recursive Counting

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- Is this correct? Why?

Yes – loop invariant

Recursive Counting

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- Is this correct? Why?
  - Yes – (loop invariant) after each iteration of the "until only one student is standing" loop, the sum of the standing students' numbers = the total number of students; i.e. every student is always accounted for

Recursive Counting

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- How many steps does this take with 1 student?

  $T(1) = 2$

- How many students are left after the first loop iteration when there are initially *n > 1* students? $S(n) =$

Recursive Counting

$\lceil 7 \rceil = 7 = \lfloor 7 \rfloor$

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- How many students are standing after the first loop iteration when there are initially *n > 1* students?

$S(n) = \lceil \frac{n}{2} \rceil$

$\lceil \frac{10}{2} \rceil = \frac{10}{2} = 5$, $\lceil \frac{11}{2} \rceil = \lceil 5.5 \rceil = 6$

Recursive Counting

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- How many steps does this take with *n* students (as a function *T(n)*)?

$T(n) = \underbrace{T(\lceil \frac{n}{2} \rceil)} + 3$

$T(1) = 2$

Recursive Counting

```
RecCount:
  Everyone set your number to 1

  Until only one student is standing:
    Everyone partner up, wait if you don't find one

    Set your number to (your number + partner's number),
    if waiting keep same number

    Sit down if you are taller than your partner (break
    ties arbitrarily)

The standing student says "the total is (their number)"
```

- How many steps does this take with *n* students?
  - $T(1) = 2$, $T(n>1) = T\left(\lceil n/2 \rceil\right) + 3$

Recursive Counting Running Time

$\frac{2^m}{2} = 2^m \cdot 2^{-1} = 2^{m-1}$ $\qquad \frac{\cancel{2} \cdot 2 \cdot 2 \cdot \ldots \cdot 2}{\cancel{2}}$

- **Recurrence:** once at least one term is given, the further terms are defined via the preceding ones

$T(2^{m-1}) = 3 + T\left(\left\lceil \frac{2^{m-1}}{2} \right\rceil\right) = 3 + T(2^{m-2})$

$T(1) = 2,\quad T(n) = 3 + T(\lceil n/2 \rceil)$

$\boxed{T(2) = ?}$

$T(2) = 3 + T\left(\left\lceil \frac{2}{2} \right\rceil\right)$

$= 3 + T(1) = 3 + 2 = 5$

$T(3) = ? = 8$

$T(4) = ? = 8$

$T(2^m) = ?$

$T(2^m) = 3 + T\left(\left\lceil \frac{2^m}{2} \right\rceil\right) = 3 + \boxed{T(2^{m-1})}$

$= 3 + \boxed{3 + T(2^{m-2})} \ldots = \underbrace{3 + \ldots + 3}_{m\ 3\text{'s}} + T(2^0)$

$\boxed{= 3m + 2}$

Running Time

- **Recurrence:** $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$

T(2) = 5

T(3) = 8

T(4) = 8

$T(2^m) = 3 + T(2^{m-1}) = 3 + 3 + T(2^{m-2})$

$= 3 + \ldots + 3 + T(2^0) = 3m + 2$

Our First Proof

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$

Proof by Induction

- We will prove our claim using **induction**

- **Induction:**
  - Used to prove a claim *H* is true for every natural number *i* starting at a first value (usually *0* or *1*) – *H(i)* is true ∀ *i*
  - How:
    - 1. Base case – prove directly for *H(1)* (or whatever the base case(s) is/are)
    - 2. Inductive step – For general *k,* show that if *H(k-1)* is true, then *H(k)* is true. The assumption that *H(k-1)* is true is the **inductive hypothesis (IH).**

  - Why:
    - Suppose we want to prove *H(100)*. First, we can use the base case to show *H(1)* holds. Then, because *H(1)* is true, *H(2)* is true via inductive step, and then *H(3)* is true, and so on, all the way to *H(100)* (or whatever value!).

- Problems: counting students

- Alg. techniques:

- Analysis:

- Proof techniques: **induction**

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$

- **Induction:** "automatically" prove for every $\boldsymbol{m}$
  - Let $H(m)$ be the statement $T(2^m) = 3m + 2$
  - **Base Case:** Show $H(0)$ is true
  - **Inductive Step:** For every $m \geq 1$, can assume $H(m-1)$ is true to show $H(m)$ is true
  - **Conclusion:** statement is true for every m

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$
- Let $H(m)$ be the statement $T(2^m) = 3m + 2$
- **pf. Base:** need to show *H(0)* is true,

  $H(0)$:

  Inductive: can assume $H(m-1)$ is true to show $H(m)$ is true, that $T(2^m) = 3m + 2$

Proof by Induction

- **Claim:** Recurrence $T(1) = 2, T(n) = 3 + T(\lceil n/2 \rceil)$ has closed form $T(n) = 3m + 2$ for every number of students $n = 2^m$
- Let $H(m)$ be the statement $T(2^m) = 3m + 2$
- **pf. Base:** need to show *H(0)* is true,

$H(0)$: $T\left(2^0\right) = 3(0) + 2$

$T(1) \;\; = 2$ **;** so *H(0) is true*

Inductive: can assume $H(m-1)$ is true to show $H(m)$ is true, that $T(2^m) = 3m + 2$

$T(2^m)$ **=** *3 +* $T(\lceil 2^m/2 \rceil)$ *= 3 +* $T\left(2^{m-1}\right)$

$= 3 + 3(m-1) + 2$ *= 3m + 2;* *so H(m) is true*

Comparing to Simple Counting

- **# of steps in RecCount:** $T(n) = 3m + 2$ for every number of students $n = 2^m$

- $m = log_2 n$ $\rightarrow T(n) = 3log_2 n + 2$

Running Time

- **Simple counting:** $T(n) = 3n$ steps
- **Recursive counting:** $T(n) = 3\log_2 n + 2$ steps

- Which was faster in class?

- Which requires more steps?

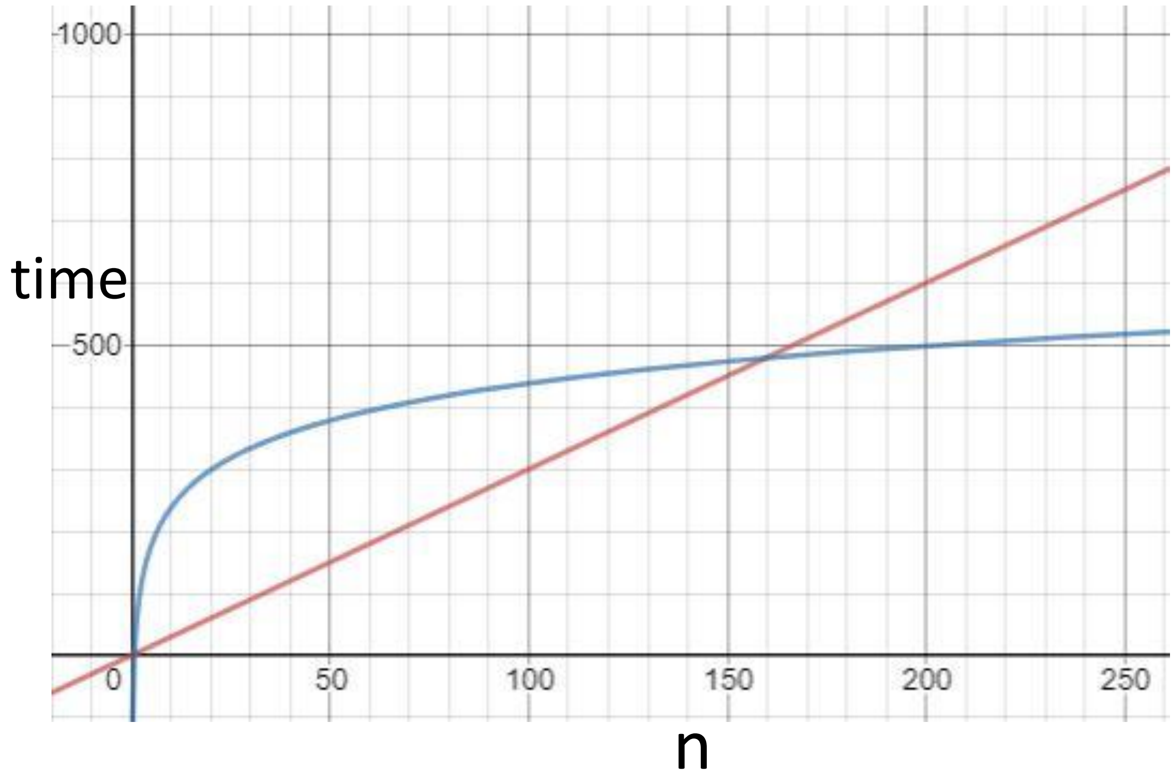
Running Time

- **Simple counting:** $T(n) = 3n$ steps
- **Recursive counting:** $T(n) = 3\log_2 n + 2$ steps

- Simple counting had more steps, but was faster???

Running Time

- **Simple counting:**
  $3n$ time
- **Recursive counting:**
  $60 \log_2 n + 40$ time



- **Compare algorithms by asymptotics!**
  - Log-time beats linear-time as $n \to \infty$