CS4800: Algorithms & Data
Jonathan Ullman

Lecture 1:
- Course Overview (Warning: slightly dry)
- Induction

Jan 9, 2018

Me

- Name: Jonathan Ullman
  - Feel free to call me Jon
  - NEU since 2015
  - Office: 623 ISEC
  - Office Hours: Tuesday 1:30-3pm
- Research:
  - Privacy, Crypto, Machine Learning, Game Theory
  - Algorithms are at the core of all of these!

Our Esteemed TAs

- Vikrant Singhal
  - Office Hours: Thu 4-6pm
  - Location: 6^{th} Floor ISEC

    ISEC 605

- Konstantin Gizdarski
  - Office Hours: 3-5pm Wed
  - Location: WVH Atrium

Algorithms

- What is an algorithm?

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

  -Jeff Erickson

  - Examples: Sort a list of numbers, find the shortest route home, find web pages about algorithms
  - Essentially all computer programs (and more) are algorithms for some computational problem.

Algorithms

- What is "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 the properties of algorithms
    - Most often correctness, running time, space usage

Algorithms

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

- Build Intuition:
  - How/why do algorithms really work?
  - How to attack new problems?
  - Which design techniques work well?
  - How to compare different solutions?
  - How to know if a solution is the best possible?

Algorithms

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

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

Algorithms

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

- Learn Problem Solving / Ingenuity /Creativity
  - "Algorithms are little packets of brilliance." -Olin Shivers

Algorithms

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

- You can only gain these skills with practice!
    - HW
    - Doing practice problems
    - Going to OH
    - Reading

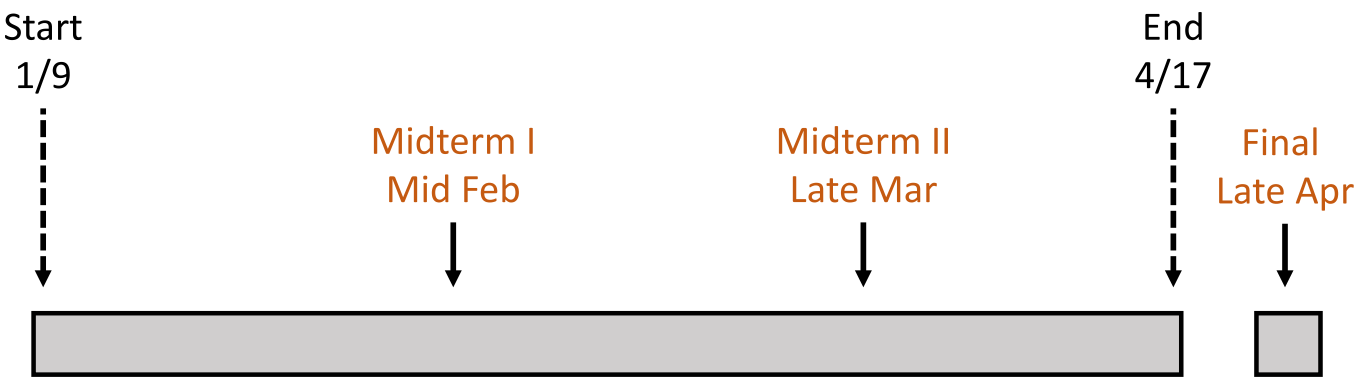
Algorithms

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

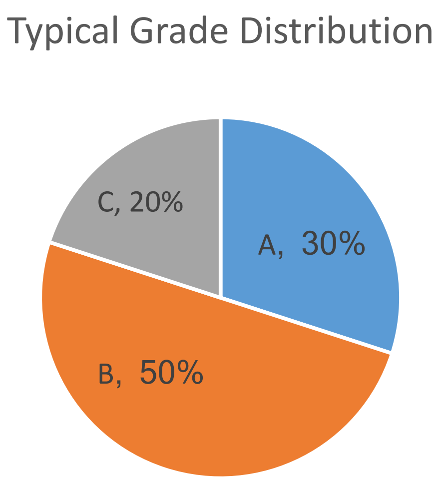
- Get Rich:
  - Many of the world's largest companies (e.g. Google, Akamai,...) began with algorithms.
- Understand the natural world:
  - Brains, cells, networks, etc. often viewed as algorithms.
- Fun:
  - Yes, seriously, fun.

30%

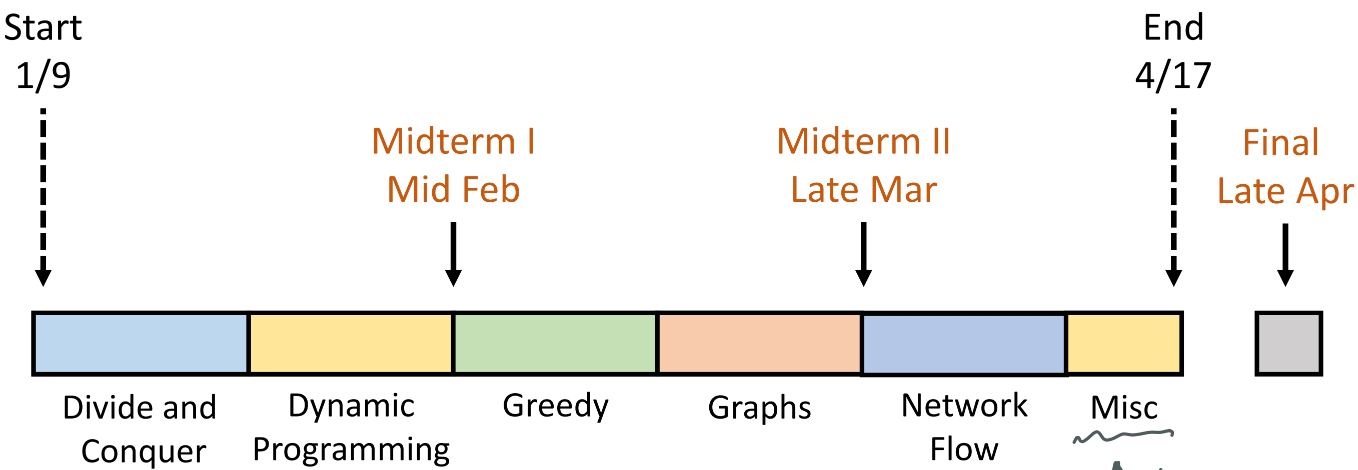
Course Structure



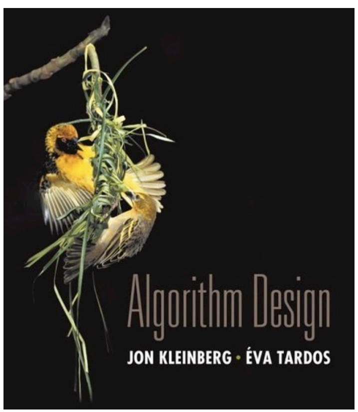
- HW = 50%
- Exams = 50%
  - Midterm I = 15%
  - Midterm II = 15%
  - Final = 20%



Course Structure



I may solicit feedback



Textbook:
Algorithm Design by Kleinberg and Tardos

- Erickson. Algorithms, Etc. (online)
- CLRS
- Math for Computer Science. Meyer, Leighton (online)

Homework

- Weekly HW Assignments (50% of grade)
  - Due Fridays by 4:59pm
  - **HW1 out now! Due Fri 1/19**
  - No extensions*, no late work
  - Lowest score will be dropped

- Mostly mathematical / algorithmic problems
- 2-4 Programming problems

  throughout the semester

Homework Policies

- Homework must be typeset in LaTeX!
  - Many resources available
  - Many good editors available (TexShop, TexStudio)
  - I will provide HW source

MAC

The Not So Short
Introduction to $\LaTeX\ 2_\varepsilon$

Or $\LaTeX\ 2_\varepsilon$ 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 MV8J4R
  - Sign up today (or even right this minute)!

Homework Policies

- **You are encouraged to work with your classmates on the homework problems.**
  - **You may not "collaborate" with the internet or with students not in the class.**

- **If you do collaborate, you must write all solutions by yourself, in your own words, and are strictly forbidden from sharing any written solutions. You must list all of your collaborators.**

- **I reserve the right to ask you to explain any solution.**

What About the Other Sections?

- No formal relationship between this section and the other two sections.
  - Will cover very similar topics → especially Prof. Nguyen's sec
  - Will share some homework questions
  - Will use different exams

- You are expected to come to lectures for your section, meet with TAs for your section, collaborate with people in your section.

Discussion Forum

- We will use Piazza for discussions
  - Ask questions
  - Help your classmates
- Sign up today (or even right this minute)!

Course Website

http://www.ccs.neu.edu/home/jullman/CS4800S18/syllabus.html
http://www.ccs.neu.edu/home/jullman/CS4800S18/schedule.html

CS4800: Algorithms & Data

Syllabus Schedule

Note: this page will be updated frequently!

#	Date	Topic	Reading	HW
1	T 1/9	Course Overview Analyzing Algorithms via Induction	---	HW1 Out (pdf, tex)
2	F 1/12	Asymptotic Analysis Divide and Conquer: Karatsuba	KT 2.1-2.2 demo	---
3	T 1/16	Divide and Conquer: Mergesort, Recurrences	KT 5.1-5.2 demo	---
4	F 1/19	Divide and Conquer: Master Theorem	Erickson II.3	**HW1 Due** HW2 Out (pdf, tex)

One More Thing:
I need to count how many students are in this lecture!

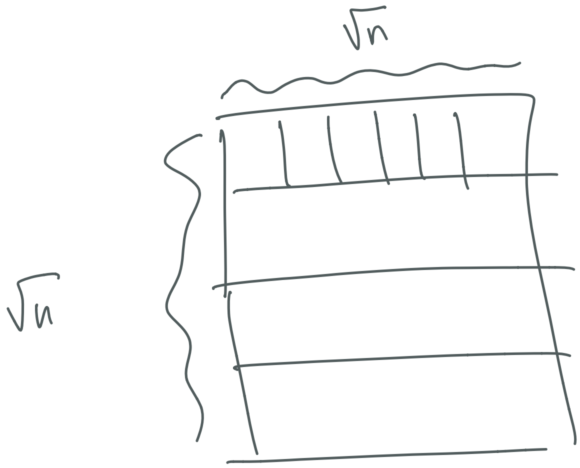
Counting People

- Simple Counting:
  1. Find the first student
  2. The first student says one
  3. Until we're out of students:
     a. Go to the next student
     b. The next student says what the last student says + one

28.04 seconds
46 students

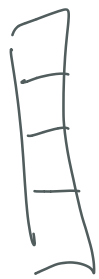
- Is this correct?
- How long does this take?
  - $T(n)$ is the time to count n students
  - $T(n) = 2n$

Elementary step
- point @ new student
- say number



$T(\sqrt{n}) = 2\sqrt{n}$

sum all rows



$T(\sqrt{n}) = 2\sqrt{n}$

$T(n) = 4\sqrt{n}$

A “Recursive” Algorithm

- Recursive Counting:
  1. Everyone stand
  2. Everyone set your “number” to one
  3. Until only one student is standing
     a. Greet a neighbor (pause if you’re the odd person out)
     b. If you are taller, give “number” and sit. If you are shorter, add up “numbers.”
  4. Say “number”

2:42:02

- Is this correct? Do you see why?

||||

Running Time

Divide-and-Conquer Algorithm

- Recursive Counting:
    1. Everyone stand
    2. Everyone set your "number" to one
    3. Until only one student is standing
        - 1 step a. Greet a neighbor (pause if you're the odd person out)
        - 1 step b. If you are taller, give "number" and sit. If you are shorter, add up "numbers."
    4. Say "number"

T(n) is the time to carry out the loop with n students.

- How long does this algorithm take?
    - $T(n)$ is the number of $\boxed{\text{steps}}$ to count n students.
    - $\underbrace{T(n) = 2 + T(\lceil n/2 \rceil)}_{\text{recurrence relation}}, \underbrace{T(1) = 3}_{\text{base case}}$

Running Time

- Recurrence $T(1) = 3, T(n) = 2 + T(\lceil n/2 \rceil)$
- Intuition (easier when $n = 2^\ell$):

so $\lceil \frac{n}{2} \rceil = \frac{n}{2}$

$$
\left.\begin{aligned}
T(2^\ell) &= 2 + T(2^{\ell-1}) \\
&= 2 + 2 + T(2^{\ell-2}) \\
&\ \ \vdots \\
&= 2 + 2 + \dots + 2 + T(1)
\end{aligned}\right\} \ell \text{ steps}
$$

$$
= \underbrace{2 + 2 + \dots + 2}_{\ell} + 3
$$

$$
= 2 \cdot \ell + 3
$$

Inductive Proofs

- Conjecture: For every number of students $n = 2^\ell$, $T(2^\ell) = 2\ell + 3$
- Can verify small cases
  - $\ell = 0$: $T(2^0) = 3 = 2 \cdot 0 + 3$ ✓
  - $\ell = 1$: $T(2^1) = 5 = 2 \cdot 1 + 3$ ✓
  - …
- We cannot do this for every n
- Induction: assume the claim is true for all $n < k$, prove that it is true for $n = k$
- $n = 1 \implies n = 2 \implies n = 3 \implies \dots$

Inductive Proofs

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

- Conjecture: For every number of students $n = 2^{\ell}$, $T(2^{\ell}) = 2\ell + 3$

Proof by Induction on ℓ:

Base Case ($\ell = 0$): $T(2^0) = T(1) = 3 = 2 \cdot 0 + 3$

Inductive Step: [If the statement is true for all $\ell < k$, then it is true for $\ell = k$.]

$$T(2^k) = \underbrace{T(2^{k-1})}_{\text{IH}} + 2 = (\underbrace{2 \cdot (k-1) + 3}_{\text{IH}}) + 2$$

$$= 2 \cdot k + 3$$

Therefore conjecture holds for all n by induction. $\square$

Running Time

96

- Simple counting: $T_{sim}(n) = 2n$ "steps"
- Recursive counting: $T_{rec}(n) = 2\log_2 n + 3$ "steps"

≤ 15

- But for this class, simple counting was faster???

Running Time

- Simple counting: $T_{sim}(n) = 2n$ sec
- Recursive counting: $T_{rec}(n) = 30\log_2 n + 45$ sec

30·6 +45 = 225

- Asymptotics!
  - Log-time beats linear-time as $n \to \infty$