CS 170
Efficient Algorithms and Intractable Problems

Lecture 1:
Logistics, Introduction, Arithmetic

Nika Haghtalab and John Wright

EECS, UC Berkeley

Today's Plan

Introductions

- Who are we?
- Who are you?
- Why are we here?

Course Overview

- Course Goals and overview
- Logistics

Arithmetic!

- Can we add and multiply?
- Can we do them fast?

Who are you?

Mostly sophomores, juniors, seniors!

Studying

- Applied Math
- Astrophysics
- Architecture
- Bioengineering
- Business Administration
- Chemical Biology
- Chemical Engineering
- Computer Science
- Data Science
- EECS
- Economics
- Environmental Sciences
- IEOR
- Cognitive Science
- Genetics & Plant Biology
- Linguistics
- Mathematics
- Mechanical Engineering
- Music
- Philosophy
- Physics
- Public Health
- Statistics
- …

Who are we?

Instructors

Prof. John Wright

Prof. Nika Haghtalab

Jonny

Ramanan

Ishaq

Ansh

Malvika

Alex

Kunhe

Ajit

Jonathan

Lance

Parth

Viraj

Andrew

Albert

Sowmya

Vaibhav

Xavier

Zenan

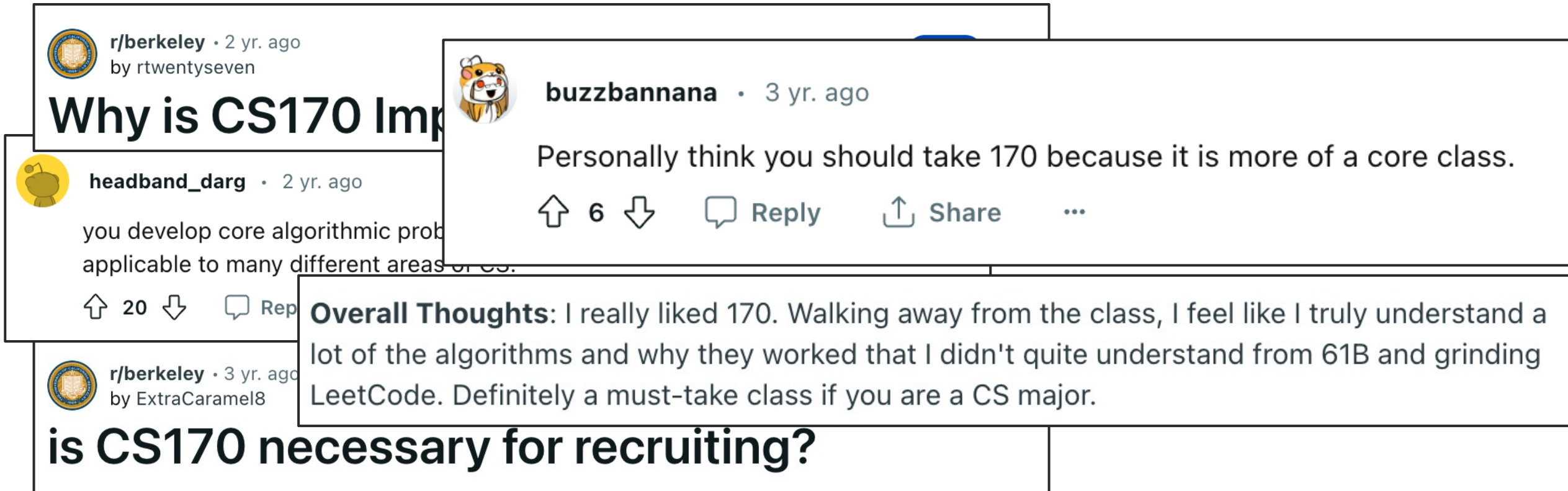
Zeeshan

Why are we here?

You need an upper div credit ...

Algorithms are fundamental and useful.

Algorithms are fun!



Course Goals

Design and analyze algorithms

In this course you will learn:

- Design: Acquire an algorithmic toolkit
- Analysis: Learn to think analytically about algorithms
- Understand limitations: Understand algorithmic limitations
- Communication: Learn to formalize your thoughts and communicate clearly about algorithms

Fundamental Questions about Algorithms

Does it work?

Is it fast?

Can we do better?

Detail-oriented

Bigger Picture

Course Logistics

Course website:

- **https://cs170.org/**
- Hosts lecture slides and notes, class calendar, assigned reading from textbook.

Lectures

- No livestream! COME TO LECTURES!
- Video recordings: available on bCourses
- Textbook readings linked on course website

Homework

- Weekly HWs.
- HW Parties on Fridays and Mondays 

Course Logistics

Discussion Sections (under the schedule tab)

- Tuesday evenings and Wednesdays (more sections will be added)
- **Coordination:** Informal signup sheet, we'll publicize # of signups
- You can go to sections without signing up too.
- **LOST Section:** Fridays, slower pace, more interactions, reinforces concepts

Contact us or each others

- Ed: Announcements and forum
- Email: cs170@Berkeley.edu for admins and logistics.

Office hours (See the schedule under the calendar tab)

Exams: 2 midterms and 1 final. **No alternate exams offered.**

Midterm 1 on October 3, Midterm 2 on Nov 7. Both 7pm-9pm

Final: Dec 15, 8am-11am

A good way to learn in this course

Lectures:

- Show up to class! Ask questions and remain engaged in class.
- Review the slides and questions after the lecture, before attempting the homework.

Discussion section:

- Attempt the discussion problems before the session.
- GO TO SECTIONS!

Assigned reading:

- Read before or soon after class. Don’t leave until the exam time.

Other resources and forms

Course Policies:

- Course policies and etiquettes will be listed on the website.
  - Academic Honesty code strictly enforced, ...
- Read them and adhere to them.

Feedback!

- Help us improve the class!
- Send us suggestions on Ed or in person.
- We will set up a midsemester anonymous feedback form.

Algorithms!

“Algorithms”

Muḥammad ibn Mūsā al-Khwārizmī or **al-Khwarizmi**, was Persian polymath from Khwarazm (today’s Uzbekistan and Turkmenistan).

Was a scholar in Baghdad contributed to Math, Astronomy, and Geography.

In Latin, al-Khwarizmi’s name gave rise to “algorithm”.

His books spread the Hindu-Arabic numeral system to Europe.

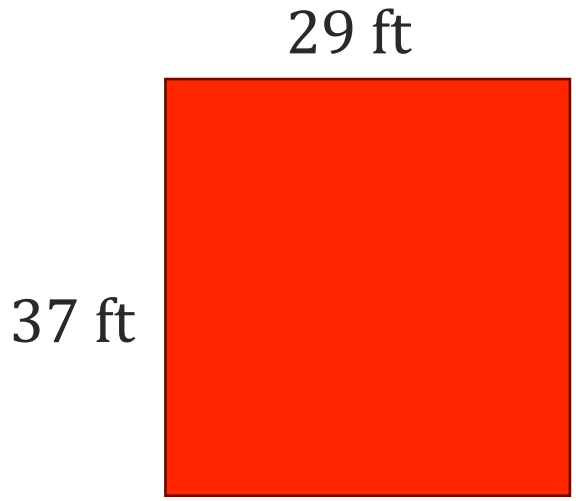
Hindu-Arabic Numeral System

Roman numerals not in any natural basis. Hard to do arithmetic.

How big is my plot of land?

$$\text{XXIX} \times \text{XXXVII} = ?$$

$$\begin{array}{r} 29 \\ \times\ 37 \\ \hline \end{array}$$



Let's go back to elementary school

How do we add integers?

$$\begin{array}{r} 12345 \\ +\ 78910 \\ \hline \end{array}$$

Discuss

How fast is the grade-school addition algorithm?

More formal: How many one-digit operations does it take?

n digits

$$\begin{array}{r} 12345678910109876543 21 \\ +\ 109876543211234567891 0 \\ \hline \end{array}$$

About n one-digit operations

Well … there are also at most n carries, but that makes it still something like $2n$, maybe $3n$ ….

Big-Oh Notation

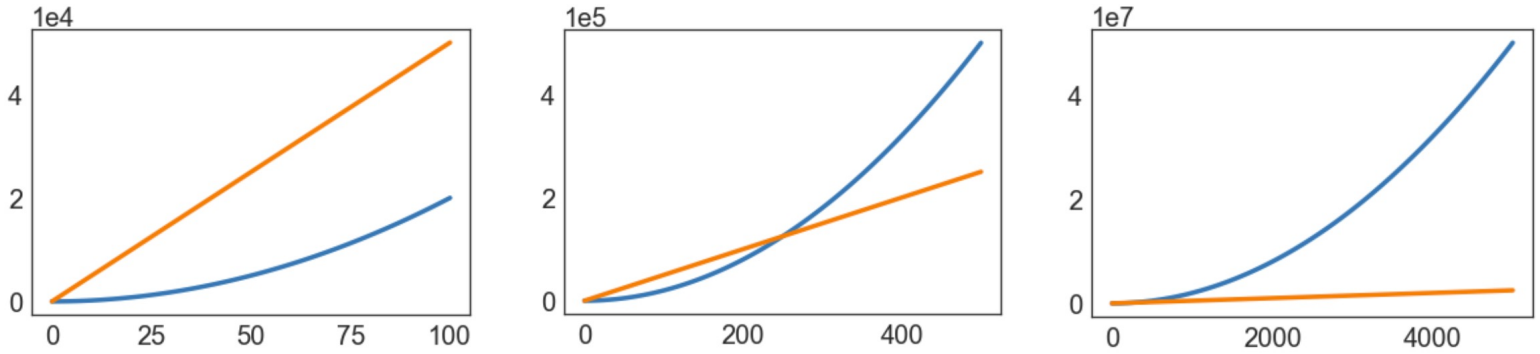
Recall $O(\cdot)$ notation from 61B!!

Asymptotic Behavior

In most cases, we care only about asymptotic behavior, i.e. what happens for very large N.

- Simulation of billions of interacting particles.
- Social network with billions of users.
- Logging of billions of transactions.
- Encoding of billions of bytes of video data.

Algorithms which scale well (e.g. look like lines) have better asymptotic runtime behavior than algorithms that scale relatively poorly (e.g. look like parabolas).



$$R(N) \in O(f(N))$$

means there exists positive constants k_2 such that:

$$R(N) \leq k_2 \cdot f(N)$$

for all values of N greater than some N_0.

i.e. very large N

Big-Oh Notation

Recall $O(\cdot)$ notation from 61B!

- Ignore constants and focus on the largest dependence on n.

We say that addition of 2 numbers with n digits

"runs in time O(n)"

Still don't remember $O(\cdot)$ notation well?

- We'll dig deeper more formally next time. Also GO TO SECTIONS!

What about multiplication?

Discuss

How fast is grade school integer multiplication?

n digits

$$\begin{array}{r} 12345678910\,10987654321 \\ \times\ 10987654321\,12345678910 \\ \hline \end{array}$$

It runs in time $O(n^2)$!

Well … there are at most n^2 1-digit multiplications, at most n^2 carries to be added, and then we have to add n numbers, each with at most $2n$ digits ….

Can we do better?

Easier question: Can we do better than $O(n)$?

- No! It takes at least n steps to just read the numbers.

One other fun algorithm for multiplications:

27 x 19

Egyptian multiplication / Russian Peasant Algorithm

1. Repeat: Halve 1st number (floor) and double the second number, until we get 0 in the first column.
2. Remove any rows where the first column is even.
3. Add all remaining rows.

At home, prove why this algorithm is correct and what its runtime is.

There is a way to do better!

- **Karatsuba** (1960): $O(n^{1.6})$! We'll see this algorithm!

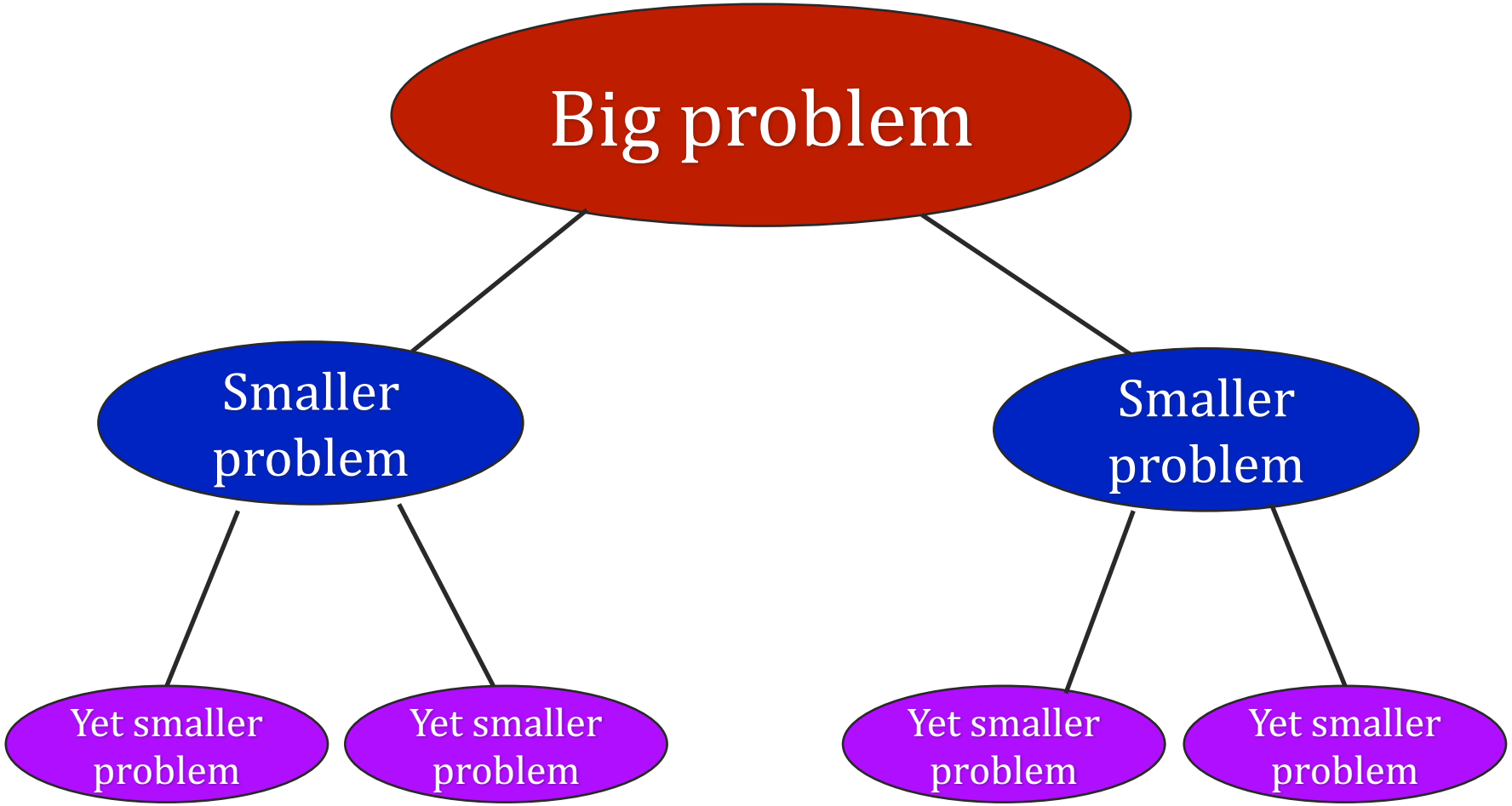
- **Toom-3/Toom-Cook** (1963): $O(n^{1.465})$

  Uses the same technical tool as Karatsuba's.

- **Schönhage–Strassen** (1971):
  - Runs in time $O(n \log(n) \log\log(n))$
- **Furer** (2007)
  - Runs in time $n \log(n) \cdot 2^{O(\log^*(n))}$
- **Harvey and van der Hoeven** (2019)
  - Runs in time $O(n \log(n))$

Divide and Conquer

Breaking up a big problem into smaller subproblems, recursively.



Divide and Conquer for Multiplication

Break up the multiplication of two integers with n digits into multiplication of integers with $n/2$ digits:

$$1234 = 12 \times 100 + 34$$

$$1234 \times 5678 =$$

(simplify: assume even n)

The algorithm

Break up the multiplication of two integers with n digits into multiplication of integers with $n/2$ digits:

$$[x_1 x_2 \cdots x_n] = \overbrace{[x_1, x_2, \cdots, x_{n/2}]}^{a} \times 10^{\frac{n}{2}} + \overbrace{[x_{n/2+1} x_{n/2+2} \cdots x_n]}^{b}$$

$$\begin{aligned} x \times y &= \left(a \times 10^{\frac{n}{2}} + b\right)\left(c \times 10^{\frac{n}{2}} + d\right) \\ &= \underbrace{(a \times c)}_{\text{P1}} 10^n + \underbrace{(a \times d}_{\text{P2}} + \underbrace{c \times b)}_{\text{P3}} 10^{n/2} + \underbrace{(b \times d)}_{\text{P4}} \end{aligned}$$

One n-digit multiplication → Four $n/2$-digit multiplications

Multiply two 4-digit Numbers

$$1234 \times 5678$$

We broke 1 multiplication of 4-digit numbers to 4 multiplications of 2-digit numbers.

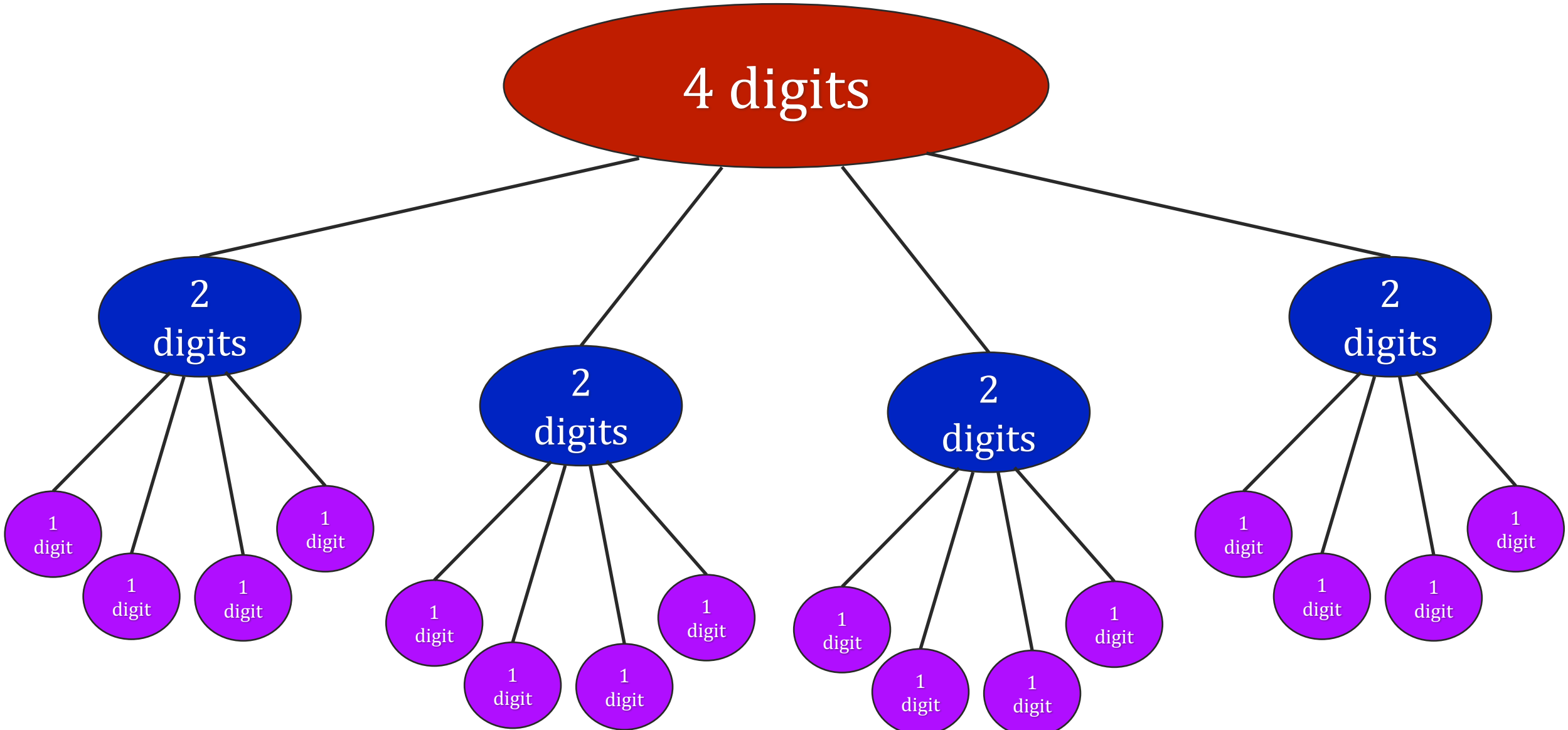
We wanted to count 1-digit operations. So, what should we do now?

Recurse!

Break up each of the 2-digit multiplication problems, to 4 multiplications with 1-digit numbers.

Write the pseudo-code, handling corner cases and odd ns too.

Recursion tree for 4-digit numbers



What is the running time of this algorithm?

We saw that multiplying two 4-digit numbers resulted in 16 one-digit multiplications.

- How many one-digit multiplications for multiplying two 8-digit numbers?

- What about multiplying n-digit numbers?

Running time of the algorithm

Claim: The runtime of the algorithm is $O(n^2)$.

Claim: We are creating $O(n^2)$ number of 1-digit operations.

So, was there a point to Divide and Conquer?

Karatsuba's Clever Trick

Divide and Conquer indeed can lead to a faster algorithm!

$$x \times y = \left(a \times 10^{\frac{n}{2}} + b\right)\left(c \times 10^{\frac{n}{2}} + d\right)$$

$$= \underbrace{(a \times c)}_{\text{P1}} 10^{n} + \underbrace{(a \times d}_{\text{P2}} + \underbrace{c \times b)}_{\text{P3}} 10^{n/2} + \underbrace{(b \times d)}_{\text{P4}}$$

The issue is that we are creating 4 sub-problems.
What if we could create fewer subproblems?

Main idea: Could we write P2+P3 using what we compute in P1 and P4, and at most one other $n/2$-digit multiplication?

Karatsuba's Clever Trick

Let us only compute 3 things:

- Q1: $a \times c$
- Q2: $b \times d$
- Q3: $(a+b)(c+d)$

Expressing P2+P3 differently

$$a \times d + c \times b = (a+b)(c+d) - \text{ac} - \text{bd}$$

Three subproblems

$$x \times y = \left(a \times 10^{\frac{n}{2}} + b\right)\left(c \times 10^{\frac{n}{2}} + d\right)$$

$$= \underbrace{(a \times c)}_{\text{Q1}} 10^{n} + \underbrace{(a \times d + c \times b)}_{\text{Q3}-\text{Q1}-\text{Q2}} 10^{n/2} + \underbrace{(b \times d)}_{\text{Q2}}$$

What is the running time of Karatsuba's Alg?

What is the running time of Karatsuba's Alg?

What about binary representation?

We used base 10 so far

→Counted the # of 1-digit operations, assuming adding/multiplying single digits is easy (memorized our multiplication table!)

What if we use base 2?

→We would want to count # of 1-bit operations.

How do we alter Karatsuba's algorithm for binary numbers?

N-bit integer multiplications

Easy to compute 10^k in base 10. In base 2, it is easy to compute 2^k.

$$[b_1 b_2 \cdots b_n] = [b_1, b_2, \cdots, b_{n/2}] \times 2^{n/2} + [b_{n/2+1} b_{n/2+2} \cdots b_n]$$

$$\underbrace{b_1 \quad b_2 \quad b_3 \quad \cdots}_{b^L} \;\big|\; \underbrace{\cdots \; b_{n-2} \quad b_{n-1} \quad b_n}_{b^R}$$

$$a \times b = (a^L \times b^L)2^n + (a^L \times b^R + a^R \times b^L)2^{n/2} + (a^R \times b^R)$$

$$= \cdots$$

Practice: Complete this equation the Karatsuba's way and rederived $O(n^2)$ runtime for multiply two n-bit numbers.

Other Algorithms

- **Karatsuba** (1960): $O(n^{1.6})$! Saw this!
- **Toom-3/Toom-Cook** (1963): $O(n^{1.465})$

(advanced)

Divide and conquer too! Instead of breaking into three n/2-sized problems, break into five n/3-sized problems.

Hint: Start with 9 subproblems and reduce it to 5 subproblems.

- **Schönhage–Strassen** (1971):
  - Runs in time $O(n \log(n) \log\log(n))$
- **Furer** (2007)
  - Runs in time $n \log(n) \cdot 2^{O(\log^*(n))}$
- **Harvey and van der Hoeven** (2019)
  - Runs in time $O(n \log(n))$

Wrap up

Divide and conquer:

- A useful and fundamental algorithmic tool. Fun too!

Karatsuba Integer Multiplication:

- You can do better than grade school multiplication!
- Example of divide-and-conquer in action

Next time

- Big-Oh and Asymptotic notations more formally
- Divide and Conquer some more
  - Matrix multiplications!