

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!**

r/berkeley • 2 yr. ago  
by rtwentyseven

## Why is CS170 Impor...

headband\_darg • 2 yr. ago  
you develop core algorithmic prob  
applicable to many different areas

Overall Thoughts: I really liked 170. Walking away from the class, I feel like I truly understand a lot of the algorithms and why they worked that I didn't quite understand from 61B and grinding LeetCode. Definitely a must-take class if you are a CS major.

is CS170 necessary for recruiting?

# 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



Detail-oriented

Does it work?

Is it fast?

Can we do better?



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](mailto: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”

Muhammad 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.



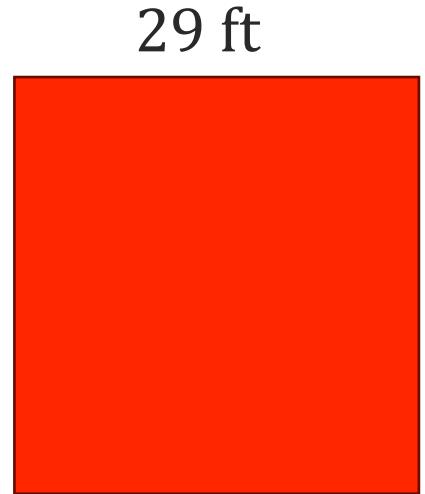
# Hindu-Arabic Numeral System

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

How big is my plot of land?

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

$$\begin{array}{r} & ^2\cancel{^6} \\ & 29 \\ \times & 37 \\ \hline + & \begin{array}{r} 203 \\ 870 \\ \hline 1073 \end{array} \end{array}$$



# Let's go back to elementary school

How do we add integers?

Discuss

How fast is the grade-school addition algorithm?

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

$n$  digits

$$\begin{array}{r} 1234567891010987654321 \\ + 1098765432112345678910 \\ \hline \end{array}$$

About  $n$  one-digit operations

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

$$\begin{array}{r} 12345 \\ + 78910 \\ \hline \dots 55 \end{array}$$

# Big-Oh Notation

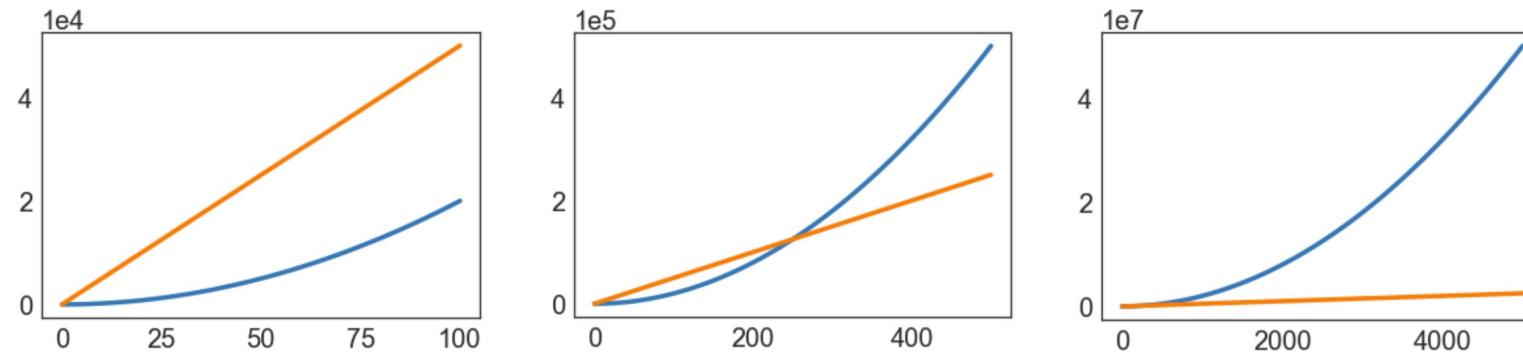
Recall  $O(\cdot)$  notation from 61B!!

## 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?

$$\begin{array}{r} n \text{ digits} \\ \overbrace{1234567891010987654321} \\ \times 1098765432112345678910 \\ \hline \end{array}$$

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:

$$\begin{array}{r} \text{y} & \text{l} & \text{2} \\ 27 & \times & 19 \\ \text{l} & & \\ \text{l} & 13 & 38 \\ \text{l} & & \\ \text{o} & 6 & 76 \\ \text{l} & & \\ \text{l} & 3 & 152 \\ \text{l} & & \\ \text{l} & 1 & 304 \\ \text{l} & & \\ \text{o} & 0 & 608 \\ \hline & & 513 \end{array}$$

**Egyptian multiplication / Russian Peasant Algorithm**

1. Repeat: Halve 1<sup>st</sup> 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 (2nd col)



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)

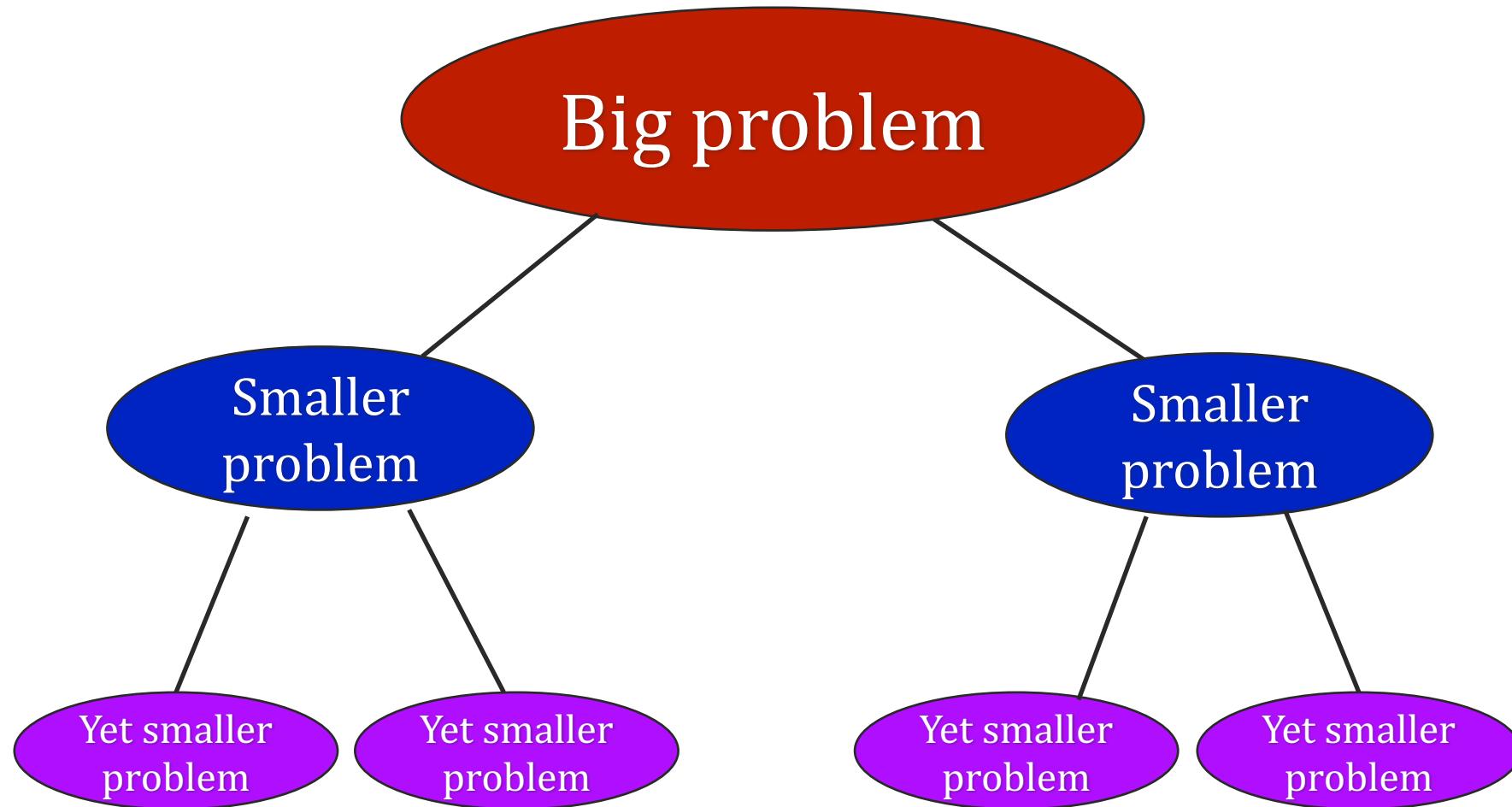
- Runs in time  $n \log(n) \cdot 2^{O(\log^*(n))}$

- **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:

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

$$\begin{aligned} 1234 \times 5678 &= (12 \times 100 + 34) \times (56 \times 100 + 78) \\ &= (\underbrace{12 \times 56}_{\textcircled{1}}) \underbrace{10^{\boxed{4}}}_{\textcircled{2}} + (\underbrace{12 \times 78}_{\textcircled{3}} + \underbrace{34 \times 56}_{\textcircled{4}}) \underbrace{100^{\boxed{1}}}_{\textcircled{5}} + 34 \times 78 \end{aligned}$$

One 4-digit multiply  $\Rightarrow$  Four 2-digit multipliers

(simplify: assume even  $n$ )

# The algorithm

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

$$[x_1 x_2 \cdots x_n] = \underbrace{[x_1, x_2, \dots, x_{n/2}]}_a \times 10^{\frac{n}{2}} + [x_{n/2+1} x_{n/2+2} \cdots x_n]_b$$

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

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?

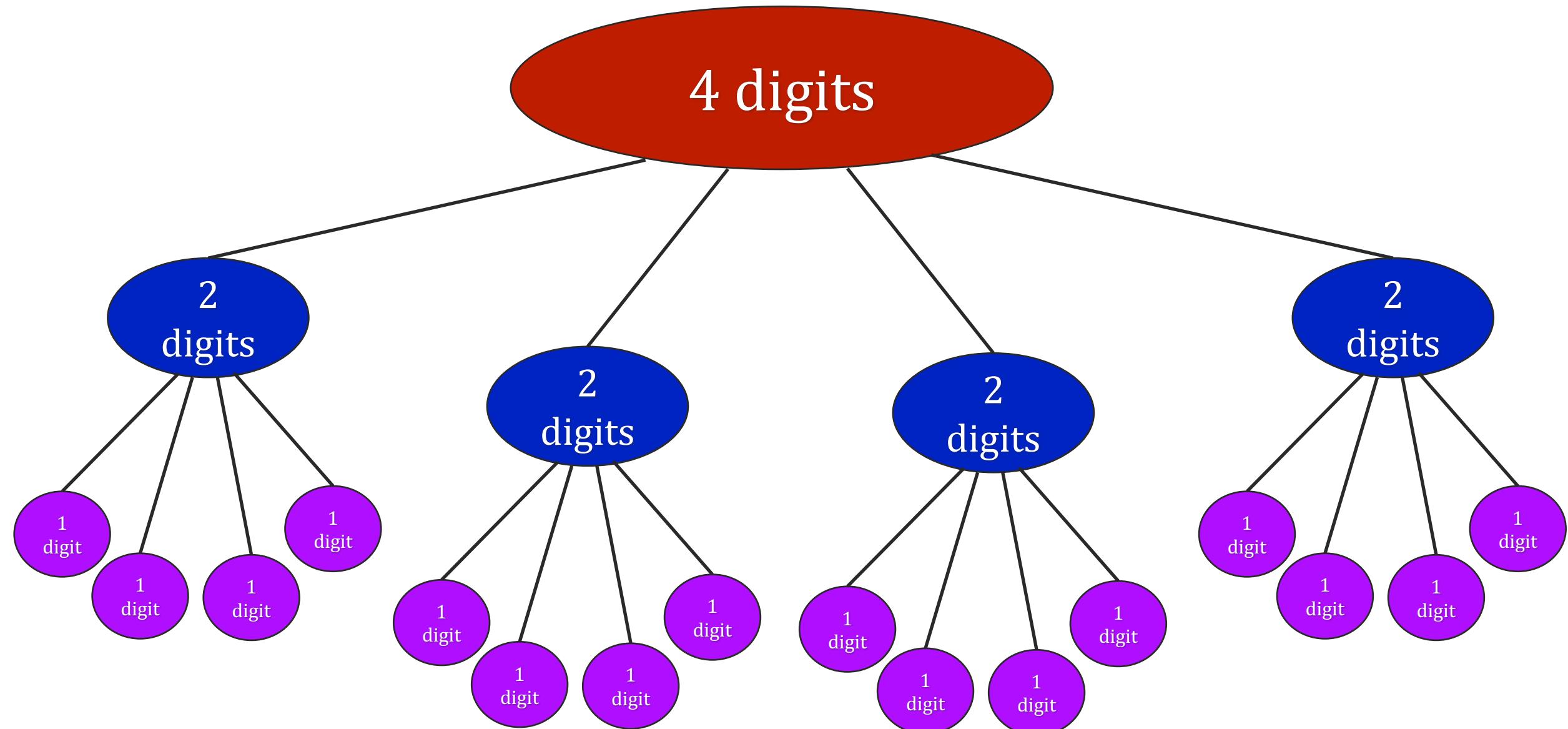
**Recurse!**

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.

layer	$\downarrow$ problems	size	What's the depth of the tree?
0	1	n	
1	4	$n/2$	$\frac{n}{2^t} = 1 \Rightarrow t = \log_2(n)$
2	16	$n/4$	How many leaves:
t:	$4^t$	$n/t$	$4^t = 4^{\log_2(n)} = n^{\log_2(4)} = n^2$
	?	1	Unfortunate!

So, was there a point to Divide and Conquer?

(end of lecture 1)