# CS 170
# Efficient Algorithms and Intractable Problems

## Lecture 1:
### Logistics, Introduction, Arithmetic

Nika Haghtalab and John Wright

EECS, UC Berkeley

(Some slides and material are inspired by courses taught by Nelson, Raghavendra, Tal, Vazirani, Haghtalab, Wright (UC Berkeley) and Wootters (Stanford))

# 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 junior (~35%), senior (~26%), and sophomore (~23%)

Studying

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

# Who are we?

## Instructors

Prof. Nika Haghtalab

Prof. John Wright


Jonny

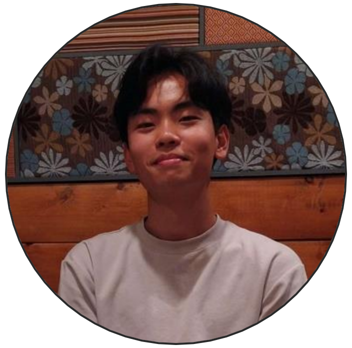

Carolyn


David W


Eric


Meghal


Andrew


Bill


Diana


Jessica L


Ryan


Xavier

Aaryan

Ajay

Alex

Anushka

David Y

Divya

George

Jeffery

Jessica L

Richik


Shu

Thomas

Yamuna

Will Y

# Why are we all here?

You need an upper div credit ...

Algorithms are fundamental and useful.

Algorithms are fun!

r/berkeley •
by rtwentyseven

**Why is CS170 Imp**

**buzzbannana** •

Personally think you should take 170 because it is more of a core class.

6 Reply Share ...

**headband_darg** •

you develop core algorithmic proble
applicable to many different areas of CS.

20 Rep

**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.

r/berkeley •
by ExtraCaramel8

**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


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 (released on Sundays, due Saturdays)
- HW Parties on Fridays 🥳

# Course Logistics

Discussion Sections

- Schedule TBD: Check the “Discussions” tab under the website
- **Discussions don’t replace lectures:** We assume you have already attended the lecture and reviewed the material before coming to the discussion.
- **LOST Section:** There will be a section with slower pace, more interactions, reinforces concepts

Contact us or each others

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

# More Course Logistics

Office hours (See the schedule under the calendar tab)

- Nika's OH: After lecture on Tuesdays or by appointment

  → meet at the class entrance
- John's OH: TBD

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

Midterm 1 on Feb 25, Midterm 2 on April 3. Both 7pm-9pm

Final: May 12, 11:30am-2:30pm

# 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.

# A good way to learn in this course

**Lectures:**

- GO TO LECTURES! Ask questions and remain engaged in class.
  - Attendance is not mandatory, but highly encouraged. Help us record your attendance!
- Review the slides and questions after the lecture, **before attempting the homework**.

Assigned reading:

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

Discussion section:

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

# 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

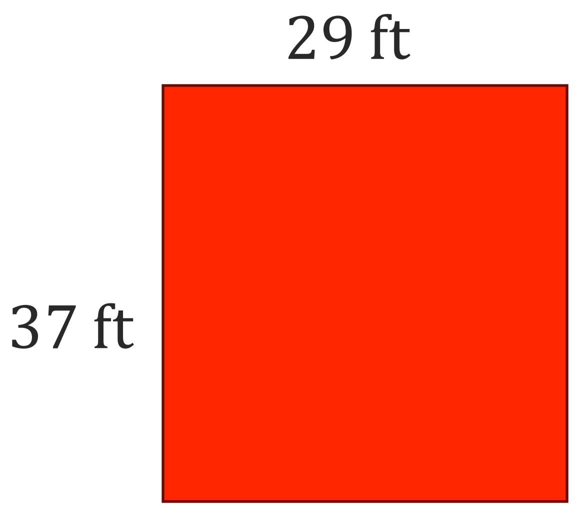
Roman numerals not in any natural basis. Hard to do arithmetic.

How big is my plot of land?

XXIX × XXXVII = ?

~~2~~ 6

$$
\begin{array}{r}
29 \\
\times\ 37 \\
\hline
203 \\
870 \\
\hline
1073
\end{array}
$$



# 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?

$$\begin{array}{r} \overbrace{1234567891010987654321}^{n \text{ digits}} \\ +\ 1098765432112345678910 \\ \hline 231 \end{array}$$

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

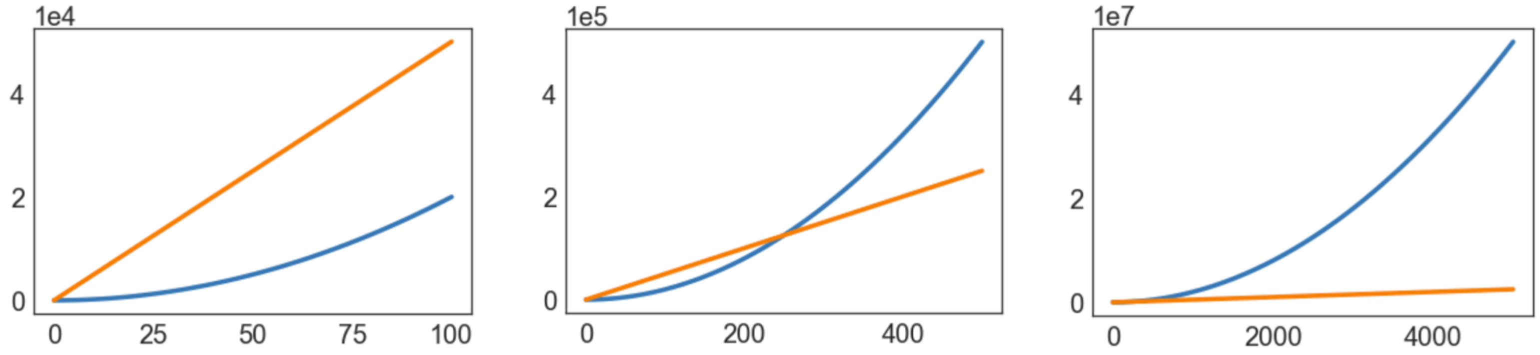
Recall $O(\cdot)$ notation from 61B!!

## 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 $N_0$.

$N > N_0$

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?

$$
\begin{array}{r}
\overbrace{12345678910109876543 21}^{n \text{ digits}} \\
\times\ 10987654321123456789 10 \\
\hline
\end{array}
$$

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:

12

27 x 19 ✓

1 13 38 ✓

~~0 6 76~~

1 3 152 ✓

1 1 304 ✓

~~0 0 608~~

513

**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.

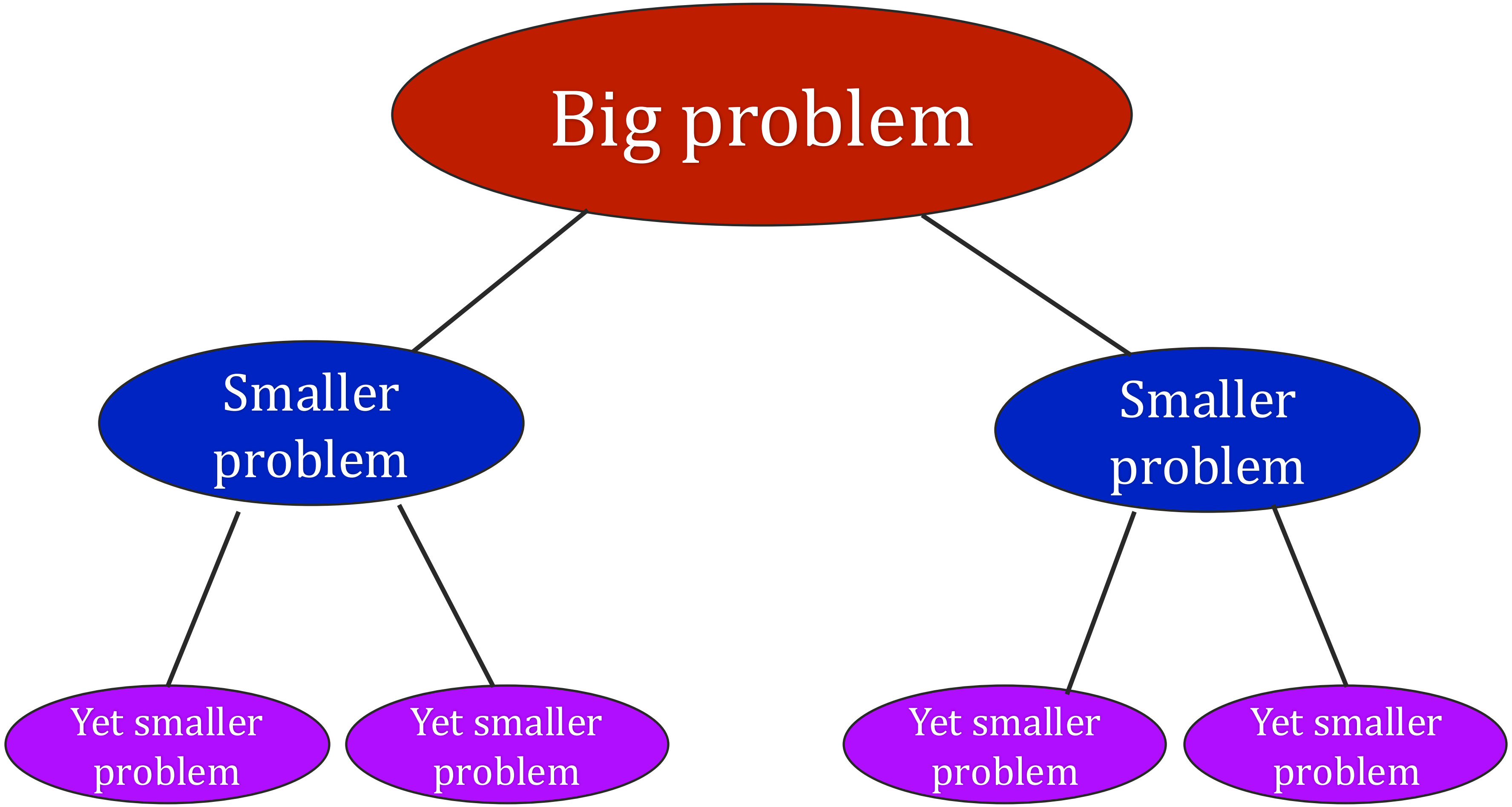
# There is a way to do better!

- **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:

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

$$1234 \times 5678 = (12 \times 100 + 34) \times (56 \times 100 + 78)$$

$$= \underbrace{(12 \times 56)}_{1} \times 10^4 + (\underbrace{12 \times 78}_{2} + \underbrace{34 \times 56}_{3}) \times 100 + \underbrace{(34 \times 78)}_{4}$$

2-digit multiplication

One 4-digit product $\Longrightarrow$ Four 2-digit products

(simplify: assume even $n$)

# The algorithm

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

(simplify: assume even $n$)

# The algorithm

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

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

$y = c \times 10^{n/2} + d$

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

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

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.

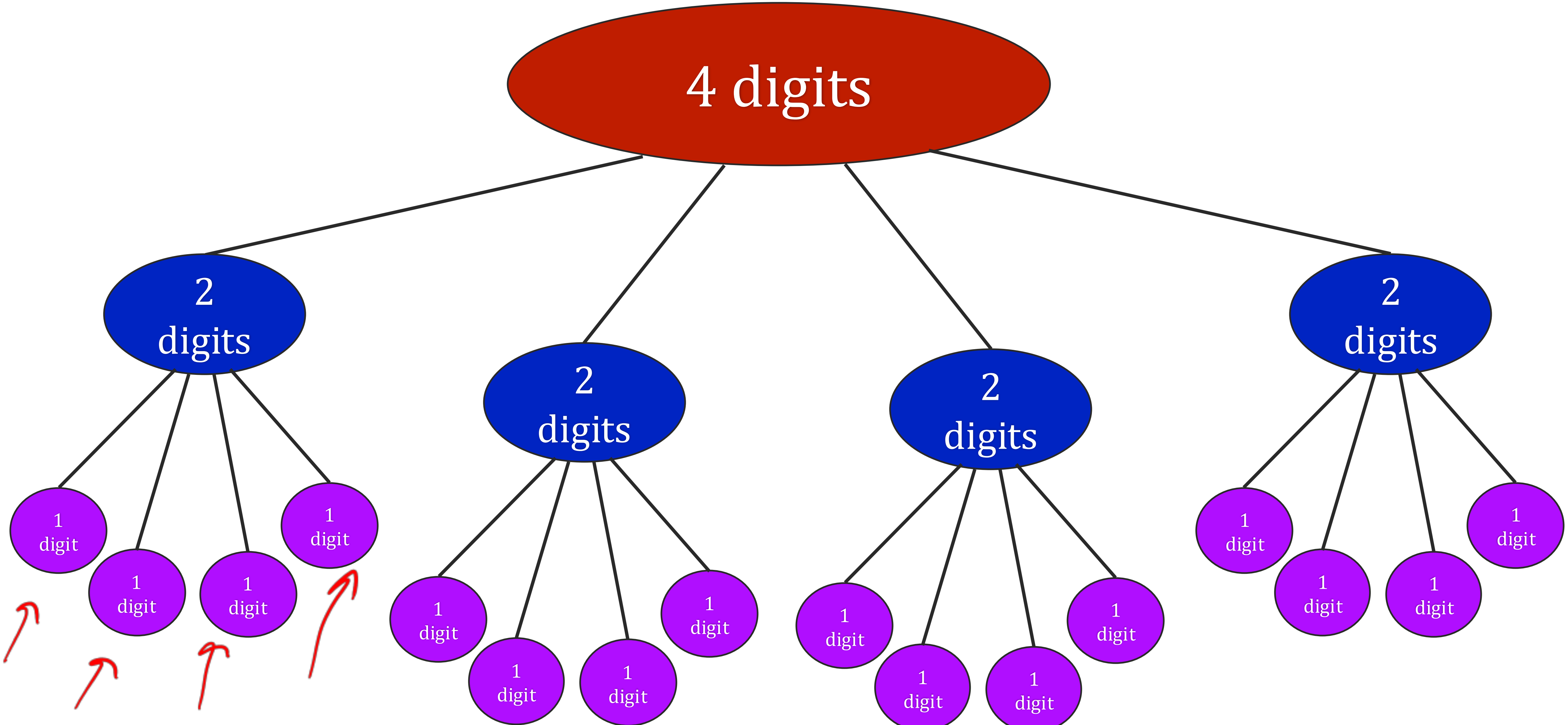
We wanted to count 1-digit operations. So, what should we do now?

**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 $n$s 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.


$$\boxed{2^{\log_2(n)} = n}$$

$$\frac{n}{2^t} = 1 \Rightarrow n = 2^t$$

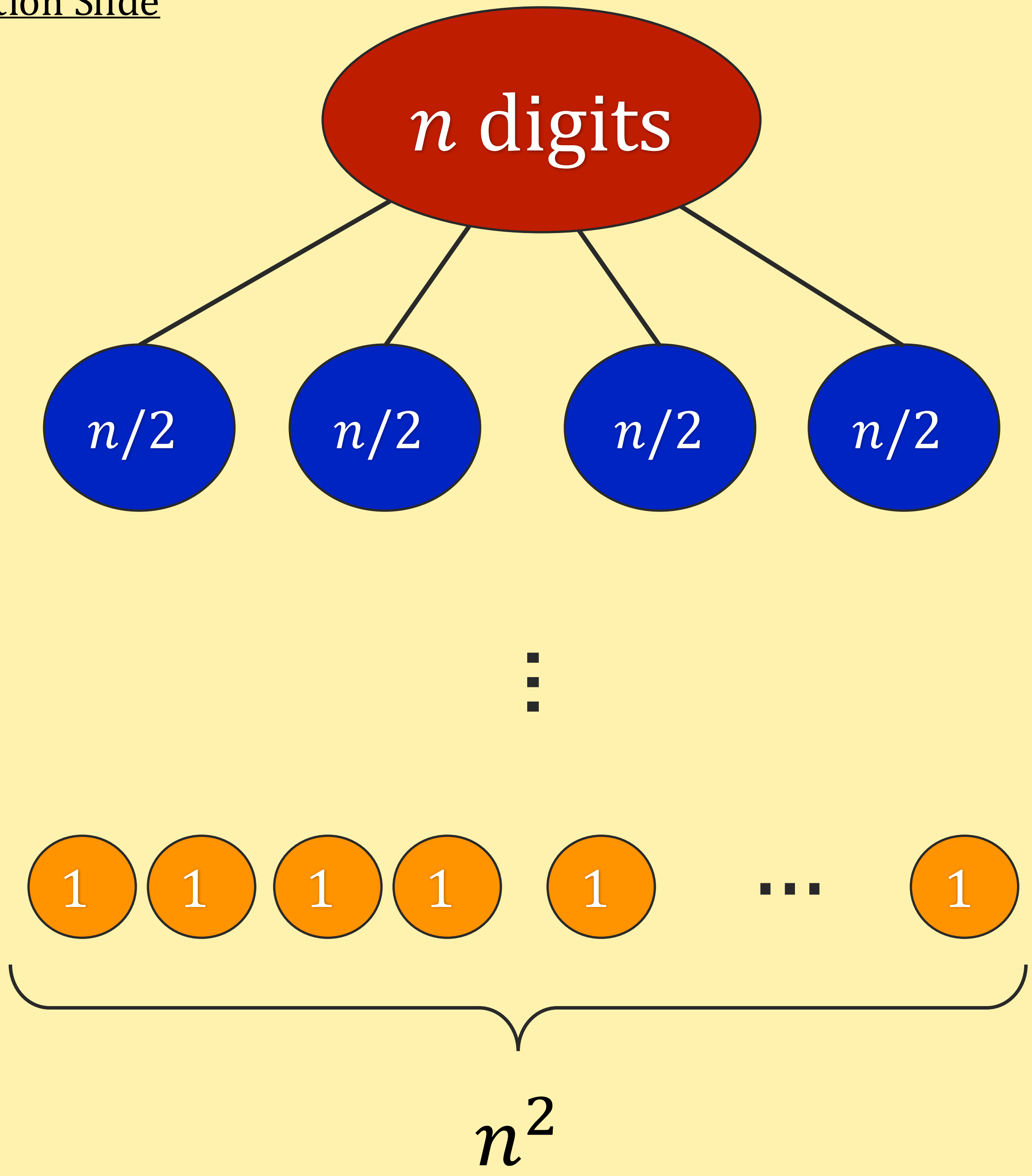
$$4^t = 4^{\log_2(n)} = n$$

$\log_2(4)$ $\quad 2$

$= n$

| layer # | # digits | # problem/nodes |
|---|---|---|
| 0 | $n$ | 1 |
| 1 | $n/2$ | 4 |
| 2 | $n/4$ | 16 |
| ⋮ | | |
| $t$ | $\frac{n}{2^t}$ | $4^t$ |
| $\log_2(n)$ | $\frac{n}{2^t} = 1$ | ? |

Solution Slide




| Layer | # of digits | # problems |
|---|---|---|
| $0$ | $n$ | $1$ |
| $1$ | $n/2$ | $4$ |
| ⋮ | ⋮ | ⋮ |
| $t$ | $\frac{n}{2^t}$ | $4^t$ |
| ⋮ | ⋮ | ⋮ |
| $\log_2(n)$ | $1$ | $4^{\log_2 n} = n^2$ |

# So, was there a point to Divide and Conquer?

# Wrap up

Integer Multiplication:

- We just need 1 more trick on top of Divide and Conquer to do better than grade school multiplication!

Divide and conquer:

- A useful and fundamental algorithmic tool. Fun too!

**Next time**

- Big-Oh and Asymptotic notations more formally
- Divide and Conquer some more
  - Continue with integer multiplication
  - Matrix multiplications!