Algorithm

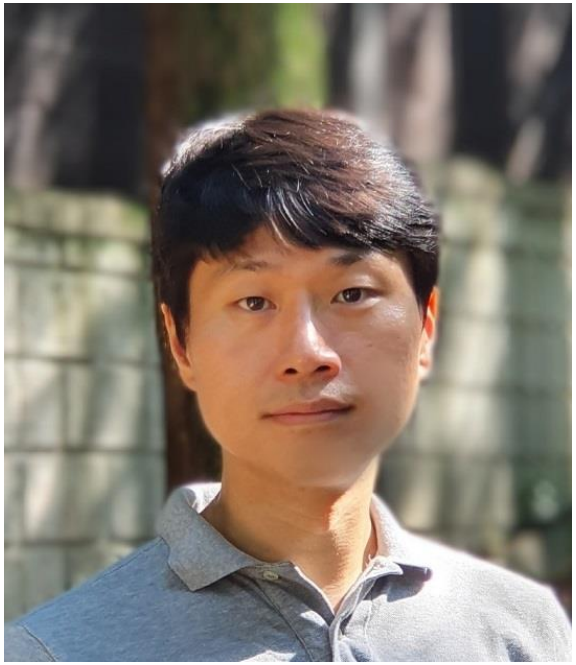
Lecture 1:

Introduction!

Today

- Course introduction?
  - Professor, policy, grade?
- Why are we here?
  - Why learn about algorithms?
- What is going on?
  - What is this course about?
- Can we multiply integers?
  - And can we do it quickly?

Professor



Jongbin Ryu

- Assistant professor, Ajou University
- Office: Paldal Hall. 604
- E-mail: jongbin.ryu@gmail.com / jongbinryu@ajou.ac.kr
- Web: https://sites.google.com/view/jongbinryu/
- Research interests: Deep learning and computer vision related problems

Why are you here?

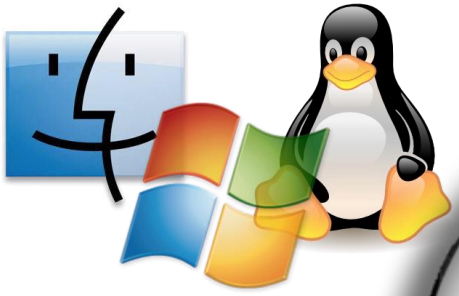
You are better equipped to answer this question than I am, but I'll give it a go anyway…

- Algorithms are fundamental.
- Algorithms are useful.
- Algorithms are fun!
- Algorithm class is a required course.

Why is Algorithm class required?

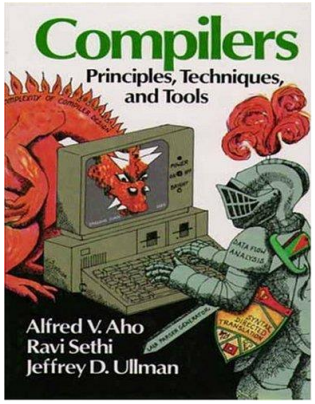
- Algorithms are fundamental.
- Algorithms are useful.
- Algorithms are fun!

Algorithms are fundamental



Operating Systems

Cryptography



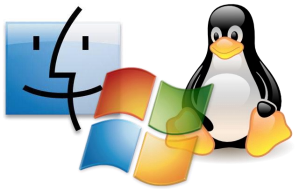
Compilers

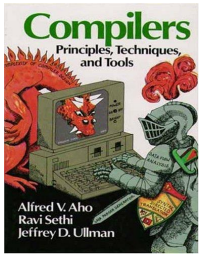
Networking

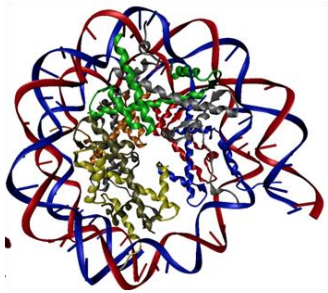
Computational Biology

Algorithms are useful

- All those things without the course numbers.
- As inputs get bigger and bigger, having good algorithms becomes more and more important!







Algorithms are fun!

- Algorithm design is both an art and a science.
- Many surprises!
- Many exciting research questions!

What’s going on?

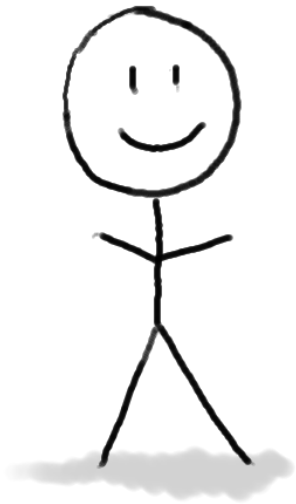
- Course goals/overview
- Policy
- Grade

Course goals

- The design and analysis of algorithms
  - These go hand-in-hand

- In this course you will:
  - Learn to think analytically about algorithms
  - Learn to communicate clearly about algorithms

Our guiding questions:



Does it work?

Is it fast?

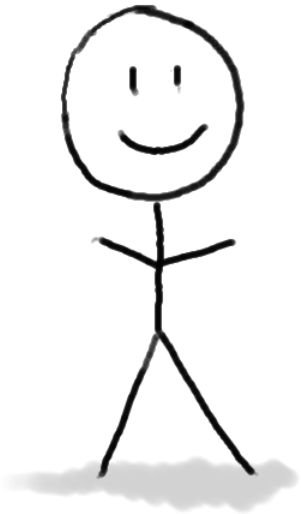
Can I do better?

Our internal monologue…

What exactly do we mean by better? And what about that corner case? Shouldn't we be zero-indexing?

Detail-oriented
Precise
Rigorous

Does it work?
Is it fast?
Can I do better?



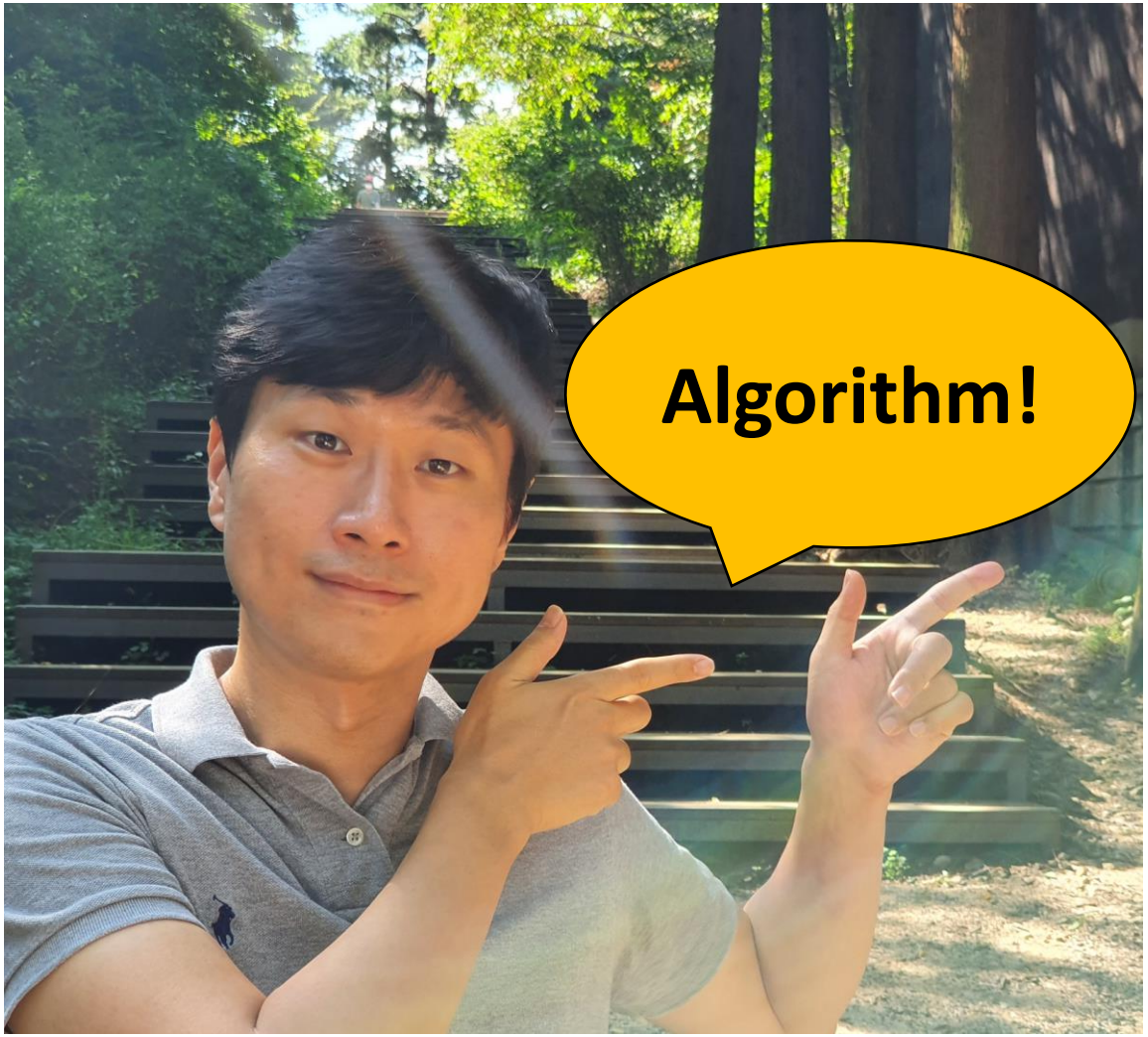
Dude, this is just like that other time. If you do the thing and the stuff like you did then, it'll totally work real fast!

Big-picture
Intuitive
Hand-wavy

Both sides are necessary!

Welcome to the lecture!

1. Work hard
2. Work smart
3. Ask for help



Policy

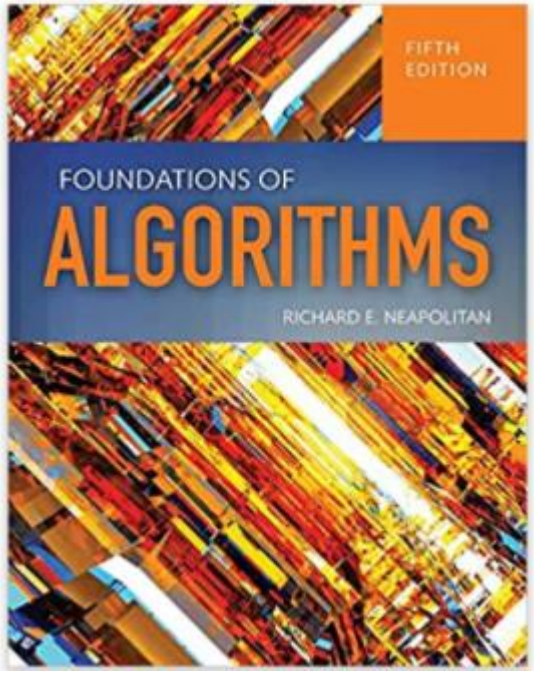
- Examination
  - Mid-term and final-term

- Home-work
  - Exercise (4times)
  - Lecture-summary (each lecture)

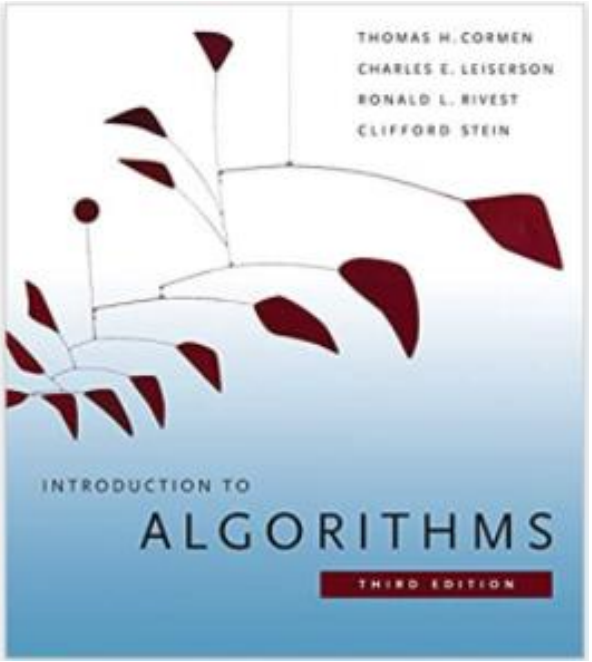
Syllabus

주	강 의 주 제	언어	담당교수	수업방법	평가방법	준비사항
1	Introduction, Asymptotics, MergeSort		유종빈	멀티미디어 활용 강의		
2	Solving Recurrences and the Master Theorem		유종빈	멀티미디어 활용 강의		
3	Median and Selection		유종빈	멀티미디어 활용 강의	보고서 평가	
4	QuickSort and BucketSort		유종빈	멀티미디어 활용 강의		
5	Binary Search Trees		유종빈	멀티미디어 활용 강의		
6	Graphs, BFS and DFS		유종빈	멀티미디어 활용 강의	보고서 평가	
7	Graphs, BFS and DFS / Recap		유종빈	멀티미디어 활용 강의		
8	Midterm Exam		유종빈		중간지필평가	
9	Dijkstra and Bellman-Ford		유종빈	멀티미디어 활용 강의		
10	Dynamic Programming		유종빈	멀티미디어 활용 강의		
11	Dynamic Programming		유종빈	멀티미디어 활용 강의	보고서 평가	
12	Greedy Algorithms		유종빈	멀티미디어 활용 강의		
13	Backtracking / Branch-and-Bound		유종빈	멀티미디어 활용 강의		
14	NP-hard problem		유종빈	멀티미디어 활용 강의	보고서 평가	
15	Minimum Spanning Trees and Recap		유종빈	멀티미디어 활용 강의		
16	Final Exam		유종빈		기말지필평가	

Textbook and lecture material



Foundations of Algorithms,
5th edition



Introduction to Algorithms,
3rd Edition

** Lecture material **

- Many lecture slides are imported from Prof. Mary Wootters at Stanford University

Grade

- Attendance: 10%
- Home-work: 30%
- Mid-term: 20%
- Final-term: 40%

Kakao Talk open-chat

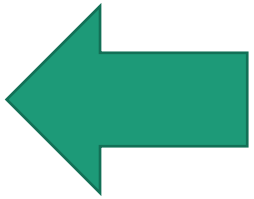
- url: https://open.kakao.com/o/블랙보드 공지 참조

- Join open-chat as your name.

Course goals

- Think analytically about algorithms
- Learn to communicate clearly about algorithms

Today's goals

- Karatsuba Integer Multiplication 
- Algorithmic Technique:
  - Divide and conquer
- Algorithmic Analysis tool:
  - Intro to asymptotic analysis

Let’s start at the beginning

This was kind of a big deal

$$\begin{array}{r} 97 \\ \times\ 15 \\ \hline \end{array}$$

Integer Multiplication

$$\begin{array}{r} 97 \\ \times\ 15 \\ \hline \end{array}$$

Integer Multiplication

$$
\begin{array}{rr}
 & 12345678959314 13 \\
\times & 4563823520395533 \\
\hline
\end{array}
$$

Integer Multiplication

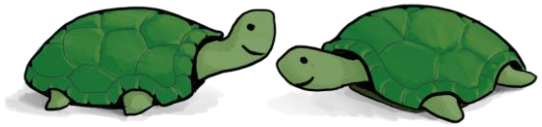
n

```
  12339257207527523846237642835683649183745238562 98
x 45623235823423952856234672350191307501335001375 3
```

???

How fast is the grade-school multiplication algorithm?

(How many one-digit operations?)



Think-pair-share Terrapins

About n^2 one-digit operations

Plucky the Pedantic Penguin

At most n^2 multiplications,
and then at most n^2 additions (for carries)
and then I have to add n different 2n-digit numbers…

Big-Oh Notation

- We say that Grade-School Multiplication

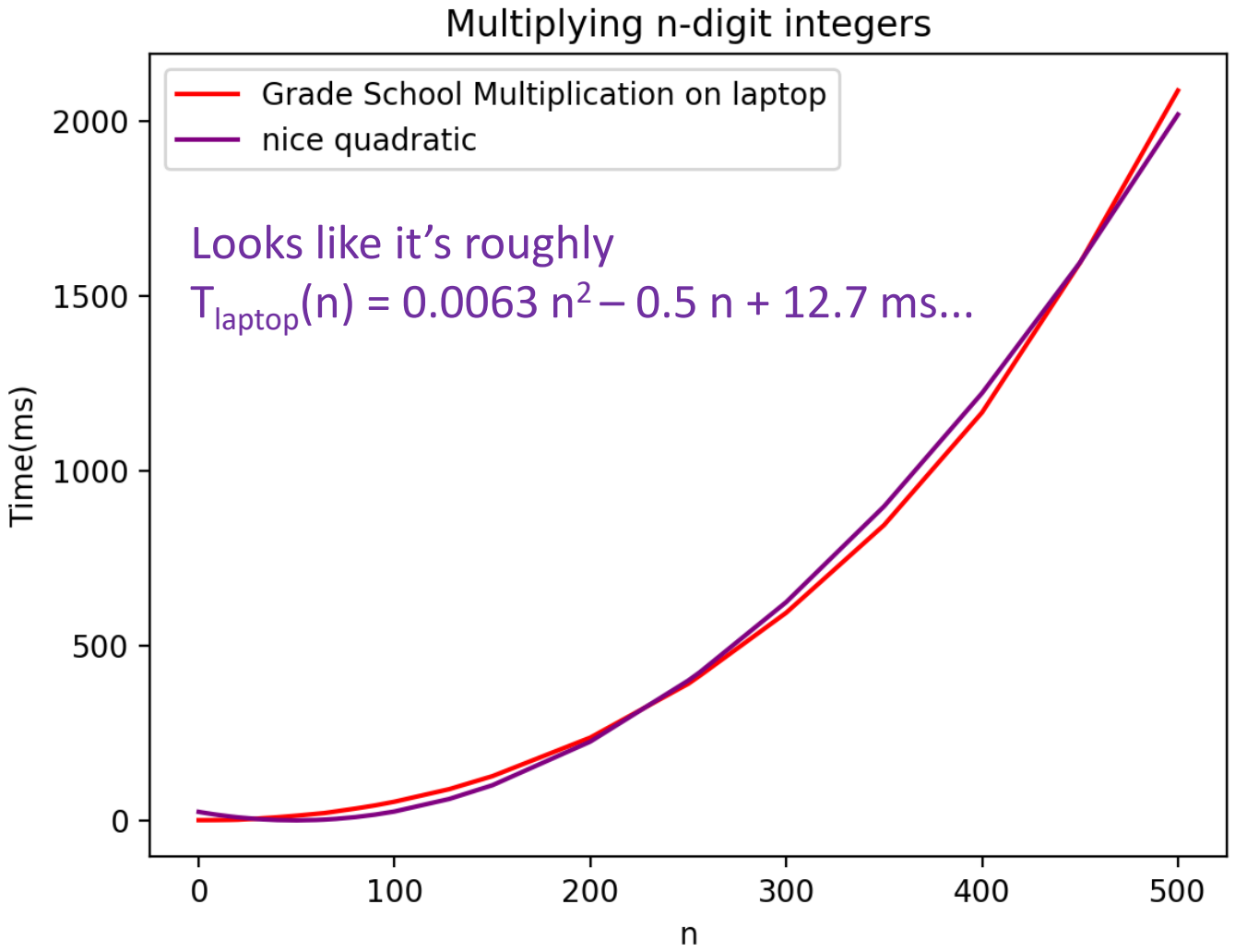
  "runs in time $O(n^2)$"

- Informally, big-Oh notation tells us how the running time scales with the size of the input.

highly non-optimized

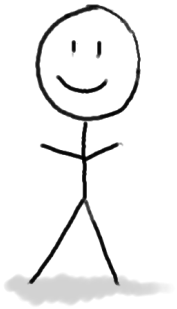
Implemented in Python, on a laptop

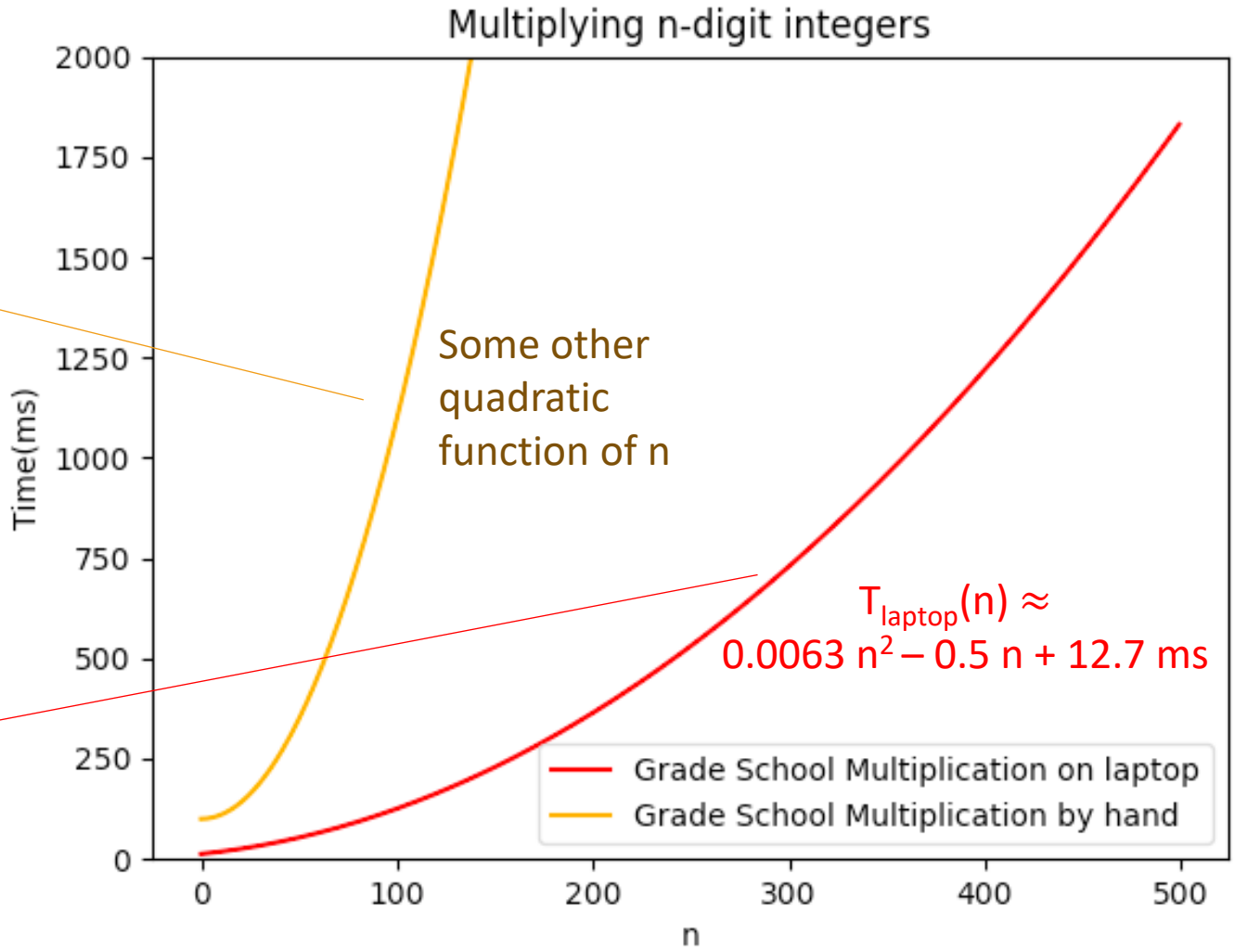
The runtime "scales like" n^2



Implemented by hand

The runtime still "scales like" n^2



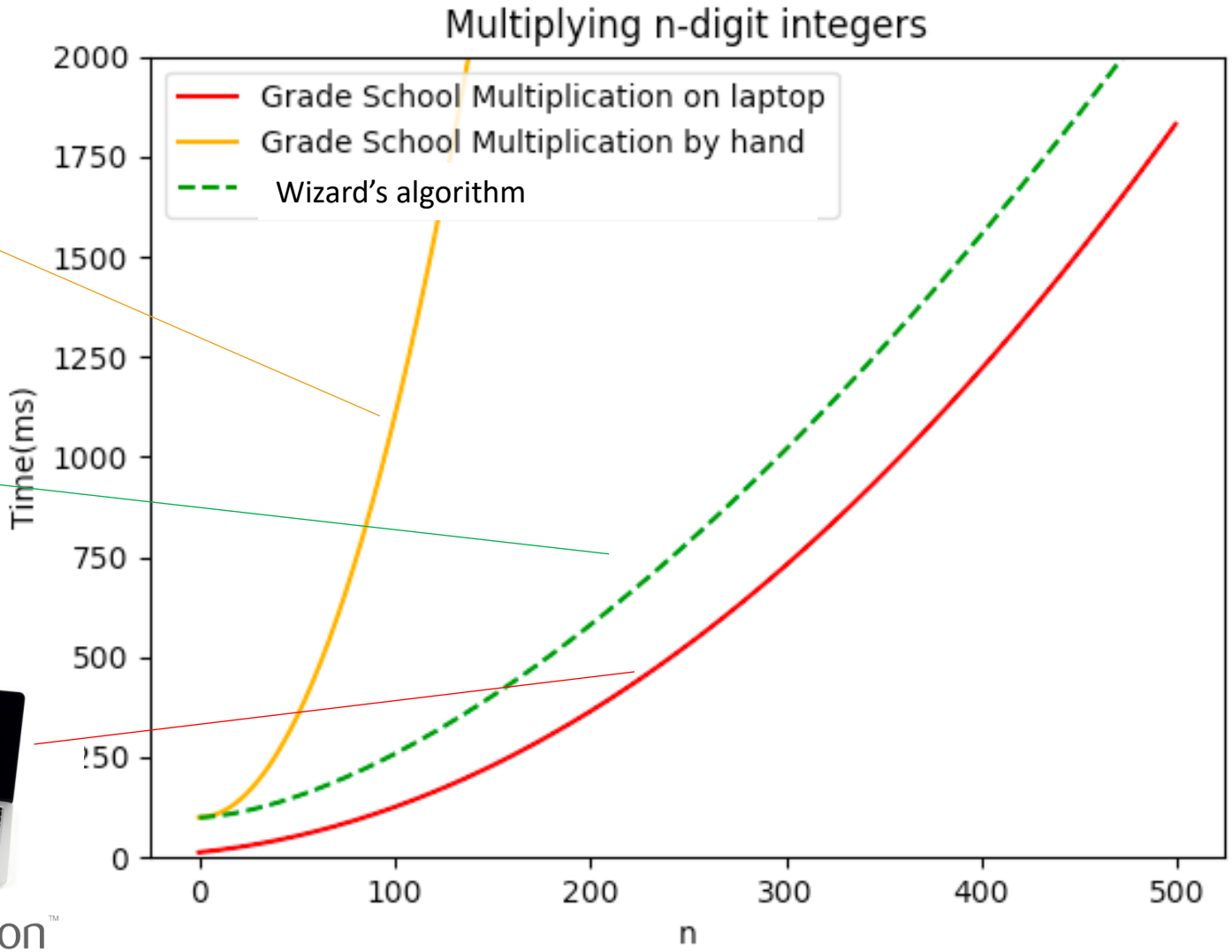


python™

Why is big-Oh notation meaningful?

$\approx \frac{n^{1.6}}{10} + 100$

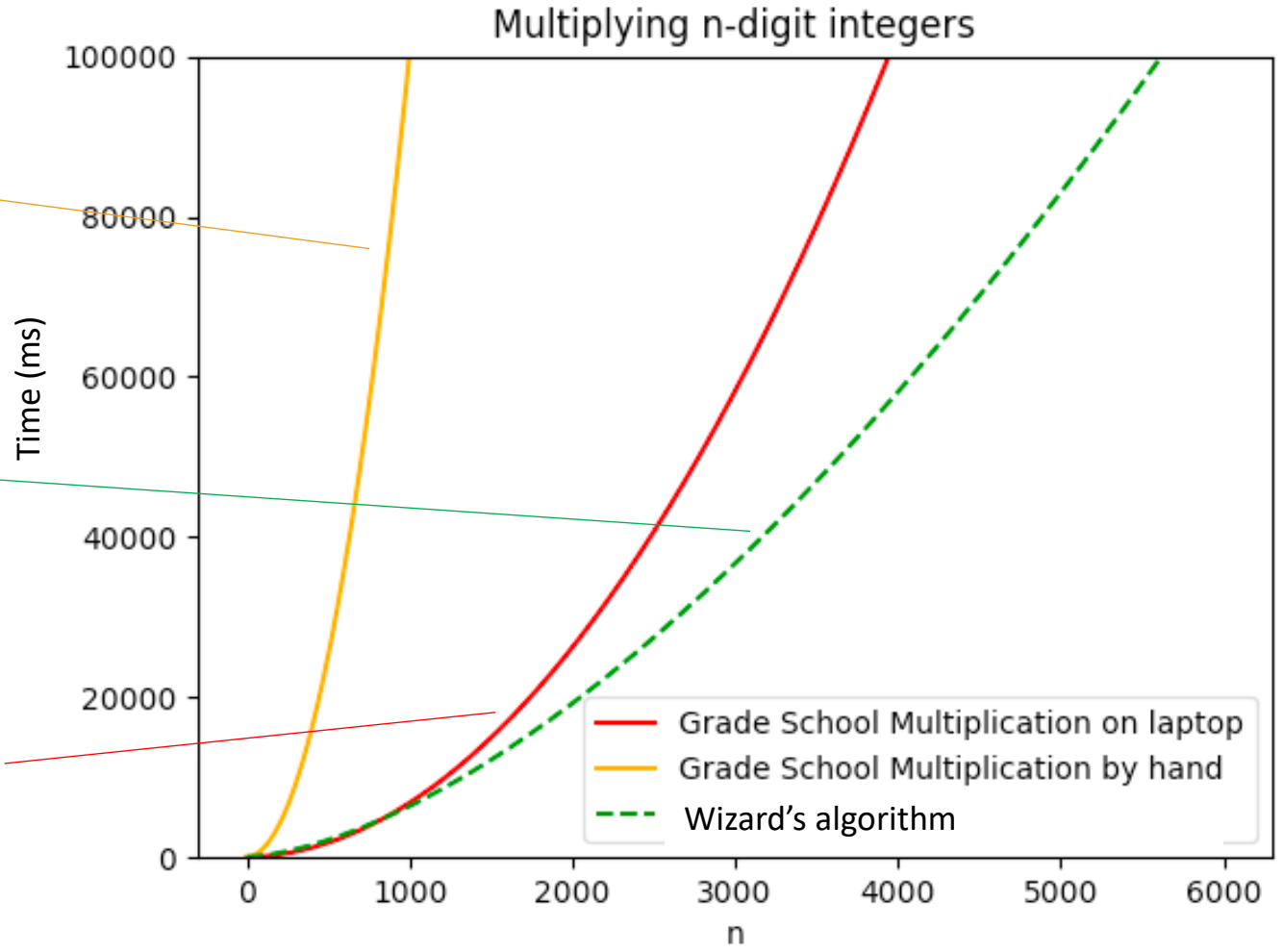
$\approx .0063n^2$



Let n get bigger…

$\approx \frac{n^{1.6}}{10} + 100$

$\approx .0063n^2$



Take-away

- An algorithm that runs in time $O(n^{1.6})$ is "better" than an an algorithm that runs in time $O(n^2)$.

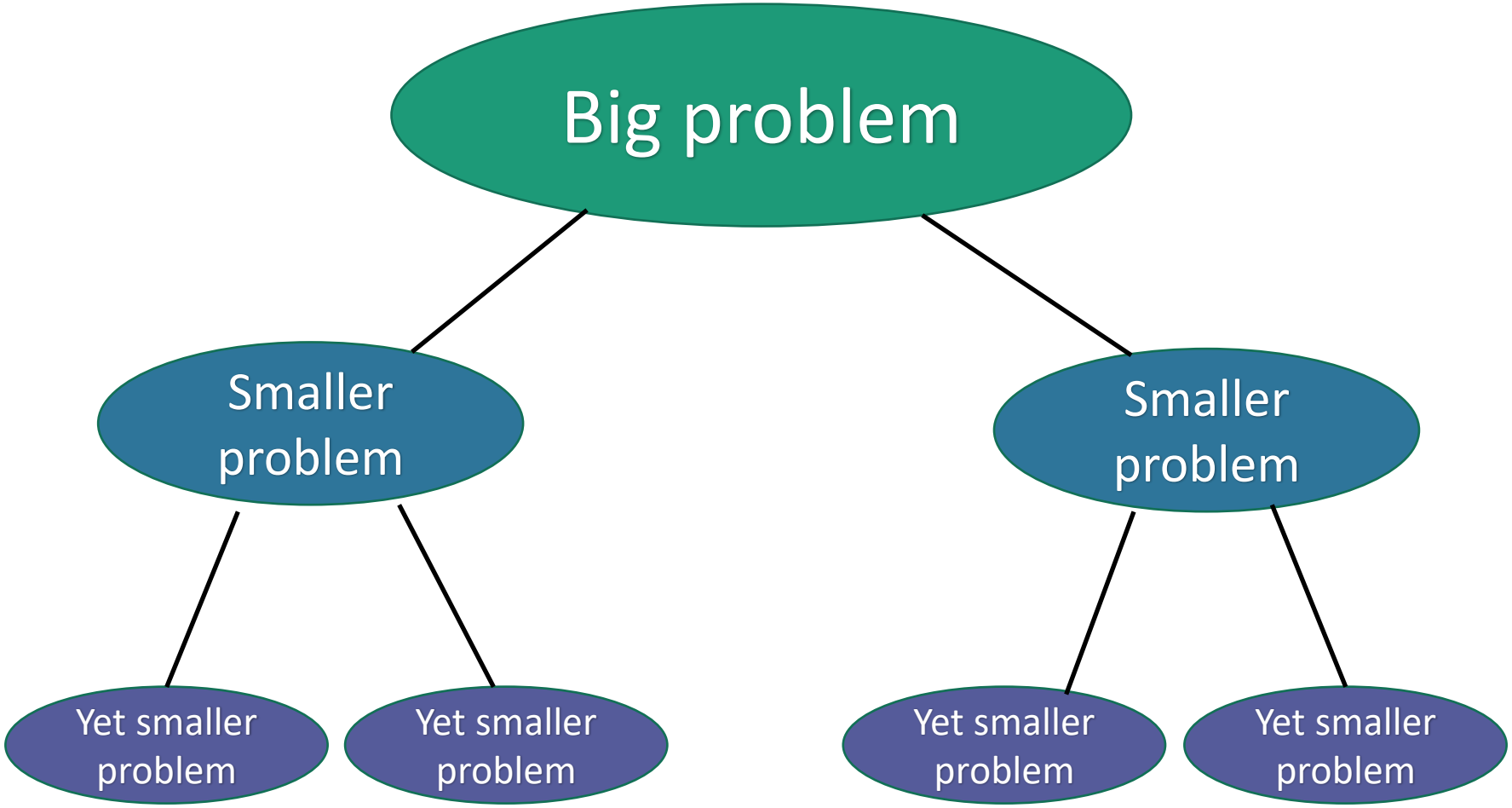
- So the question is…

Can we do better?

Can we multiply n-digit integers faster than $O(n^2)$?

Divide and conquer

Break problem up into smaller (easier) sub-problems



Divide and conquer for multiplication

Break up an integer:

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

$$1234 \times 5678$$

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

$$= \underbrace{(12 \times 56)}_{1} 10000 + \underbrace{(34 \times 56}_{2} + \underbrace{12 \times 78)}_{3} 100 + \underbrace{(34 \times 78)}_{4}$$

One 4-digit multiply  Four 2-digit multiplies

More generally

Suppose n is even

Break up an n-digit integer:

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

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

$$= \underbrace{(a \times c)10^n}_{1} + \underbrace{(a \times d}_{2} \underbrace{+ c \times b)10^{n/2}}_{3} + \underbrace{(b \times d)}_{4}$$

One n-digit multiply → Four (n/2)-digit multiplies

Divide and conquer algorithm

not very precisely…

(Assume n is a power of 2…)

x,y are n-digit numbers

Multiply(x, y):

- **If** n=1:
  - **Return** xy
- Write $x = a\ 10^{\frac{n}{2}} + b$
- Write $y = c\ 10^{\frac{n}{2}} + d$
- Recursively compute ac, ad, bc, bd:
  - ac = **Multiply**(a, c), etc..
- Add them up to get xy:
  - xy = ac 10^n + (ad + bc) $10^{n/2}$ + bd

Base case: I've memorized my 1-digit multiplication tables…

a, b, c, d are n/2-digit numbers

Make this pseudocode more detailed! How should we handle odd n? How should we implement "multiplication by 10^n"?

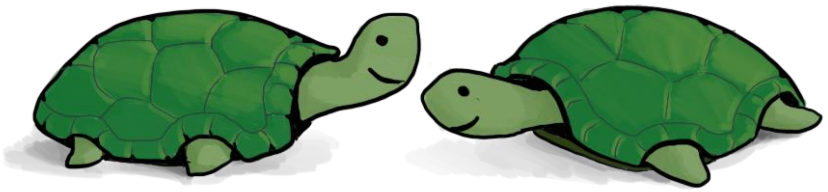
Siggi the Studious Stork

Think-Pair-Share

- We saw that this 4-digit multiplication problem broke up into four 2-digit multiplication problems

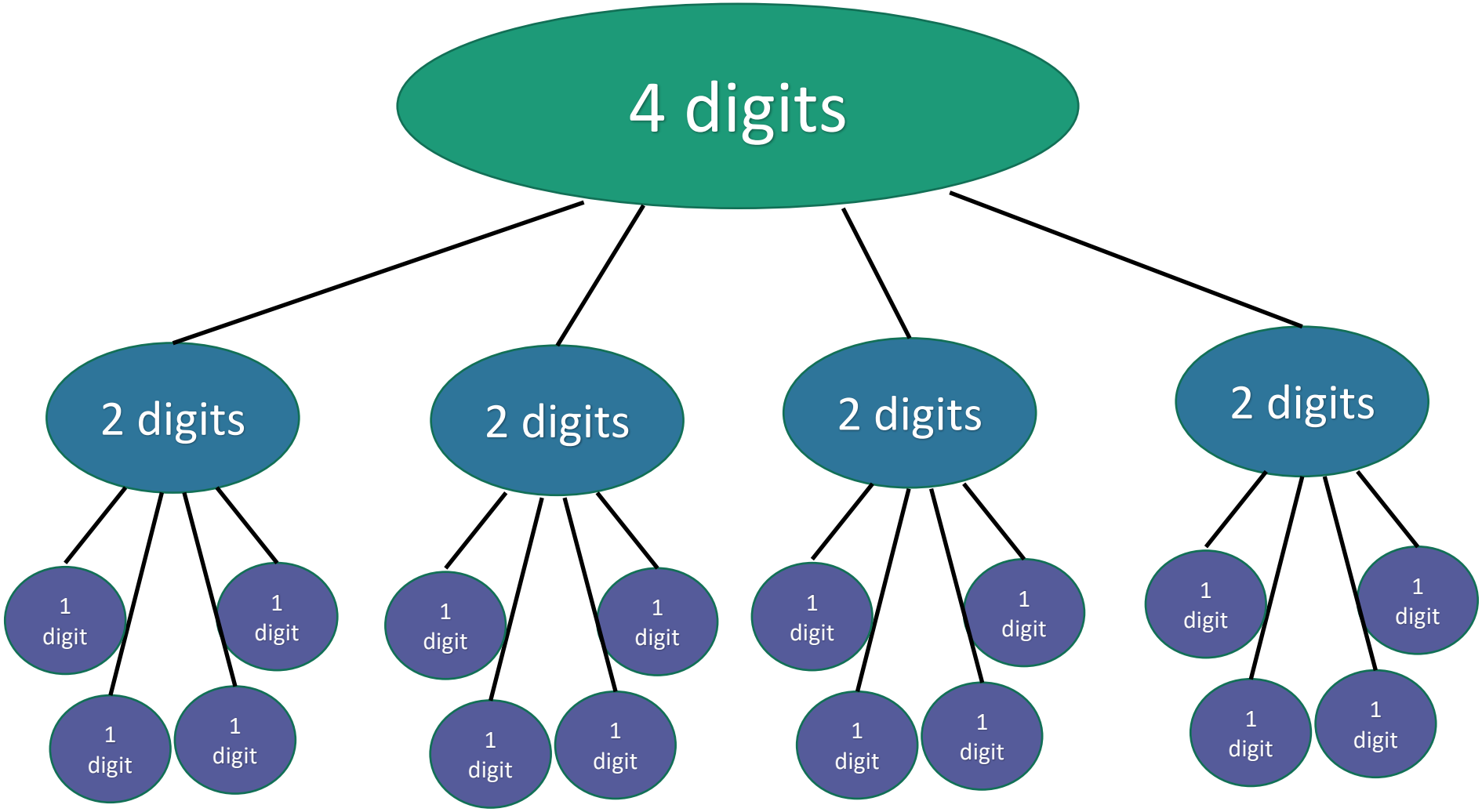
$$1234 \times 5678$$

- If you recurse on those 2-digit multiplication problems, how many 1-digit multiplications do you end up with total?



Recursion Tree

16 one-digit multiplies!

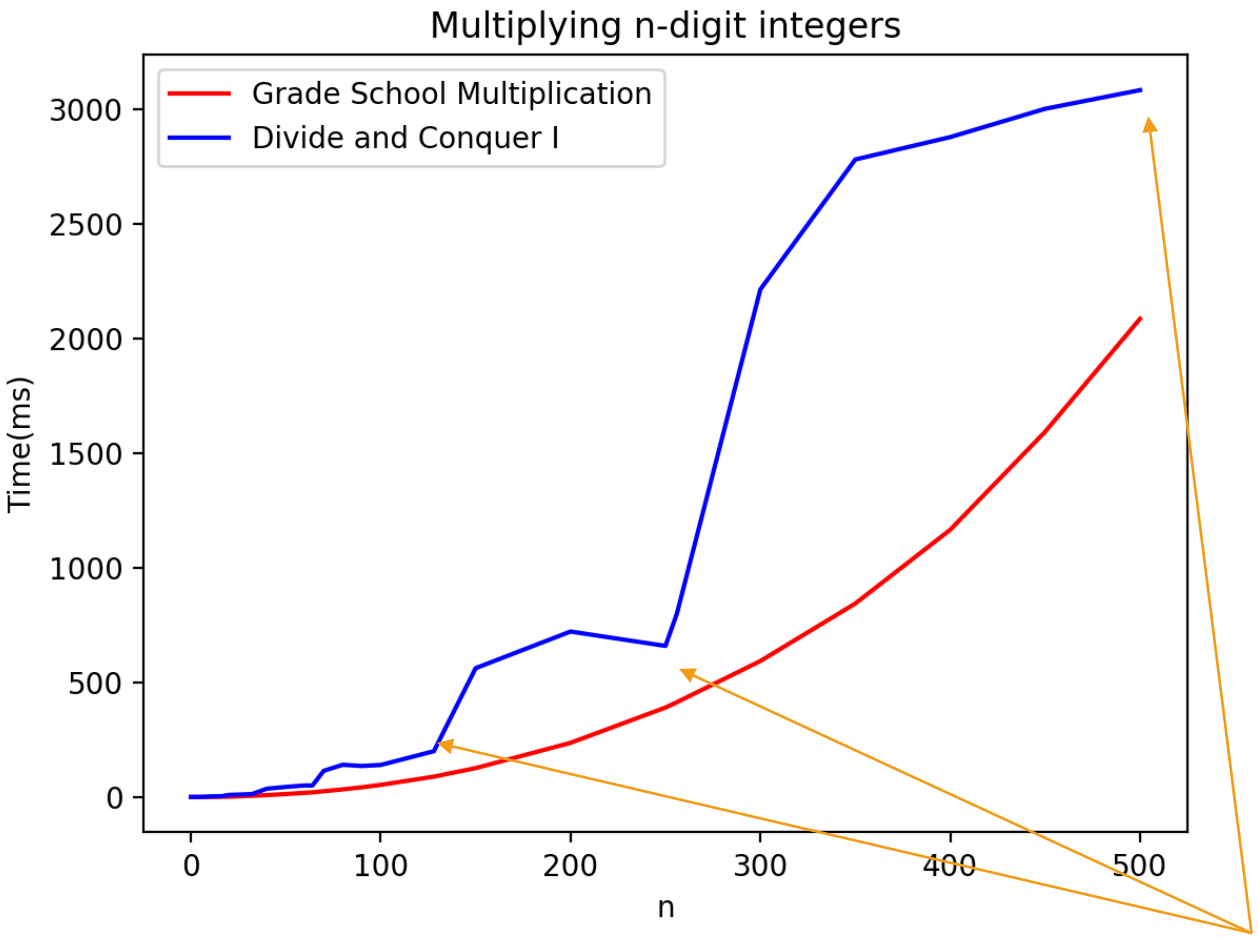


What is the running time?

- Better or worse than the grade school algorithm?

- How do we answer this question?
  1. Try it.
  2. Try to understand it analytically.

1. Try it.



Conjectures about running time?

Doesn't look too good but hard to tell…

Maybe one implementation is slicker than the other?

Maybe if we were to run it to n=10000, things would look different.

Something funny is happening at powers of 2…