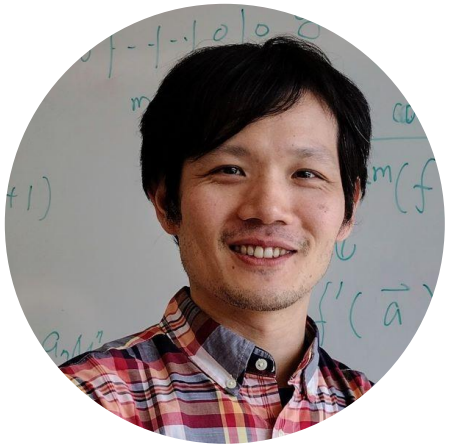
CS 3120 (DMT2)
Theory of Computation

Wei-Kai Lin

Jan 13. 2026

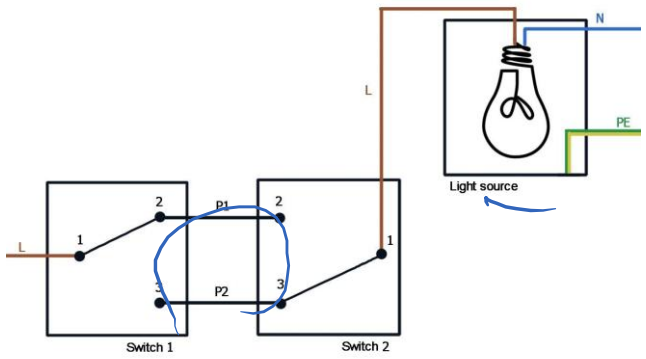


High-Level Introduction

(A couple of examples)

Goals (syllabus in one sentence)

To understand the power and limitations of computation



circuits

~~laptops~~ Turing machines

Quantum computers

Our approach to this goal

- Define computation formally
- Answer two important questions:
  - Q: What is computable with “unlimited resources” (computability)
  - Q: What is computable with “limited resources” (complexity)

Concrete analogy: LEGO

Model

- Build cool stuff  from bricks 

Computable (result)

Computation & Resources

Main example: Add and Multiply

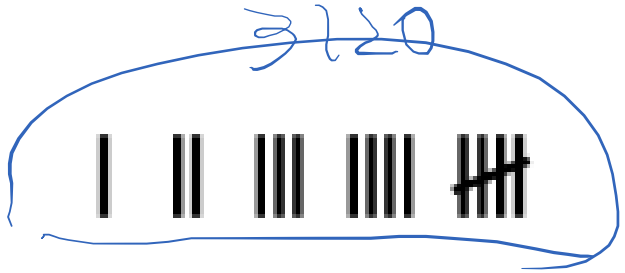
Step 0: deciding about representation

write nums

- How to write (represent / encode) numbers?
  - Only then can we give them as inputs to “algorithms.”
- How should we represent a (possibly large) number?

Bad representations

- Stone age: tally marks (30,000 years ago)



- Roman numerals

39 = XXX + IX = **XXXIX**.

246 = CC + XL + VI = **CCXLVI**.

789 = DCC + LXXX + IX = **DCCLXXXIX**.

2,421 = MM + CD + XX + I = **MMCDXXI**.

What's the largest number represented by a single Roman numeral?

AI Overview

The largest number represented by a single, standard Roman numeral character is **M**, which stands for **1,000**; larger numbers are formed by combining these symbols (like MMMCMXCIX for 3,999) or by using a vinculum (overline) to multiply a numeral by 1,000 (e.g., $\overline{M}$ for a million).

Standard Roman Numerals:

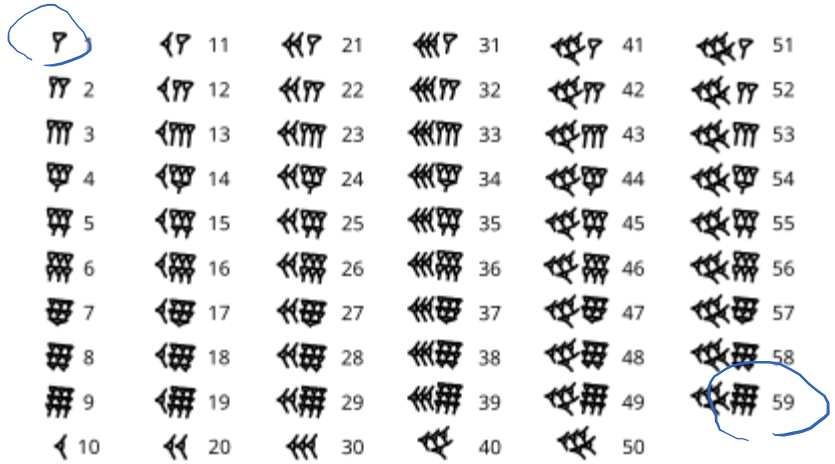
- **I** = 1
- **V** = 5
- **X** = 10
- **L** = 50
- **C** = 100
- **D** = 500
- **M** = 1,000 (the largest single symbol)

MM ··· M = 12,000
12

- In "additive" systems like Roman's, characters carry their value.
  - Sun to earth distance in kilometers will require > 100,000 symbols!

Place-value number system

- Babylonian (2000BC)



- The number: $x_k x_{k-1} x_{k-2} \ldots x_2 x_1 x_0$

  Means: $\sum x_i \cdot b^i$

b=10

- Extremely important development in history!

Is the Babylonian representation “optimal”?

- Optimal: represent more numbers in the same length

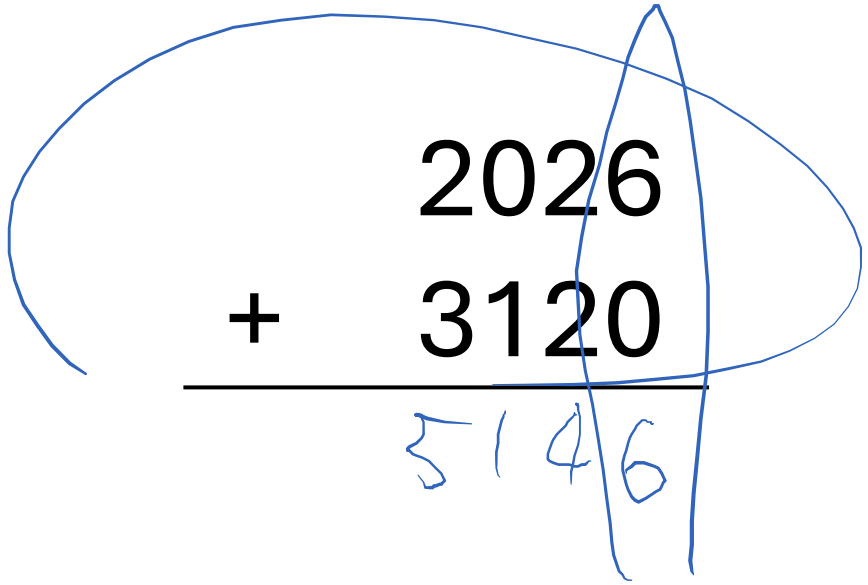
- Length k is k digits
- We can write 2^k distinct numbers (b = 2)
- We can not write $2^k + 1$ distinct numbers (by pigeon hole principle)

b^k v.s. tally marks.

- Yes, optimal

Integer Addition

- The grade school algorithm:



- Is this algorithm "optimal"?

YES

How much time to add two n-digit nums:

$2n+7 = O(n)$ time,

need $\Omega(n)$ time to read input

Integer Multiplication via repeated addition

Input: Non-negative integers x,y

Output: Product x·y

2026 · 3120

1. Let result←0
2. for{i=1,…,y}
   result ← result + x
3. endfor
4. return result

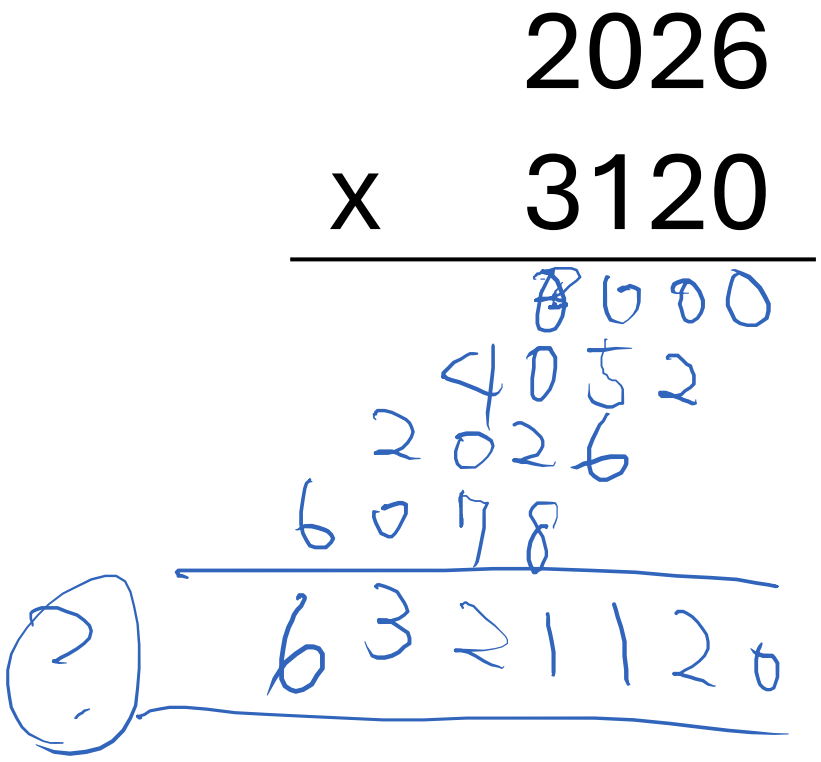
Simple: Add x for y times

Grade-school multiplication

1. Write $x = x_{n-1}x_{n-2}\cdots x_0$ and $y = y_{m-1}y_{m-2}\cdots y_0$ in decimal place-value notation. (x_0 is the ones digit of x, x_1 is the tens digit, etc.)
2. Let result←0
3. for{i=0,…,n−1}
   a) for{j=0,…,m−1}

      result ← result + $10^{i+j} \cdot x_i \cdot y_j$

   b) endfor
4. endfor
5. return result



Comparing Algorithms

- Suppose we multiply 2 numbers, each of 100 digits
- Method 1 requires about 10^{100} operations (additions).
- Method 2 requires about 100×100 "simple operations".
- Universe's age is < 10^{18} seconds

Kolmogorov (1960) conjecture

Can we do better?

NO

Grade-school multiplication

time $O(n^2)$

$$\begin{array}{cc} \bar{x} & \underline{x} \\ \times \\ \bar{y} & \underline{y} \\ \hline \bar{x}\bar{y} & \underline{x}\underline{y} \\ + \\ \bar{x}\bar{y} & \underline{x}\bar{y} \\ \hline \bar{x}\bar{y} & (\bar{x}\underline{y}+\underline{x}\bar{y}) \quad \underline{x}\underline{y} \end{array}$$

Yes!

a few weeks

Karatsuba's multiplication

20 yr old graduate student

$$\begin{array}{cc} \bar{x} & \underline{x} \\ \times \\ \bar{y} & \underline{y} \\ \hline & (\bar{x}+\underline{x})(\bar{y}+\underline{y}) \quad \underline{x}\underline{y} \\ + \\ \bar{x}\bar{y} & -\bar{x}\bar{y}-\underline{x}\underline{y} \\ \hline \bar{x}\bar{y} & (\bar{x}\underline{y}+\underline{x}\bar{y}) \quad \underline{x}\underline{y} \end{array}$$

Karatsuba's, recursion

Base case:

$\bar{x}$ and $\underline{x}$ are 1 digit each, and so do $\bar{y}$ and $\underline{y}$.

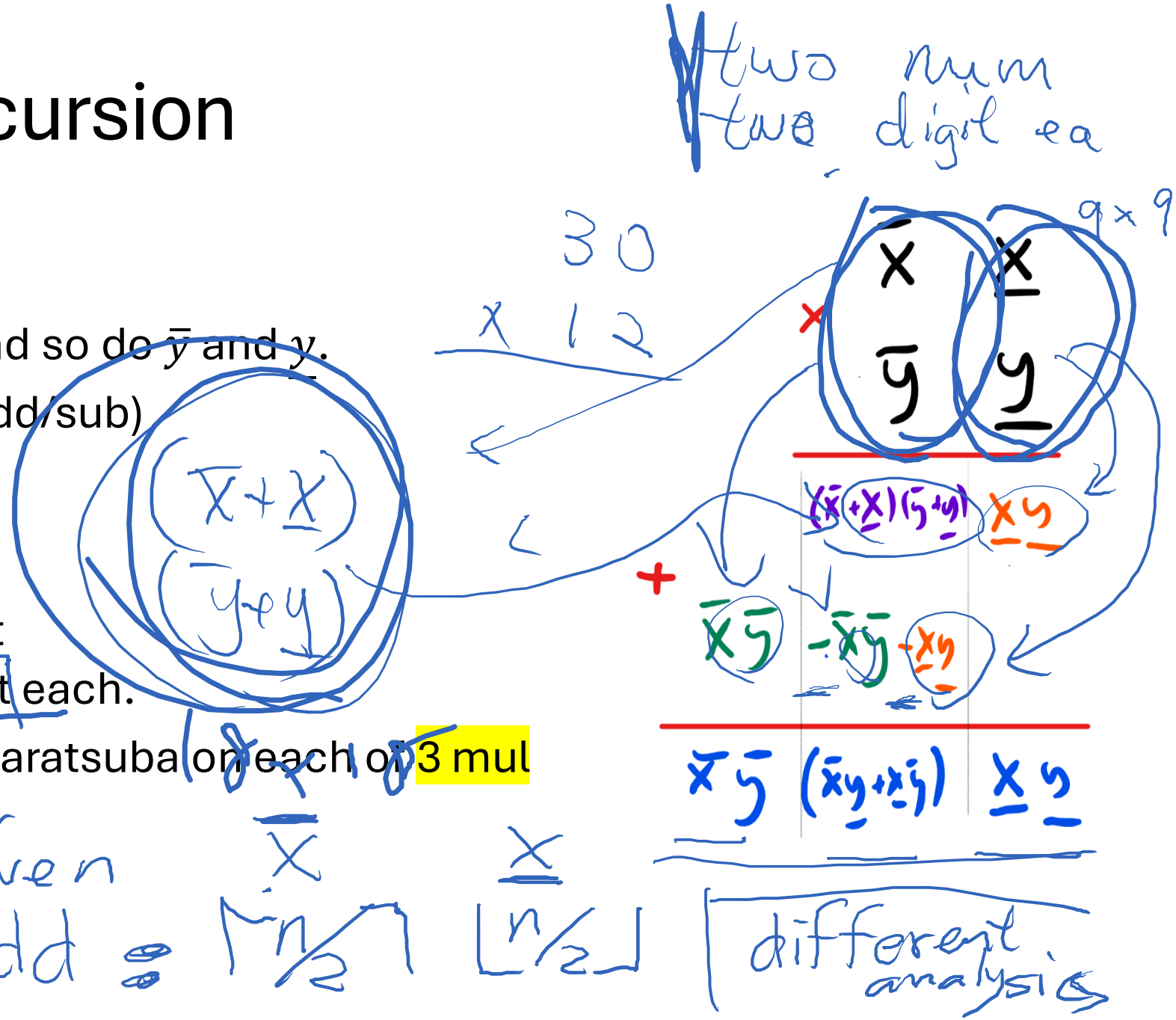
Perform 3 mul (and then add/sub)

Recursive case:

For n-bit multiplication, let

$\bar{x}$, $\underline{x}$, $\bar{y}$, and $\underline{y}$ are n/2-digit each.

Recursively call n/2-digit Karatsuba on each of 3 mul (and then add/sub)



Analyzing Karatsuba's method

- Let $C \cdot n$ be the time it takes to add two n digit numbers for some constant C (depending on how we do addition)
- Let $T(n)$ be the time it takes to multiply two n digit numbers.
- Recursion:

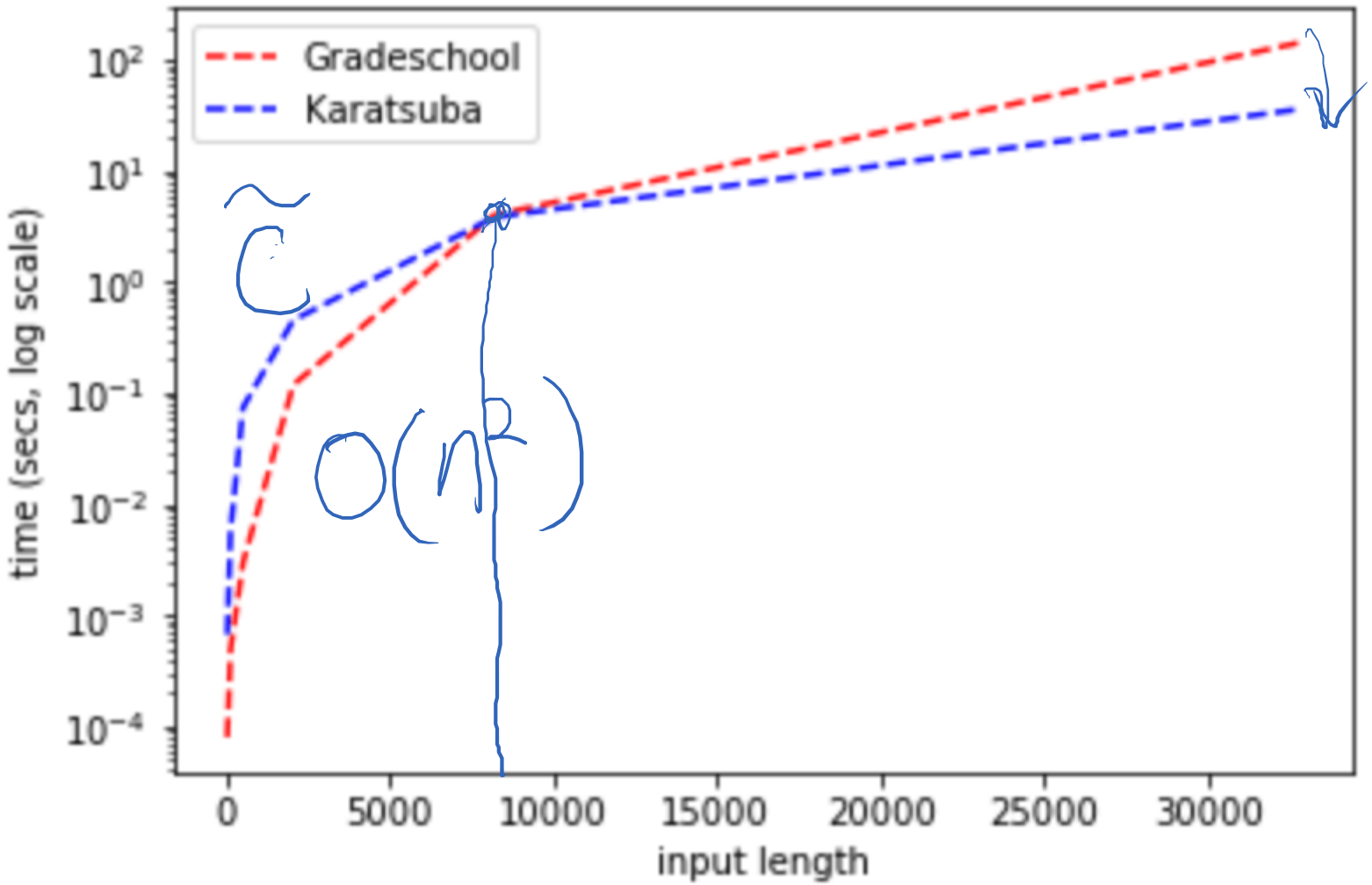
$$T(n) \leq 3\,T\left(\frac{n}{2}\right) + C \cdot n$$

base case

What details are missing?

- Solving the recursion, gives $T(n) \leq n^{\log_2 3} \leq O(n^{1.58})$

$O(n^{\log_2 3}) = \tilde{C} \cdot n^{\log_2 3}$ · const > 10


Gradeschool
Karatsuba
time (secs, log scale)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5000
10000
15000
20000
25000
30000
input length
$O(n^2)$

Can we do even better?

- Yes! using the so-called "Fourier analysis"
- Fourier Transform: map a number x into $FFT(x)$ such that

$$x \cdot y = FFT^{-1}\big(FFT(x) + FFT(y)\big)$$

- $FFT(\dots), FFT^{-1}(\dots)$ can be computed in time $\sim \boldsymbol{n \log n}$ where n is the bit length of x (this way of computing is known as fast Fourier Transform)

Can we do even better than $O(n \log n)$?

trivial LB $\Omega(n)$

It is a major open question!

It is a recent result to obtain O(n log n) *exactly*:
Integer multiplication in time $O(n \log n)$. David Harvey, Joris van der Hoeven. Annals of Mathematics, Pages 563-617 from Volume 193 (2021), Issue 2.

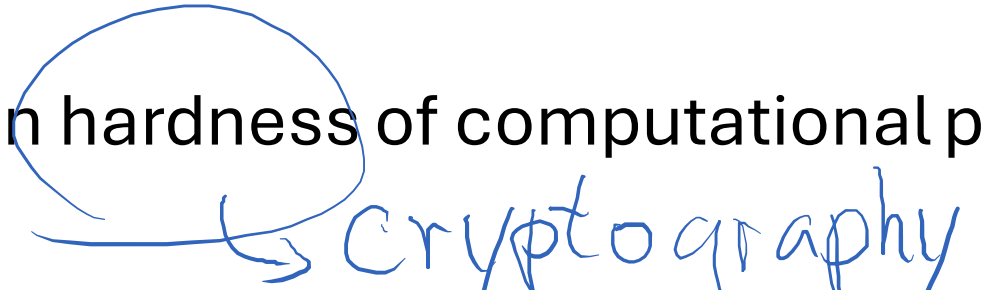
Examples of Other Computational Tasks

- Adding two (say n digit) numbers
- Multiplying two integers
- Factoring integers into primes?
- Solving linear sets of equations (with noisy equations?)
- Solving the Travelling Salesman Problem
- Solving polynomial equations for integer solutions

- Software verification
  (eg, never writes into a dangerous memory location)

Further questions: important but not our focus

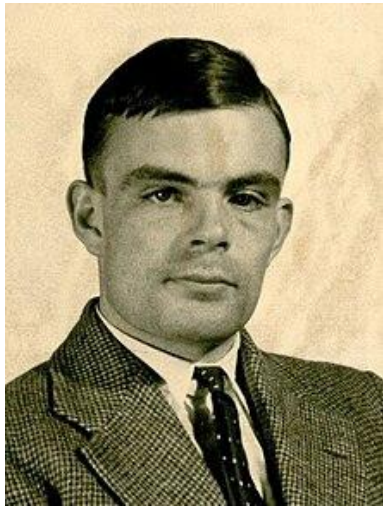
- Is randomness useful?
- Do problems become easier if we just want "approximate" solutions?
- Is there any "benefit" in hardness of computational problems?
- Do laws of physics help with computational power?



quantum

Roadmap for this course

1. Preliminary mathematics
2. Simple (but already strong) circuits model
3. Finite automata and regular expressions
4. Free unlimited accessible memory (Turing/RAM machines)
5. Efficient computation using limited resources



Main messages of today

- It is both intellectually and financially worth to understand computation
- The efficiency gain through algorithmic design could be significantly more than hardware improvement
- We want to understand the optimality of our results. Understanding the limitations is also called: “negative results”

Logistics and more on Syllabus

Meetings

- Tuesdays and Thursdays, 9:30 – 10:45pm in Olsson Hall 120

- https://weikailin.github.io/cs3120-toc

- We have talked about the objectives and highlights

- Best way to learn:
  participate in classes, ideally ask questions

Wei-Kai will be remote until February

- Due to the waiting time for US visa
  - Tentative (depending on the processing time)

- We encourage in-person participation
  - Lectures are Zoomed in the classroom, interruptions are encouraged
  - Does anyone volunteer to set up Zoom (bonus)?

- Fair to join remotely during this period (but less encouraged)
  - https://virginia.zoom.us/j/91325337637?pwd=caaOlObbQeK920eB1wEPIZmbH5zJTt.1

Preparation

- **Official Prerequisites:** To enroll in cs3120, you must have completed CS 3100 (DMT 1) and CS 3010 (DSA2) with a grade of C- or better.
- **Expected Background:** We expect you entering cs3120 to be comfortable using **proof techniques** from DMT1 (e.g., proof-by-contradiction, quantifiers, and induction). From DSA2, we expect you to have good understanding of the most common **asymptotic** operators and using them
- **Programming:** We also expect you to be able to read and write short programs. We will use the **Python** programming language for some assignments.

Textbook/resources

- Boaz Barak, Introduction to Theoretical Computer Science.
  https://introtcs.org/public/index.html
  https://files.boazbarak.org/introtcs/lnotes_book.pdf
  - (Differ from Prof. Floryan and Prof. Pettit)
- Other resources might be posted too, occasionally.

- The slides will be posted on the course's page after meetings
- The (zoom) video recordings will be posted.

Teaching Team

- 4 amazing TAs!

Chase

Shiyu

Ethan

Eric

- Office hours and locations are posted on the course Google calendar

Communication

- **Calendar:** We will keep course deadlines, office hours, and other events on a public **google calendar**. You are expected to subscribe to this calendar.
  - https://calendar.google.com/calendar/embed?src=292e8ddb35fbfb071102435db2822547ef891ab66dfd7eb00e075e0ea54a45bd%40group.calendar.google.com

interactive

announce

- **Piazza:** We will use the website for most other course communications. We expect you to receive messages we send to the “Announcement” as well as any direct messages we send to you.
  Link to join: https://piazza.com/virginia/spring2026/cs3120

Assignments

- **Homework assignments, 10.** Due most weeks in the course (typically on Mondays at 10:00pm). We expect students to read and follow these carefully.
  You will be writing LaTeX and submitting your PDFs.
- **Quizzes, 10.** You are asked to pre-read and reflect on course material, and you are asked to answer simple questions, almost weekly.

Python prog

Exams

- **Exams.** We have five exams in the course.
- Midterms are all in class. We count the best 3 out of 4.
  - Midterm 0: Tuesday, Feb 3, 2026.
  - Midterm 1: Tuesday, Feb 24, 2026.
  - Midterm 2: Tuesday, Mar 24, 2026.
  - Midterm 3: Tuesday, Apr 14, 2026.
- Final: Monday, 4 May, 2:00pm - 5:00pm. (UVA schedule)

Item	Standard Weighting
Homework (10 expected)	27%
Quizzes	5%
Midterm	45%
Final Exam	23%

Honor expectation

- We believe strongly in the value of a community of trust and expect all the students in this class to contribute to strengthening and enhancing that community.
- All students are required to read, understand, and sign the course pledge.

Additional Info

- Special Circumstances: If you require access accommodation, please contact the Student Disability Access Center (SDAC)

- Other Accommodations: It is the University's policy and practice to reasonably accommodate students so that they do not experience an adverse academic consequence when serious personal issues conflict with academic requirements. Religious reasons, family obligations, personal crises, and extraordinary opportunities could all be potentially valid reasons for accommodations.

What to do next?

- Quiz 0:
  Registration survey, to be posted on Gradescope.
  https://www.gradescope.com/courses/1223997
  Due next Monday, ~~Jan 9~~, 10pm.

Jan 19

calendar link

Piazza