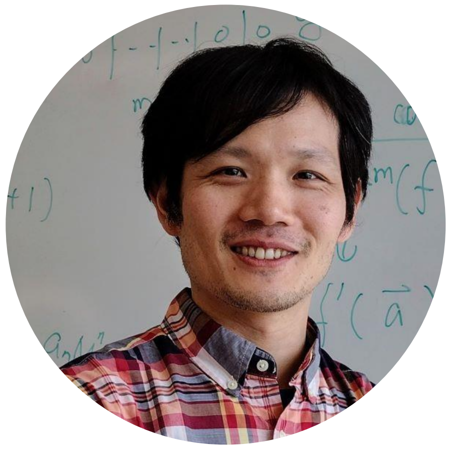
CS 3120 (DMT2) Theory of Computation

Wei-Kai Lin

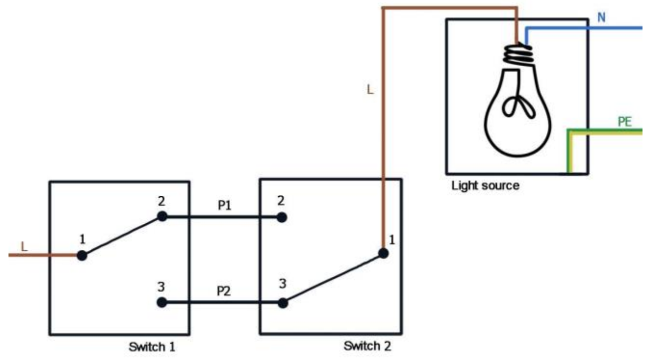


High-Level Introduction

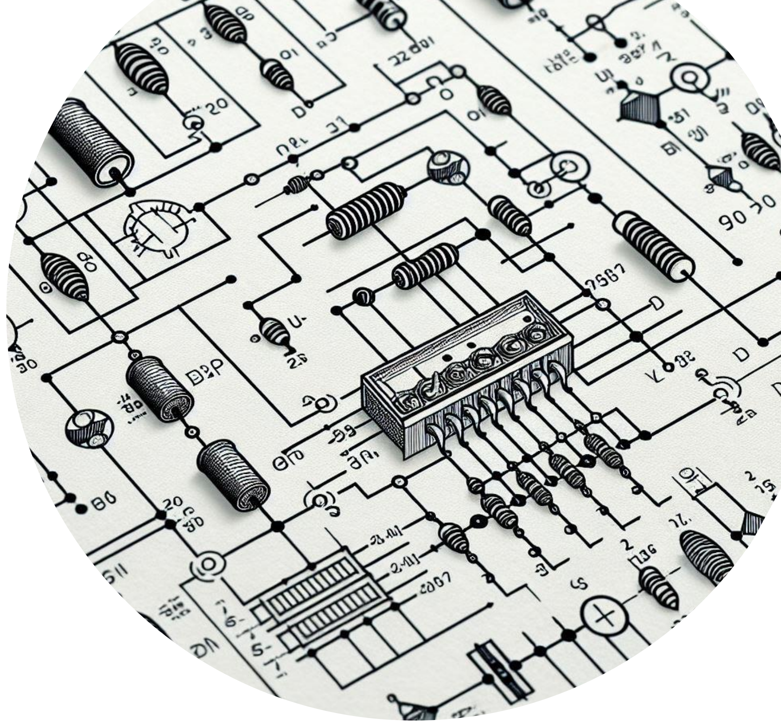
(A couple of examples)

Goals (syllabus in one sentence)

To understand the power and limitations of computation



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

- Build cool stuff from bricks

Computable
(result)

Computation
&
Resources

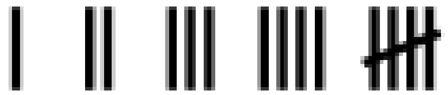
Main example: Add and Multiply

Step 0: deciding about representation

- 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**.

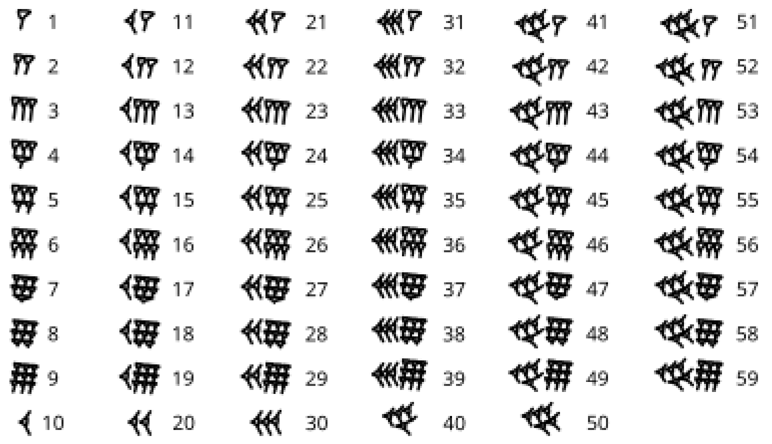
789 = DCC + LXXX + IX = **DCCLXXXIX**.

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

- 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)


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

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

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

- 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)

- Yes, optimal

Integer Addition

- The grade school algorithm:

$$\begin{array}{r} {}^{0}\ {}^{0}{}^{0}{}^{0}{}^{0} \\ 1234 \\ +\quad 3120 \\ \hline 4354 \end{array}$$

- Is this algorithm "optimal"? Yes

n digit input ⇒ O(n) time

No less "resource" bcs need to read all input digits

$\underset{n}{\wedge}$

Can we do $\sqrt{n}$ time? Parallel comp.

Integer Multiplication via repeated addition

Input: Non-negative integers x,y

Output: Product $x \cdot y$

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

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 + $10^{i+j} \cdot$ ~~$x^i \cdot y^j$~~   $x_i \cdot y_j$
   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

Can we do better?

Grade-school multiplication

$O(n^2)$

$$\begin{array}{r|c|c}
 & \bar{x} & \underline{x} \\
\times & \bar{y} & \underline{y} \\
\hline
 & \bar{x}\underline{y} & \underline{x}\underline{y} \\
+\ \bar{x}\bar{y} & \underline{x}\bar{y} & \\
\hline
\bar{x}\bar{y} & (\bar{x}\underline{y}+\underline{x}\bar{y}) & \underline{x}\underline{y}
\end{array}$$

Yes!

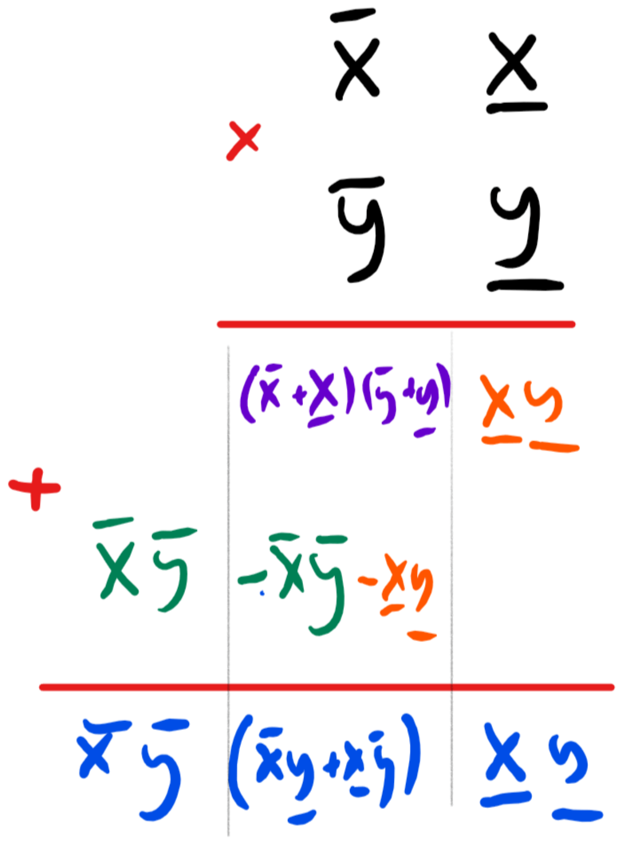
Karatsuba's multiplication

$$\begin{array}{r|c|c}
 & \bar{x} & \underline{x} \\
\times & \bar{y} & \underline{y} \\
\hline
 & (\bar{x}+\underline{x})(\bar{y}+\underline{y}) & \underline{x}\underline{y} \\
+\ \bar{x}\bar{y} & -\bar{x}\bar{y}-\underline{x}\underline{y} & \\
\hline
\bar{x}\bar{y} & (\bar{x}\underline{y}+\underline{x}\bar{y}) & \underline{x}\underline{y}
\end{array}$$

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)


$\bar{x}$ $\underline{x}$
$\times$ $\bar{y}$ $\underline{y}$
$(\bar{x}+\underline{x})(\bar{y}+\underline{y})$ $\underline{x}\underline{y}$
$+$ $\bar{x}\bar{y}$ $-\bar{x}\bar{y}-\underline{x}\underline{y}$
$\bar{x}\bar{y}$ $(\bar{x}\underline{y}+\underline{x}\bar{y})$ $\underline{x}\underline{y}$

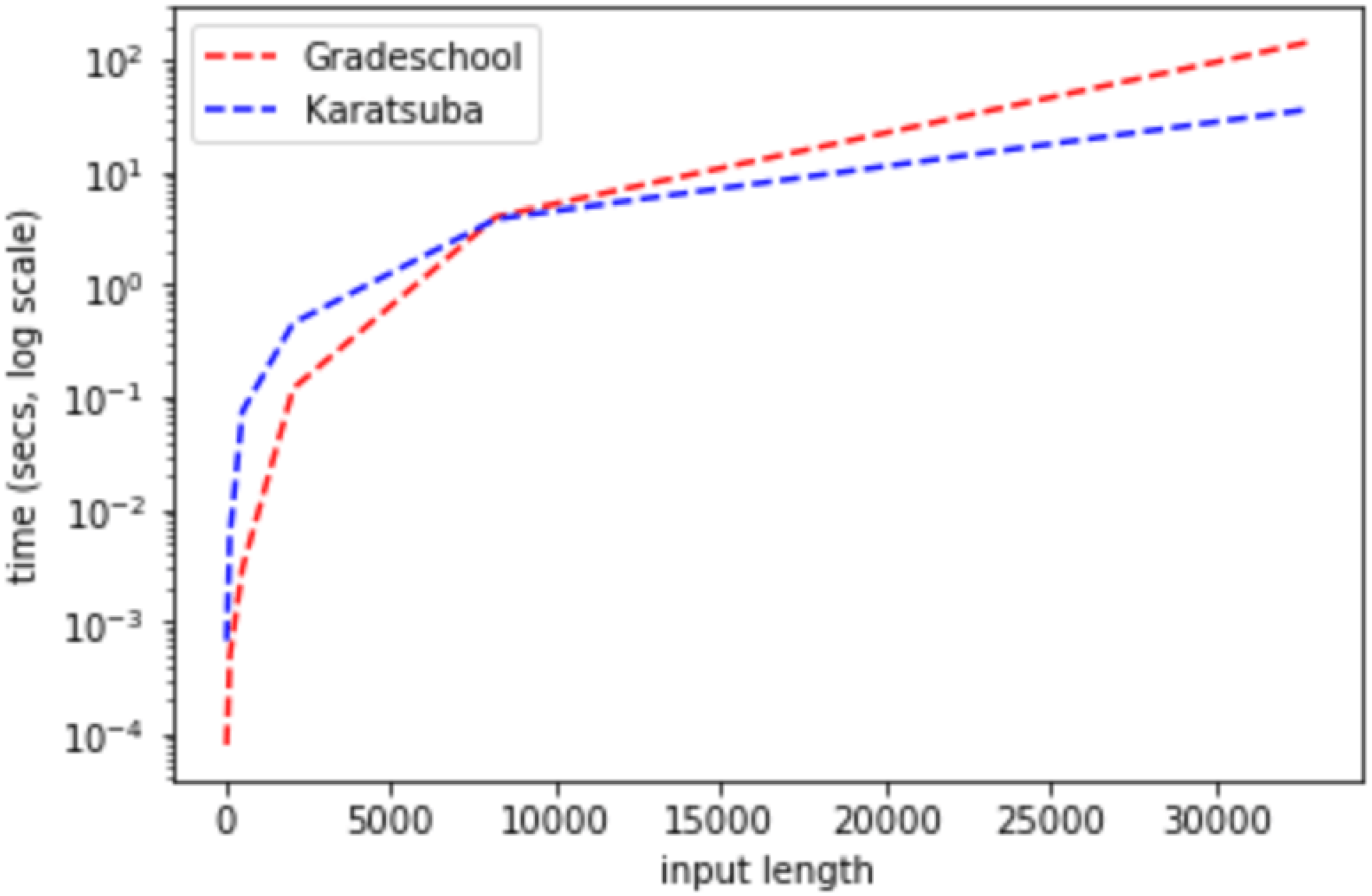
Analyzing Karatsuba's method

- Let $T(n)$ be the time it takes to multiply two n digit numbers:

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

  for a constant C depending on how we do addition

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


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

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 $\boldsymbol{\sim n \log n}$ where n is the bit length of x (this way of computing is known as fast Fourier Transform)

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

It is a major open question!

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

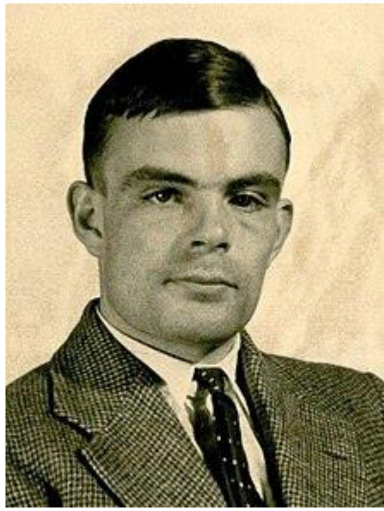
- Software verification
  (eg, never writes into a dangerous memory location)

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?

Roadmap for this course

1. Start from the simple (but not too limiting) circuits model
2. Finite automata and regular expressions
3. Free unlimited accessible memory (Turing/RAM machines)



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, 12:30 – 1:45pm in Rice 130
- https://weikailin.github.io/cs3120-toc
- We have implicitly talked about the objectives and highlights
- Best way to learn:
  participate in classes, ideally ask questions

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)
- Other resources might be posted too, occasionally.

- The slides will be posted on the course's page
- The (zoom) video recording of the classes accessible through Canvas.
  The Zoom link is NOT for attending remotely.

Teaching Team

- 5 amazing TA!

Haolin

Alice

Liran

Ethan

Nicole

- Office hours is on course Google calendar (to schedule rooms)

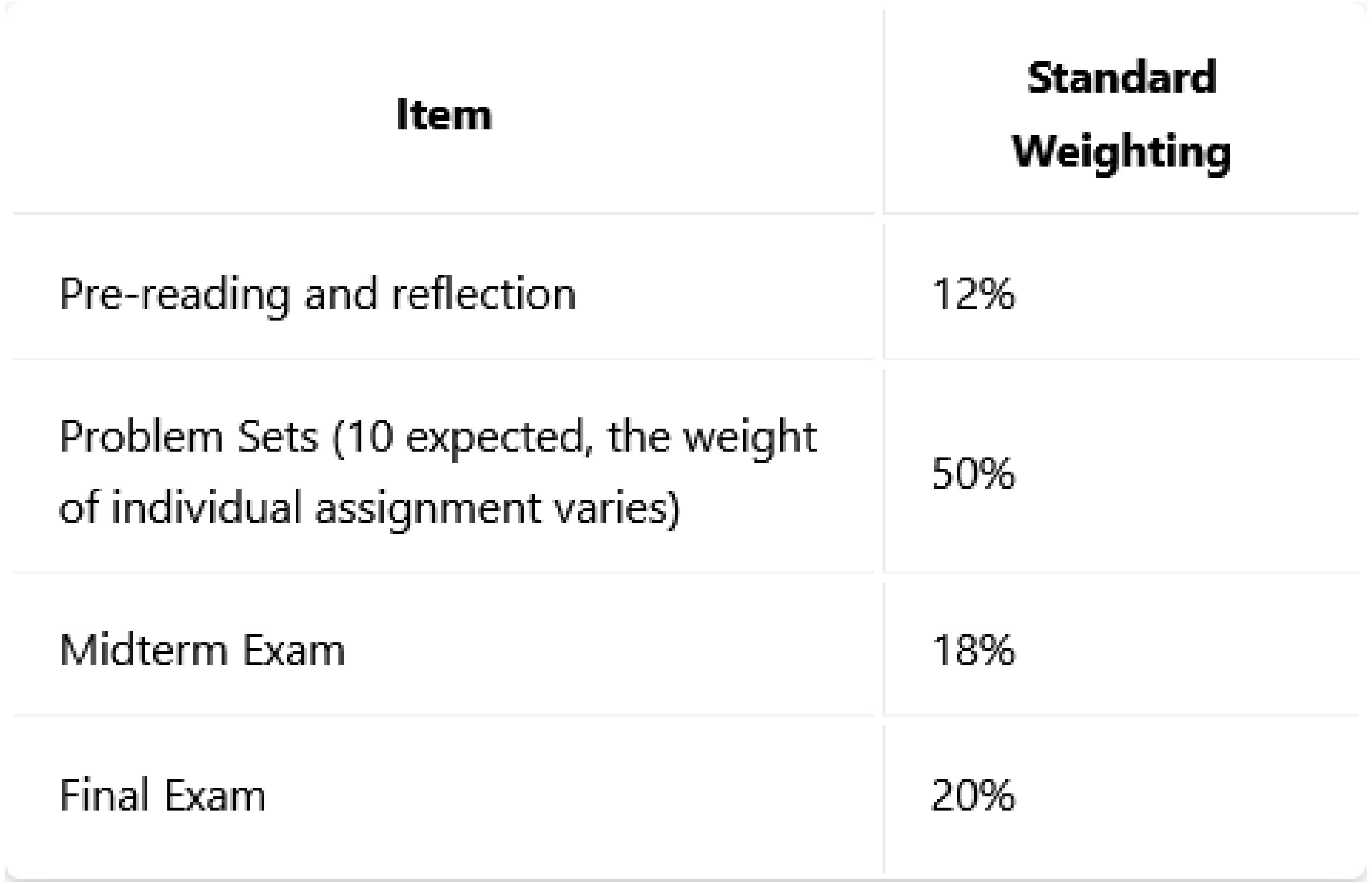
Communication

- **Calendar:** We will keep course deadlines, office hours, and other events on a public **google calendar**. You are expected to subscribe to this calendar. (survey)

- **Ed Education:** We will use the website for most other course communications. We expect you to receive messages we send to the "General" channel as well as any direct messages we send to you.
  Link to join: https://edstem.org/us/join/3DvewM (survey)

Assignments and Exams

- **Problem Sets** are due most weeks in the course (typically on Mondays at 10:00pm). We expect students to read and follow these carefully.
  You will be given Overleaf templates and will submit your pdfs.
- **Pre-reading and reflection.** You are asked to pre-read and reflect on course material, and you are asked to answer simple questions, almost weekly. You shall submit at least 12 times.
- **Exams.** We will have two exams in the course:
  Midterm: In class on Thursday, 6 March 2023.
  Final: Thursday, 2 May, 2:00pm - 5:00pm.



Item	Standard Weighting
Pre-reading and reflection	12%
Problem Sets (10 expected, the weight of individual assignment varies)	50%
Midterm Exam	18%
Final Exam	20%

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. (survey)

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?

- Reflection:
  Registration Survey, due tomorrow (Wednesday) 10pm:
  https://forms.office.com/r/pPBLzW1TSD