Module I: The Mathematical Foundation of Computation, Lecture I.a

Chi-Ning Chou @ 2022 January Mini-Course "What is Computation? From Turing Machines to Black Holes and Neurons"

Departure: Reasoning about Computation via Mathematics

Module I: The Mathematical Foundation of Computation

"If we understand an idea then it is only by mathematically recreating it."

– Misha Gromov

Chi-Ning Chou @ 2022 January Mini-Course "What is Computation? From Turing Machines to Black Holes and Neurons"

Abstraction

Mathematics

&

Computation

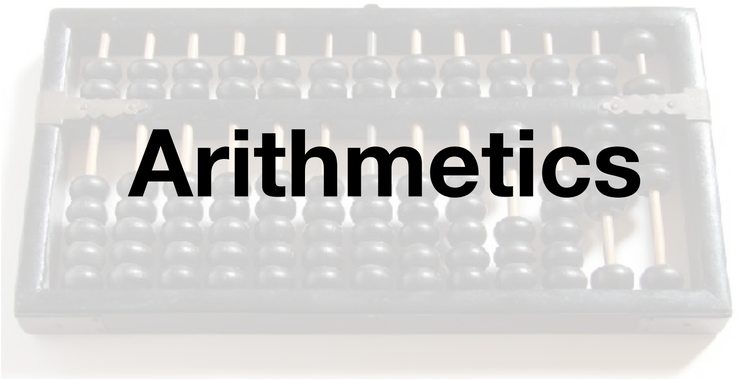
Tools

Why Math?

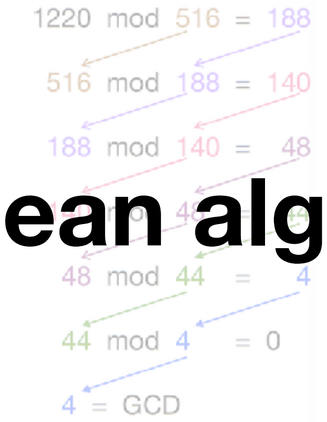
I encourage you to follow your instinct and feel the underlying story like watching an art movie!

Computations in Mathematics

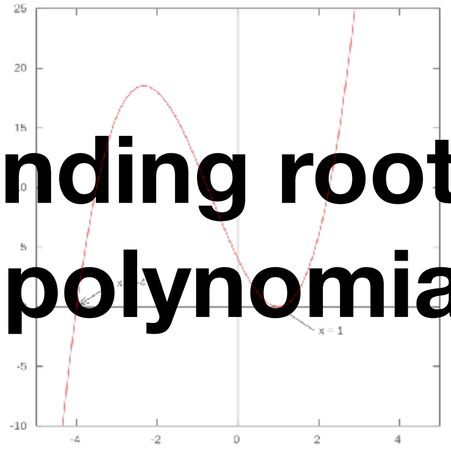
Computation → Mathematics



Arithmetics



Euclidean algorithm



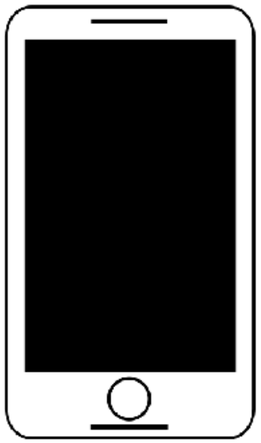
Finding roots of polynomials

$$\frac{d}{dx}(\sin x) = \cos x$$
$$\frac{d}{dx}(\cos x) = -\sin x$$
$$\frac{d}{dx}(\tan x) = \sec^2 x$$
$$\frac{d}{dx}(\sec x) = \sec x \tan x$$
$$\frac{d}{dx}(\cot x) = -\csc^2 x$$

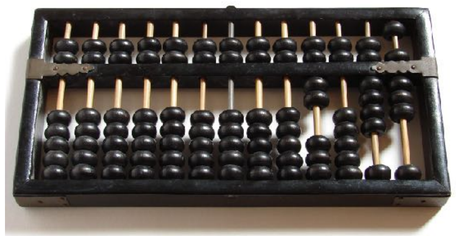
Calculus

Studying Computation Using Mathematics?

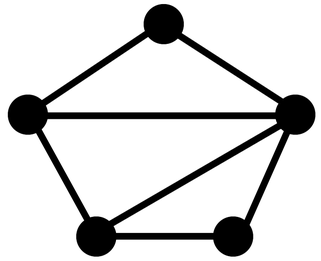
Mathematics → Computation

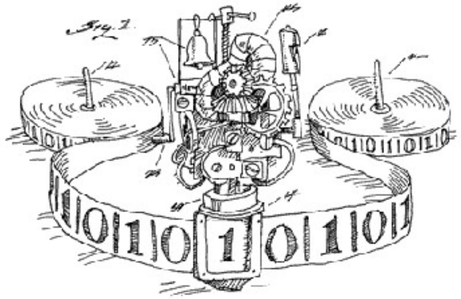


$$\frac{d}{dx}(\sin x) = \cos x$$
$$\frac{d}{dx}(\cos x) = -\sin x$$
$$\frac{d}{dx}(\tan x) = \sec^2 x$$
$$\frac{d}{dx}(\csc x) = -\csc x \cot x$$
$$\frac{d}{dx}(\sec x) = \sec x \tan x$$
$$\frac{d}{dx}(\cot x) = -\csc^2 x$$



Abstraction





$$f : \{0,1\}^* \rightarrow \{0,1\}^*$$

Mathematically Model Computational Problems?

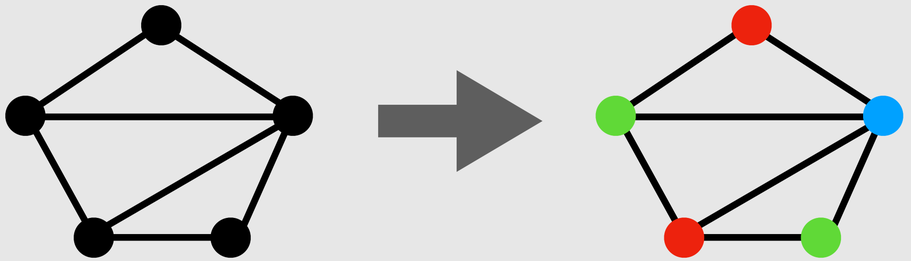
Decision Problem
(e.g., prime number)

Is 47 a prime?

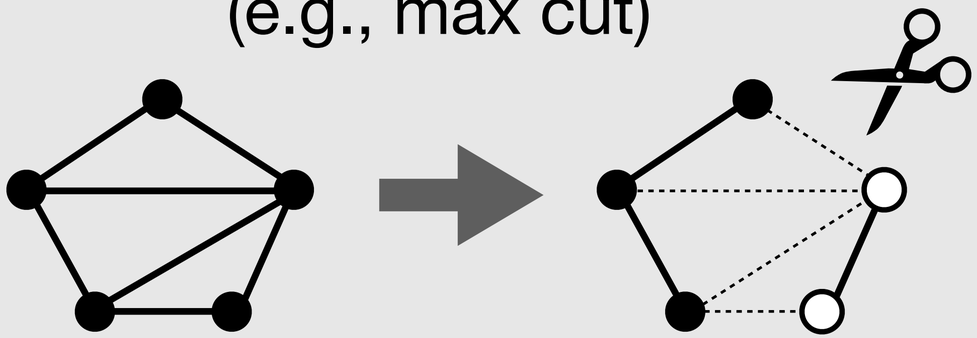
Counting Problem
(e.g., #roots of a polynomial)

How many roots does $x^2 - 7x + \frac{49}{4}$ have?

Search Problem
(e.g., graph coloring)



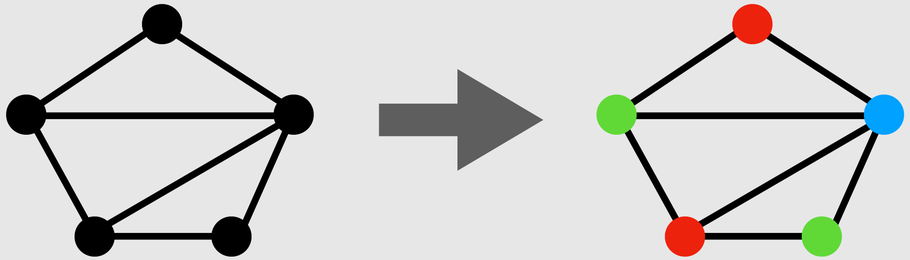
Optimization Problem
(e.g., max cut)



* There are more types of computational problems (e.g., sampling problem) but I skip here for simplicity.

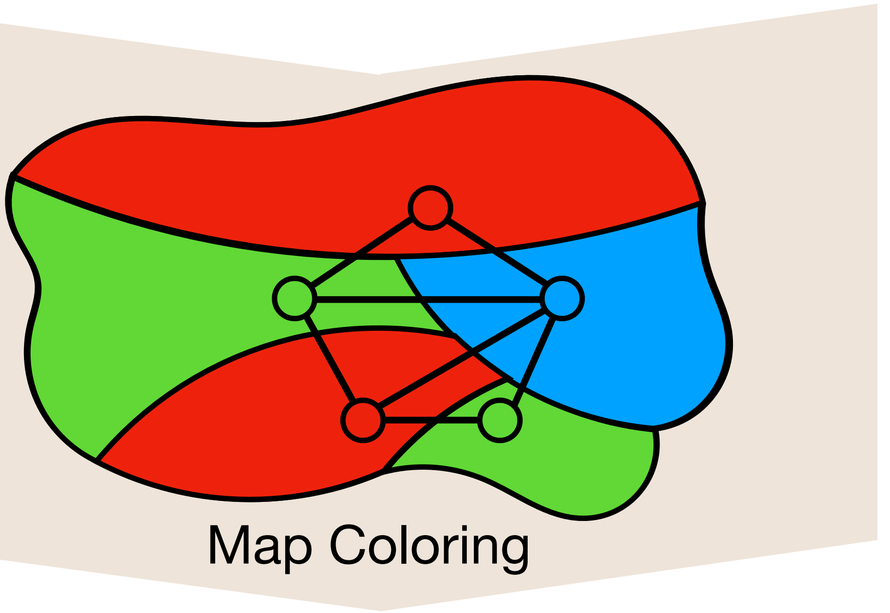
Okay… These Seem to be Quite Abstract…!?

Search Problem
(e.g., graph coloring)



Prahlad
(Jan. 12 9am-10am ET)

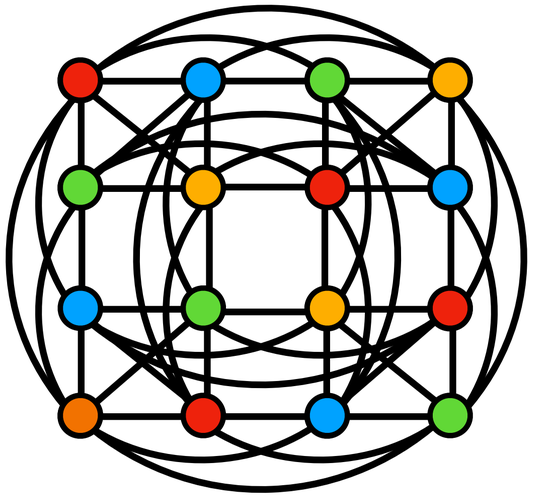
“The Four Color Theorem”



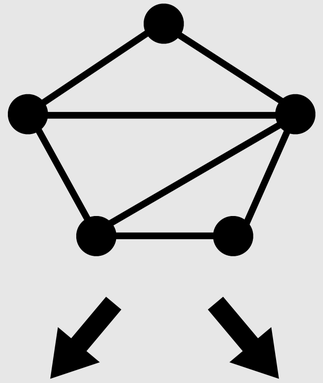
Map Coloring

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

Sudoku



Math as a Language for Studying Computation



100101010111 827492939441

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

Natural number, i.e., 0,1,2,3,…

Equivalent!

47

10111 47

$$f : \{0,1\}^* \rightarrow \{0,1\}^*$$

Boolean strings of finite length

Use $\mathbb{N}$ or $\{0,1\}^*$ to encode input and output!

Mathematical Abstraction for Computation

David Hilbert 1862-1943

Entscheidungsproblem

Is there a “mechanical procedure” s.t. given a “logical expression” as input, output “Yes” or “No” according to its universal validity?

Q: How to model “mechanical procedure”?

Modeling Mechanical Procedure

Which now has a fancy modern name: "algorithm"

Attempt 1:

(Logical systems)

$$\frac{\begin{array}{c} P_1 \to P_2 \\ P_2 \to P_3 \\ P_3 \to P_4 \\ \vdots \\ P_{99} \to P_{100} \end{array}}{P_1 \to P_{100}}$$

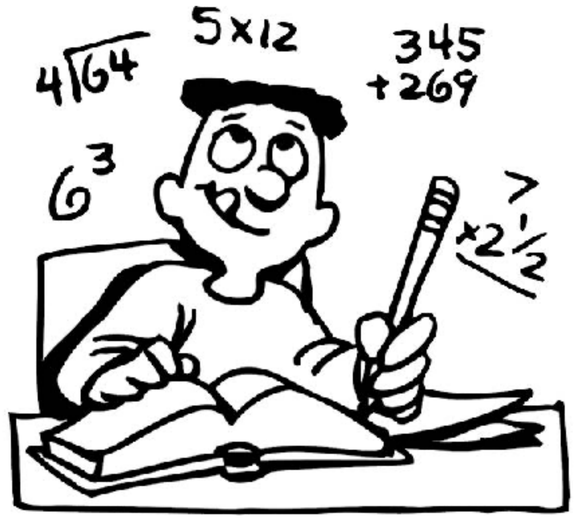
Examples:

- Propositional logic
- First-order logic
- Mathematical induction
- More…

Human goes beyond logical systems?

Attempt 2:

(Mathematicians)

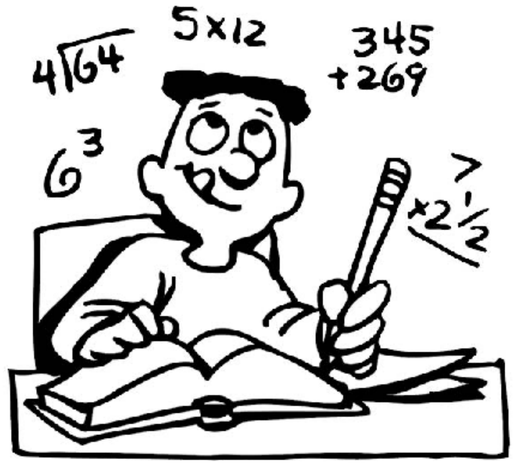


Turing machine!

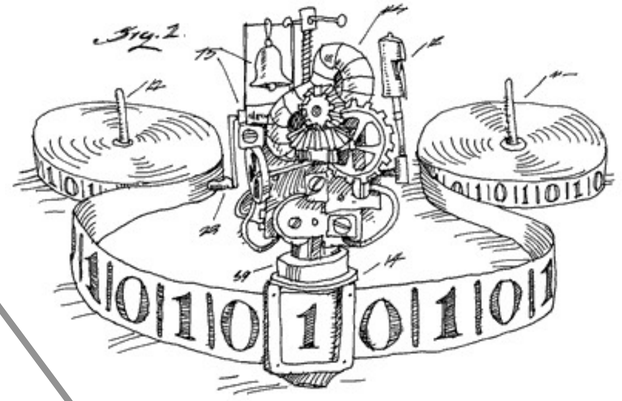
* More mathematical details in advanced sections.

Turing Machine

How to model mathematician proving theorems?



Mathematician	Turing machine
Many papers	An infinite tape
Read and write on papers	Read and write on the tape
State of the mind	State of the machine
Finite brain	Finite transition rules

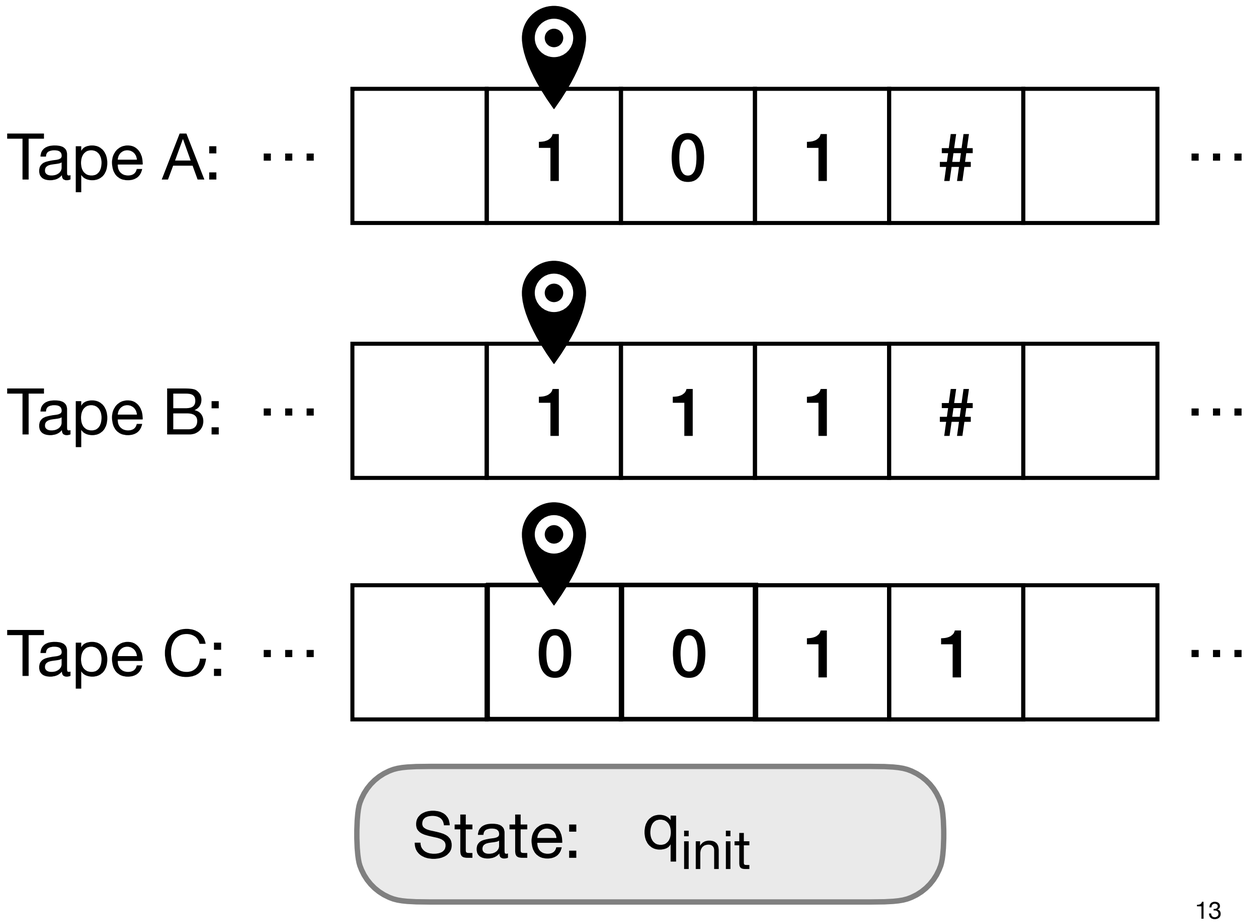


states & # symbols are finite!

* More interpretations in later slides and more mathematical details in advanced sections.

Example: A Turing Machine for Addition

Input: 101+111(in binary form, reverse direction on the tapes)



State	Read	Write	Move & Next State
q_{init}	(A, 0)		(A, R); q_0
q_{init}	(A, 1)		(A, R); q_1
q_{init}	(A, #)		q_{skip}
q_{carry}	(A, 0)		(A, R); q_1
q_{carry}	(A, 1)		(A, R); q_2
q_{carry}	(A, #)		$q_{skip+carry}$
q_0	(B, 0)	(C, 0)	(B, R); (C, R); q_{init}
q_0	(B, 1)	(C, 1)	(B, R); (C; R); q_{init}
q_0	(B, #)	(C, 0)	(C, R); q_{init}
q_1	(B, 0)	(C, 1)	(B, R); (C, R); q_{init}
q_1	(B, 1)	(C, 0)	(B, R); (C, R); q_{carry}
q_1	(B, #)	(C, 1)	(C, R); q_{init}
q_2	(B, 0)	(C, 0)	(B, R); (C, R); q_{carry}
q_2	(B, 1)	(C, 1)	(B, R); (C, R); q_{carry}
q_2	(B, #)	(C, 0)	(C, R); q_{carry}
q_{skip}	(B, 0)	(C, 0)	(B, R); (C, R); q_0
q_{skip}	(B, 1)	(C, 1)	(B, R); (C, R); q_0
q_{skip}	(B, #)		q_{end}
$q_{skip+carry}$	(B, 0)	(C, 1)	(B, R); (C, R); q_0
$q_{skip+carry}$	(B, 1)	(C, 0)	(B, R); (C, R); q_1
$q_{skip+carry}$	(B, #)	(C, 1)	q_{end}

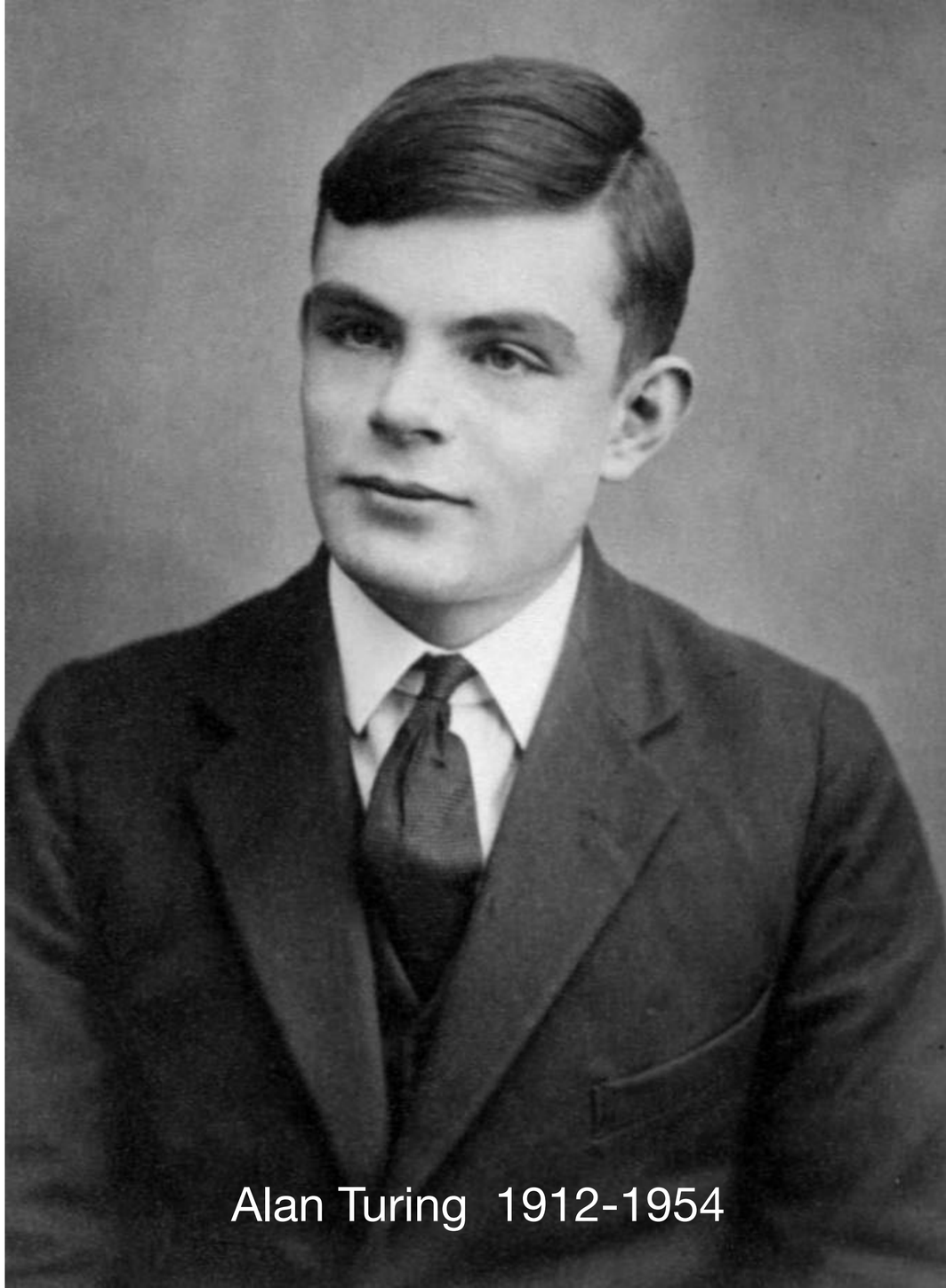
This should be the last time you see how a Turing machine work…

Feel comfortable to think of Turing machine as
your favorite programming language!

In summary, Turing machine…

Generalizes the notion of logical systems &

Aims to capture all possible mechanical procedures.


Alan Turing 1912-1954

* More on the significances of Turing machine in later slides.

Is Turing Machine Powerful Enough?

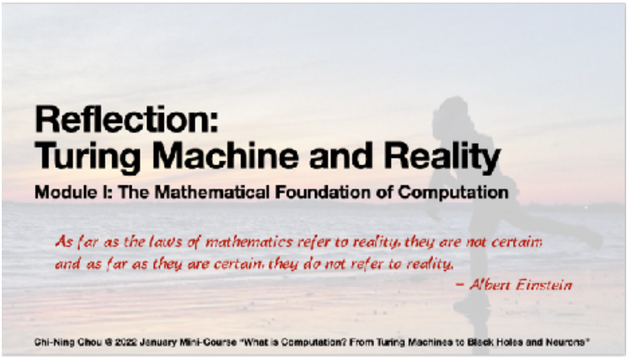
Entscheidungsproblem

Is there a “mechanical procedure” s.t. given a “logical expression” as input, output “Yes” or “No” according to its universal validity?

- Mathematically, Turing machine is equivalent to various computational models, e.g., λ-calculus.
- Philosophically/Physically,

Church-Turing Thesis

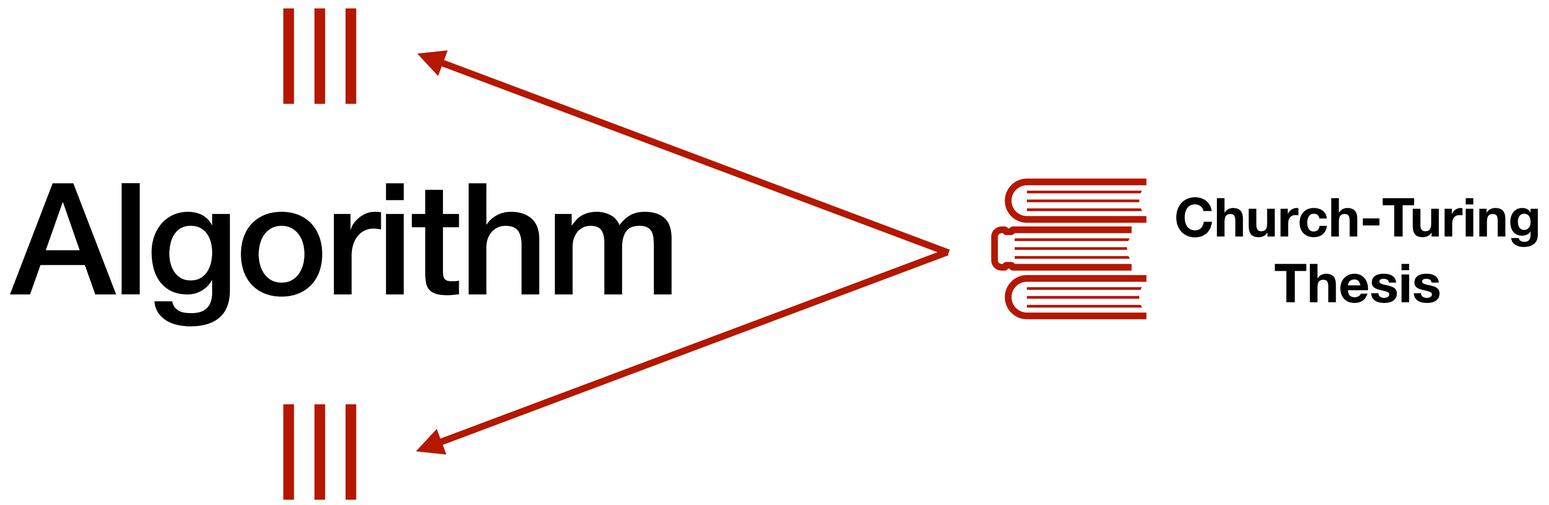
All realizable computation in the physical world can be done by a Turing machine.



Lecture I.c

(Jan. 17 10am-11am ET)

Turing Machine



Mechanical Procedure

Answer to Entscheidungsproblem…

A Negative Answer to Entscheidungsproblem

Gödel's Incompleteness Theorems [Gödel 1931]

For every *powerful enough* logical system $\mathcal{S}$, there exists a logical claim $C(\mathcal{S})$ whose validity cannot be determined within $\mathcal{S}$.

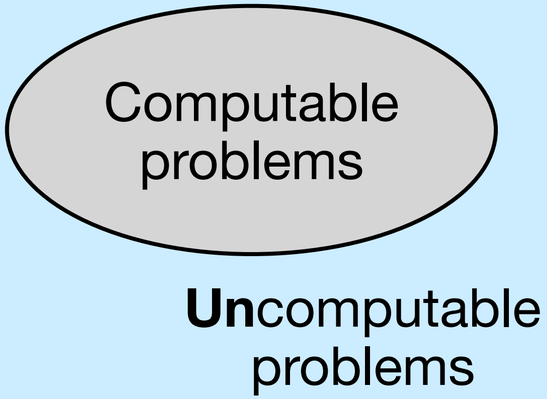
Uncomputability Theorems [Church 1936, Turing 1936]

There exists a computational problem that cannot be computed by any Turing machine. Specifically, the *Halting Problem* is uncomputable.

* Halting Problem will be defined in a moment and see more mathematical details in advanced sections.

What Do these Theorems Teach Us?

- There exists uncomputable problem.
- In fact, most problems are uncomputable.
- Once you can formalize your computational model, then it has a limit.
- Philosophically, …



"Our knowledge can only be finite, while our ignorance must necessarily be infinite."

– Karl Popper

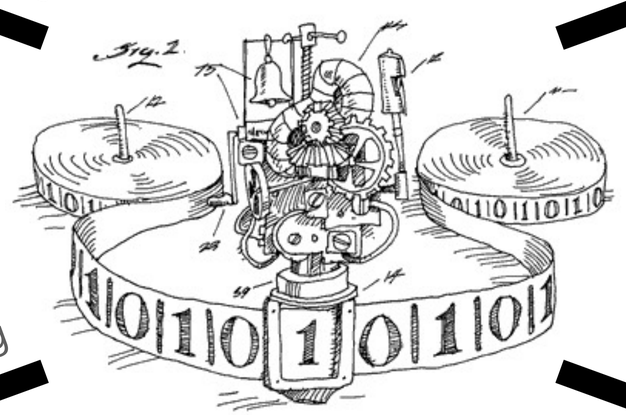
The End? Or the Beginning?

Theory of Computing

Mathematics

Philosophy

Validity of Church-Turing Thesis?



Engineering

Others

Digital Computers

Computational Thinking in Other Fields

A Glimpse into the Uncomputability Theorems

For those with less math background, please view the next few minutes as watching an art movie :)

Key Ingredients in the Proof

Uncomputability Theorems [Church 1936, Turing 1936]

There exists a computational problem that cannot be computed by any Turing machine. Specifically, the *Halting Problem* is uncomputable.

Halting Problem:

- Input: The description of a Turing machine  and an input  to it.
- Output: 1 if the state of  on input  becomes q_{end} at some point.

* Note that a Turing machine might not halt!

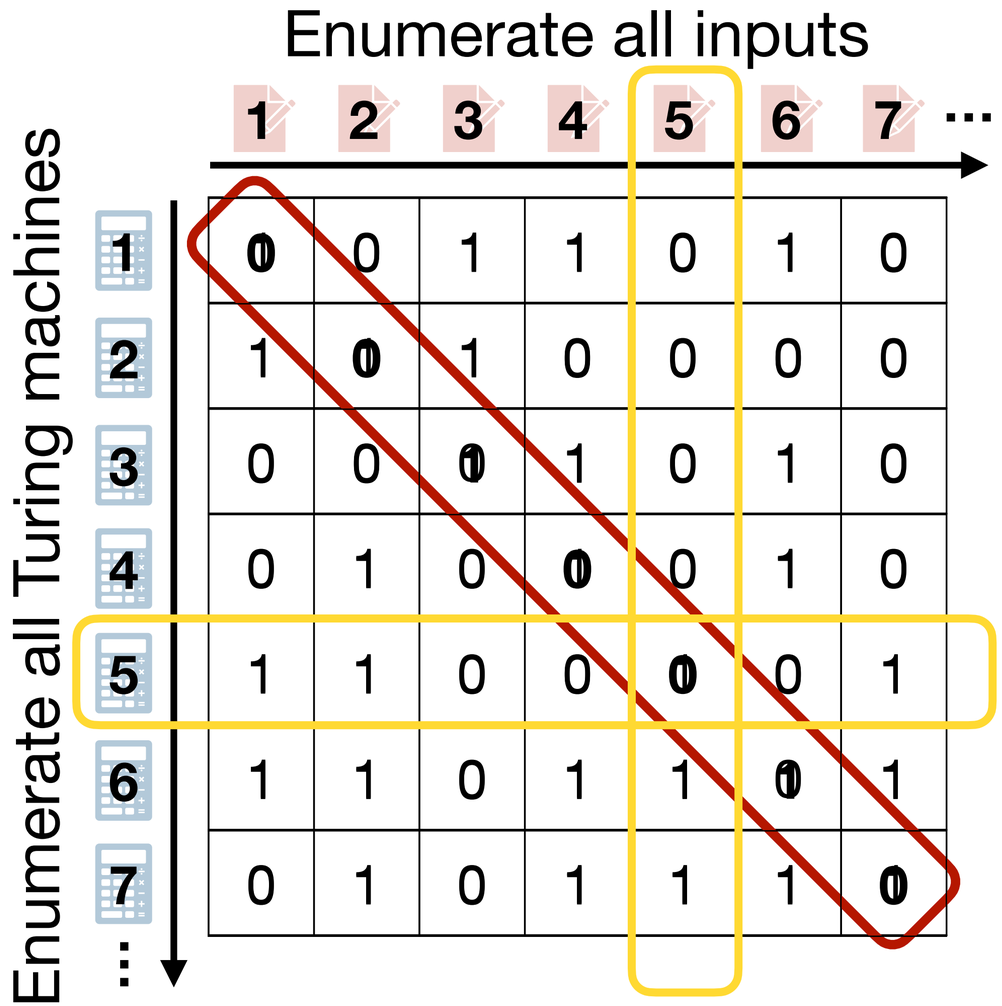
Example:

```
int main() {
    while(1);
}
```

Proof Sketch

A Turing machine

An input



Assume: There's a Turing machine **H** for the Halting Problem.

Step 1: Enumerate all i and all j.

Step 2: Use **H** to compute i(j). If it does not halt, then set the output to be 0.

Step 3: Design a Turing **T** machine using Step 2 to flip the diagonal of the table.

Contradiction: Say the Turing machine **T** designed in Step 3 is 5 :

If 5(5)=1 ⇒ Step 2+3 gives 5(5) =0 →←

If 5(5)=0 ⇒ Step 2+3 gives 5(5) =1 →←

Summary of Uncomputability Theorems

It's okay if you were not completely following the proof!

Enumerating all Turing machines

The description of an algorithm is finite => enumeration is possible!

Universal Turing Machine

A special Turing machine that can simulate all algorithms!

"Diagonalization"

An elegant mathematical idea for "proof by contradiction".

Rice Theorem

We say f is semantic if $f($  $) = f($  $)$ whenever  $($  $) =$  $($  $)$ for all  .

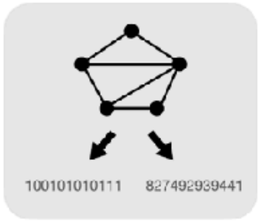
Then, every semantic and non-constant function f is uncomputable.

Summary

In mathematics, people model computation as something that can be done by a mechanical process.

Key Concepts

Math as a Language for Studying Computation



Mathematics to Computation

$$f : \mathbb{N} \to \mathbb{N}$$

Equivalent!

Natural number, i.e., 0,1,2,3,...

$$f : \{0,1\}^* \to \{0,1\}^*$$

Boolean strings of finite length

Use $\mathbb{N}$ or $\{0,1\}^*$ to encode input and output!

Turing Machine

|||

Algorithm — Church-Turing Thesis

|||

Mechanical Procedure

A Negative Answer to Entscheidungsproblem

Gödel's Incompleteness Theorems [Gödel 1931]

For every *powerful enough* logical system $\mathcal{S}$, there exists a logical claim $C(\mathcal{S})$ whose validity cannot be determined within $\mathcal{S}$.

Uncomputability Theorems [Church 1936, Turing 1936]

There exists a computational problem that cannot be computed by any Turing machine. Specifically, the *Halting Problem* is uncomputable.

The End? Or the Beginning?

Theory of Computing — Mathematics — Philosophy — **Validity of Church-Turing Thesis?**

Digital Computers — Engineering — Others — **Computational Thinking in Other Fields**

Food for Thought

Q: Do you think Turing machine capture the essence of computation?

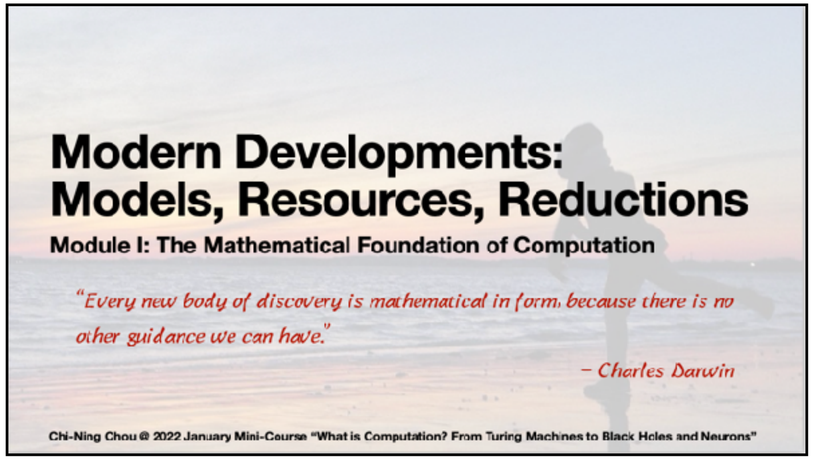
Q: If no, what aspect of computation do you think Turing machine does not capture?

Q: Turing machine seems very “dumb”, can other computational models solve some problems significantly faster than any Turing machine?

Exercise

- Try to explain one or few things you find interesting to your friends or families.
- If you find some topics that particularly interest you, take a glimpse into some references and explore!
- Think about is there anything that doesn’t really make sense to you, if so try to formulate it and ask me during Coffee chat or through email!

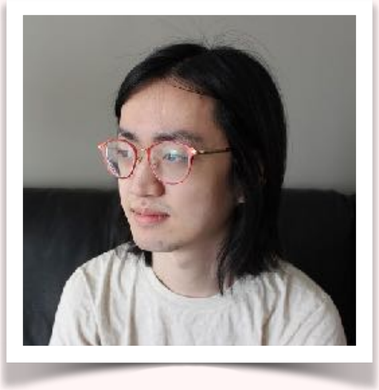
Next



Lecture I.b
(Jan. 11 10am-11am ET)

Today at 2pm ET!

"Efficiency of Algorithms and the Sorites Paradox"



Lijie Chen
(Jan. 11 11am-12pm ET)

Reijo
(Jan. 11
2pm-3pm ET)

"Undecidability of the Halting Problem and Gödel's incompleteness Theorem"

Prahlad
(Jan. 12
9am-10am ET)

"The Four Color Theorem"

References

Articles:

- Horswill, Ian. “What is computation?”, 2008, link.
- Wigderson, Avi. “The Godel Phenomenon in Mathematics: A Modern View.” Kurt Gödel and the Foundations of Mathematics (2011): 475, link.

Introductory Books:

- Moore, Cristopher, and Stephan Mertens. The nature of computation. OUP Oxford, 2011, link.
- Barak, Boaz. Introduction to Theoretical Computer Science. Online, link.
- Rautenberg, Wolfgang. A concise introduction to mathematical logic. New York, NY: Springer, 2006, link.

Advanced Books:

- Wigderson, Avi. Mathematics and computation. Princeton University Press, 2019, link.
- Arora, Sanjeev, and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009, link.

Fun reads:

- Aaronson, Scott. “Why philosophers should care about computational complexity.” Computability: Turing, Gödel, Church, and Beyond 261 (2013): 327, link.