Lecture 2

Recap: Cardinality of Infinite Sets

- We proved the following are countably infinite: $\{0,1\}^*, \mathbb{Z}, \mathbb{Q}, \mathbb{N}^2, \mathbb{N}^3, \cdots$

⇒ cardinality of these = $|\mathbb{N}|$

- We concluded that a set B is countably infinite if you can enumerate the elements of $B = \{x_1, x_2, x_3, \cdots\}$.

- The above is equivalent to finding a one-one function $f : B \to \mathbb{N}$.

The ordering of the elements is determined by f.

This Lecture

1. Proof of $|\mathbb{R}| > |\mathbb{N}|$
2. Computational Problems
    a. Simplifying Restrictions
    b. Computational Problems as languages and strings.
    c. Cardinality of languages.
3. Our first impossibility result.

This Lecture

1. Proof of $|\mathbb{R}| > |\mathbb{N}|$
2. Computational Problems
   a. Simplifying Restrictions
   b. Computational Problems as languages and strings.
   c. Cardinality of languages.
3. Our first impossibility result.

Definitions

- **Definition:** A set A is uncountable if it is not countable (i.e. $|A| > |\mathbb{N}|$).

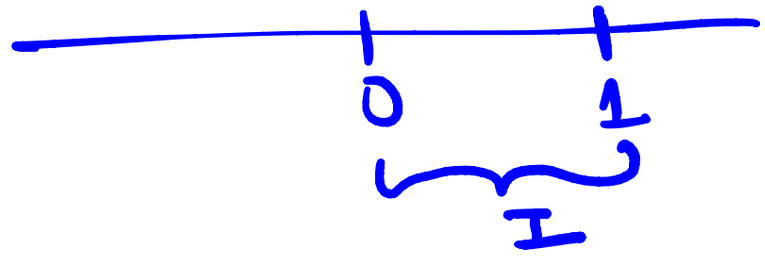
- **Definition:** A set A is uncountable if there is no one-one function from A to $\mathbb{N}$.

Claim: $|\mathbb{R}| > |\mathbb{N}|$

(1) We would like to show that $\mathbb{R}$ is uncountable.

(2) Equivalently, there is no one-one function $f: \mathbb{R} \to \mathbb{N}$.

Claim: $|\mathbb{R}| > |\mathbb{N}|$



To show $|\mathbb{R}| > |\mathbb{N}|$, it is sufficient to show $|I| > |\mathbb{N}|$, where $I = \{x \in \mathbb{R} \mid 0 < x < 1\}$.

From this claim, we have, as $I \subseteq \mathbb{R}$, and $|I| > |\mathbb{N}|$
$\Rightarrow |\mathbb{R}| > |\mathbb{N}|$ as well.

Proof: (By contradiction)

Suppose I is countable, then, there is a one-one function $f: I \longrightarrow \mathbb{N}$. and equivalently, we can enumerate all elements of $I = \{x_1, x_2, x_3, x_4, \ldots\}$

Claim: $|\mathbb{R}| > |\mathbb{N}|$

Let us work out the x_i's in their decimal expansion

$x_1 = 0.\boxed{x_{11}}\, x_{12}\, x_{13}\, x_{14} \cdots$

$x_2 = 0.\, x_{21}\, \boxed{x_{22}}\, x_{23}\, x_{24} \cdots$

$x_3 = 0.\, x_{31}\, x_{32}\, \boxed{x_{33}}\, x_{34} \cdots$

$x_4 = 0.\, x_{41}\, x_{42}\, x_{43}\, \boxed{x_{44}} \cdots$

$\vdots$

$x_k = 0.\, x_{k1}\, x_{k2}\, x_{k3}\, x_{k4} \cdots$

We will define $y \in I$ as follows:

$y = 0.y_1 y_2 y_3 \cdots$

$$y_i = \begin{cases} 3 \text{ if } x_{ii} \neq 3 \\ 4 \text{ if } x_{ii} = 3. \end{cases}$$

Claim: $|\mathbb{R}| > |\mathbb{N}|$

To show $|\mathbb{R}| > |\mathbb{N}|$, it is sufficient to show $|I| > |\mathbb{N}|$, where $I = \{x \in \mathbb{R} \mid 0 < x < 1\}$.

For example

$x_1 = 0.\textcircled{3}214\ldots$

$x_2 = 0.2\textcircled{1}234\ldots$

$x_2 = 0.15\textcircled{8}4\ldots$

$y = 0.433\ldots$

By construction,
$y \neq x_k$ for any $k \in \omega$

$y_k \neq x_{kk}$.

(y disagrees w/ x_k at the kth position in its decimal expansion)

Claim: $|\mathbb{R}| > |\mathbb{N}|$

As $y \neq x_k$ for any k

$\Rightarrow$ $y \notin \{x_1, x_2, x_3, \ldots\}$

However, by construction, $y \in I$.

(as $y = 0.y_1 y_2 y_3 \ldots$)!

$\Rightarrow I \neq \{x_1, x_2, x_3 \ldots\}$.

Claim: $|\mathbb{R}| > |\mathbb{N}|$

To show $|\mathbb{R}| > |\mathbb{N}|$, it is sufficient to show $|I| > |\mathbb{N}|$, where $I = \{x \in \mathbb{R} \,|\, 0 < x < 1\}$.

Key Questions in This Course

- What problems can you solve with a computer?
  - **Computability Theory**
- Why are some problems harder to solve than others?
  - **Complexity Theory**
- What are some tools we will use to formalize our answers to these questions?
  - **Discrete Math**

This Lecture

1. Proof of $|\mathbb{R}| > |\mathbb{N}|$
2. Computational Problems
   a. Simplifying Restrictions
   b. Computational Problems as languages and strings.
   c. Cardinality of languages.
3. Our first impossibility result.

Computational Problem

A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

Computational Problem

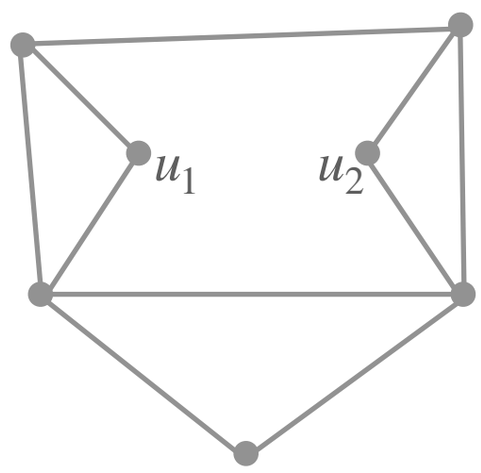
A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2

Computational Problem

A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

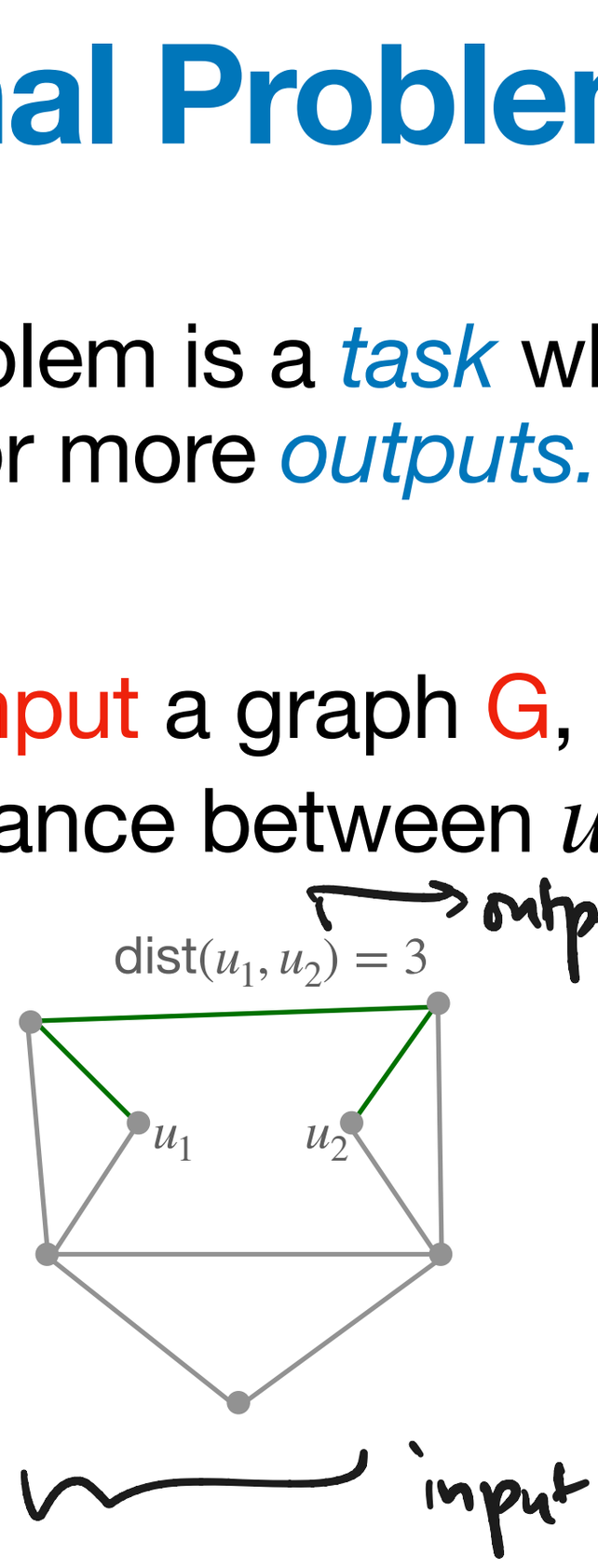
Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2



Computational Problem

A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

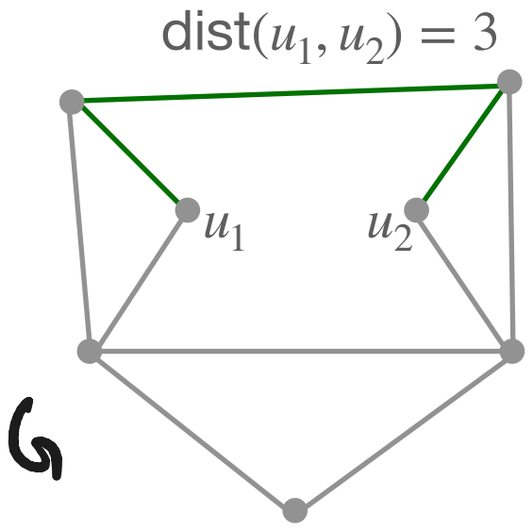
Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2.

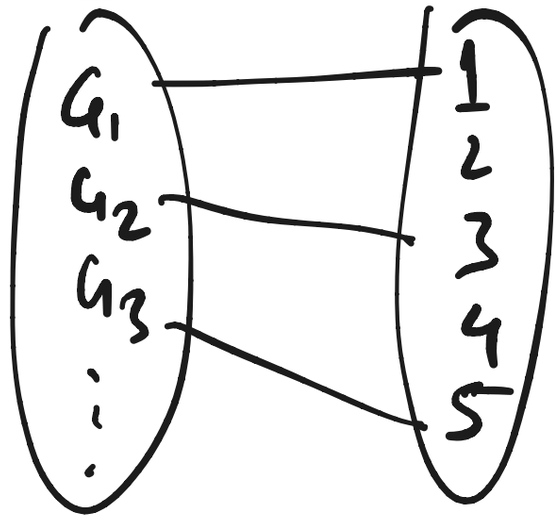


Computational Problem

A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2.





In other words, a computational problem is a relation from the set of valid inputs to the set of outputs.

Computational Problem

A computational problem is relation from the set of valid inputs to the set of outputs. Too general!

→ i.e. $\{0,1\}^*$ is the set of inputs.

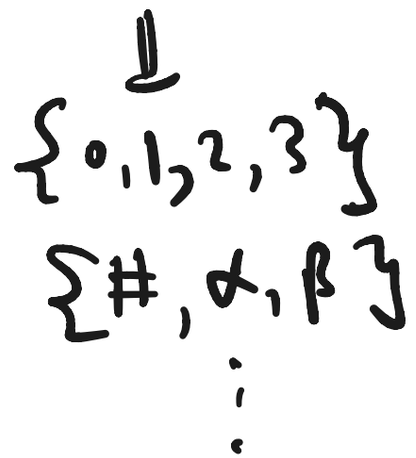
Simplifying Restriction 1: We represent inputs as binary strings.

Simplifying Restriction 2: We will consider only decision problems. That is the output is YES/NO.

Strings

- We use $\{0,1\}$ to denote the two elements of the binary alphabet.
- $\{0,1\}^n$ is denoted as the set of all binary strings of length n.
- The unique string in $\{0,1\}^0$ is the empty string, denoted ϵ.

  (common in many programming languages as “”)
- Let x be a binary string, then $|x|$ denotes the length of the string.
- We use $\{0,1\}^*$ to denote the collection of all strings of finite length. That is, $\{0,1\}^* = \bigcup_{n \in \mathbb{N}} \{0,1\}^n$.
- All the above definitions can be applied to any alphabet Σ. An alphabet is just a finite collection of symbols.



Back to Restriction 1

Simplifying Restriction 1: We represent inputs as binary strings.

Why is this reasonable?

Answer:

- We can encode any object with a finite description with a binary string.

Proposition: For every finite set $\mathcal{X}$ with k elements, there is a one-to-one encoding function $h : \mathcal{X} \to \{0,1\}^{\lceil \log k \rceil}$.

For example $H = \{1, 2, 3, \ldots, k\}$

Proof:

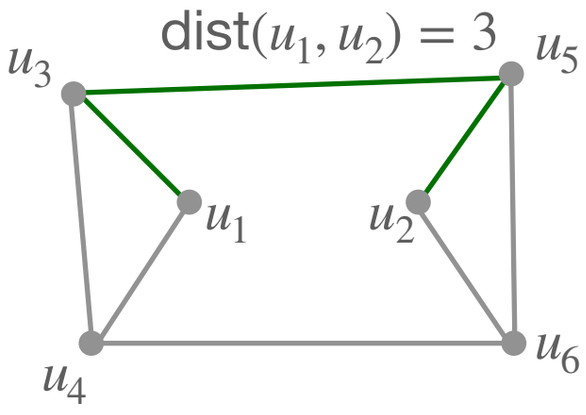
(a) Order elements of $H = \{a_1, a_2, a_3 \ldots a_k\}$

(b) Define $h: H \to \{0,1\}^{\lceil \log k \rceil}$ as follows:

$h(a_i) = \langle i \rangle$ ($\langle i \rangle$ refers to the binary representation of i.)

Restriction 1: Example

Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2.



Graph G

$M(u_i, u_j) = 1 \Leftrightarrow (u_i, u_j)$ is an edge in G.

	u_1	u_2	u_3	u_4	u_5	u_6
u_1	0	0	1	1	0	0
u_2	0	0	0	0	1	1
u_3	1	0	0	1	1	0
u_4	1	0	1	0	0	0
u_5	0	1	1	0	0	1
u_6	0	1	0	0	1	0

$= M$

(adjacency matrix of G)

The binary encoding of G denoted, $\langle G \rangle \in \{0,1\}^{n^2}$ obtained by unraveling the adjacency matrix. (read off the matrix row by row)

For any object B, we will denote its binary encoding by $\langle B \rangle$.

Restriction 2

Simplifying Restriction 2: We will consider only decision problems. That is the output is YES/NO.

Objection: There are many natural problems that interest us for which the desired output is not simply a Yes or No answer.

(Partial) Answer:

In many cases, even if the problem we are interested in is not a decision problem, we can formulate a single or multiple related decision problems that "capture the essence of" the problem.

Example: Given as input a graph G and vertices u_1, u_2, is there a path of length at most 3 between u_1, u_2?

Restriction 2

Consider the following non-decision problem:

"Given a graph G, and vertices u_1, u_2, output the distance b/w u_1, u_2 in G"

This is equivalent to solving the following decision problems: Here, n = # vertices in G.

(1) Is the distance between $u_1, u_2 \leq n$? ✓

(2) Is the distance between $u_1, u_2 \leq n-1$? ✓

(3) Is the distance b/w $u_1, u_2 \leq n-2$? ✓ $\Rightarrow d(u_1, u_2) = i+1$.

⋮

Is the distance b/w $u_1, u_2 \leq i+1$? ✓

Is the distance b/w $u_1, u_2 \leq i$? ✗

This Lecture

1. Proof of $|\mathbb{R}| > |\mathbb{N}|$
2. Computational Problems
    a. Simplifying Restrictions
    b. Computational Problems as languages and strings.
    c. Cardinality of languages.
2. Our first impossibility result.

Functions and Languages

- Given Restriction 1 and Restriction 2, a *decision problem* where all the inputs are binary strings of length n can be described with a *Boolean function:* $f : \{0,1\}^n \to \{0,1\}$.

$f(x) = 1 \iff$ The decision problem outputs YES on x.

Example: Given as input a graph $G = (V, E)$ with $|V| = n$ and vertices u_1, u_2, is there a path of length at most 3 between u_1, u_2?

Can be described as a Boolean function $f : \{0,1\}^{n^2+2\lceil \log n \rceil} \to \{0,1\}$.

The input to the decision problem is $\langle G \rangle, \langle u_1 \rangle, \langle u_2 \rangle$

As, $|\langle G \rangle| = n^2$, $|\langle u_1 \rangle| = \lceil \log n \rceil$ and $|\langle u_2 \rangle| = \lceil \log n \rceil$

Functions and Languages

- In this course, we will consider ways to solve a computational problems using a computational model.
- But a “computer code” doesn’t take a fixed length input!
- So, we represent decision problems where input can be of any length as:

- Family of Boolean Functions: $\cup_{n \geq 0} f_n$ where $f_n : \{0,1\}^n \rightarrow \{0,1\}$.
- As languages: a language $L \subseteq \{0,1\}^*$ is a set of all valid inputs on which we return YES.

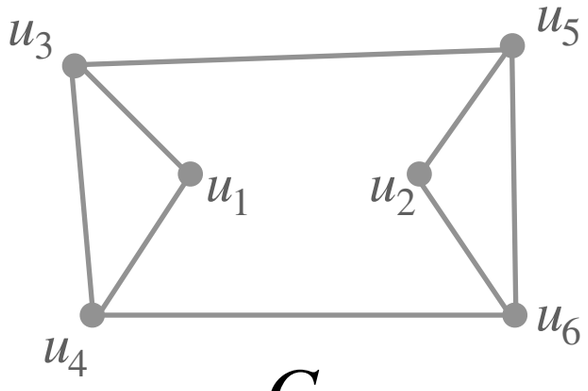
A decision problem D corresponds to the following language

$L_D = \{ x \in \{0,1\}^* \mid D \text{ outputs YES on } x \}$.

Functions and Languages

Example: Given a graph G, output whether or not it contains a cycle of length 3.

Corresponding Language: $L = \{\langle G \rangle \mid G \text{ has a cycle of length } 3\}$.

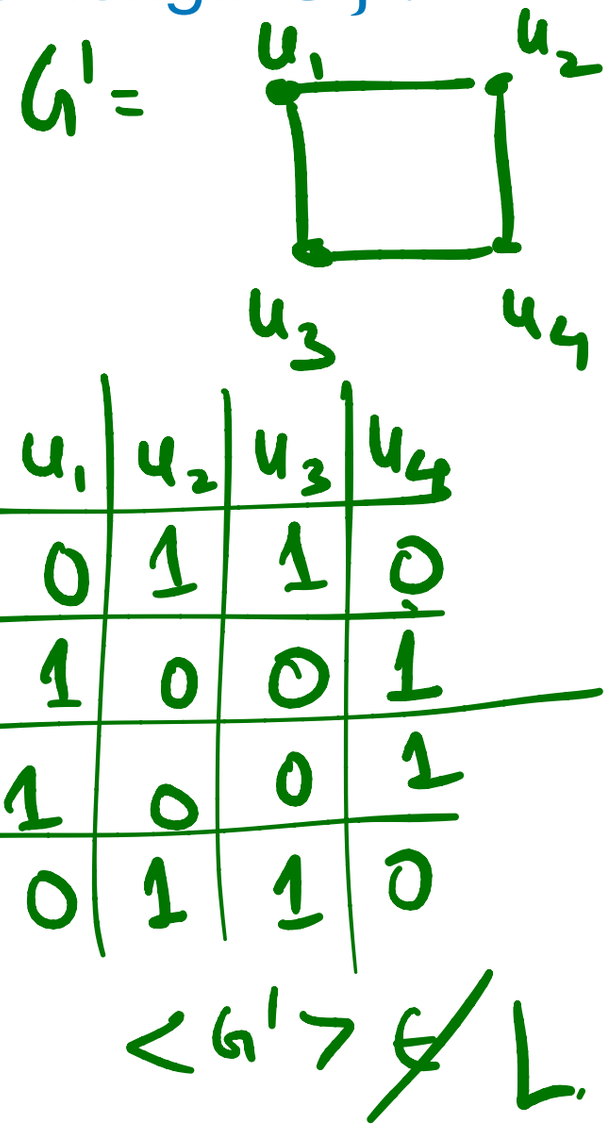


G

	u_1	u_2	u_3	u_4	u_5	u_6
u_1	0	0	1	1	0	0
u_2	0	0	0	0	1	1
u_3	1	0	0	1	1	0
u_4	1	0	1	0	0	0
u_5	0	1	1	0	0	1
u_6	0	1	0	0	1	0

$\langle G \rangle$

$\langle G \rangle \in L$



Functions and Languages

Equivalence of the two representations:

- Given a language L_D representing a decision problem D. Then, all strings of length n in L_D can be represented by a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that $f(x) = 1 \iff x \in L_D$.

- Similarly, any family of Boolean functions $\cup_{n \geq 0} f_n$ where, $f_n : \{0,1\}^n \rightarrow \{0,1\}$ corresponds to the language $L = \cup_{n \geq 0} f^{-1}(1)$.

This Lecture

1. Computational Problems
    a. Simplifying Restrictions
    b. Computational Problems as languages and strings.
    c. Cardinality of languages.
2. Our first impossibility result.

Review: Countability

- Claim 1: For a fixed $n \geq 0$, the set $\{0,1\}^n$ is finite.

$|\{0,1\}^n| = 2^n$.

- Claim 2: The set $\{0,1\}^*$ is countably infinite.

↓ showed in previous lecture:

$$\{0,1\}^* = \bigcup_{n \in \mathbb{N}} \{0,1\}^n.$$

Review: Countability

- Claim 1: For a fixed $n \geq 0$, the set $\{0,1\}^n$ is finite.
- Claim 2: The set $\{0,1\}^*$ is countably infinite.

Note: any language $L \subseteq \{0,1\}^*$, thus it is countable.

(As L is a subset of a countable set)

Cardinality of Languages

- **Claim:** The set of all languages L is uncountable.
- Proof by Diagonalization! (We will see more of this technique.)

(By contradiction)

$\mathbb{L} = \{ L \mid L \subseteq \{0,1\}^* \}$ (The set of all languages)

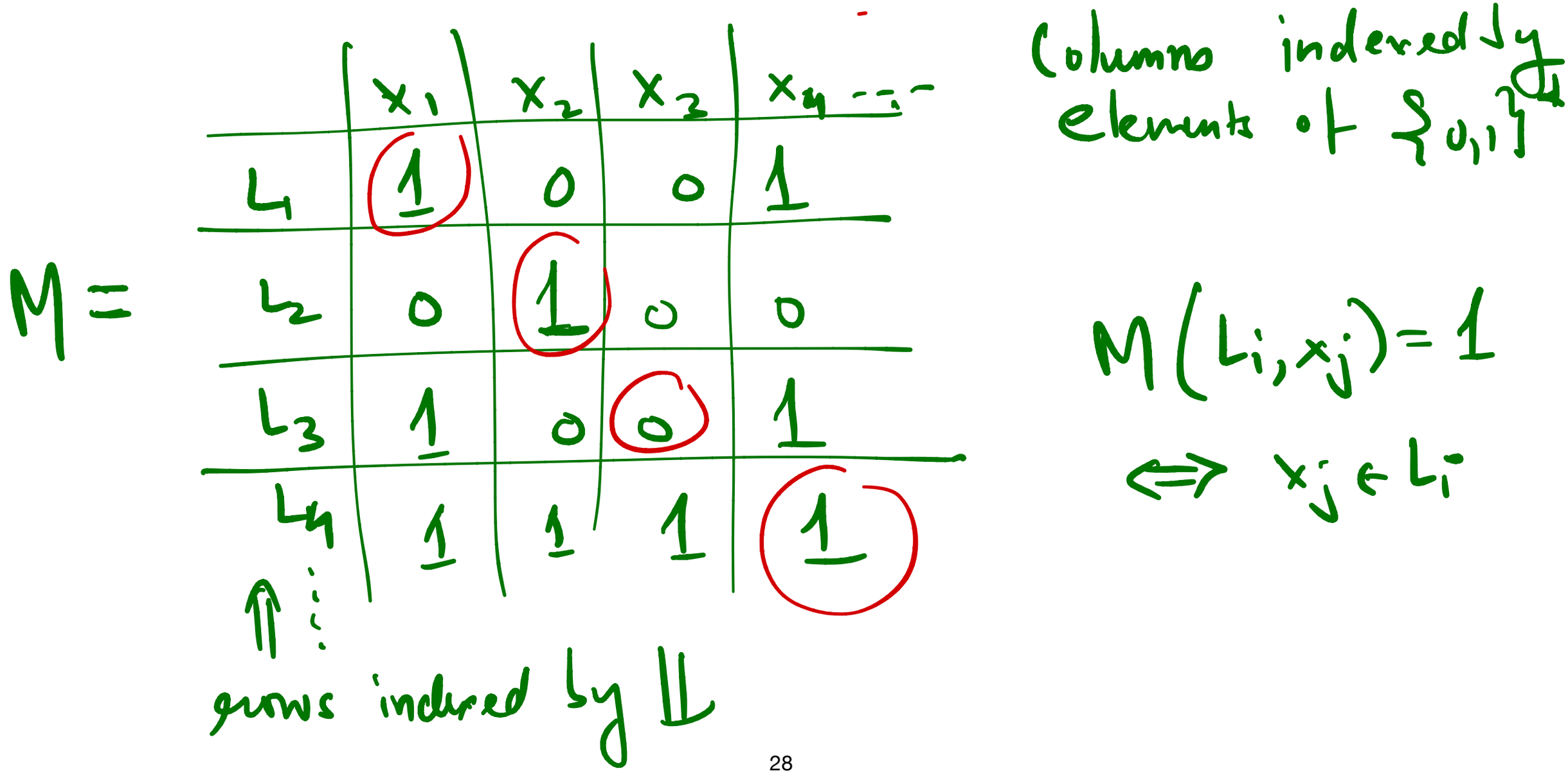
Assume $\mathbb{L}$ is countable $\Rightarrow \mathbb{L} = \{L_1, L_2, L_3 \ldots\}$

(That is elements of $\mathbb{L}$ can enumerated)

Recall $\{0,1\}^*$ countable $\Rightarrow \{0,1\}^* = \{x_1, x_2, x_3 \ldots\}$

Cardinality of Languages

- **Claim:** The set of all languages L is uncountable.
- Proof by Diagonalization! (We will see more of this technique.)



Cardinality of Languages

- **Claim:** The set of all languages L is uncountable.
- Proof by Diagonalization! (We will see more of this technique.)

Define L_{diag} as follows:

$x_i \in L_{diag} \iff x_i \notin L_i \quad \forall x_i \in \{0,1\}^*$

Note: $L_{diag} \subseteq \{0,1\}^* \Rightarrow L_{diag} \in \mathbb{L}$. (1)

But $L_{diag} \notin \{L_1, L_2, L_3, L_4, \ldots\}$ (2)

as L_{diag} "disagrees" w/ L_i at x_i

That is, $x_i \in L_{diag} \iff x_i \notin L_i$. So, $L_{diag} \neq L_i$

from (1) and (2), we have,

$\mathbb{L} \neq \{L_1, L_2, L_3, \ldots\}$

This Lecture

1. Computational Problems
   a. Simplifying Restrictions
   b. Computational Problems as languages and strings.
   c. Cardinality of languages.
2. Our first impossibility result.

Can all computational problems be solved?

- What does it mean for a computational problems to be solved?
- Let's say it means that we can write a python code to solve it.
- Alternatively, a language L can be solved if and only if there is a python code P_L such that $P_L(x) = 1 \iff x \in L$.

- **Claim:** There exists a language L for which there is no python code P such that $P(x) = 1 \iff x \in L$.

Can all computational problems be solved?

- **Claim:** There exists a language L for which there is no python code P such that $P(x) = 1 \iff x \in L$.

(1) A python code is a finite length code.

(2) A python code P therefore has a finite binary description $\langle P \rangle \in \{0,1\}^k$ for some k.

(3) Consider $\mathcal{P} = \{ \langle P \rangle \mid P \text{ is a python code} \}$

Note that $\mathcal{P} \subseteq \{0,1\}^*$

(So, $\mathcal{P}$ is countable) → (a)

Can all computational problems be solved?

- **Claim:** There exists a language L for which there is no python code P such that $P(x) = 1 \iff x \in L$.

On the other hand $\mathbb{L}$ is uncountable.
(This means there is no one-one function from $\mathbb{L}$ to $\mathbb{N}$.)

Assume the claim is true. That is for each $L \in \mathbb{L}$
$\exists$ a python code $P_L \in \mathcal{P}$ s.t. $P_L(x) = 1 \iff x \in L$.

Then, we have a one-one $f: \mathbb{L} \longrightarrow \mathcal{P}$
where $f(L) = \langle P_L \rangle$

↳ (b)

Can all computational problems be solved?

- **Claim:** There exists a language L for which there is no python code P such that $P(x) = 1 \iff x \in L$.

From (a) $\exists\ g : \mathcal{P} \to \mathbb{N}$ where one-one.

From f, g, we can conclude there exists

$h : \mathbb{L} \to \mathbb{N}$ s.t. h is one-one.

$\Rightarrow$ $\mathbb{L}$ is countable (contradiction)

Some Thoughts on the Proof

- There is nothing special about Python. You can replace it with a programming language or machine model of your choice.
- The key step: any algorithm has a finite description and we can therefore encode them as strings.
- Objection 1: The proof is non-constructive. What are the problems that cannot be solved by a python code? Are they interesting?
- Objection 2: The proof says: for every computational model there is a problem that cannot be solved by it. Is there a problem that cannot be solved by all computational models simultaneously?

Summary

1. **Computational Problems:** is a *task* which for each *valid input* produces one or more *outputs.*
2. **Two Simplifying Restrictions:**
   - Inputs are binary strings
   - Outputs are 0/1 or YES/NO.
3. **Decision Problems = Languages:**
   $L \subseteq \{0,1\}^*, x \in L \iff$ answer on x is YES.
4. **Cardinality of Languages:** The set $\{0,1\}^*$ is countable but, the set of all languages is uncountable.
5. **Our First Impossibility Result:** The set of all Python programs is countable but the set of all languages is uncountable. So, there are languages which cannot be solved by any Python program.