Lecture 2

Recap: Cardinality of Infinite Sets

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

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

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

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

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

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

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

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

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

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

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.
2. Our first impossibility result.

Computational Problem

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

Computational Problem

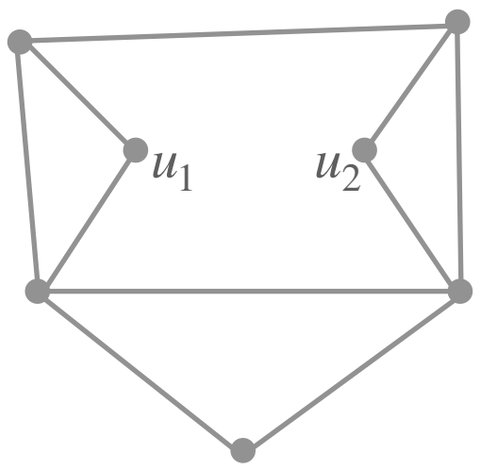
A computational problem is a *task* which for each *valid input* produces one or more *outputs.*

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.*

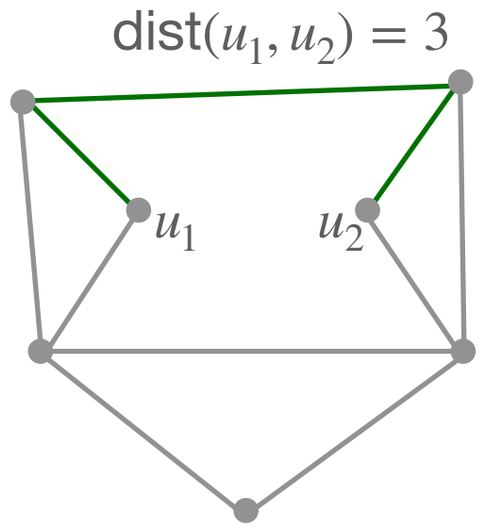
Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2



Computational Problem

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

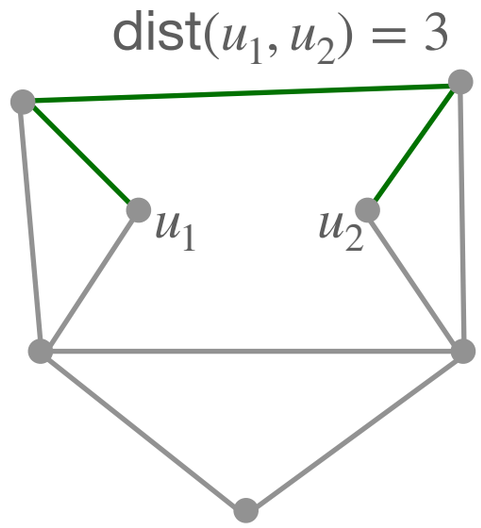
Example: Given as input a graph G, and pair of vertices u_1, u_2 output the distance between u_1, u_2.



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!

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?

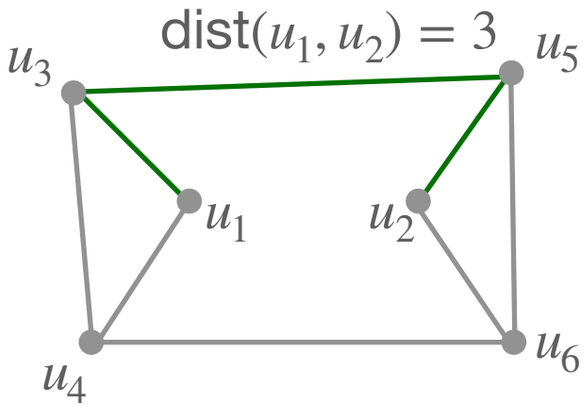
Answer:

- 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 $h : \mathcal{X} \to \{0,1\}^{\lceil \log k \rceil}$.

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

	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

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

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

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

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 $f : \{0,1\}^{n^2+2\lceil \log n \rceil} \to \{0,1\}$.

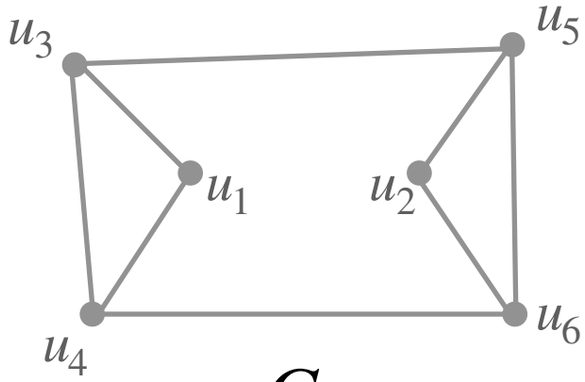
Functions and Languages

- 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.

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

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.

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.)

Cardinality of Languages

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

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 $P(x) = 1 \iff x \in L$.

Can all computational problems be solved?

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

Can all computational problems be solved?

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

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.