Time Complexity

Co-hosted by Paul

Time Complexity Class

- Let $f: \mathbb{N} \to \mathbb{R}^+$ be a function. Define the time complexity class $TIME(f(n))$ to be the collection of all languages that are decidable by an $O(f(n))$ Turing Machine.

- The class above is a class of **languages**. Does this seem to cause loss of generality?

- For example, the sorting problem (or function), can it be regarded as a language?

Languages are general enough

- Let's look at the sorting example. Let $\mathtt{Sort}$ be the machine that does the sorting

- $\mathtt{Sort}$ is a function that takes a tuple as input and outputs a tuple of the same size

- As we learnt before, a function is a set of ordered pairs. Here,

$\mathtt{Sort}$ is a subset of $\mathbb{N}^* \times \mathbb{N}^*$.

Every function problem can be turned into a decision problem

- Suppose we are given a tuple $\bar{x} = (x_1, \ldots, x_n)$ and that we want to compute $\mathsf{Sort}(\bar{x})$.

- $\mathbb{N}^*$ is c.e., and so we can computably list it, say: $(\bar{y})_1, (\bar{y})_2, \ldots$

- And keep checking: is $(\bar{x}, (\bar{y})_1) \in \mathsf{Sort}$?, is $(\bar{x}, (\bar{y})_2) \in \mathsf{Sort}$?, …. until one of them is Yes

Vice versa

- Every decision problem is a function problem. Why?

- Answer: the characteristic function

- Note that the transition from a function problem to a decision problem does not necessarily preserve the TIME complexity class

- Function Problem: Consider a machine $\mathtt{Solver}$ which can take an equation as an input (say quadratic equations), and outputs solutions.

- Decision Problem: Given an equation and a value x, decide whether x is a solution for the equation or not.

Sort again (thoughts)

- When we listed $(\bar{y})_1, (\bar{y})_2, \ldots$ one may chose to do that smartly so the sorted tuple shows up faster

- For example, we could only list tuple of length n

- With a deeper look, finding a smart way to list the tuples is a process that has its own running time

Break

Hope we are comfortable with the fact that complexity theory is developed through decision problems

The class P

- $P = \{L: L \text{ is a language decidable by some polytime TM}\}$

- Note that $P = \bigcup_k TIME(n^k)$

- Why is

$$\bigcup_{k \in \mathbb{N}} TIME(n^k) = \{L: L \text{ is a language decidable by some polytime TM}\}$$

How do we prove equality of sets?

- $\subseteq$:

Let L be an arbitrary language from $\bigcup_{k \in \mathbb{N}} TIME(n^k)$

- $\supseteq$

Let L be an arbitrary language decidable by some polytime TM

The *PATH* problem

- Given a directed graph G and two nodes s, t in G. Consider the following question: Is there a path from s to t?

- $PATH =$ $\{(G, s, t): G \text{ is a directed graph that has a directed path from } s \text{ to } t\}$

- This set (or relation) *PATH* is an example of what we mean by a problem in the context of complexity

Unfolding *PATH*

- The question we first asked is equivalent to the following decision problem

$$\text{Is } (G, s, t) \in PATH?$$

- This question unfolds into:

$$\text{Is } G \text{ } a \text{ directed graph?}$$
$$\text{Are } s, t \text{ vertices in } G?$$

Is there a number n, and vertices $v_1, v_2, \ldots v_n$ such that the edges $(s, v_1), (v_1, v_2), \ldots (v_n, t)$ are edges in G?