Lecture 2 + 3, Classes P and NP

Turing Machine:

Q, Γ, Σ, δ, q_{start}, q_{accept}, q_{reject}

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\mathsf{L}, \mathsf{S}, \mathsf{R}\}^k$$

Basic steps of a Turing Machine: reading, writing, moving pointer.

Church Turing Thesis: Any algorithm has an equivalent Turing Machine*.
*except possibly Quantum algorithms.

Turing Machine model is robust to variation including:

Larger Alphabet, Multitape, RAM.

A Turing Machine can be encoded as a string.

High Level Language = Turing Machine.

Computing a function and running time

Let $f : \{0, 1\}^* \to \{0, 1\}$ and $T : N \to N$ be some functions, and let M be a ™.

1. We say that M computes f if for every $x \in \{0, 1\}^*$, whenever M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape.
2. We say M computes f in T(n)-time if its computation on every input x requires at most T(|x|) steps.

Decides: We say that a TM ***decides*** a language $L \subseteq \{0, 1\}^*$, if it computes the function $f_L : \{0, 1\}^* \to \{0, 1\}$, where $f_L(x) = 1 \Leftrightarrow x \in L$.

Definition 1.12 *(The class* **DTIME***)* Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in $\mathbf{DTIME}(T(n))$ iff there is a Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L. ◇

The Class P (feasible decision problems)

Definition 1.13 *(The class* **P***)*
$\mathbf{P} = \cup_{c \geq 1} \mathbf{DTIME}(n^c)$.

EXAMPLE 1.14 *(Graph connectivity)*

In the *graph connectivity* problem, we are given a graph G and two vertices s, t in G. We have to decide if s is connected to t in G. This problem is in **P**. The algorithm that shows this uses *depth-first search*, a simple idea taught in undergraduate courses. The algorithm

Complexity of DFS: O(V+E)

The class contains only decision problems. Thus we cannot say, for example, that "integer multiplication is in P."

Read about the CT Thesis from book p 26.

The **strong form of the CT thesis** says that every physically realizable computation model can be simulated by a TM with polynomial overhead (in other words, t steps on the model can be simulated in tc steps on the TM, where c is a constant that depends upon the model). If true, it implies that the class P defined by the aliens will be the same as ours.

~~Read from **section 1.6.3 Criticisms of P and some efforts to address them.**~~

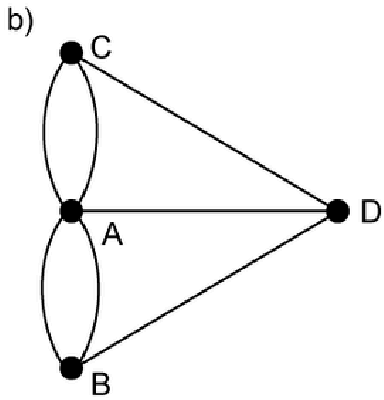
Agrawal-Kayal-Saxena Primality Test (2002):

p is prime if $(x - 1)^p - (x^p - 1)$ has all its coefficients divisible by p.

first primality-proving algorithm to be simultaneously *general*, *polynomial-time*, *deterministic*, and *unconditionally correct*.

$O(n^6)$

Euler Circuit: A circuit that visits every edge exactly once.



A graph has an Euler Circuit iff every vertex has even degree (O(v))

Hamiltonian Circuit : A circuit that visits every vertex exactly one.

We don't have a polytime algorithm for Hamiltonian Circuit!

Definition 2.1 *(The class* **NP***)*
A language $L \subseteq \{0,1\}^*$ is in **NP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M (called the *verifier* for L) such that for every $x \in \{0,1\}^*$,

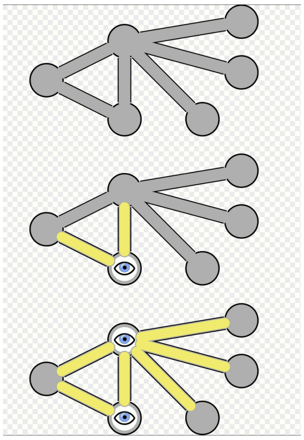
$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a *certificate* for x (with respect to the language L and machine M).

Intuition about P vs NP: Would you prefer to write a proof or verify a proof?

Example 1: Hamiltonian Circuit ∈ NP

Example 2: Vertex Cover ∈ NP
A set of vertices such that every edge has at least 1 of its endpoints in the vertex cover



VC = {(G, k) | G contains a vertex cover of size k }

Example 3: Independent Set ∈ NP
A set of vertices such that no two vertices are adjacent
IS = {(G, k) } G contains an independent set of size k }

Clearly, P ⊆ NP since the polynomial function p(|x|) is allowed to be 0.

Question for class: Show that TSP ∈ NP Traveling salesperson: Given a set of n nodes, (n choose 2) denoting distances between nodes, and a number k, decide if there is a closed circuit that visits every node exactly once and has total length at most k.

Claim 2.4 *Let* **EXP** $= \bigcup_{c>1}$. *Then* **P** $\subseteq$ **NP** $\subseteq$ **EXP**. ◇

PROOF: (**P** $\subseteq$ **NP**): Suppose $L \in$ **P** is decided in polynomial-time by a TM N. Then $L \in$ **NP**, since we can take N as the machine M in Definition 2.1 and make $p(x)$ the zero polynomial (in other words, u is an empty string).

(**NP** $\subseteq$ **EXP**): If $L \in$ **NP** and $M, p()$ are as in Definition 2.1, then we can decide L in time $2^{O(p(n))}$ by enumerating all possible strings u and using M to check whether u is a valid certificate for the input x. The machine accepts iff such a u is ever found. Since $p(n) = O(n^c)$ for some $c > 1$, the number of choices for u is $2^{O(n^c)}$, and the running time of the machine is similar. ■

Definition 2.5 For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in$ **NTIME**$(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$-time NDTM M such that for every $x \in \{0,1\}^*$, $x \in L \Leftrightarrow M(x) = 1$. ◇

Theorem 2.6 **NP** $= \cup_{c \in \mathbb{N}}$**NTIME**$(n^c)$.

Proof that the 2 definitions of NP are the same.

→ If we have a certificate that can be verified in polynomial time, then the same certificate can be guessed on one branch of the NTM and then verified in polynomial time.

← If we have a NTM, then we can try all possible solutions and if any one of them is correct, then we can use that as a certificate.