BU CS 332 – Theory of Computation

Lecture 20:

- More on NP

Reading:

Sipser Ch 7.3-7.4

Ran Canetti

November 19, 2020

Goals of complexity theory

Ultimate goal: Classify problems according to their feasibility and inherent computational difficulty

$\mathrm{P} \approx$ Decision problems which can be solved "*efficiently*".

Are there decidable problems not in P?

Yes: Some problems provably require exponential time! (Chapter 9)

Can we decide if a given problem is in P?

How about problems where solutions are efficiently verifiable?

Nondeterministic Time and NP

NTIME explicitly

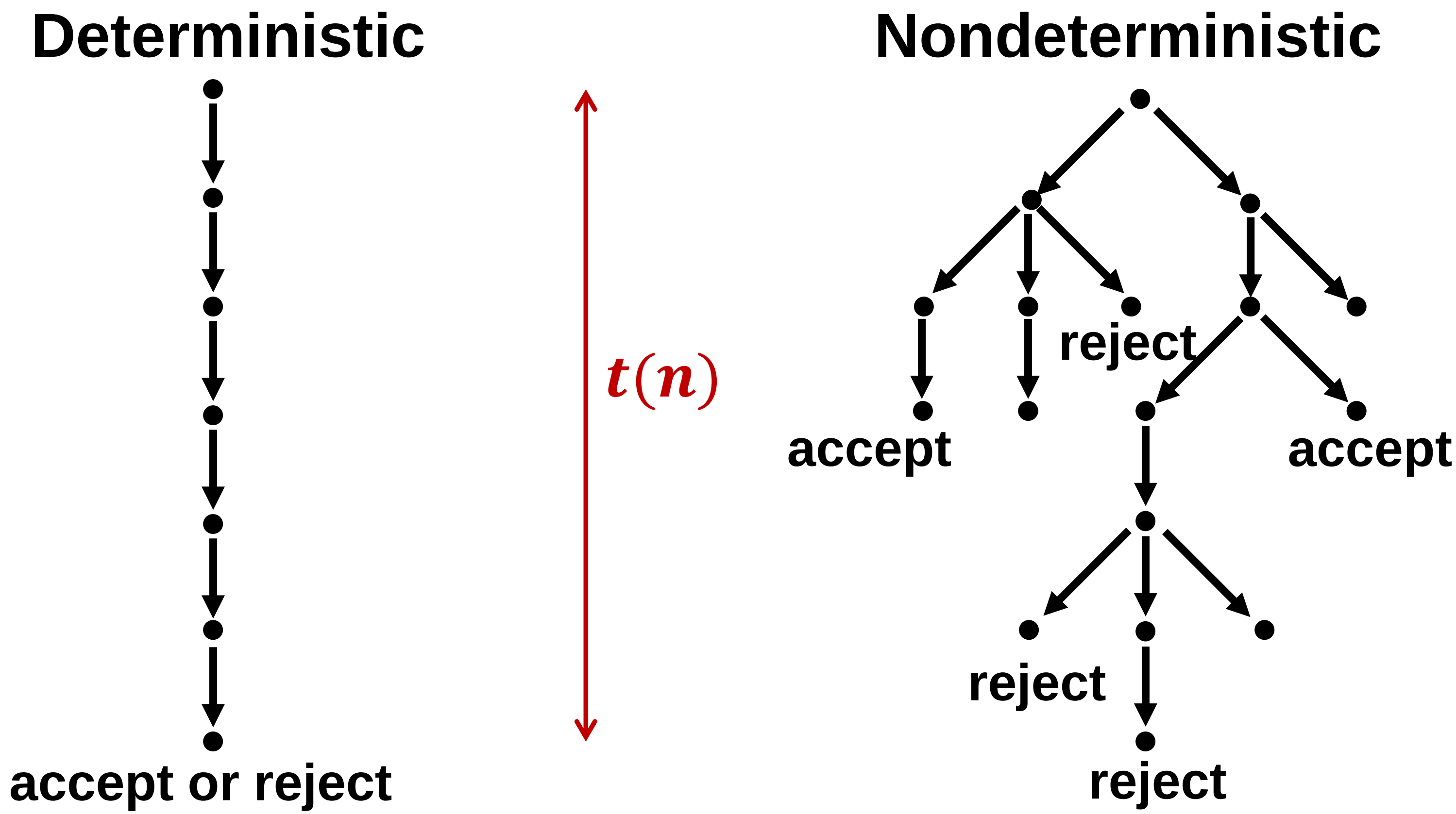
A language $A \in \text{NTIME}(f(n))$ if there exists an NTM M such that, on every input $x \in \Sigma^*$

1. Every computational branch of M halts in either the accept or reject state within $f(|x|)$ steps

2. $x \in A$ iff there exists an accepting computational branch of M on input w

3. x$\notin A$ iff every computational branch of M rejects on input x (or dies with no applicable transitions)

Deterministic vs. nondeterministic time



Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent single-tape TM running in time $2^{O(t(n))}$

Complexity class NP

Definition: NP is the class of languages decidable in polynomial time on a nondeterministic TM

$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k)$$

An alternative characterization of $\mathbf{NP}$

Definition: A TM V is a verifier for language L if:

- For any $x \in L$, $\exists w \;\; s.t. \;\; V(x, w) = 1$
- If $x \notin L$, $then \; \forall w, \;\; V(x, w) = 0$

We say that V is polynomial-time if its runtime is polynomial in the length of its first input(i.e., length of x).

An alternative characterization of NP

Definition: A TM V is a verifier for language L if:

- For any $x \in L$, $\exists w$ $s.t.$ $V(x,w) = 1$
- If $x \notin L$, $then$ $\forall w$, $V(x,w) = 0$

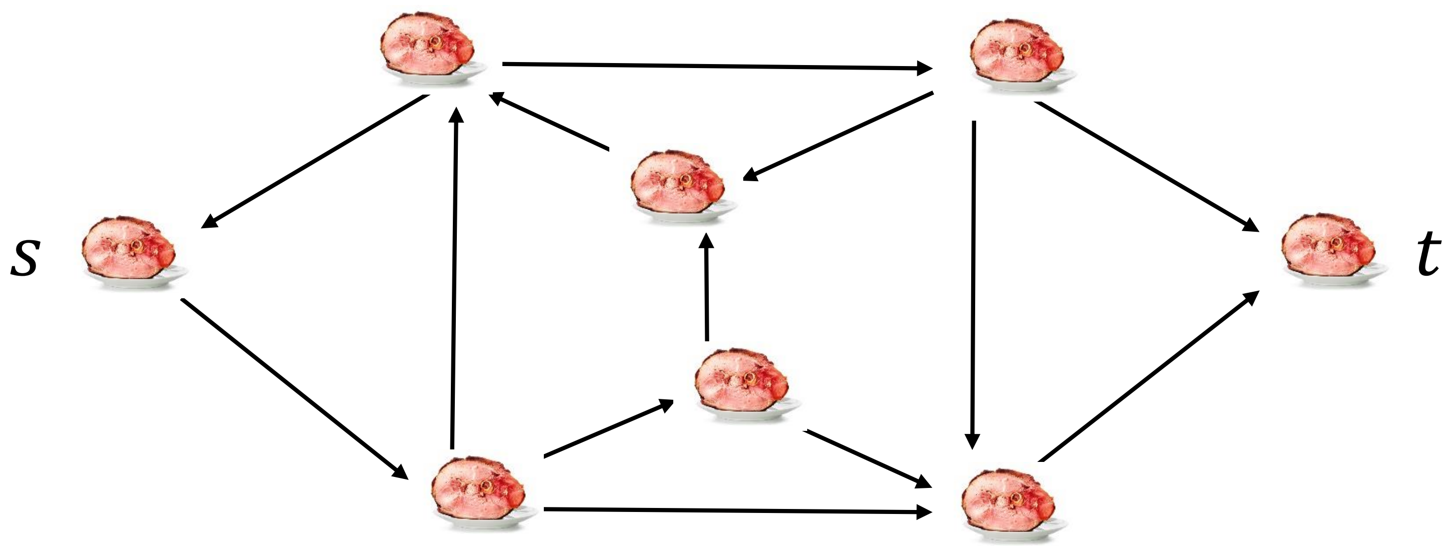
We say that V is polynomial-time if its runtime is polynomial in the length of its first input(i.e., length of x).

Theorem: A language $L \in \mathrm{NP}$ iff there is a polynomial-time verifier for L

Theorem: A language $L \in \mathrm{NP}$ iff there is a polynomial-time verifier for L

Problem in NP: Hamiltonian Path

$HAMPATH = \{\langle G, s, t\rangle \mid G$ is a directed graph and there is a path from s to t that passes through every vertex exactly once$\}$



$HAMPATH$ has a polynomial-time verifier

Certificate c:

Verifier V:

On input $\langle G, s, t; c \rangle$: (Vertices of G are numbers $1, \ldots, k$)

1. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number $1, \ldots, k$ appears exactly once
2. Check that $c_1 = s$, $c_k = t$, and there is an edge from every c_i to c_{i+1}
3. Accept if all checks pass, otherwise, reject.

Examples of NP languages: SAT

"Is there an assignment to the variables in a logical formula that make it evaluate to true?"

- Boolean variable: Variable that can take on the value true/false (encoded as 0/1)
- Boolean operations: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$
- An assignment of 0s and 1s to the variables satisfies a formula φ if it makes the formula evaluate to 1
- A formula φ is satisfiable if there exists an assignment that satisfies it

Examples of NP languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ Satisfiable?

Ex: $(x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge \overline{x_2}$ Satisfiable?

$$SAT = \{\langle\varphi\rangle | \varphi \text{ is a satisfiable formula}\}$$

Claim: $SAT \in \mathrm{NP}$

Examples of $\mathbf{NP}$ languages: TSP

"Given a list of cities and distances between them, is there a 'short' tour of all of the cities?"

More precisely: Given

- A number of cities m
- A function $D: \{1, \ldots, m\}^2 \to \mathbb{N}$ giving the distance between each pair of cities
- A distance bound B

$$TSP = \{\langle m, D, B\rangle \mid \exists \text{ a tour visiting every city with length} \leq B\}$$

P vs. NP

Question: Does $\mathrm{P} = \mathrm{NP}$?

Philosophically: Can every problem with an efficiently verifiable solution also be solved efficiently?

A central problem in mathematics and computer science

Millennium Problems

Yang–Mills and Mass Gap

Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

Riemann Hypothesis

The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Navier–Stokes Equation

This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certitude, but also understanding.

Hodge Conjecture

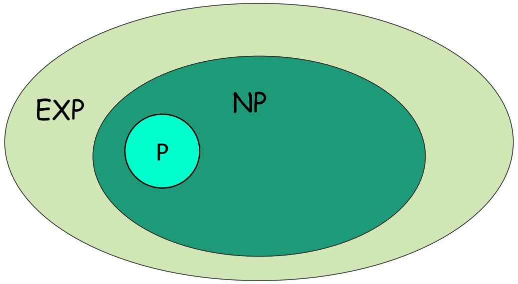
The answer to this conjecture determines how much of the topology of the solution set of a system of algebraic equations can be defined in terms of further algebraic equations. The Hodge conjecture is known in certain special cases, e.g., when the solution set has dimension less than four. But in dimension four it is unknown.

Poincaré Conjecture

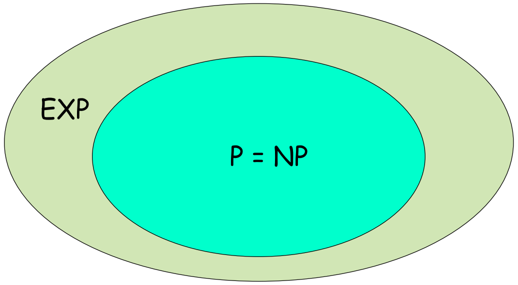
In 1904 the French mathematician Henri Poincaré asked if the three dimensional sphere is characterized as the unique simply connected three manifold. This question, the Poincaré conjecture, was a special case of Thurston's geometrization conjecture. Perelman's proof tells us that every three manifold is built from a set of standard pieces, each with one of eight well-understood geometries.

Birch and Swinnerton-Dyer Conjecture

Supported by much experimental evidence, this conjecture relates the number of points on an elliptic curve mod p to the rank of the group of rational points. Elliptic curves, defined by cubic equations in two variables, are fundamental mathematical objects that arise in many areas: Wiles' proof of the Fermat Conjecture, factorization of numbers into primes, and cryptography, to name three.



If $\mathrm{P} \neq \mathrm{NP}$



If $\mathrm{P} = \mathrm{NP}$

A world where $\mathrm{P} = \mathrm{NP}$

- Many important decision problems can be solved in polynomial time ($HAMPATH$, SAT, TSP, etc.)

- Many search problems can be solved in polynomial time (e.g., given a natural number, ***find*** a prime factorization)

- Many optimization problems can be solved in polynomial time (e.g., find the lowest energy conformation of a protein)

A world where $\mathrm{P} = \mathrm{NP}$

- Secure cryptography becomes impossible

  An NP search problem: Given a ciphertext C, find a plaintext m and encryption key k that would encrypt to C

- AI / machine learning become easy: Identifying a consistent classification rule is an NP search problem

- Finding mathematical proofs becomes easy: NP search problem: Given a mathematical statement S and length bound k, is there a proof of S with length at most k?

General consensus: $\mathrm{P} \neq \mathrm{NP}$

NP Completeness

What about a world where $\mathrm{P} \neq \mathrm{NP}$

Believe this to be true, but very far from proving it

$\mathrm{P} \neq \mathrm{NP}$ implies that there is a problem in NP which cannot be solved in polynomial time, but it might not be a useful one

Question: What would $\mathrm{P} \neq \mathrm{NP}$ allow us to conclude about problems we care about?

Idea: Identify the "hardest" problems in NP

Find $L \in \mathrm{NP}$ such that $L \in \mathrm{P}$ iff $\mathrm{P} = \mathrm{NP}$

Recall: Mapping reducibility

Definition:

A function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.

Definition:

Language A is mapping reducible to language B, written

$$A \leq_{\mathrm{m}} B$$

if there is a computable function $f: \Sigma^* \to \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \Leftrightarrow f(w) \in B$

Polynomial-time reducibility

Definition:

A function $f: \Sigma^* \to \Sigma^*$ is polynomial-time computable if there is a polynomial-time TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.

Definition:

Language A is polynomial-time mapping reducible to language B, written

$$A \leq_{\mathrm{p}} B$$

if there is a polynomial-time computable function $f: \Sigma^* \to \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \Leftrightarrow f(w) \in B$

Implications of poly-time reducibility

Theorem: If $A \leq_{\mathrm{p}} B$ and $B \in P$, then $A \in P$

Proof:

Cook-Levin Theorem and NP-Complete Problems

Cook-Levin Theorem

Theorem: SAT (Boolean satisfiability) is NP-complete

Proof: Already know $SAT \in \mathrm{P}$. Need to show every problem in NP reduces to SAT (later?)

Stephen A. Cook (1971)

Leonid Levin (1973)

New NP-complete problems from old

Lemma: If $A \leq_{\mathrm{p}} B$ and $B \leq_{\mathrm{p}} C$, then $A \leq_{\mathrm{p}} C$

(poly-time reducibility is transitive)

Theorem: If $C \in \mathrm{NP}$ and $B \leq_{\mathrm{p}} C$ for some NP-complete language B, then C is also NP-complete

New NP-complete problems from old

All problems below are NP-complete and hence poly-time reduce to one another!