BU CS 332 – Theory of Computation

Lecture 22:

- Nondeterministic time, NP
- NP-completeness

Reading:

Sipser Ch 7.3-7.4

OH: delay to 6-7 pm today

Mark Bun

April 14, 2021

Big-Oh, formally

$f(n) = O(g(n))$ means:

There exist constants $c > 0, n_0 > 0$ such that

$f(n) \leq cg(n)$ for every $n \geq n_0$

Example: Show that $7n^2 \cdot 3^n = 2^{O(n)}$

Means: $\exists\, g(n)$ s.t. $g(n) = O(n)$ and $7n^2 \cdot 3^n = 2^{g(n)}$

$\Leftrightarrow \log(7n^2 \cdot 3^n) = \log(2^{g(n)})$ where $g(n) = O(n)$

$\Leftrightarrow \log(7n^2 \cdot 3^n) = O(n)$

$\Leftrightarrow \underbrace{n \log 3}_{(\log 3)\cdot n} + \underbrace{2 \log n}_{\leq 2 \cdot n} + \underbrace{\log 7}_{\leq (\log 7)\, n} = O(n)$ (all hold for all $n \geq 1$)

LHS $\leq \underbrace{(\log 3 + 2 + \log 7)}_{c} \cdot n$ $(\forall\, n \geq \underbrace{1}_{n_0})$

Big-Oh, formally

Ex: $n = O(2^{\sqrt{n}})$

$n \leq 2^{\sqrt{n}}$ is true for "large enough n"

if $n_0 = 16$ then $n \leq 2^{\sqrt{n}}$ for all $n \geq n_0 = 16$

$f(n) = O(g(n))$ means:

There exist constants $c > 0, n_0 > 0$ such that

$f(n) \leq c g(n)$ for every $n \geq n_0$

Example: Show that $7n^2 \cdot 3^n = 2^{O(n)}$

Proof: Set $c = \log 3 + 2 + \log 7$, $n_0 = 1$

Set $g(n) = \log(7n^2 \cdot 3^n)$

Suffices to show that $g(n) = O(n)$ [since $7n^2 \cdot 3^n = 2^{O(n)} \Leftrightarrow g(n) = O(n)$]

$g(n) = \log(7n^2 \cdot 3^n) = \log 7 + 2\log n + n\log 3$

$\leq c \cdot n \quad (\forall\, n \geq n_0 = 1)$ $\square$

Nondeterministic time

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

A NTM M runs in time $f(n)$ if on every input $w \in \Sigma^n$, M halts on w within at most $f(n)$ steps on every computational branch

$\mathrm{NTIME}(f(n))$ is a class (i.e., set) of languages:

A language $A \in \mathrm{NTIME}(f(n))$ if there exists an NTM M that

1) Decides A, and
2) Runs in time $O(f(n))$

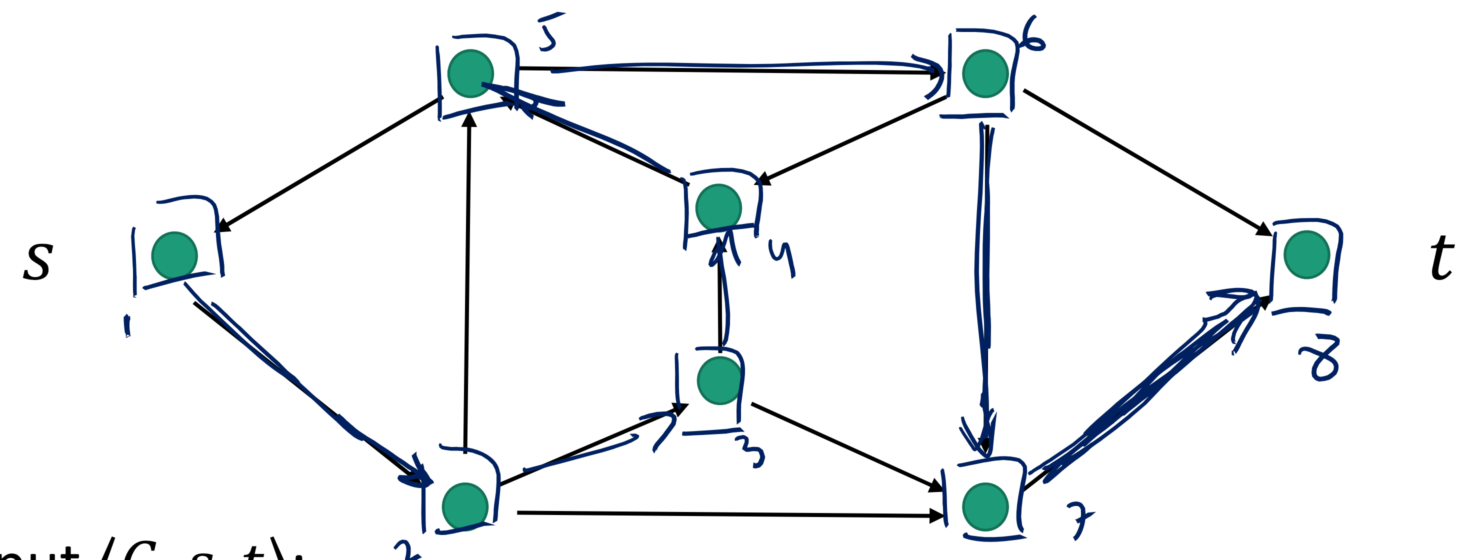
Complexity class NP

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

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

$HAMPATH \in \mathrm{NP}$

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



On input $\langle G, s, t\rangle$:

1. **Nondeterministically** guess a sequence of vertices
2. Check that the guess forms a Hamiltonian path from s to t

An alternative characterization of NP

$w \in L \Rightarrow \exists c\ V(\langle w,c\rangle)$ accepts

$w \notin L \Rightarrow \forall c\ V(\langle w,c\rangle)$ reject

"Languages with polynomial-time verifiers"

A verifier for a language L is a deterministic algorithm V such that $w \in L$ iff there exists a string c such that $V(\langle w, c\rangle)$ accepts

"certificate", "witness", "proof"

Running time of a verifier is only measured in terms of $|w|$

V is a polynomial-time verifier if it runs in time polynomial in $|w|$ on every input $\langle w, c\rangle$

(Without loss of generality, $|c|$ is polynomial in $|w|$, i.e., $|c| = O(|w|^k)$ for some constant k)

$HAMPATH$ has a polynomial-time verifier

Certificate c: $c_1, \ldots, c_k \in [k]$

Poly run time:
1) $c_1, \ldots, c_k$ has length poly$(|\langle G,s,t\rangle|)$
2) Checking certificate takes poly takes poly time

Verifier V: $\underbrace{}_{}$ instance w, certificate

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

Check that $c_1, \ldots, c_k$ form a Ham path from s to t

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.

Correctness:
- If $\langle G,s,t\rangle \in HAMPATH$, $\exists$ Ham path $c_1, \ldots, c_k$. This choice of certificate causes verifier to accept.
- If $\langle G,s,t\rangle \notin HAMPATH$, then all sequences $c_1, \ldots, c_k$ either fail to hit every vertex or are not an s-t path, so verifier rejects.

NP is the class of languages with polynomial-time verifiers

L decidable in poly time on an NTM

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

Proof: $\Leftarrow$ Let L have a poly-time verifier $V(\langle w, c\rangle)$

Idea: Design NTM N for L that nondeterministically guesses a certificate

N = "On input w:

1) Nondeterministically guess certificate c (of appropriate length)

2) Run V(⟨w, c⟩), return result"

Correctness: $w \in L \iff \exists c$ s.t. $V(\langle w,c\rangle)$ accepts $\iff N$ accepts w

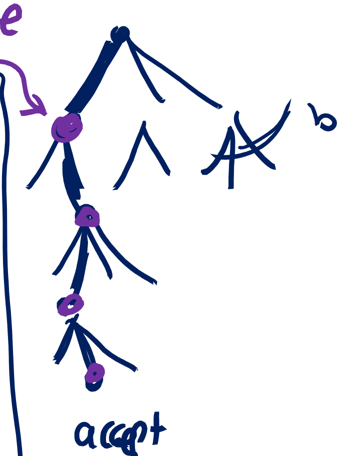
Poly run time:
1) c has to have poly(|w|) length for V to be efficient
2) V itself runs in poly time $\Rightarrow$ poly time NTM

NP is the class of languages with polynomial-time verifiers

$\Rightarrow$ Let L be decided by an NTM N making up to b nondeterministic choices in each step , running $T(n)$

Idea: Design verifier V for L where certificate is sequence of "good" nondeterministic choices

certificate



$V =$ " On input $\langle w, c \rangle$ where $c = (c_1, \dots, c_{T(|w|)}) \in \{b\}^{T(|w|)}$

1. Simulate N on w using c as the sequence of nondet. choices
2. Accept if accepts, reject otherwise "

- Need to show correctness i.e. $w \in L \iff \exists\, c$ s.t. $V(\langle w, c \rangle)$ accepts
- Need to show poly-time: V runs in time poly$(|w|)$

Alternative proof of $\mathrm{NP} \subseteq \mathrm{EXP}$

input $\langle w, c \rangle$

One can prove $\mathrm{NP} \subseteq \mathrm{EXP}$ as follows. Let V be a verifier for a language L running in time $T(n)$. We can construct a $2^{O(T(n))}$ time algorithm for L as follows.

input w

a) On input $\langle w, c \rangle$, run V on $\langle w, c \rangle$ and output the result ✗

b) On input w, run V on all possible $\langle w, c \rangle$, where c is a certificate. Accept if any run accepts.

c) On input w, run V on all possible $\langle w, c \rangle$, where c is a certificate of length at most $T(|w|)$. Accept if any run accepts. ✓

running time $\approx T(n) \cdot 2^{T(n)}$ exp alg. where T is poly

d) On input w, run V on all possible $\langle x, c \rangle$, where x is a string of length $|w|$ and c is a certificate of length at most $T(|w|)$. Accept if any run accepts.

WARNING: Don't mix-and-match the NTM and verifier interpretations of NP

To show a language L is in NP, do exactly one:

instance x to problem specified by L

Guess a string $c_1, \ldots, c_k \in \{0,1\}^k$ in time $O(k)$

1) Exhibit a poly-time NTM for L

   N = "On input x:

   <Do some nondeterministic stuff>…"

OR

instance x

2) Exhibit a poly-time (deterministic) verifier for L

   V = "On input x and certificate c:

   <Do some deterministic stuff>…"

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) Ex: x_1, x_2 x, y, z
- Boolean operations: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ $=: \varphi$

  $\neg x_2$
- An assignment of 0s and 1s to the variables satisfies a formula φ if it makes the formula evaluate to 1

  Assignment $x_1 = 0, x_2 = 1, x_3 = 1$ does not satisfy φ, but $x_1 = 1, x_2 = 1, x_3 = 1$ does satisfy φ
- A formula φ is satisfiable if there exists an assignment that satisfies it

Examples of NP languages: SAT

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

Assignment $x_1 = 1, x_2 = 1, x_3 = 1$ satisfies it

Ex: $(x_1 \vee x_2) \wedge \overline{x_1} \wedge \overline{x_2}$ Not satisfiable Satisfiable?

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

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

NTM for SAT:

N = "On input $\varphi(x_1, \ldots, x_n)$:
1. Nondet. guess $\hat{x}_1, \ldots, \hat{x}_n \in \{0,1\}^n$
2. If $\varphi(\hat{x}_1, \ldots, \hat{x}_n) = 1$, accept. Else reject."

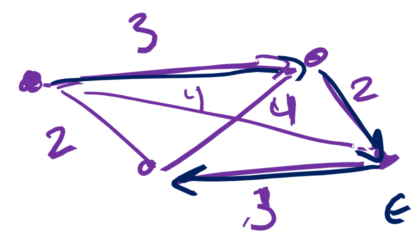
Verifier for SAT:

certificate: $c_1, \ldots, c_n \in \{0,1\}^n$

V = "On input $\varphi(x_1, \ldots, x_n), (c_1, \ldots, c_n)$:
1. If $\varphi(c_1, \ldots, c_n) = 1$, accept. Else, reject"

Examples of **NP** languages: Traveling Salesperson

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



$m = 4$

$B = 10$

$\in TSP$

More precisely: Given

- A number of cities m
- A function $D: \{1, \dots, m\}^2 \to \mathbb{N}$ 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

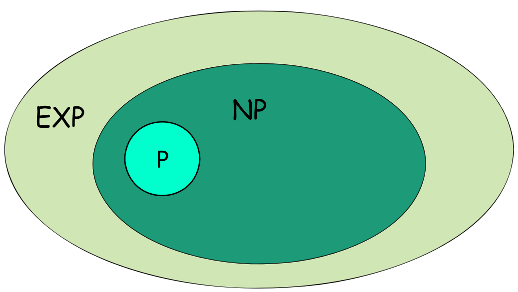
Question: Does $\mathrm{P} = \mathrm{NP}$?

know $P \subseteq NP$

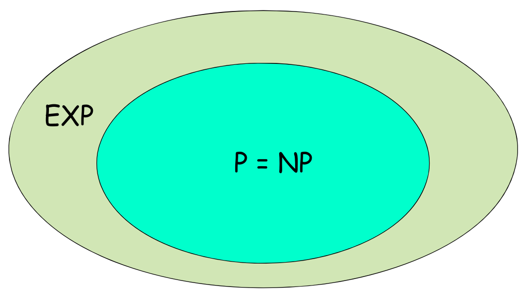
Question: is $NP \subseteq P$?

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

A central problem in mathematics and computer science



If $P \neq NP$



If $P = NP$

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.

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