BU CS 332 – Theory of Computation

Lecture 21:

- NP-Completeness
- Cook-Levin Theorem
- Reductions

HW 8 due on Tuesday
4/21

Reading:

Sipser Ch 7.3-7.5

Mark Bun

April 15, 2020

Last time: Two equivalent definitions of NP

1) NP is the class of languages decidable in polynomial time on a nondeterministic TM

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

2) A polynomial-time verifier for a language L is a deterministic $\mathrm{poly}(|w|)$-time algorithm V such that $w \in L$ iff there exists a string c such that $V(\langle w, c\rangle)$ accepts

certificate, proof, witness ($w \in L \iff \exists c$ s.t. $V(\langle w,c\rangle)$ accepts)

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

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: $\land$ (AND), $\lor$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $(x_1 \lor \overline{x_2}) \land x_3$
- 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

Ex: $x_1 = 0, x_2 = 1, x_3 = 1$ is not satisfying

Ex: $x_1 = 1, x_2 = 1, x_3 = 1$ is satisfying

Is satisfiable

Examples of NP languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ YES: $x_1 = 1, x_2 = 1, x_3 = 1$ Satisfiable?

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

$x_1 = 1, x_2 = 0$

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

$x_1 \wedge \overline{x_1} \notin SAT$

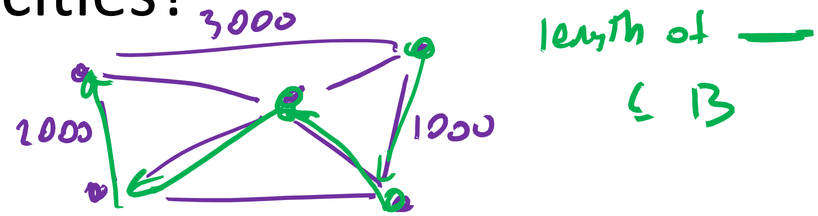
Claim: $SAT \in \mathrm{NP}$

On input $\langle\varphi\rangle$:

1) Guess an assignment to vars $x_1, \ldots, x_k \in \{0,1\}^k$
2) Check that assignment is satisfying. Accept if yes, reject o.w.

Examples of 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

Often believed:
- SAT requires time $2^{\Theta(n)}$ [Exponential Time Hypothesis (ETH)]
- SAT requires time $\Theta(2^{n(1-\delta)})$ for every $\delta > 0$ [Strong ETH]
- SAT requires time $\Theta(2^{\frac{n}{2}(1-\delta)})$ $\forall \delta > 0$ even on quantum computers [Quantum strong ETH]

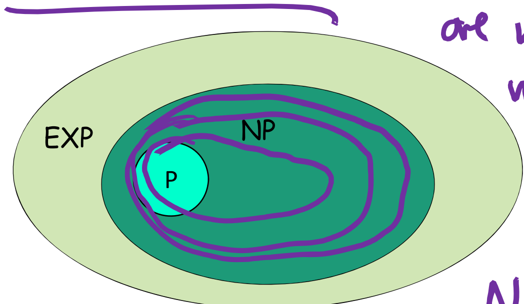
Question: Does $\mathrm{P} = \mathrm{NP}$?

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

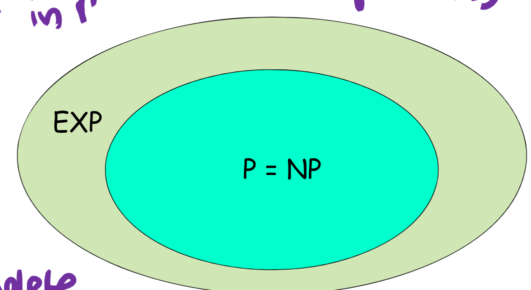
Compositeness testing $\in$ NP (also in P)

A central problem in mathematics and computer science

Ladner's theorem: If $P \neq NP$, $\exists$ "NP-intermediate problems" are neither in P nor NP-complete



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

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 $f: \Sigma^* \rightarrow \Sigma^*$ 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 $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \Leftrightarrow f(w) \in B$

Polynomial-time reducibility

Definition:

deterministic

A function $f: \Sigma^* \rightarrow \Sigma^*$ 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 reducible to language B, written

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

if there is a polynomial-time computable function $f: \Sigma^* \rightarrow \Sigma^*$ 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: Let M decide B in poly time, and let f be a poly-time reduction from A to B. The following TM decides A in poly time:

On input w:

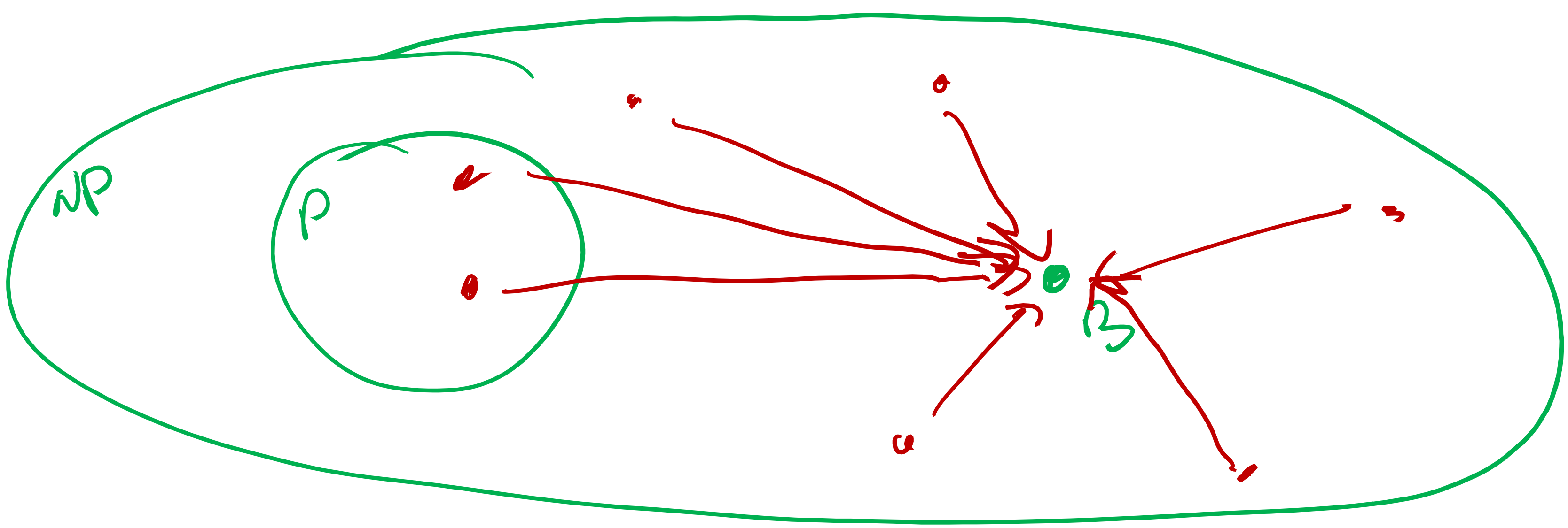
1) Run machine computing reduction to produce $f(w)$
2) Run M on $f(w)$
3) If M accepts, accept. o.w. reject.

Runs in poly time: M is poly-time, run on poly-length input $f(w)$

NP-completeness

Definition: A language B is NP-complete if

1) $B \in \mathrm{NP}$, and
2) Every language $A \in \mathrm{NP}$ is poly-time reducible to B, i.e., $A \leq_{\mathrm{p}} B$ ("B is NP-hard")



Implications of NP-completeness

Theorem: Suppose B is NP-complete.

Then $B \in \mathrm{P}$ iff $\mathrm{P} = \mathrm{NP}$

Proof:

$\Leftarrow$) If $P = NP$ and $B \in NP$, then $B \in P$

$\Rightarrow$) If $B \in P$, let A be an arbitrary language $A \in NP$

$A \leq_p B \Rightarrow A \in P$ (because of Thm + $B \in P$)

$\Rightarrow P = NP$

Implications of NP-completeness

Theorem: Suppose B is NP-complete.

Then $B \in \mathrm{P}$ iff $\mathrm{P} = \mathrm{NP}$

Consequences of B being NP-complete:

1) If you want to show $\mathrm{P} = \mathrm{NP}$, you just have to show $B \in \mathrm{P}$
2) If you want to show $\mathrm{P} \neq \mathrm{NP}$, you just have to show $B \notin \mathrm{P}$
3) If you already believe $\mathrm{P} \neq \mathrm{NP}$, then you believe $B \notin \mathrm{P}$

Cook-Levin Theorem and NP-Complete Problems

Cook-Levin Theorem

Theorem: SAT (Boolean satisfiability) is NP-complete

Proof: Already know $SAT \in \mathrm{NP}$. 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)

SAT $\leq_p$ C

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

Proof. Need to show $A \leq_p C$ for every $A \in NP$

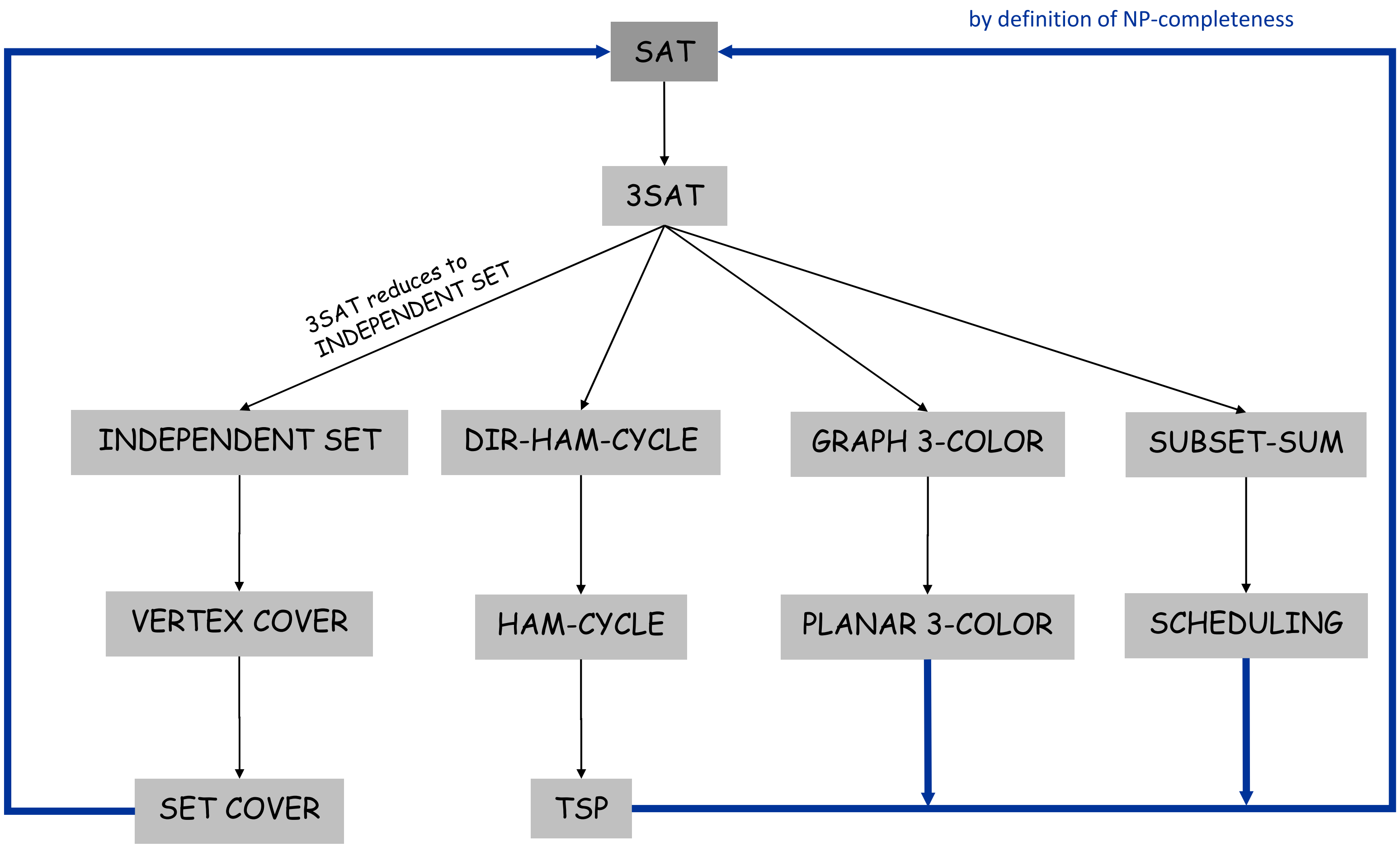
If $A \in NP$, $A \leq_p B$ (since B NP-complete)

$B \leq_p C$ (hypothesis)

$\Rightarrow A \leq_p C$ (transitivity)

New NP-complete problems from old

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



$3SAT$ (3-CNF Satisfiability)

Definition(s):

- A literal either a variable or its negation x_5 , $\overline{x_7}$
- A clause is a disjunction (OR) of literals Ex. $x_5 \vee \overline{x_7} \vee x_2$
- A 3-CNF is a conjunction (AND) of clauses where each clause contains exactly 3 literals

Ex. $C_1 \wedge C_2 \wedge \ldots \wedge C_m =$

$$(x_5 \vee \overline{x_7} \vee x_2) \wedge (\overline{x_3} \vee x_4 \vee x_1) \wedge \cdots \wedge (x_1 \vee x_1 \vee x_1)$$

$$3SAT = \{\langle\varphi\rangle | \varphi \text{ is a satisfiable } 3-\text{CNF}\}$$

$3SAT$ is NP-complete

1) Use de Morgan's rules to push all negations to 1.teal level

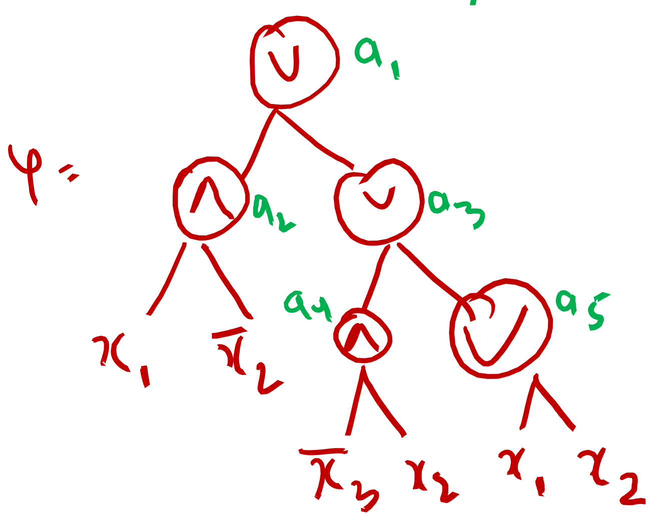
Theorem: $3SAT$ is NP-complete

Proof idea: 1) $3SAT$ is in NP (why?)

2) Show that $SAT \leq_p 3SAT$

Idea of reduction: Given a poly-time algorithm converting an arbitrary formula φ into a 3CNF ψ such that φ is satisfiable iff ψ is satisfiable

$\varphi =$



$$\psi = a_1 \wedge (a_1 \equiv a_2 \vee a_3) \wedge (a_2 \equiv x_1 \wedge \overline{x_2}) \wedge (a_3 \equiv a_4 \vee a_5) \wedge (a_4 \equiv \overline{x_3} \wedge x_2) \wedge (a_5 \equiv x_1 \wedge x_2)$$

In general: Can convert any function $f: \{0,1\}^3 \to \{0,1\}$ into a 3CNF