BU CS 332 – Theory of Computation

https://forms.gle/Jqs5498YecVX9Q5v6

Lecture 22:

- NP-completeness

Reading:

Sipser Ch 7.4-7.5

Mark Bun

April 24, 2024

Last time: Two equivalent definitions of $\mathbf{NP}$

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 certificate c
such that $V(\langle w, c\rangle)$ accepts

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

Examples of NP languages

- Hamiltonian path

  Given a graph G and vertices s, t, does G contain a Hamiltonian path from s to t?

- Clique

  Given a graph G and natural number k, does G contain a clique of size k?

- Subset Sum

  Given a list of natural numbers $x_1, \ldots, x_k, t$ is there a subset of the numbers $x_1, \ldots, x_k$ that sum up to exactly t?

- Boolean satisfiability (SAT)

  Given a Boolean formula, is there a satisfying assignment?

- Vertex Cover

  Given a graph G and natural number k, does G contain a vertex cover of size k?

- Traveling Salesperson

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) x, y, z x_1, x_2, x_3
- Boolean operations: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $\varphi(x_1, x_2, x_3) = (x_1 \vee \overline{x_2}) \wedge x_3$
- An assignment of 0s and 1s to the variables satisfies a formula φ if it makes the formula evaluate to 1

  $x_1 = 0$ $x_2 = 0$ $x_3 = 1$ $\Rightarrow \varphi(0,0,1) = (0 \vee 1) \wedge 1 = 1$

- A formula φ is satisfiable if there exists an assignment that satisfies it φ is satisfiable because $0,0,1$ satisfies it.

Examples of NP languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ $x_1 = 0,\ x_2 = 0,\ x_3 = 1$ satisfies this Satisfiable?

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: $SAT \in \mathrm{NP}$

Poly-time NTM

On input formula $\varphi(x_1, \ldots, x_n)$

1. Nondeterministically guess assignment $c_1, \ldots, c_n \in \{0,1\}^n$
2. Evaluate $\varphi(c_1, \ldots, c_n)$. If $=1$, accept. If $=0$, reject.

Deterministic Verifier

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

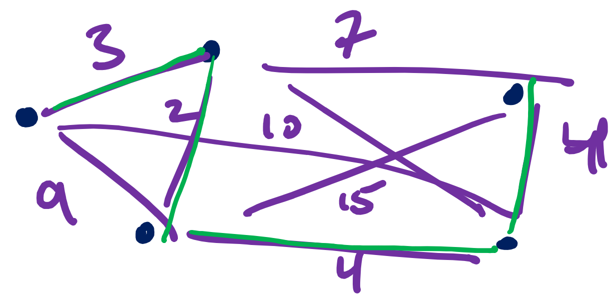
Det. Verifier:

On input φ and cert. c:

1. Evaluate $\varphi(c_1, \ldots, c_n)$. If $=1$, accept. If $=0$, 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?"



B = 28

3+2+4+4 = 13 ≤ 28

More precisely: Given

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

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

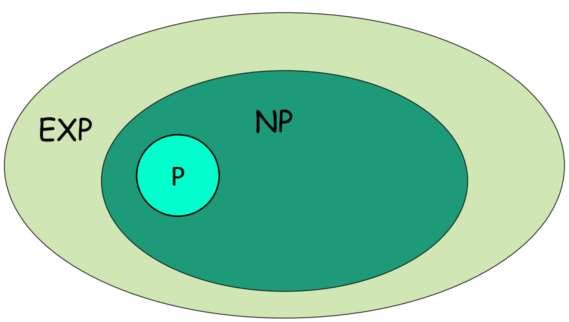
Nondet. guess tour; check it visits all cities & has length ≤ B.

P vs. NP

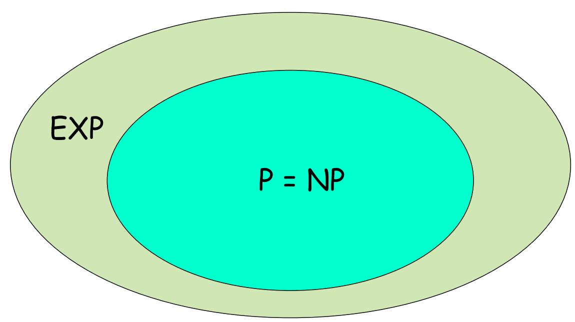
Question: Does $\mathrm{P} = \mathrm{NP}$?

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

A central problem in mathematics and computer science



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



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

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.

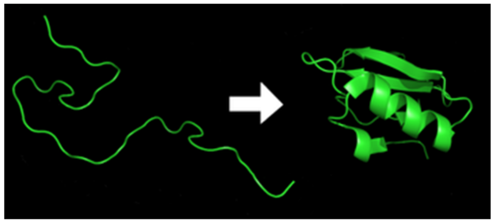
Birch and Swinnerton-Dyer Conjecture
Supported by much experimental evidence, this conjecture relates the number of points on an elliptic curve mod p to the rank of the group of rational points. Elliptic curves, defined by cubic equations in two variables, are fundamental mathematical objects that arise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

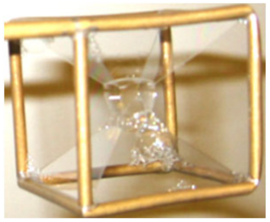
In a world where $\mathrm{P} = \mathrm{NP}$:

- 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)





In a world where $\mathrm{P} = \mathrm{NP}$:

- Secure cryptography (as we know it) 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

Understanding the $\mathbf{P}$ vs. $\mathbf{NP}$ question

Most believe $\mathrm{P} \neq \mathrm{NP}$, but we are very far from proving it

Question 1: How can studying specific computational problems help us get a handle on resolving P vs. NP?

Question 2: What would $\mathrm{P} \neq \mathrm{NP}$ allow us to conclude about specific problems we care about?

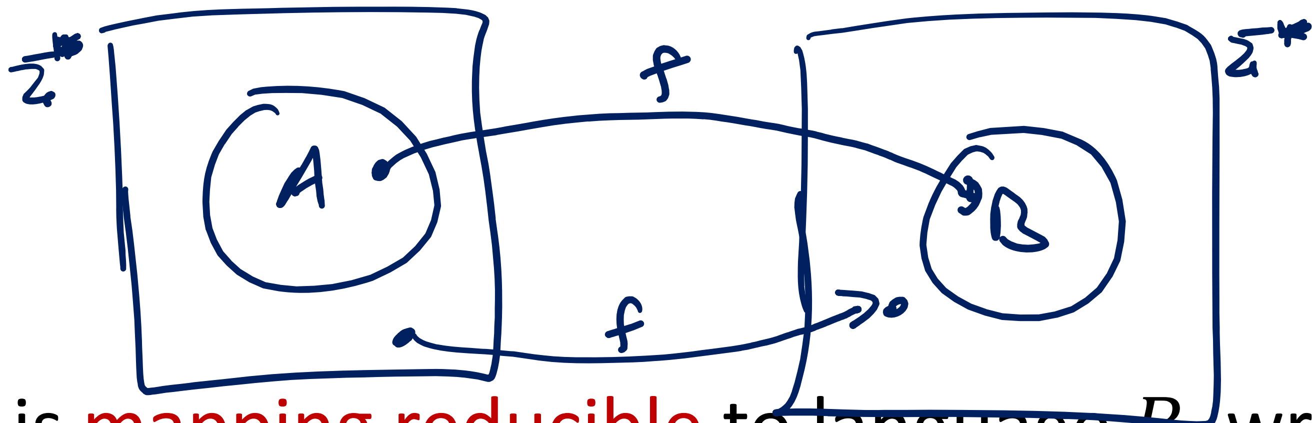
Idea: Identify the "hardest" problems in NP

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

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 reducible to language B, written

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

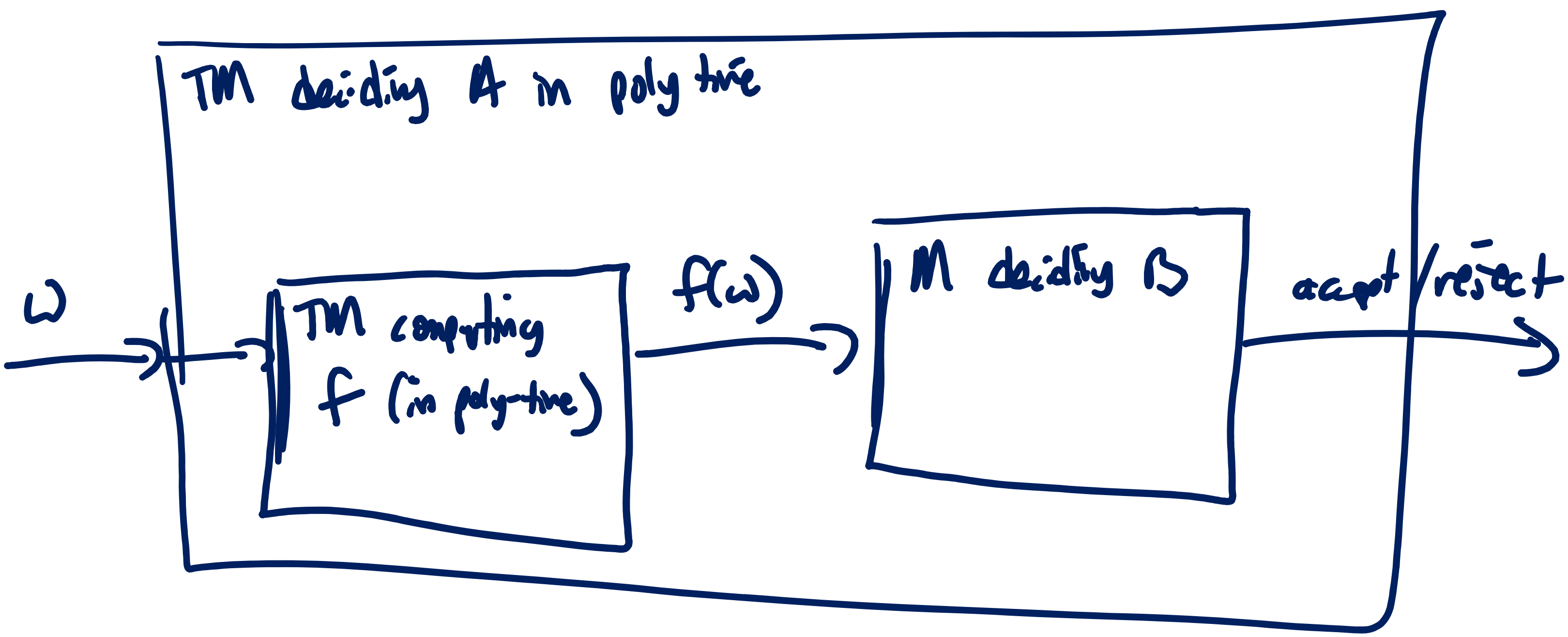
if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \Leftrightarrow f(w) \in B$

Implications of poly-time reducibility

cf. If $A \leq_m B$ and B is decidable, then A is decidable

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

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:



Thm: If $A \le_p B$ and B decidable, then A is decidable

Proof. Let TM M decide B in poly time. The following TM N decides A in poly time:

TM N:

On input w:

1. Compute $f(w)$
2. Run M on input $f(w)$.
3. If accepts, accept. If rejects, reject.

Runtime:

1. Computing $f(w)$ takes poly($|w|$) time
2. $|f(w)| \le$ poly($|w|$) since f poly-time computable
   $\Rightarrow M(f(w))$ takes poly($|w|$) time.

Correctness:

If $w \in A$:

$f(w) \in B$ [correctness of poly-time reduction]

$\Rightarrow M$ accepts $f(w)$ [correctness of M]

If $w \notin A$:

$f(w) \notin B$ [correctness of reduction]

$\Rightarrow M$ rejects $f(w)$.

Is NP closed under poly-time reductions?

If $A \leq_{\mathrm{p}} B$ and B is in NP, does that mean A is also in NP?

{Analogous to $A \leq_m B$, B recognizable $\Rightarrow A$ recognizable}

a) Yes, the same proof works using NTMs instead of TMs

b) No, because the new machine is an NTM instead of a deterministic TM

c) No, because the new NTM may not run in polynomial time

d) No, because the new NTM may accept some inputs it should reject

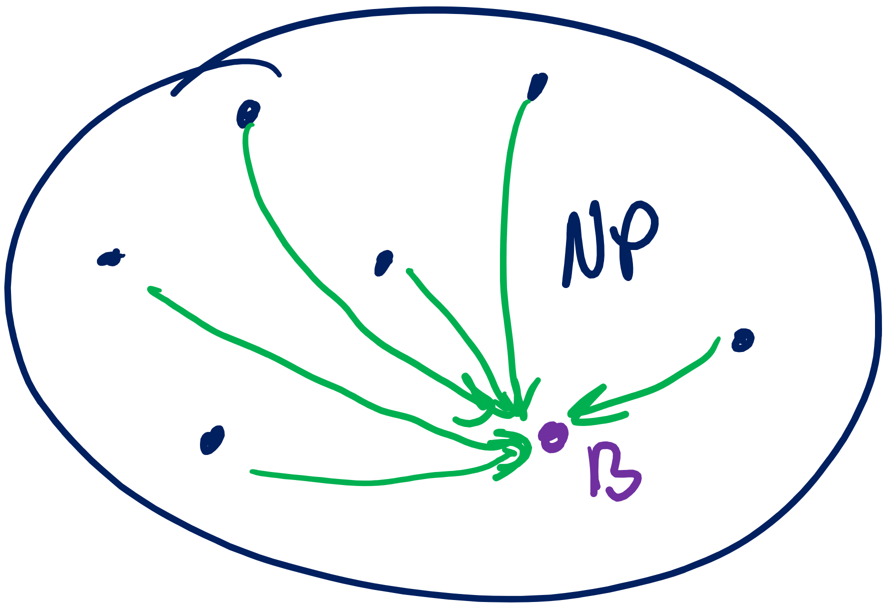
e) No, because the new NTM may reject some inputs it should accept

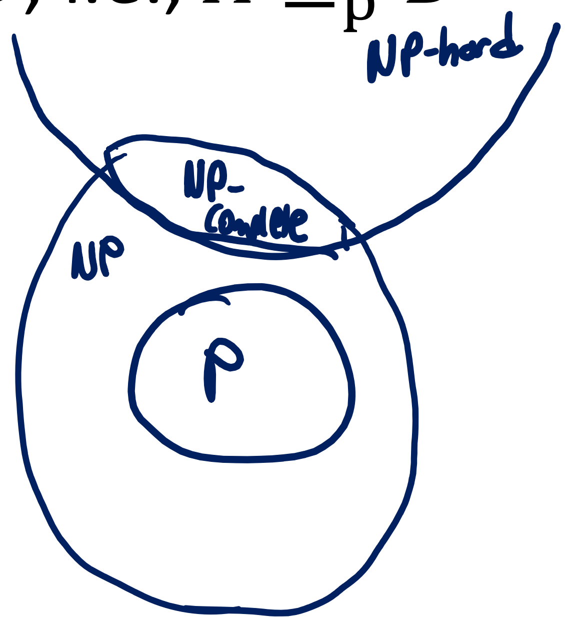
NP-completeness

Definition: A language B is NP-complete if

1) $B \in \mathrm{NP}$, and

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





Implications of NP-completeness

Theorem: Suppose B is NP-complete. [$B \in NP$ <u>and</u> $\forall A \in NP,\ A \le_p B$]

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

Proof: $\Leftarrow$ | If $P = NP$, then $B \in NP = P$

$\Rightarrow$ | If $B \in P$: Goal is to show $NP \subseteq P$

Let $A \in NP$. Then $A \le_p B$ [B is NP-hard]

$\Rightarrow A \in P$.

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 prove $\mathrm{P} = \mathrm{NP}$, you just have to prove $B \in \mathrm{P}$
2) If you want to prove $\mathrm{P} \neq \mathrm{NP}$, a good candidate is to try to show that $B \notin \mathrm{P}$
3) If you believe $\mathrm{P} \neq \mathrm{NP}$, then you also believe $B \notin \mathrm{P}$

Cook-Levin Theorem and NP-Complete Problems

Do NP-complete problems exist?

← time bound t encoded in unary

Theorem: $TMSAT = \{\langle N, w, 1^t \rangle \mid$ NTM N accepts input w within t steps$\}$ is NP-complete

Proof sketch: 1) $TMSAT \in \mathrm{NP}$: Homework 10, Problem 3a

2) $TMSAT$ is NP-hard: Let $L \in \mathrm{NP}$ be decided by NTM N running in time $T(n)$. The following poly-time TM shows $L \leq_{\mathrm{p}} TMSAT$:

some polynomial

"On input w (an instance of L):

Output $\langle N, w, 1^{T(|w|)} \rangle$."

↑ instance of TMSAT

correctness:

$w \in L \iff N$ accepts w within $T(|w|)$ steps [correctness of N + runtime of N]

$\iff \langle N, w, 1^{T(|w|)} \rangle \in TMSAT$.

Cook-Levin Theorem (Sipser Ch. 7.4)

Theorem: SAT (Boolean satisfiability) is NP-complete

"Proof": Already know $SAT \in \mathrm{NP}$. (Much) harder direction: Need to show every problem in NP reduces to SAT

Stephen A. Cook (1971)

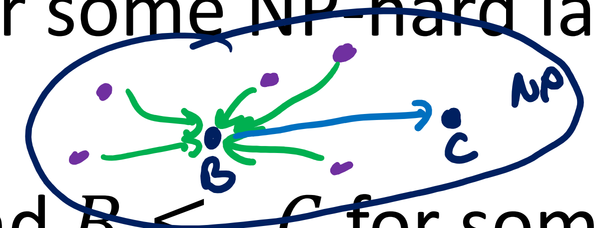
Leonid Levin (1973)

New NP-complete problems from old

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

(poly-time reducibility is <u>transitive</u>)

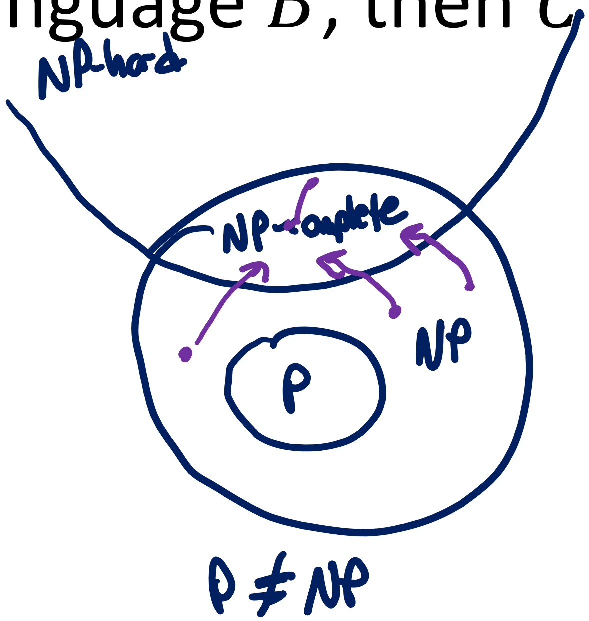
Theorem: If $B \leq_{\mathrm{p}} C$ for some NP-hard language B, then C is also NP-hard

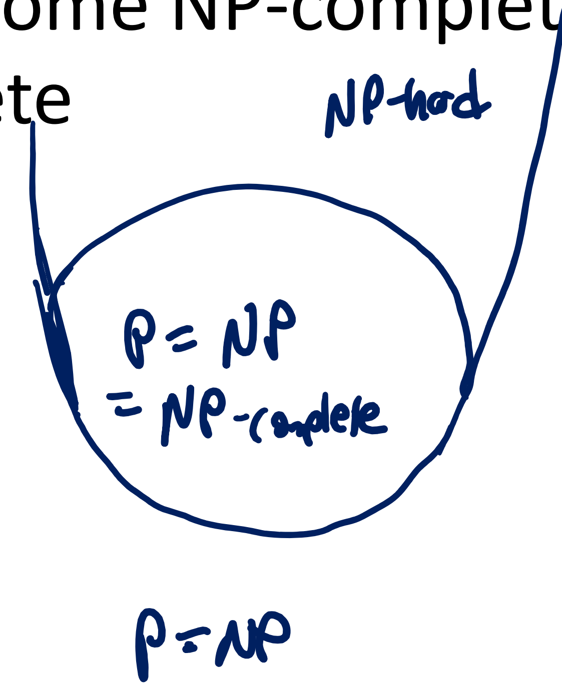


Corollary: If $C \in \mathrm{NP}$ and $B \leq_{\mathrm{p}} C$ for some NP-complete language B, then C is also NP-complete

- hard

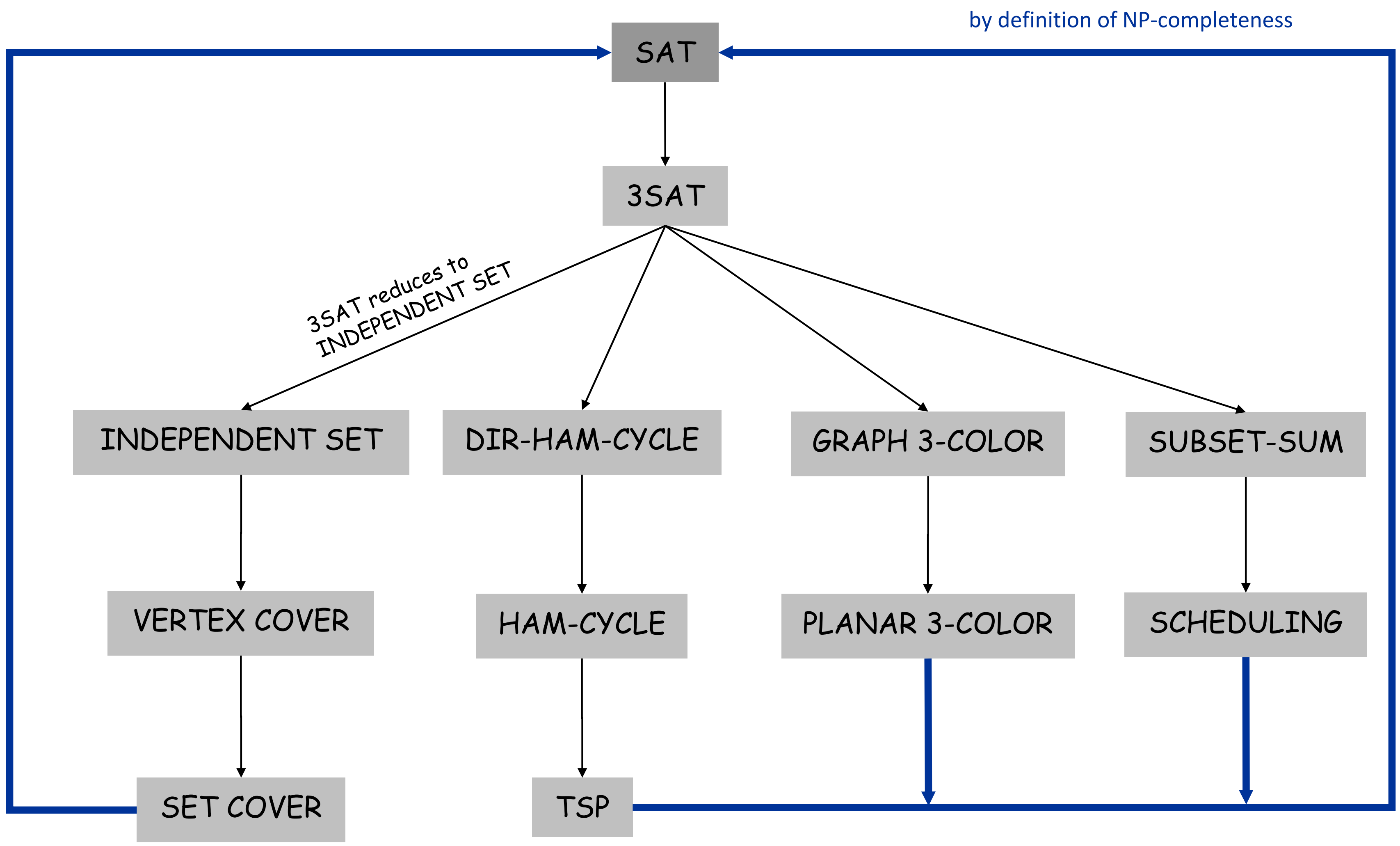
ok, but B is NP-complete wLOG





New NP-complete problems from old

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



$3SAT$ (3-CNF Satisfiability)

Definitions:

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

$$\underbrace{(x_5 \vee \overline{x_7} \vee x_2)}_{C_1} \wedge \underbrace{(\overline{x_3} \vee x_4 \vee x_1)}_{C_2} \wedge \cdots \wedge \underbrace{(x_1 \vee x_1 \vee x_1)}_{C_m}$$

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

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

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

2) Show that $SAT \leq_p 3SAT$

know SAT is NP-complete by Cook-Levin

Your classmate suggests the following reduction from SAT to $3SAT$: "On input φ, a 3-CNF formula (an instance of $3SAT$), output φ, which is already an instance of SAT." Is this reduction correct?

a) Yes, this is a poly-time reduction from SAT to $3SAT$

b) No, because φ is not an instance of the SAT problem

c) No, the reduction does not run in poly time

d) No, this is a reduction from $3SAT$ to SAT; it goes in the wrong direction

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

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

2) Show that $SAT \leq_{\mathrm{p}} 3SAT$

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

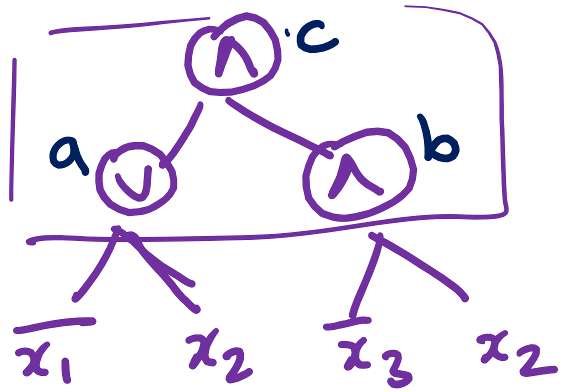
Want: $\varphi \in SAT \Longleftrightarrow \psi \in 3SAT$

Illustration of conversion from φ to ψ

"Proof by example"

1. Given φ, "push all negations to the bottom"

$$\overline{a \wedge b} = \bar{a} \vee \bar{b}$$



2. Add auxilliary variable to capture constraints of satisfiability at each gate

$$(a = \bar{x}_1 \vee x_2) \wedge (b = \bar{x}_3 \wedge x_2) \wedge (c = a \wedge b) \wedge c$$

3. replace every "pseudo constraint" of the form $\underbrace{(a = b \wedge c)}_{f(a,b,c)}$ w/ an equivalent 3CNF formula

Some general reduction strategies

- Reduction by simple equivalence

  Ex. $IND - SET \leq_{\mathrm{p}} VERTEX - COVER$

  $VERTEX - COVER \leq_{\mathrm{p}} IND - SET$

- Reduction from special case to general case

  Ex. $VERTEX - COVER \leq_{\mathrm{p}} SET - COVER$

  $3SAT \leq_{\mathrm{p}} SAT$

- "Gadget" reductions

  Ex. $SAT \leq_{\mathrm{p}} 3SAT$

  $3SAT \leq_{\mathrm{p}} IND - SET$