BU CS 332 – Theory of Computation

https://forms.gle/Mdxu6ahHoV8fbFn1A

Lecture 23:

- More NP-completeness

Reading:

Sipser Ch 7.4-7.5

Mark Bun

April 28, 2025

NP-completeness

"The hardest languages in NP"

Definition: A language B is NP-complete if

      1) $B \in$ NP, and

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

Last time:

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

Cook-Levin Theorem:
$\{\langle \varphi \rangle \mid$ Boolean formula φ is satisfiable$\}$ is NP-complete

New NP-complete problems from old

Lemma: If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$

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

Theorem: If $B \leq_p C$ for some NP-hard language B, then C is also NP-hard

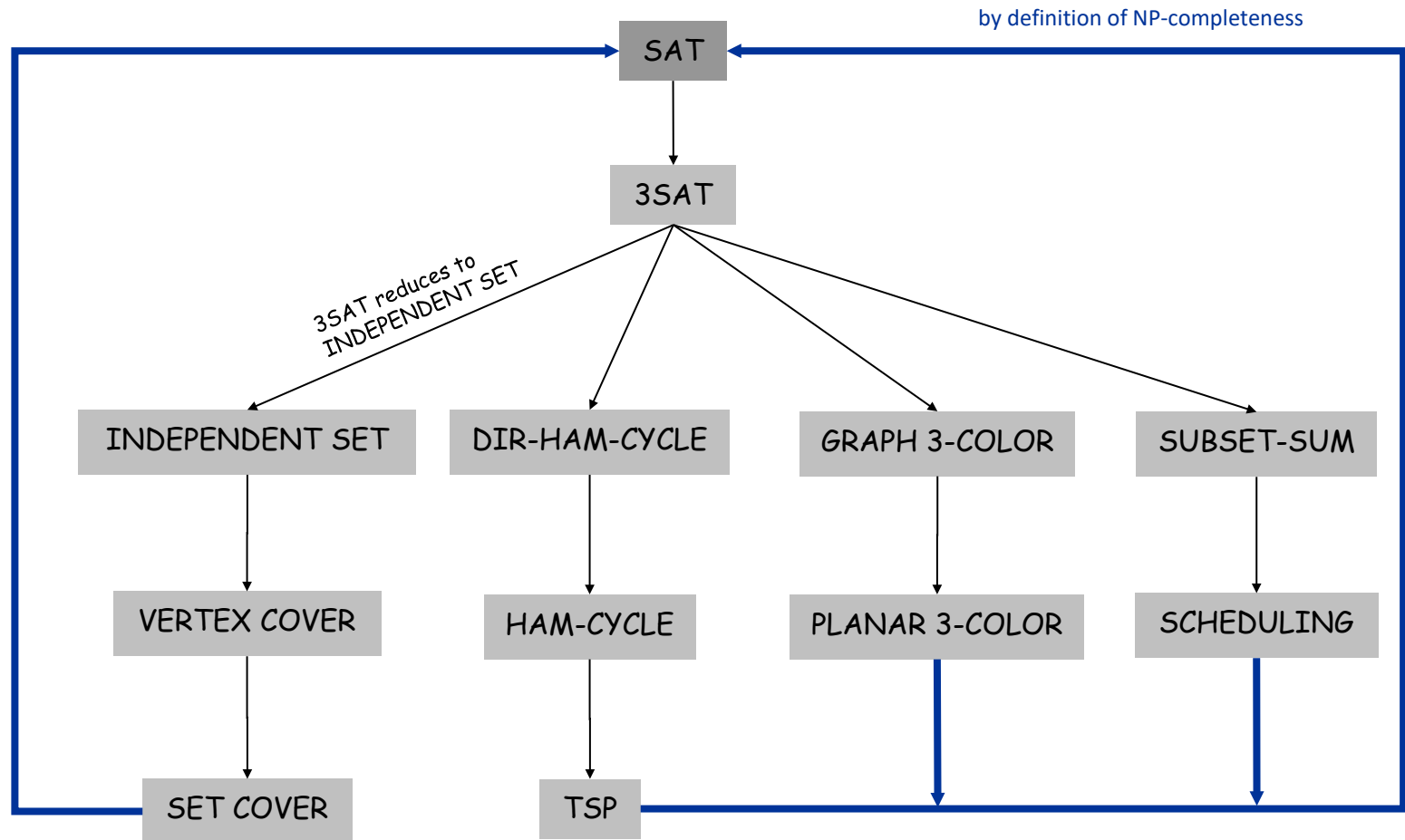The usual way to prove NP-completeness:

If

1) $C \in$ NP and

2) There is an NP-complete language B (e.g., 3SAT, VERTEX-COVER, IND-SET, …) such that $B \leq_p C$,

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!

$3SAT$ (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. $C_1 \wedge C_2 \wedge \ldots \wedge C_m =$
  $$(x_5 \vee \overline{x_7} \vee x_2) \wedge (\overline{x_3} \vee x_4 \vee x_1) \wedge \cdots \wedge (x_1 \vee x_1 \vee x_1)$$

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

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

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

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

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 $SAT \leq_p 3SAT$

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

Illustration of conversion from φ to ψ

Some general reduction strategies

- Reduction by simple equivalence

  Ex. $IND - SET \leq_p VERTEX - COVER$

  $VERTEX - COVER \leq_p IND - SET$

- Reduction from special case to general case

  Ex. $VERTEX - COVER \leq_p SET - COVER$
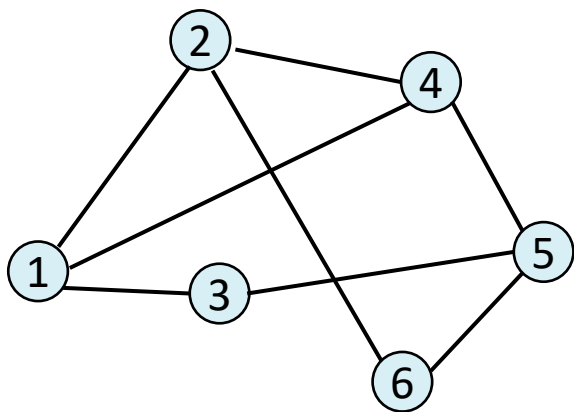
  $3SAT \leq_p SAT$

- "Gadget" reductions

  Ex. $SAT \leq_p 3SAT$

  $3SAT \leq_p IND - SET$

Independent Set

An **independent set** in an undirected graph G is a set of vertices that includes at most one endpoint of every edge.

$$IND - SET = \{\langle G, k \rangle | G \text{ is an undirected graph containing an}$$
$$\text{independent set with} \geq k \text{ vertices}\}$$

Which of the following are independent sets in this graph?

a) $\{1\}$
b) $\{1, 5\}$
c) $\{2, 3, 6\}$
d) $\{3, 4, 6\}$

Independent Set is NP-complete

1) $IND - SET \in \text{NP}$

2) Reduce $3SAT \leq_\text{p} IND - SET$

Proof of 1) The following gives a poly-time verifier for $IND - SET$

Certificate: Vertices $v_1, \dots, v_k$

Verifier:

"On input $\langle G, k; v_1, \dots, v_k \rangle$, where G is a graph, k is a natural number,

1. Check that $v_1, \dots v_k$ are distinct vertices in G

2. Check that there are no edges between the v_i's."

Independent Set is NP-complete

1) $IND - SET \in \text{NP}$

2) Reduce $3SAT \leq_\text{p} IND - SET$

Proof of 2) The following TM computes a poly-time reduction.

"On input $\langle \varphi \rangle$, where φ is a 3CNF formula,

1. Construct graph G from φ

   - G contains 3 vertices for each clause, one for each literal.
   - Connect 3 literals in a clause in a triangle.
   - Connect every literal to every appearance of its negation.

2. Output $\langle G, k \rangle$, where k is the number of clauses in φ."

Example of the reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3)$$

Proof of correctness for reduction

Let k = # clauses and l = # literals in φ

Correctness: φ is satisfiable iff G has an independent set of size k

$\Longrightarrow$ Given a satisfying assignment, select one true literal from each triangle. This is an independent set of size k

$\Longleftarrow$ Let S be an independent set in G of size k

- S must contain exactly one vertex in each triangle
- Set these literals to true, and set all other variables arbitrarily
- Truth assignment is consistent and all clauses are satisfied

Runtime: $O(k + l^2)$ which is polynomial in input size

Some general reduction strategies

- Reduction by simple equivalence
  Ex. $\boxed{\begin{array}{l} IND - SET \leq_{\mathrm{p}} VERTEX - COVER \\ VERTEX - COVER \leq_{\mathrm{p}} IND - SET \end{array}}$

- Reduction from special case to general case
  Ex. $VERTEX - COVER \leq_{\mathrm{p}} SET - COVER$
  $3SAT \leq_{\mathrm{p}} SAT$

- "Gadget" reductions
  Ex. $SAT \leq_{\mathrm{p}} 3SAT$
  $3SAT \leq_{\mathrm{p}} IND - SET$

Vertex Cover

Given an undirected graph G, a **vertex cover** in G is a subset of nodes which includes at *least* one endpoint of every edge.

$VERTEX - COVER$ = {$\langle G, k \rangle$ | G is an undirected graph which has a
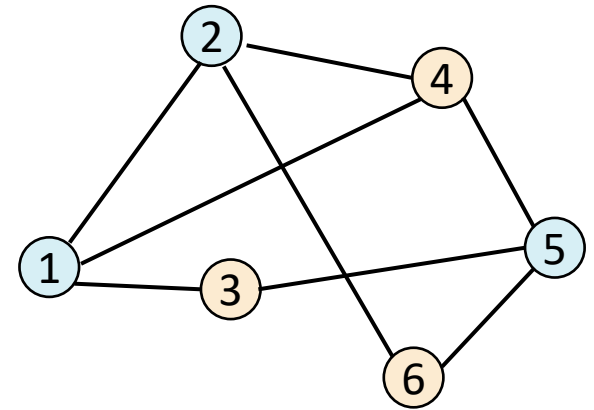
vertex cover with $\leq k$ vertices}
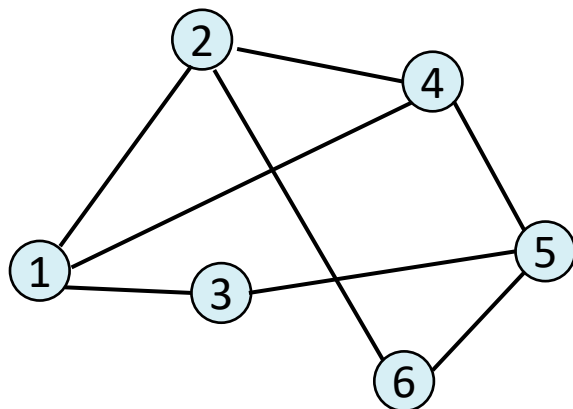
Independent Set and Vertex Cover

Claim. S is an independent set iff $V \setminus S$ is a vertex cover.

$\Longrightarrow$ Let S be any independent set.

- Consider an arbitrary edge (u, v).
- S is independent $\Longrightarrow u \notin S$ or $v \notin S \Longrightarrow u \in V \setminus S$ or $v \in V \setminus S$.
- Thus, $V \setminus S$ covers (u, v).

$\Longleftarrow$ Let $V \setminus S$ be any vertex cover.

- Consider two nodes $u \in S$ and $v \in S$.
- Then $(u, v) \notin E$ since $V \setminus S$ is a vertex cover.
- Thus, no two nodes in S are joined by an edge $\Longrightarrow S$ is an independent set.

INDEPENDENT SET reduces to VERTEX COVER

Theorem. IND-SET $\leq_p$ VERTEX-COVER.

What do we need to do to prove this theorem?

a) Construct a poly-time nondet. TM deciding IND-SET

b) Construct a poly-time deterministic TM deciding IND-SET

c) Construct a poly-time nondet. TM mapping instances of IND-SET to instances of VERTEX-COVER

d) Construct a poly-time deterministic TM mapping instances of IND-SET to instances of VERTEX-COVER

e) Construct a poly-time nondet. TM mapping instances of VERTEX-COVER to instances of IND-SET

f) Construct a poly-time deterministic TM mapping instances of VERTEX-COVER to instances of IND-SET

INDEPENDENT SET reduces to VERTEX COVER

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

Proof. The following TM computes the reduction.

"On input $\langle G, k \rangle$, where G is an undirected graph and k is an integer,

1. Output $\langle G, n-k \rangle$, where n is the number of nodes in G."

Correctness:

- G has an independent set of size at least k iff it has a vertex cover of size at most $n-k$.

Runtime:

- Reduction runs in linear time.

VERTEX COVER reduces to INDEPENDENT SET

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

Proof. The following TM computes the reduction.

"On input $\langle G, k \rangle$, where G is an undirected graph and k is an integer,

1. Output $\langle G, n - k \rangle$, where n is the number of nodes in G."

Correctness:

- G has a vertex cover of size at most k iff it has an independent set of size at least $n - k$.

Runtime:

- Reduction runs in linear time.

Subset Sum

SUBSET-SUM = $\{\langle w_1, \dots, w_m, t \rangle \mid$
there exists a subset of natural numbers $w_1, \dots, w_m$ that sum to $t\}$

Theorem: SUBSET-SUM is NP-complete

Claim 1: SUBSET-SUM is in NP

Claim 2: SUBSET-SUM is NP-hard

3SAT $\leq_p$ SUBSET−SUM

Goal: Given a 3CNF formula φ on v variables and k clauses, construct a SUBSET-SUM instance $w_1, \dots, w_m, t$ such that

φ is satisfiable   iff   there exists a subset of $w_1, \dots, w_m$ that sum to t

First attempt: Encode each literal ℓ of φ as a k-digit *decimal* number $w_\ell = c_1 \dots c_k$ where

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ \qquad 0 \text{ otherwise} \end{cases}$$

Example of the first attempt reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3)$$

3SAT $\leq_p$ SUBSET−SUM

First attempt: Encode each literal ℓ of φ as a k-digit *decimal* number $w_\ell = c_1 \ldots c_k$ where

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ \quad 0 \text{ otherwise} \end{cases}$$

Claim: If φ is satisfiable, then there exists a subset of the w_ℓ's that "digit-wise" add up to "at least" $111 \ldots 11$

Two issues:

1) Need to enforce that exactly one of $\ell, \bar{\ell}$ is set to 1

2) Need the subset to add up to exactly some target

3SAT $\leq_p$ SUBSET–SUM

Actual reduction: Encode each literal ℓ of φ as a $(v+k)$-digit *decimal* number $w_\ell = b_1 \ldots b_v \mid c_1 \ldots c_k$ where

$$b_i = \begin{cases} 1 \text{ if } \ell \in \{x_i, \overline{x_i}\} \\ \ 0 \text{ otherwise} \end{cases} \qquad c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ \qquad 0 \text{ otherwise} \end{cases}$$

Also, include two copies each of 000…0|100…0, 000…0|010…0, … 000…0|0…01

Claim: φ is satisfiable if and only if there exists a subset of the numbers that add up to $t = 111 \ldots 11 \mid 333 \ldots 33$

Example of the reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3)$$

Encode literal ℓ as $w_\ell = b_1 \ldots b_v |\ c_1 \ldots c_k$ where

$$b_i = \begin{cases} 1 \text{ if } \ell \in \{x_i, \overline{x_i}\} \\ \ 0 \text{ otherwise} \end{cases}$$

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ \quad\ 0 \text{ otherwise} \end{cases}$$

Include two copies each of $000\ldots0|100\ldots0$, $000\ldots0|010\ldots0$, $\ldots$ $000\ldots0|0\ldots01$