Intro to Theory of Computation

LECTURE 25

CS
464

Last time

- Class **NP**

Today

- Polynomial-time reductions

Adam Smith; Sofya Raskhodnikova

The classes P and NP

P is the class of languages decidable in polynomial time on a ***deterministic*** 1-tape TM:

$$\boldsymbol{P} = \bigcup_k TIME(n^k).$$

NP is the class of languages that have polynomial-time verifiers.

CS 464

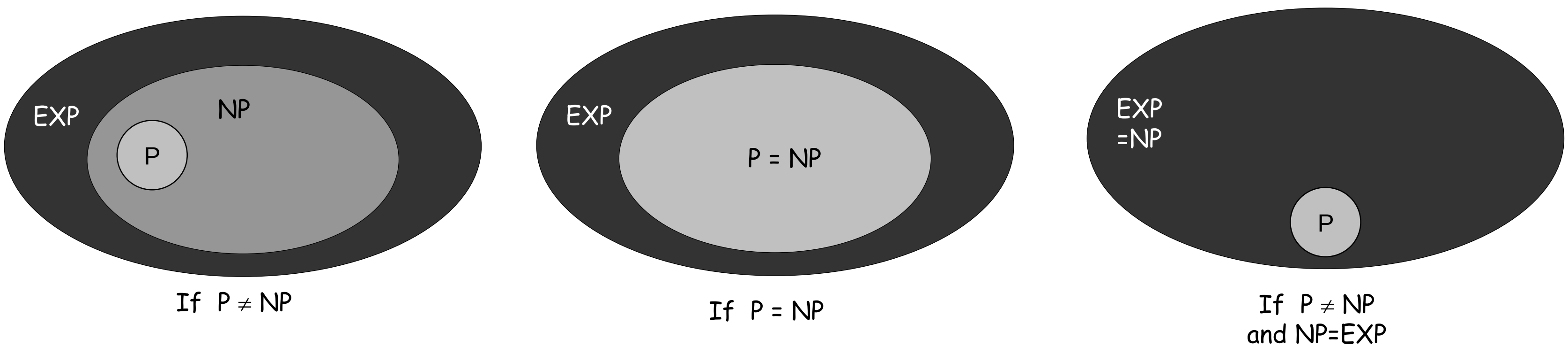
Classes P, NP, EXP

- **P**. Languages for which there is a poly-time algorithm.
- **EXP**. Languages for which there is an exponential-time algorithm.
- **NP**. Languages for which there is a poly-time verifier.

- **Lemma.** P $\subseteq$ NP.
- **Lemma.** NP $\subseteq$ EXP.
- **Lemma.** A language L is in NP iff L can be decided by a polynomial-time nondeterministic TM.

P vs. NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the verification problem?
  - Clay $1 million prize.



- If yes: Efficient algorithms for HamPath, SAT, TSP, factoring
  - Cryptography is impossible*
  - Creativity is automatable
- If no: No efficient algorithms possible for these problems.
- Consensus opinion on P = NP? Probably no.

Classify Problems

- **Desiderata:** classify problems according to those that can be solved in polynomial-time and those that cannot.
- Some problems *provably require exponential time* (next month):
  - Given a Turing machine, does it halt in at most *k* steps?
  - Given a board position in an *n*-by-*n* generalization of chess, can black guarantee a win?
- **Frustrating news:** huge number of fundamental problems have defied classification for decades.
- **Chapters 7.4-7.5 (NP-completeness):** Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one **really hard** problem.

Polynomial-time reductions

- $f: \Sigma^* \to \Sigma^*$ is polynomial-time computable if there is a poly-time TM that, on every input $w \in \Sigma^*$ halts with just $f(w)$ on its tape.
- $f: \Sigma^* \to \Sigma^*$ is a polynomial-time mapping reduction from language A to language B if:
  - f is polytime computable
  - For all $w \in \Sigma^*$: $\qquad w \in A \Leftrightarrow f(w) \in B$
- When such a reduction exists, write $A \leq_P B$

Polynomial-time reductions are the major tool we have to understand P and NP

Polynomial-time reduction

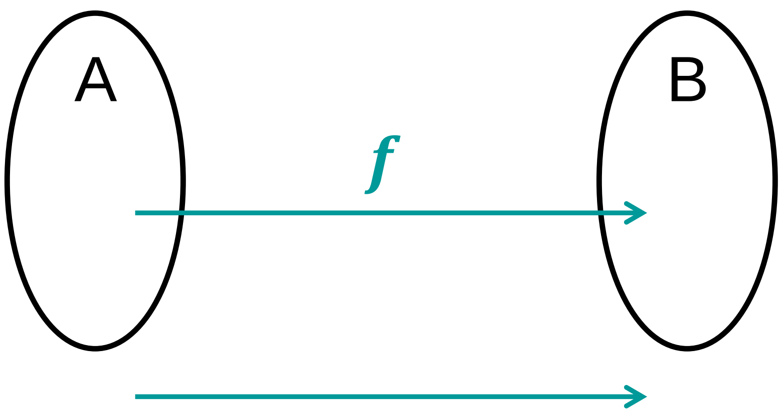
Given languages A and B,

$A \leq_p B$

if there is a ***poly-time*** computable function f,

such that for all strings w,

$w \in A$ iff $f(w) \in B$.



Implication of poly-time reductions

Theorem. If A $\leq_p$ B and B$\in$ **P** then A$\in$ **P**.

(So, if A $\leq_p$ B and A$\notin$ **P** then B$\notin$ **P.)**

Theorem. If A $\leq_p$ B and B $\leq_p$ C then A $\leq_p$ C.

(Poly-time reductions compose.)

Basic reduction strategies

Basic reduction strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

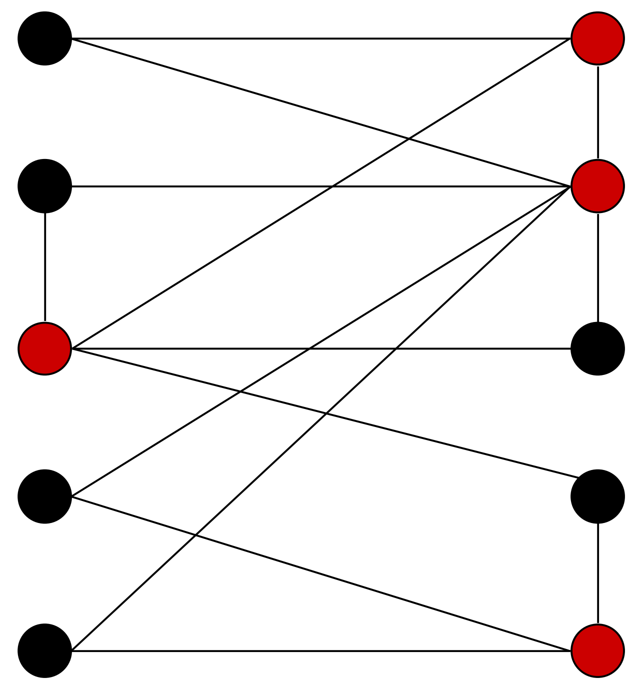
Independent Set

Given an undirected graph G, an **independent set** in G is a set of nodes, which includes at most one endpoint of every edge.

INDEPENDENT SET $= \{\langle G, k\rangle \mid G$ is an undirected graph which has an independent set with k nodes$\}$

- Is there an independent set of size ≥ 6?
  - Yes.
- Is there an independent set of size ≥ 7?
  - No.

● independent set

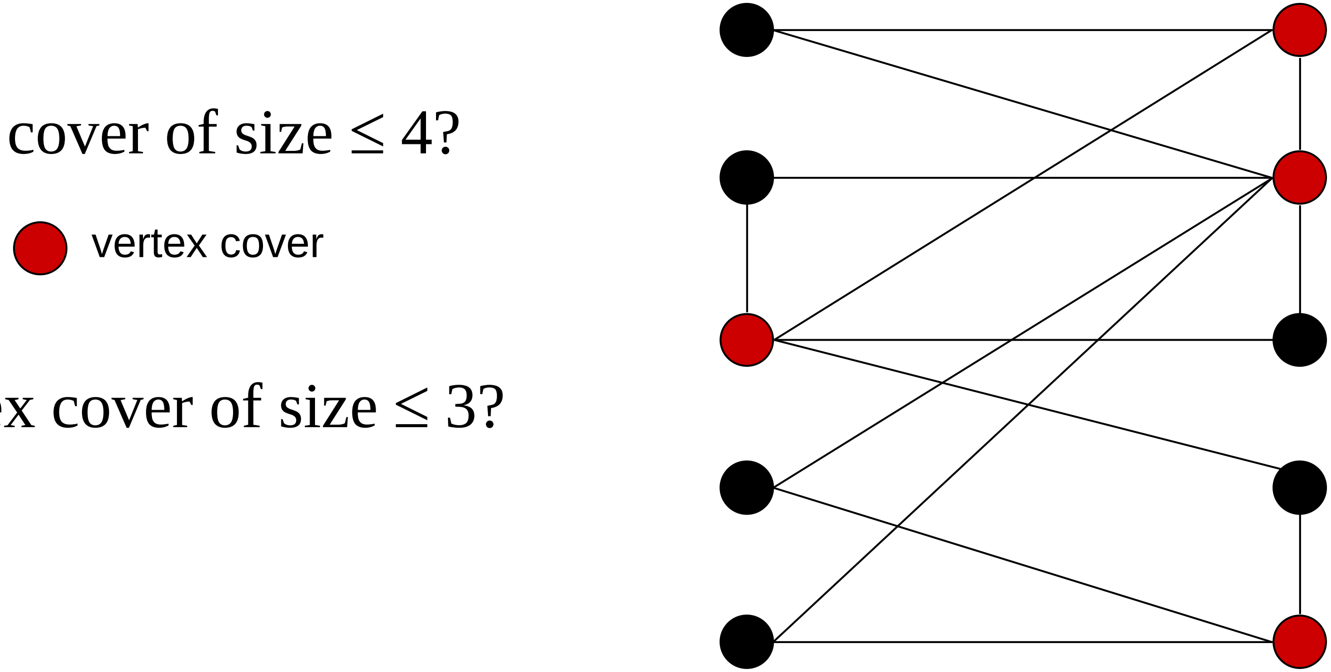


Vertex Cover

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

VERTEX COVER$=\{\langle G, k\rangle \mid G$ is an undirected graph which has a vertex cover with k nodes$\}$

- Is there vertex cover of size ≤ 4?
  - Yes.
- Is there a vertex cover of size ≤ 3?
  - No.



Independent Set and Vertex Cover

Claim. S is an independent set iff V – S is a vertex cover.

- $\Rightarrow$
  - Let S be any independent set.
  - Consider an arbitrary edge (u, v).
  - S is independent $\Rightarrow$ u $\notin$ S or v $\notin$ S $\Rightarrow$ u $\in$ V – S or v $\in$ V – S.
  - Thus, V – S covers (u, v).
- $\Leftarrow$
  - Let V – S be any vertex cover.
  - Consider two nodes u $\in$ S and v $\in$ S.
  - Then (u, v) $\notin$ E since V – S is a vertex cover.
  - Thus, no two nodes in S are joined by an edge $\Rightarrow$ S independent set. ▪

INDEPENDENT SET reduces to VERTEX COVER

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

Proof. "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 k iff it has a vertex cover of size $n - k$.
- Reduction runs in linear time.

CS 464

Reduction from special case to general case

Basic reduction strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

Given a set U, called a *universe*, and a collection of its subsets $S_1, S_2, \dots, S_m$, a **set cover** of U is a subcollection of subsets whose union is U.

- SET COVER=$\{\langle U, S_1, S_2, \dots, S_m; k\rangle \mid$ U has a set cover of size $k\}$

U = { 1, 2, 3, 4, 5, 6, 7 }

k = 2

S_1 = {3, 7}	S_4 = {2, 4}
S_2 = {3, 4, 5, 6}	S_5 = {5}
S_3 = {1}	S_6 = {1, 2, 6, 7}

- Sample application.
  - m available pieces of software.
  - Set U of n capabilities that we would like our system to have.
  - The ith piece of software provides the set $S_i \subseteq$ U of capabilities.
  - Goal: achieve all n capabilities using fewest pieces of software.

VERTEX COVER reduces to SET COVER

Theorem. VERTEX-COVER $\leq_P$ SET-COVER.

Proof. "On input $\langle G, k \rangle$, where $G = (V, E)$ is an undirected graph and k is an integer,

1. Output $\langle U, S_1, S_2, \ldots, S_m; k \rangle$, where U=E and

S_v = {e ∈ E : e incident to v }"

Correctness:

- G has a vertex cover of size k iff U has a set cover of size k.
- Reduction runs in linear time.

.

Reduction by encoding with gadgets

Basic reduction strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

CS 464 Satisfiability

- **Boolean variables:** variables that can take on values T/F (or 1/0)
- **Boolean operations:** ∨, ∧, and ¬
- **Boolean formula:** expression with Boolean variables and ops

SAT = {⟨Φ⟩ | Φ is a satisfiable Boolean formula}

- **Literal:** A Boolean variable or its negation.
- **Clause:** OR of literals.
- **Conjunctive normal form (CNF):** AND of clauses.

x_i or $\overline{x_i}$

$C_j = x_1 \vee \overline{x_2} \vee x_3$

$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

3SAT = {⟨Φ⟩ | Φ is a satisfiable Boolean CNF formula, where each clause contains exactly 3 literals}

each corresponds to a different variable

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

Yes: x_1 = true, x_2 = true x_3 = false.

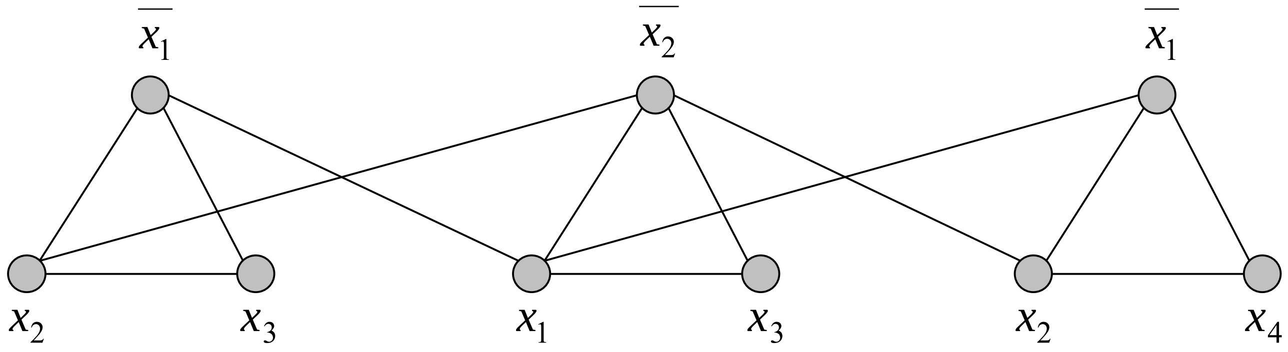
3SAT reduces to INDEPENDENT SET

Theorem. 3-SAT $\leq_P$ INDEPENDENT-SET.

Proof. "On input $\langle \Phi \rangle$, 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 literal to each of its negations.
2. Output $\langle G, k \rangle$, where k is the number of clauses in G."

G



k = 3

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3SAT reduces to INDEPENDENT SET

Correctness. Let k = # of clauses and ℓ= # of literals in Φ.

Φ is satisfiable iff G contains an independent set of size k.

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

- $\Leftarrow$ Let S be an independent set of size k.
  - S must contain exactly one vertex in each triangle.
  - Set these literals to true, and other literals in a consistent way.
  - Truth assignment is consistent and all clauses are satisfied.

Run time. O($k + \ell^2$), i.e. polynomial in the input size.

Summary

- Basic reduction strategies.
  - Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
  - Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
  - Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

- **Transitivity**. If $X \leq_P Y$ and $Y \leq_P Z$, then $X \leq_P Z$.
- Proof idea. Compose the two algorithms.
- Ex: 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.