The Classes P and NP

- P is the class of all languages that can be decided in polynomial-time by a DTM
- NP is the class of all languages that can be decided in polynomial-time by a NTM
- We do not know whether P = NP! Million dollar open problem.
- If we cannot prove that some problems are not poly-time solvable, how do we classify the computational hardness of problems?
NP Completeness
Recall: Reducibility

- Problem $A$ reduces to Problem $B$
- If we can use the solution to $B$ to solve $A$, that is,
- Solution to $B$ leads to Solution to $A$
Recall: Mapping Reducibility

\[ A \leq_m B \]
Polynomial-Time Mapping Reduction

\[ A \leq_p B \]

Input to A \( w \) \rightarrow Compute \( f(w) \) \rightarrow Decider for A

Input to B \( f(w) \) \rightarrow Decider for B

Yes \rightarrow Yes

No \rightarrow No
Polynomial-Time Reduction Implications?

- If $A \leq_p B$ and suppose $B \in \mathbb{P}$, what can we conclude about $A$?

$A \leq_p B$
Quick Exercise

Suppose $A \leq_p B$, then which of the following statements are true?

(a) If $A \in P$, then $B \in P$

(b) $A \in P$ iff $B \in P$

(c) If $A \notin P$ then $B \notin P$

(d) If $B \notin P$ then $A \notin P$
Importance of Polynomial-time Reductions

Intuitively, if \( A \leq_p B \) then solving \( A \) is no harder than solving \( B \).

**Designing algorithms.**
If \( A \leq_p B \) and \( B \) can be solved in poly-time, then \( A \) can be solved in poly-time.

**Establishing intractability.**
If \( A \leq_p B \) and \( A \) cannot be solved in polynomial time, then \( B \) cannot be solved in polynomial time.
Practice with Reductions

An **independent set** $S$ in an undirected graph $G = (V, E)$ is a subset of $V$ such that no two nodes in $S$ have an edge between them.

$$\text{INDSET} = \{<G, k> \mid G \text{ is an undirected graph with an independent set of size at least } k\}$$

Let $G$ be the graph shown.

**Questions.**

- Is $<G, 6>$ in INDSET?
- Is $<G, 7>$ in INDSET?
- Is INDSET in NP?
Vertex Cover

A vertex cover $C$ in an undirected graph $G = (V, E)$ is a subset of $V$ such that every edge in $E$ is incident on some node in $C$.

$\text{VCOVER} = \{ <G, k> \mid G \text{ is an undirected graph with a vertex cover of size at most } k \}$

Let $G$ be the graph shown.

Questions.

- Is $<G, 4>$ in $\text{VCOVER}$?
- Is $<G, 3>$ in $\text{VCOVER}$?
- Is $\text{VCOVER}$ in NP?
Reducing Vertex Cover to Independent Set

- Given an instance $< G, k >$ of vertex-cover, can you construct an instance $< G', k' >$ of independent set such that $\langle G, k \rangle \in \text{VCOVER} \iff \langle G', k' \rangle \in \text{INDSET}$
Reducing Vertex Cover to Independent Set

- Given an instance $< G, k >$ of vertex-cover, can you construct an instance $< G', k' >$ of independent set such that $\langle G', k \rangle \in \text{VCOVER} \iff \langle G', k' \rangle \in \text{INDSET}$

- Let $G' = G$, $k' = n-k$

- Claim. $\langle G, k \rangle \in \text{VCOVER} \iff \langle G, n - k \rangle \in \text{INDSET}$
Reducing Vertex Cover to Independent Set

- Given an instance \( < G, k > \) of vertex-cover, can you construct an instance \( < G', k' > \) of independent set such that

\[
\langle G', k \rangle \in \text{VCOVER} \iff \langle G', k' \rangle \in \text{INDSET}
\]

- Let \( G' = G, \ k' = n-k \)

- **Claim.** \( \langle G, k \rangle \in \text{VCOVER} \iff \langle G, n - k \rangle \in \text{INDSET} \)

- If \( G \) has a vertex-cover \( C \) of size \( k \), then \( G \) has an independent set \( S = V-C \) of size \( n-k \)

- If \( G \) has an independent set \( S \) of size \( n-k \), then \( G \) has a vertex-cover \( C = V-S \) of size \( k \)
Conclusion

- Solving vertex cover \textit{is no harder than} solving independent set

- Or if someone gave me an efficient algorithm to solve independent set, I could use it to solve vertex cover

- I want to prove that problem $X$ is hard to solve, and someone told me problem $Y$ is well-known to be hard to solve, which problem should I reduce to which?
NP-Completeness

A language $L$ is *NP-complete* if it satisfies two conditions:

- $L$ is in NP, and
- $L$ is NP-hard, that is, every language $A$ in NP is polynomial time reducible to $L$. 
NP-Completeness

A language $L$ is **NP-complete** if it satisfies two conditions:

- $L$ is in NP, and
- $L$ is NP-hard, that is, every language $A$ in NP is polynomial time reducible to $L$.

**Question.** Could we conclude anything about Independent Set if we knew that Vertex Cover is NP-complete?
First NP-Complete Problem

Satisfiability (SAT). Given a Boolean formula $B$ in conjunctive normal form (CNF), is there a truth assignment that satisfies $B$?

\[
\Phi = \left( \bar{x}_1 \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \bar{x}_2 \lor x_3 \right) \land \left( \bar{x}_1 \lor x_2 \lor x_4 \right)
\]

yes instance: $x_1 = \text{true}$, $x_2 = \text{true}$, $x_3 = \text{false}$, $x_4 = \text{false}$
Theorem. SAT is NP-complete.

Proof:

- (Easy part) SAT is in NP. Why?
- (Hard part) Every language \( A \) in NP reduces to SAT in polynomial time.
Proving NP-Completeness

To prove a given language $X$ is NP-Complete.

1. Prove $X$ is in NP.

2. Prove $X$ is NP-hard:
   
   a. Every language $A$ in NP reduces to $X$ in polynomial time
Proving NP-Completeness

To prove a given language $X$ is NP-Complete.

1. Prove $X$ is in NP.

2. Prove $X$ is NP-hard:
   a. Choose a language $Y$ that is known to be NP-complete.
   b. Show that $Y \leq_p X$, that is, give a poly-time reduction from $Y$ to $X$. 
Is \( P = NP \)?

**Theorem.** If \( A \) is NP-complete and \( A \in P \), then \( P = NP \).
Exercise

Show that Vertex Cover is in NP.

Show $\text{Clique} \leq_{p} \text{VCOVER}$.

Given instance $< G, k >$ of Clique, construct instance $< H, k' >$ of vertex cover such that...
Exercise

$$HALF-CLIQUE = \{ <G> \mid G \text{ has a clique of size } m/2 \text{ where } m \text{ is the number of nodes in } G \}$$

Show $HALF-CLIQUE$ is NP-Complete.