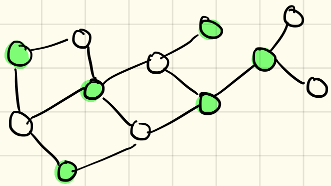
# Reduction Example: Independent Set → Vertex Cover

Def: A vertex cover of a graph is a set $S$ of nodes such that each edge has at least one endpoint in $S$

Intuitively, we try to cover all edges of the graph by choosing some set of endpoints

E.g.



Here, the green nodes constitute a vertex cover of the graph

✓

Problem: Given a graph $G$ and a number $k$, does $G$ contain a vertex cover of size at most $k$?

↑ smaller covers are harder to find

Problem (Independent Set): Given a graph $G$ and a number $k$, does $G$ contain a set of at least $k$ independent vertices?

↑ larger independent sets are harder to find

Can we reduce Independent Set to Vertex Cover?

# Relationship between Independent Set and vertex Cover

**Theorem:** If $G=(V,E)$ is a graph, then $S \subseteq V$ is an independent set $\iff$ $V-S$ is a vertex cover.

**Proof:** $\Rightarrow$ Suppose $S$ is an independent set, and let $e=(u,v)$ be some edge. Only one of $u,v$ can be in $S$, hence at least one of $u,v \in V-S$. So $V-S$ is a vertex cover.

**Proof:** $\Leftarrow$ Suppose $V-S$ is a vertex cover and let $u,v \in S$. There can't be an edge between $u$ and $v$ (otherwise that edge wouldn't be covered in $V-S$). So, $S$ is an independent set.

# Independent Set $\leq_p$ Vertex Cover

Given an arbitrary instance of Independent Set $\langle G, k \rangle$

- Ask vertex cover algorithm if there is a vertex cover $V-S$ of size $\leq |V|-k$

By $S$ is an IS iff $V-S$ is a VC

If the VC algo said yes: $S$ must be IS $\geq k$
no: There is no VC of size $\leq |V|-k$, hence no IS of size $\geq k$.

Actually, we also have VC $\leq_p$ IS

Reduction: To decide if $G$ has a VC of size $k$, ask if it has an IS of size $n-k$.

So VC and IS are equivalently difficult.

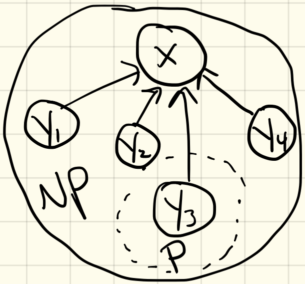
NP-completeness: Now, we can define what it means for a problem to be NP-complete.

Def: We say $X$ is NP-complete if:

1) $X \in NP$
2) For all $Y \in NP$, $Y \leq_p X$

If these hold, an algorithm for $X$ could be used to solve all problems in NP.

So, $X$ is at least as hard as any problem in NP.



Theorem: If X is NP-complete, then X is solvabale in polynomial time iff P = NP

Proof: If P = NP, X is solvable in polynomial time

Suppose X is solvabale in poly-time, let Y be any problem in NP. We can solve Y in poly-time by reduction to X.

Therefore, every problem in NP would have a poly-time algo and we would have P = NP.

All of this relies on having some first NPC problem. Finding that first problem is the result of the Cook-Levin theorem (will mention briefly later).

For now, let's look at another reduction.

Problem (Set Cover): Given a universe (set) $U$ of elements and a collection $S_1, \dots, S_m$ of subsets of $U$, is there a collection of at most $K$ of these subsets whose union equals $U$?

Goal: Show that Set Cover is NP-complete. To show that we need to show

1) Set Cover $\in$ NP
2) Some NP-complete problem reduces to Set Cover (we'll use vertex cover)

For 1, consider the collection of $\leq K$ subsets as the certificate. Clearly, we can verify a set cover instance in poly-time.

Theorem: Vertex Cover $\leq_p$ Set Cover

Proof: Let $\langle G=(V,E), K \rangle$ be an arbitrary instance of Vertex cover. Create the following instance of set cover:

- $U = E$
- Create a subset $S_i$ for all $i \in V$ where $S_i$ contains the edges adjacent to vertex $i$.

$U$ can be covered by $\leq K$ sets iff $G$ has a VC of size $\leq K$. Why?

$\Rightarrow$ Let $S_1, \ldots, S_j$ be a set cover of size $\leq k$. Then select vertices $1, \ldots, j$ in $G$. By the construction of our set cover instance, they constitute a VC of $G$. Since every $u \in \mathcal{U}$ is covered and $\mathcal{U} = E$ then every $e \in E$ must be adjacent to a chosen vertex.

$\Leftarrow$ Let $G$ have a vertex cover of size $\leq k$, and let the set of vertices be given by $C^*$. Since $C^*$ is a VC, every $e \in E$ is adjacent to some $v \in C^*$. However, by our reduction, we have $\mathcal{U} = E$ and we also have that for all $e$ adjacent to $i \in V$ then $e \in S_i$ in our set cover instance. Hence $\bigcup_{i \in C^*} S_i = \mathcal{U}$.

Summary: To show a problem is NP-complete, you must show it is in NP, and must reduce a known NP-complete problem <u>to</u> your new problem.

## Some more problems

### Boolean Formulas:

Variables: $x_1, x_2, \ldots$ (can be either true or false)

Terms: $t_1, t_2, \ldots$: $t_j$ is either $x_j$ or $\overline{x_j}$ (i.e. either $x_j$ or not $x_j$)

Clauses: $t_1 \vee t_2 \vee \ldots \vee t_\ell$: ($\vee$ stands for "or"). a clause is true if any of its terms are true

E.g. $(x_1 \vee \overline{x}_2)$, $(\overline{x}_1 \vee \overline{x}_3)$, $(x_2 \vee \overline{x}_3)$, $(x_1 \vee x_2 \vee \overline{x}_3)$

Def: A truth assignment is a choice of true or false for each variable i.e. a function

$$v: X \to \{\text{true}, \text{false}\}$$

Def: A Conjunctive Normal Form (CNF) formula is a conjunction (and-ing) of clauses:

$$C_1 \wedge C_2 \wedge \ldots \wedge C_k$$

Def: A truth assignment is a <u>satisfying assignment</u> for such a formula if it makes every clause <u>true</u>.

SAT and 3-SAT

Problem [Satisfiability (SAT)]: Given a set of clauses $C_1, \dots, C_k$ over variables $X = \{x_1, \dots, x_n\}$, is there a satisfying assignment?

Problem [3-SAT]: Given a set of clauses $C_1, \dots, C_k$, each of length 3 (i.e. containing 3 terms), over variables $X = \{x_1, \dots, x_n\}$, is there a satisfying assignment?

<u>Cook-Levin Theorem</u> shows that SAT is NP-complete.
Richard Karp showed (1972) that SAT $\leq_p$ 3-SAT. He, in fact, showed via reduction, the NP-completeness of 21 different problems.

The Garey and Johnson text "Computers and Intractability" shows $> 300$ NP-complete problems.

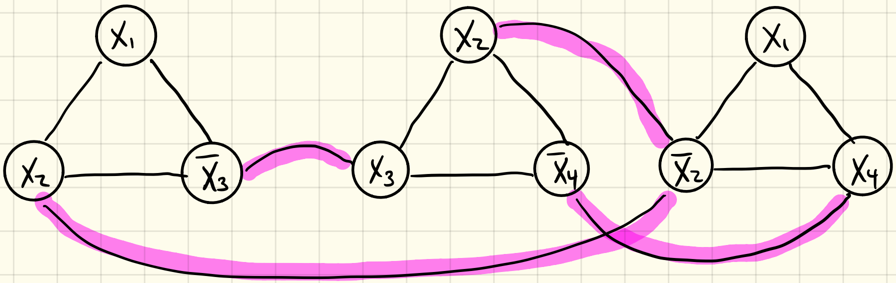
The CL theorem gives us the first "hook" on which to hang new NPC proofs (reductions).

Anoter (non-covering) reduction.

**Theorem**: 3-SAT $\leq_p$ Independent Set

**Proof**: Consider the following mapping from clauses in a 3-SAT instance to a graph

$$(X_1 \vee X_2 \vee \overline{X_3}) \wedge (X_2 \vee X_3 \vee \overline{X_4}) \wedge (X_1 \vee \overline{X_2} \vee X_4)$$




The — edges are called "conflict links". We just draw them like this; in G they are regular edges.

That is, we create a triangle for each clause where the vertices are labeled with the terms and there are edges between all terms in a clause. Additionally, we add an edge between each vertex labeled with a term and each instance of a vertex labeled with the negation of that term (e.g. $X_2$ — $\overline{X_2}$)

Claim: This graph has an IS of size $\geq k$ iff the formula is satisfiable.

Proof: $\Rightarrow$ If formula is satisfiable, there is at least 1 true literal in each clause. Let S be the set of one such literal from each clause. $|S| = k$ and no two nodes in S are connected by an edge.

$\Leftarrow$ If the graph has an IS S of size k, we know that it has 1 node from each "clause triangle" (since we can have at most 1 node chosen from each fully connected triangle in S). Set those terms to <u>true</u>. This is possible because no two terms in S are negations of eachother (because of conflict links). ∎

General Proof Strategy for Showing a problem is NP-complete:

1) Show $X \in NP$ by finding an efficient certifier
2) Look for some known NP-complete problem (there are many) $Y$ that seems "similar" to your problem $X$ in some way.

3) Show that $Y \leq_p X$

One way to show $Y \leq_p X$:

1) Let $I_y$ be an <u>arbitrary</u> instance of problem $Y$

2) Show how to construct an instance $I_x$ of your problem $X$ in <u>polynomial time</u> such that:

- if $I_y \in Y$ then $I_x \in X$
- if $I_x \in X$ then $I_y \in Y$

} <u>Need both</u>

Striking dichotomy between which problems are NP-complete vs in P

| NP-Complete | in P |
|---|---|
| 3-SAT | 2-SAT |
| TSP | MST |
| Longest Path | Shortest Path |
| 3D matching | Bipartite Matching |
| Knapsack | Unary Knapsack |
| Independent Set | Independent Set on trees |
| Integer Linear Programming | Linear Programming |
| Hamiltonian Path | Eulerian Path |
| Balanced Cut | Minimum Cut |