Outline

Polynomial Time Reductions

NP Completeness
Problem 1: Clique

Definition
A clique of size $k$ in a graph $G$ is a completely connected subgraph of $G$ with $k$ vertices.

- Input: Given graph $G = (V, E)$ and integer $k$.
- Question: Does $G$ contain a clique of size $k$?
Problem 2: 3-SAT

► Input: A boolean formula \( \phi(x_1, \ldots, x_n) \) in \textit{conjunctive normal form}\footnote{This means the formula is a conjunction (AND) of clauses, each of which is a disjunction (OR) of literals.}, with \( m \) clauses and 3 literals per clause, e.g.,

\[
(x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2 \lor \bar{x}_3)
\]

where \( \bar{x}_i \) is “not \( x_i \)”, \( \land \) is “and”, \( \lor \) is “or.” We call \( x_i \) and \( \bar{x}_i \) \textit{literals}.

► Question: Is there a setting of each \( x_i \) to TRUE or FALSE such that the formula is satisfied.
A Polynomial Time Reduction for 3-SAT to Clique

We’ll show that if you have a polynomial time algorithm for Clique, then you also have a polynomial time algorithm for 3-SAT.

Given formula 3-SAT

$$\phi = (l_{1,1} \lor l_{1,2} \lor l_{1,3}) \land (l_{2,1} \lor l_{2,2} \lor l_{2,3}) \land \ldots \land (l_{m,1} \lor l_{m,2} \lor l_{m,3})$$

in poly-time, we can construct $G_\phi = (V_\phi, E_\phi)$:

$$V_\phi = \{l_{i,j} : i \in [m], j \in [3]\}$$

$$E_\phi = \{(l_{i,j}, l_{k,l}) : i, k \in [m], j \in [3], i \neq k, l_{i,j} \neq \bar{l}_{k,l}\}$$

We’ll show $\phi$ is satisfiable iff $G_\phi$ has a clique of size $m$
φ is satisfiable iff $G_\phi$ has a clique of size $m$

Suppose $\phi$ is satisfiable:
1. In a satisfying assignment, at least one literal is true in each clause
2. Pick one true literal per clause: let $Y$ be set of corresponding nodes
3. $G_\phi[Y]$ is a clique because $x_k$ and $\bar{x}_k$ can’t both be in $Y$ for any $k$

Suppose $G_\phi$ has a clique of size $m$:
1. Let $Y$ be the clique of size $m$
2. For each clause:
   - Exactly one node $l$ from $i$-th clause is in $Y$
   - Set $x_k = \text{TRUE}$ if $l = x_k$ and set $x_k = \text{FALSE}$ if $l = \bar{x}_k$
3. We can’t set $x_k$ to be true and false because literals $x_k$ and $\bar{x}_k$ can’t both be in $Y$
Polynomial Time Reduction

Definition
Π is a decision problem if it only has a “yes” or “no” answer.

Definition
Given two decision problems Π₁, Π₂ we say Π₂ is polynomial time reducible to Π₁ iff there exists a polynomial time algorithm f that transforms any instance X of Π₂ to an instance f(X) of Π₁ such that:

(X is a “yes” instance of Π₂) ⇐⇒ (f(X) is a “yes” instance of Π₁)

We write Π₂ \leq_p Π₁ to denote “Π₂ is polynomial time reducible to Π₁”.

Some Examples:
- INDEPENDENT-SET \leq_p CLIQUE
- VERTEX-COVER \leq_p SET-COVER
- VERTEX-COVER \leq_p INDEPENDENT-SET
Outline

Polynomial Time Reductions

NP Completeness
P and NP Definitions

**Definition (P)**

\( \Pi \in P \) iff there exists a polynomial time algorithm \( A \) such that:

\[
(X \text{ is a "yes" instance of } \Pi) \iff (A(X) = "yes")
\]

**Definition (NP)**

\( \Pi \in NP \) iff there exists a polynomial time algorithm \( A \) such that:

\[
(X \text{ is a "yes" instance of } \Pi) \implies (\exists Y : |Y| = \text{poly}(|X|), A(X, Y) = "yes")
\]

\[
(X \text{ is a "no" instance of } \Pi) \implies (\nexists Y : |Y| = \text{poly}(|X|), A(X, Y) = "yes")
\]

We call \( Y \) a **witness**.
Example: Clique

- **Input:** Given graph $G = (V, E)$ and integer $k$.
- **Question:** Does $G$ contain a clique of size $k$?

**Lemma**

*Clique is in NP.*

**Proof.**

1. Suppose the witness $Y$ encodes a set of $k$ nodes in $V$ and $A(G, Y)$ checks if the induced graph on $Y$, $G[Y]$ is a clique.
2. $A$ is a polynomial time algorithm.
3. If there exists a clique of size $k$, there exists $Y$ of size $k$ such that $A(G, Y)$ outputs “yes”
4. If there doesn’t exist a clique of size $k$, there doesn’t exist $Y$ of size $k$ such that $A(G, Y)$ outputs “yes”

Example for a problem that is not known to be in NP: Is a quantified boolean formula, e.g., $\forall x \exists y \exists z \; ((x \lor z) \land y)$, true?
NP-Completeness

Definition
A decision problem \( \Pi \) is NP-Hard iff for all \( \Pi' \in NP \), \( \Pi' \leq_P \Pi \).

Definition
A decision problem \( \Pi \) is NP-Complete iff it is both NP-Hard and in NP.

Remark 1: If \( \Pi \) is NP-Complete and \( \Pi \in P \) then \( P = NP \)

Remark 2: In 1971, Cook showed 3-SAT is NP-Complete. Because

\[ \text{CLIQUE} \in NP \text{ and } 3\text{-SAT} \leq_P \text{CLIQUE} \]

we now know CLIQUE is NP-Complete.