Lecture 22: NP-Completeness II

Michael Dinitz

November 11, 2021
601.433/633 Introduction to Algorithms
Introduction

Last time: Definition of $\mathbf{P}$, $\mathbf{NP}$, reductions, $\mathbf{NP}$-completeness. Proof that Circuit-SAT is $\mathbf{NP}$-complete.

Today: more $\mathbf{NP}$-complete problems.

Definition

A decision problem $\mathbf{Q}$ is in $\mathbf{NP}$ (nondeterministic polynomial time) if there exists a polynomial time algorithm $\mathbf{V(I,X)}$ (called the verifier) such that

1. If $\mathbf{I}$ is a YES-instance of $\mathbf{Q}$, then there is some $\mathbf{X}$ (usually called the witness, proof, or solution) with size polynomial in $|\mathbf{I}|$ so that $\mathbf{V(I,X)} = \text{YES}$.
2. If $\mathbf{I}$ is a NO-instance of $\mathbf{Q}$, then $\mathbf{V(I,X)} = \text{NO}$ for all $\mathbf{X}$. 
Reducions

Definition

A Many-one or Karp reduction from \( A \) to \( B \) is a function \( f \) which takes arbitrary instances of \( A \) and transforms them into instances of \( B \) so that

1. If \( x \) is a YES-instance of \( A \) then \( f(x) \) is a YES-instance of \( B \).
2. If \( x \) is a NO-instance of \( A \) then \( f(x) \) is a NO-instance \( B \).
3. \( f \) can be computed in polynomial time.

Definition

Problem \( Q \) is \textbf{NP-hard} if \( Q' \leq_p Q \) for all problems \( Q' \) in \textbf{NP}. Problem \( Q \) is \textbf{NP-complete} if it is \textbf{NP-hard} and in \textbf{NP}.
Circuit-SAT

**Definition**

*Circuit-SAT*: Given a boolean circuit of AND, OR, and NOT gates, with a single output and no loops (some inputs might be hardwired), is there a way of setting the inputs so that the output of the circuit is 1?

**Theorem**

*Circuit-SAT* is **NP-complete**.
3-SAT

Boolean formula:
- Boolean variables $x_1, \ldots, x_n$
- Literal: variable $x_i$ or negation $\bar{x}_i$
- AND: $\land$ OR: $\lor$
- $x_1 \lor (\bar{x}_5 \land x_7) \land (\bar{x}_2 \lor (x_6 \land \bar{x}_3)) \ldots$

Conjunctive normal form (CNF): AND of ORs (clauses)
- $(x_1 \lor \bar{x}_2 \lor x_4) \land (x_2 \lor x_3) \land (x_1 \lor x_4 \lor \bar{x}_6) \ldots$
3-SAT

Boolean formula:
- Boolean variables $x_1, \ldots, x_n$
- Literal: variable $x_i$ or negation $\bar{x}_i$
- AND: $\land$  OR: $\lor$
- $x_1 \lor (\bar{x}_5 \land x_7) \land (\bar{x}_2 \lor (x_6 \land \bar{x}_3)) \ldots$

Conjunctive normal form (CNF): AND of ORs (clauses)
- $(x_1 \lor \bar{x}_2 \lor \bar{x}_4) \land (x_2 \lor x_3) \land (x_1 \lor x_4 \lor \bar{x}_6) \ldots$
3-SAT

Boolean formula:
- Boolean variables $x_1, \ldots, x_n$
- Literal: variable $x_i$ or negation $\bar{x}_i$
- AND: $\land$ OR: $\lor$
- $x_1 \lor (\bar{x}_5 \land x_7) \land (\bar{x}_2 \lor (x_6 \land \bar{x}_3)) \ldots$

Conjunctive normal form (CNF): AND of ORs (clauses)
- $(x_1 \lor \bar{x}_2 \lor \bar{x}_4) \land (x_2 \lor x_3) \land (x_1 \lor x_4 \lor \bar{x}_6) \ldots$

Definition

**3-SAT**: Instance is 3CNF formula $\phi$ (every clause has $\leq 3$ literals). YES if there is assignment where $\phi$ evaluates to True (satisfying assignment), NO otherwise.
3-SAT

Theorem

3-SAT is \textbf{NP-complete}.
3-SAT

**Theorem**

3-SAT is **NP-complete**.

3-SAT in **NP**: 

- A witness is an assignment.
- The verifier checks that the formula evaluates to True on the assignment.

3-SAT is **NP-hard**: 

- Show Circuit-SAT \( \leq_p \) 3-SAT.
- Don't need to show that \( A \leq_p 3\text{-SAT} \) for arbitrary \( A \in \text{NP} \): already known that \( A \leq_p \text{Circuit-SAT} \! \).
- So start with circuit. Want to transform to 3-CNF formula.
3-SAT

Theorem

3-SAT is NP-complete.

3-SAT in NP: witness is assignment, verifier checks that formula evaluates to True on assignment.
3-SAT

Theorem

3-SAT is \textbf{NP-complete}.

3-SAT in \textbf{NP}: witness is assignment, verifier checks that formula evaluates to True on assignment.

3-SAT is \textbf{NP-hard}:
3-SAT

Theorem
3-SAT is \textbf{NP}-complete.

3-SAT in \textbf{NP}: witness is assignment, verifier checks that formula evaluates to True on assignment.

3-SAT is \textbf{NP}-hard: Show Circuit-SAT ≤_p 3-SAT.
3-SAT

Theorem

3-SAT is \textbf{NP-complete}.

3-SAT in \textbf{NP}: witness is assignment, verifier checks that formula evaluates to True on assignment.

3-SAT is \textbf{NP-hard}: Show Circuit-SAT $\leq_p$ 3-SAT.

\begin{itemize}
  \item Don’t need to show that $A \leq_p$ 3-SAT for arbitrary $A \in \textbf{NP}$: already know that $A \leq_p$ Circuit-SAT!
\end{itemize}
3-SAT

Theorem

3-SAT is \textbf{NP}-complete.

3-SAT in \textbf{NP}: witness is assignment, verifier checks that formula evaluates to True on assignment.

3-SAT is \textbf{NP}-hard: Show Circuit-SAT $\leq_p$ 3-SAT.

$\rightarrow$ Don’t need to show that $A \leq_p$ 3-SAT for arbitrary $A \in \textbf{NP}$: already know that $A \leq_p$ Circuit-SAT!

So start with circuit. Want to transform to 3-CNF formula.
Transformation to NANDs

For simplicity, transform into a circuit with one type of gate: NAND (NOT AND)

- AND/OR/NOT universal, but so is just NAND!

\[
\begin{align*}
&\text{AND} \quad \Rightarrow \quad \text{NAND} \\
&\text{OR} \quad \Rightarrow \quad \text{NAND} \\
&\text{NOT} \quad \Rightarrow \quad \text{NAND}
\end{align*}
\]
Transformation to NANDs

For simplicity, transform into a circuit with one type of gate: NAND (NOT AND)

- AND/OR/NOT universal, but so is just NAND!

So given circuit $C$, first transform it into NAND-only circuit.

Input:

- $n$ "input wires" $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$
  - $g_1 = \text{NAND}(x_1, x_3)$,
  - $g_2 = \text{NAND}(g_1, x_4)$, \ldots
  - WLOG, $g_m$ is the “output gate”
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.
Reduction to 3-SAT

So given as input a circuit \( C \):

- \( n \) “input wires” \( x_1, x_2, \ldots, x_n \)
- \( m \) NAND gates: \( g_1, \ldots, g_m \). Output gate \( g_m \)

Need to construct many-one reduction \( f \) to 3-SAT: in polynomial time, construct 3-CNF formula \( f(C) \) such that \( f(C) \) has a satisfying assignment if and only if \( C \) has an input where it outputs 1.

Variables: \( y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m} \) (one for each wire)
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

\[
\begin{align*}
\overline{y_i} &\lor y_j \lor \overline{y_k} & & \text{(if } y_j = 0 \text{ and } y_k = 0 \text{ then } y_i = 1) \\
\overline{y_i} &\lor \overline{y_j} \lor y_k & & \text{(if } y_j = 1 \text{ and } y_k = 0 \text{ then } y_i = 1) \\
\overline{y_i} &\lor y_j \lor \overline{y_k} & & \text{(if } y_j = 0 \text{ and } y_k = 1 \text{ then } y_i = 1) \\
\end{align*}
\]
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

\[
\begin{align*}
&y_i \\
&\quad \lor y_j \\
&\quad \lor \bar{y}_k \\
&\quad \lor y_j \\
&\quad \lor \bar{y}_k \\
&\quad \lor y_i \\
&\quad \lor \bar{y}_j \\
&\quad \lor \bar{y}_k
\end{align*}
\]
Reduction to 3-SAT

So given as input a circuit \( C \):

- \( n \) “input wires” \( x_1, x_2, \ldots, x_n \)
- \( m \) NAND gates: \( g_1, \ldots, g_m \). Output gate \( g_m \)

Need to construct many-one reduction \( f \) to 3-SAT: in polynomial time, construct 3-CNF formula \( f(C) \) such that \( f(C) \) has a satisfying assignment if and only if \( C \) has an input where it outputs 1.

**Variables:** \( y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m} \) (one for each wire)

**Clauses:** For every NAND gate \( y_i = \text{NAND}(y_j, y_k) \), create clauses:

\[
\begin{align*}
y_i &\lor y_j \lor y_k \quad \text{if } y_j = 0 \text{ and } y_k = 0 \text{ then } y_i = 1 \\
y_{n+i} &\lor y_{n+j} \lor y_{n+k} \quad \text{if } y_j = 1 \text{ and } y_k = 1 \text{ then } y_i = 0
\end{align*}
\]
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \bar{y}_j \lor y_k$
- $\bar{y}_i \lor \bar{y}_j \lor \bar{y}_k$ (if $y_j = 1$ and $y_k = 1$ then $y_i = 0$)
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

\[
\begin{align*}
&y_i \lor y_j \lor y_k \quad \text{(if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)} \\
&y_i \lor \overline{y}_j \lor y_k \quad \text{(if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)}
\end{align*}
\]
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

Variables: $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

Clauses: For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \neg y_j \lor y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor y_j \lor \neg y_k$
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \bar{y}_j \lor y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor y_j \lor \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \bar{y}_j \lor y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor y_j \lor \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
- $\bar{y}_i \lor \bar{y}_j \lor \bar{y}_k$
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \bar{y}_j \lor y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor y_j \lor \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
- $\bar{y}_i \lor \bar{y}_j \lor \bar{y}_k$ (if $y_j = 1$ and $y_k = 1$ then $y_i = 0$)
Reduction to 3-SAT

So given as input a circuit $C$:

- $n$ “input wires” $x_1, x_2, \ldots, x_n$
- $m$ NAND gates: $g_1, \ldots, g_m$. Output gate $g_m$

Need to construct many-one reduction $f$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $C$ has an input where it outputs 1.

**Variables:** $y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}$ (one for each wire)

**Clauses:** For every NAND gate $y_i = \text{NAND}(y_j, y_k)$, create clauses:

- $y_i \lor y_j \lor y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor \overline{y_j} \lor y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \lor y_j \lor \overline{y_k}$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
- $\overline{y_i} \lor \overline{y_j} \lor \overline{y_k}$ (if $y_j = 1$ and $y_k = 1$ then $y_i = 0$)

Also add clause $(y_{m+n})$ (want output gate to be 1)
Analysis

Theorem

*This is a many-one reduction from Circuit-SAT to 3-SAT.*
Analysis

**Theorem**

*This is a many-one reduction from Circuit-SAT to 3-SAT.*

Polytime: ✓
Analysis

Theorem

This is a many-one reduction from Circuit-SAT to 3-SAT.

Polytime: ✓

Suppose $C$ YES of Circuit-SAT

$\implies$ There exists a setting $x$ of input wires so $g_m = 1$

$\implies$ There exists an assignment of $y_1, \ldots, y_{m+n}$ so that all clauses are satisfied:

$\cdot$ $y_i = x_i$ if $i \leq n$

$\cdot$ $y_i = g_{i-n}$ if $i > n$

$\implies$ $f(C)$ YES of 3-SAT
Analysis

Theorem

This is a many-one reduction from Circuit-SAT to 3-SAT.

Polytime: ✓

Suppose \( C \) YES of Circuit-SAT

\[\implies \exists \text{ setting } x \text{ of input wires so } g_m = 1\]

\[\implies \exists \text{ assignment of } y_1, \ldots, y_{m+n} \text{ so that all clauses are satisfied:}\]

\[y_i = x_i \text{ if } i \leq n\]

\[y_i = g_{i-n} \text{ if } i > n\]

\[\implies f(C) \text{ YES of 3-SAT}\]

Suppose \( f(C) \) YES of 3-SAT

\[\implies \exists \text{ assignment } y \text{ to variables so that all clauses satisfied}\]

\[\implies \exists \text{ setting } x \text{ of input wires so } g_m = 1:\]

\[x_i = y_i\]

\[\text{Output of gate } g_i = y_{i+n} \text{ (by construction)}\]

\[\text{So } g_m = 1 \text{ (since } (y_{m+n}) \text{ is a clause)}\]

\[\implies C \text{ a YES instance of Circuit-SAT}\]
General Methodology to Prove \textbf{Q NP}-Complete

1. Show \textbf{Q} is in \textbf{NP}
   - Can verify witness for YES
   - Can catch false witness for NO (or contrapositive: if witness is verified, then a YES instance)

2. Find some \textbf{NP}-hard problem \textbf{A}. Reduce \textit{from} \textbf{A} \textit{to} \textbf{Q}:
   - Given instance \textbf{I} of \textbf{A}, turn into \textbf{f}(\textbf{I}) of \textbf{Q} (in time polynomial in $|\textbf{I}|$)
   - \textbf{I} YES of \textbf{A} if and only if \textbf{f}(\textbf{I}) YES of \textbf{Q}
General Methodology to Prove \( Q \) NP-Complete

1. Show \( Q \) is in \( \text{NP} \)
   - Can verify witness for YES
   - Can catch false witness for NO (or contrapositive: if witness is verified, then a YES instance)

2. Find some \( \text{NP} \)-hard problem \( A \). Reduce \textit{from} \( A \) \textit{to} \( Q \):
   - Given instance \( I \) of \( A \), turn into \( f(I) \) of \( Q \) (in time polynomial in \( |I| \))
   - \( I \) YES of \( A \) if and only if \( f(I) \) YES of \( Q \)

Notes:
   - Careful about direction of reduction!!!!
   - Need to handle \textit{arbitrary} instances of \( A \), but can turn into very structured instances of \( Q \)
   - Often easiest to prove NO direction via contrapositive, to turn into statement about YES:
     - \( I \) YES of \( A \) \implies \( f(I) \) YES of \( Q \)
     - \( f(I) \) YES of \( Q \) \implies \( I \) YES of \( A \)
   - So proving “both directions”, but reduction only in one direction.
**Clique**

**Definition:** A *clique* in an undirected graph $G = (V, E)$ is a set $S \subseteq V$ such that $\{u, v\} \in E$ for all $u, v \in S$.

**Definition (Clique)**

Instance is a graph $G = (V, E)$ and an integer $k$. YES if $G$ contains a clique of size at least $k$, NO otherwise.
**Definition:** A *clique* in an undirected graph $G = (V, E)$ is a set $S \subseteq V$ such that $\{u, v\} \in E$ for all $u, v \in S$.

**Definition (CLIQUE)**

Instance is a graph $G = (V, E)$ and an integer $k$. YES if $G$ contains a clique of size at least $k$, NO otherwise.

**Theorem**

CLIQUE is NP-complete.
**Clique**

**Definition:** A *clique* in an undirected graph $G = (V, E)$ is a set $S \subseteq V$ such that $\{u, v\} \in E$ for all $u, v \in S$.

**Definition (Clique)**

Instance is a graph $G = (V, E)$ and an integer $k$. YES if $G$ contains a clique of size at least $k$, NO otherwise.

**Theorem**

**Clique** is **NP-complete**.

In **NP**:
Clique

**Definition:** A *clique* in an undirected graph $G = (V, E)$ is a set $S \subseteq V$ such that $\{u, v\} \in E$ for all $u, v \in S$.

**Definition (Clique)**

Instance is a graph $G = (V, E)$ and an integer $k$. YES if $G$ contains a clique of size at least $k$, NO otherwise.

**Theorem**

*Clique* is **NP-complete**.

**In NP:**

- **Witness:** $S \subseteq V$
- **Verifier:** Checks if $S$ is a clique and $|S| \geq k$
  - If $(G, k)$ a YES instance: there is a clique $S$ of size $\geq k$ on which verifier returns YES
  - If $(G, k)$ a NO instance: $S$ cannot be clique of size $\geq k$, so verifier always returns NO
**Clique** is **NP-hard**

Prove by reducing 3-SAT to **Clique**

- For arbitrary $A \in \textbf{NP}$, would have $A \leq_p \text{Circuit-SAT} \leq_p \text{3-SAT} \leq_p \text{Clique}$
**Clique is NP-hard**

Prove by reducing 3-SAT to Clique

- For arbitrary \( A \in \text{NP} \), would have \( A \leq_p \text{Circuit-SAT} \leq_p 3\text{-SAT} \leq_p \text{Clique} \)

Given 3-SAT formula \( F \) (with \( n \) variables and \( m \) clauses), set \( k = m \) and create graph \( G = (V, E) \):

- For every clause of \( F \), for every satisfying assignment to the clause, create vertex
- Add an edge between consistent assignments
**Clique** is **NP**-hard

Prove by reducing 3-SAT to Clique

- For arbitrary \( A \in \text{NP} \), would have \( A \leq_p \text{Circuit-SAT} \leq_p 3\text{-SAT} \leq_p \text{Clique} \)

Given 3-SAT formula \( F \) (with \( n \) variables and \( m \) clauses), set \( k = m \) and create graph \( G = (V, E) \):

- For every clause of \( F \), for every satisfying assignment to the clause, create vertex
- Add an edge between consistent assignments

Example: \( F = (x_1 \lor x_2 \lor x_4) \land (\bar{x}_3 \lor x_4) \land (\bar{x}_2 \lor \bar{x}_3) \)
3-SAT to \textsc{Clique} reduction analysis

Polytime: ✓
3-SAT to Clique reduction analysis

Polytime: ✓

If $F$ YES of 3-SAT:
  ▶ There is some satisfying assignment $x$
  ▶ For every clause, choose vertex corresponding to $x$. Let $S$ be chosen vertices
  ▶ $|S| = m = k$, and clique since all consistent (since all from $x$)
  $\implies (G, k)$ YES of Clique
3-SAT to Clique reduction analysis

Polytime: ✓

If F YES of 3-SAT:
   ▶ There is some satisfying assignment x
   ▶ For every clause, choose vertex corresponding to x. Let S be chosen vertices
   ▶ |S| = m = k, and clique since all consistent (since all from x)

⇒ (G, k) YES of Clique

If (G, k) YES of Clique:
   ▶ There is some clique S of size k = m
   ▶ Must contain exactly one vertex from each clause (since clique of size m)
   ▶ Since clique, all assignments consistent ⇒ there is an assignment that satisfies all clauses

⇒ F YES of 3-SAT
**Independent Set**

**Definition:** $S \subseteq V$ is an *independent set* in $G = (V, E)$ if $\{u, v\} \notin E$ for all $u, v \in S$.

**Definition (Independent Set)**

Instance is graph $G = (V, E)$ and integer $k$. YES if $G$ has an independent set of size $\geq k$, NO otherwise.
### Independent Set

**Definition:** $S \subseteq V$ is an *independent set* in $G = (V, E)$ if $\{u, v\} \notin E$ for all $u, v \in S$

<table>
<thead>
<tr>
<th>Definition (INDEPENDENT SET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance is graph $G = (V, E)$ and integer $k$. YES if $G$ has an independent set of size $\geq k$, NO otherwise.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INDEPENDENT SET</strong> is <strong>NP-complete</strong>.</td>
</tr>
</tbody>
</table>
**Independent Set**

**Definition:** $S \subseteq V$ is an *independent set* in $G = (V, E)$ if $\{u, v\} \notin E$ for all $u, v \in S$

**Definition (Independent Set)**

Instance is graph $G = (V, E)$ and integer $k$. YES if $G$ has an independent set of size $\geq k$, NO otherwise.

**Theorem**

**Independent Set** is NP-complete.

In NP:
**Independent Set**

**Definition:** $S \subseteq V$ is an *independent set* in $G = (V, E)$ if $\{u, v\} \notin E$ for all $u, v \in S$

**Definition (Independent Set)**

Instance is graph $G = (V, E)$ and integer $k$. YES if $G$ has an independent set of size $\geq k$, NO otherwise.

**Theorem**

Independent Set is **NP-complete**.

In NP:

- Witness is $S \subseteq V$. Verifier checks that $|S| \geq k$ and no edges in $S$
- If $(G, k)$ a YES instance then such an $S$ exists $\Longrightarrow$ verifier returns YES on it.
- If $(G, k)$ a NO then verifier will return NO on every $S$. 
Independent Set is NP-hard

Reduce from:

Given instance \((G, k)\) of Clique, create "complement graph" \(H\): a set of vertices, with \(\{u, v\} \in E(H)\) if and only if \(\{u, v\} \notin E(G)\).
**Independent Set** is **NP-hard**

Reduce from: **Clique**
Independent Set is NP-hard

Reduce from: Clique

- Given instance \((G, k)\) of Clique, create “complement graph” \(H\): same vertex set, with \(\{u, v\} \in E(H)\) if and only if \(\{u, v\} \notin E(G)\)
- Instance \((H, k)\) of Independent Set
Independent Set is NP-hard

Reduce from: Clique

- Given instance \((G, k)\) of Clique, create “complement graph” \(H\): same vertex set, with \(\{u, v\} \in E(H)\) if and only if \(\{u, v\} \notin E(G)\)
- Instance \((H, k)\) of Independent Set

If \((G, k)\) YES of Clique:

\[\implies\] Clique \(S \subseteq V\) of \(G\) with \(|S| \geq k\)

\[\implies\] \(S\) an independent set in \(H\)
**Independent Set is NP-hard**

Reduce from: **Clique**

- Given instance \((G, k)\) of **Clique**, create “complement graph” \(H\): same vertex set, with \(\{u, v\} \in E(H)\) if and only if \(\{u, v\} \notin E(G)\)
- Instance \((H, k)\) of **Independent Set**

If \((G, k)\) YES of **Clique**:

\[\begin{align*}
\implies & \text{Clique } S \subseteq V \text{ of } G \text{ with } |S| \geq k \\
\implies & S \text{ an independent set in } H
\end{align*}\]

If \((H, k)\) YES of **Independent Set**:

\[\begin{align*}
\implies & \text{Independent set } S \subseteq V \text{ in } H \text{ with } |S| \geq k \\
\implies & S \text{ a clique in } G
\end{align*}\]
**Vertex Cover**

**Definition:** $S \subseteq V$ is a *vertex cover* of $G = (V, E)$ if $S \cap e \neq \emptyset$ for all $e \in E$.

**Definition (Vertex Cover)**

Instance is graph $G = (V, E)$, integer $k$. YES if $G$ has a vertex cover of size $\leq k$, NO otherwise.
**Vertex Cover**

**Definition:** $S \subseteq V$ is a *vertex cover* of $G = (V, E)$ if $S \cap e \neq \emptyset$ for all $e \in E$.

**Definition (Vertex Cover)**

Instance is graph $G = (V, E)$, integer $k$. YES if $G$ has a vertex cover of size $\leq k$, NO otherwise.

**Theorem**

*Vertex Cover* is **NP-complete**
**Vertex Cover**

**Definition:** $S \subseteq V$ is a *vertex cover* of $G = (V, E)$ if $S \cap e \neq \emptyset$ for all $e \in E$.

**Definition (Vertex Cover)**

Instance is graph $G = (V, E)$, integer $k$. YES if $G$ has a vertex cover of size $\leq k$, NO otherwise.

**Theorem**

*Vertex Cover* is **NP-complete**

In **NP**:
**Vertex Cover**

**Definition**: \( S \subseteq V \) is a *vertex cover* of \( G = (V, E) \) if \( S \cap e \neq \emptyset \) for all \( e \in E \)

**Definition (Vertex Cover)**

Instance is graph \( G = (V, E) \), integer \( k \). YES if \( G \) has a vertex cover of size \( \leq k \), NO otherwise.

**Theorem**

*Vertex Cover* is NP-complete

**In NP:**

- Witness is \( S \subseteq V \). Verifier checks that \( |S| \leq k \) and every edge has at least one endpoint in \( S \).
- If \((G, k)\) a YES instance then such an \( S \) exists \( \iff \) verifier returns YES on it.
- If \((G, k)\) a NO then verifier will return NO on every \( S \).
**Vertex Cover is NP-hard**

Reduce from **Independent Set**

- Given instance \((G = (V, E), k)\) of **Independent Set**, create instance \((G, n - k)\) of **Vertex Cover** (where \(n = |V|\))
**Vertex Cover** is **NP**-hard

Reduce from **Independent Set**

- Given instance \((G = (V, E), k)\) of **Independent Set**, create instance \((G, n - k)\) of **Vertex Cover** (where \(n = |V|\))

If \((G, k)\) a YES instance of **Independent Set**:

\(\implies\) \(G\) has an independent set \(S\) with \(|S| \geq k\)

\(\implies\) \(V \setminus S\) a vertex cover of \(G\) of size \(\leq n - k\)

\(\implies\) \((G, n - k)\) a YES instance of **Vertex Cover**
**Vertex Cover is NP-hard**

Reduce from **Independent Set**

- Given instance \((G = (V, E), k)\) of **Independent Set**, create instance \((G, n - k)\) of **Vertex Cover** (where \(n = |V|\))

If \((G, k)\) a YES instance of **Independent Set**:

\[ \implies G \text{ has an independent set } S \text{ with } |S| \geq k \]

\[ \implies V \setminus S \text{ a vertex cover of } G \text{ of size } \leq n - k \]

\[ \implies (G, n - k) \text{ a YES instance of } \textbf{Vertex Cover} \]

If \((G, n - k)\) a YES instance of **Vertex Cover**:

\[ \implies G \text{ has a vertex cover } S \text{ of size at most } n - k \]

\[ \implies V \setminus S \text{ an independent set of } G \text{ of size at least } k \]

\[ \implies (G, k) \text{ a YES instance of } \textbf{Independent Set} \]