Lecture 24: NP-Completeness II

Michael Dinitz

November 20, 2025

601.433/633 Introduction to Algorithms

Introduction

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

Today: more NP-complete problems.

Definition

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

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

Reductions

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 ***NP**-hard* if $Q' \leq_p Q$ for all problems Q' in NP. Problem Q is ***NP**-complete* if it is NP-hard and in 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 $\mathbf{1}$?

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: $\wedge$ OR: $\vee$
- $x_1 \vee (\bar{x}_5 \wedge x_7) \wedge (\bar{x}_2 \vee (x_6 \wedge \bar{x}_3)) \ldots$

Conjunctive normal form (CNF): AND of ORs (clauses)

- $(x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee \bar{x}_6) \ldots$

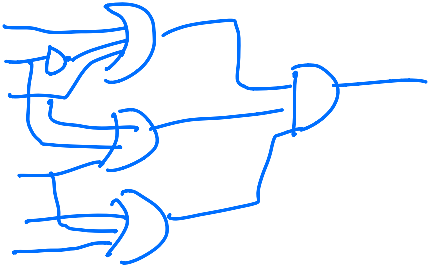
3-SAT

Boolean formula:

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

Conjunctive normal form (CNF): AND of ORs (clauses)

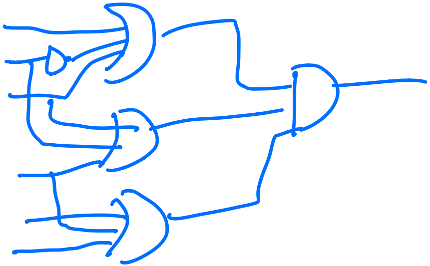
- $(x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee \bar{x}_6) \ldots$



3-SAT

Boolean formula:

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



Conjunctive normal form (CNF): AND of ORs (clauses)

- $(x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee \bar{x}_6) \ldots$

Definition

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

3-SAT

Theorem

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

3-SAT

Theorem

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

3-SAT in ***NP***:

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 ***NP****-complete.*

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

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

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 is ***NP***-hard: Show Circuit-SAT $\leq_p$ 3-SAT.

3-SAT

Theorem

3-SAT is $\boldsymbol{NP}$*-complete.*

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

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

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

3-SAT

Theorem

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

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

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

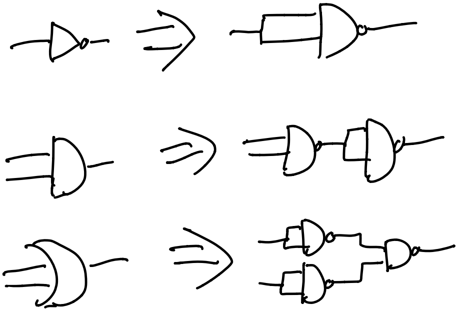
- Don't need to show that $A \leq_p$ 3-SAT for arbitrary $A \in \boldsymbol{NP}$: already know that $A \leq_p$ Circuit-SAT!

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)

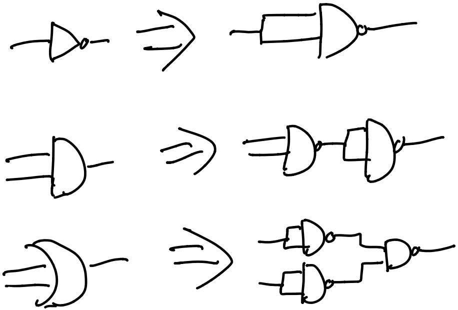
- AND/OR/NOT universal, but so is just NAND!



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" $x_1, x_2, \dots, x_n$
- m NAND gates: $g_1, \dots, g_m$
  - $g_1 = NAND(x_1, x_3)$,
    $g_2 = NAND(g_1, x_4)$, ...
- 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)

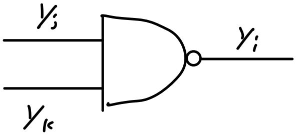
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



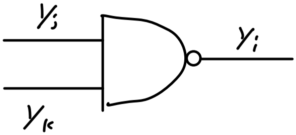
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \lor y_j \lor y_k$

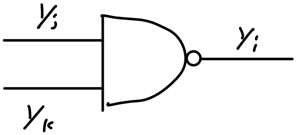
Reduction to 3-SAT

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 $\mathbf{1}$.

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 $y_i = NAND(y_j, y_k)$, create clauses:



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

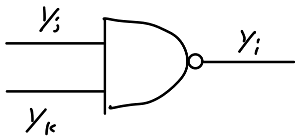
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$

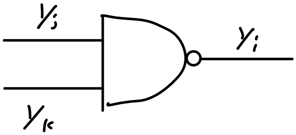
Reduction to 3-SAT

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 $\mathbf{1}$.

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)

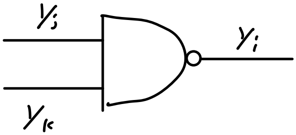
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee y_j \vee \bar{y}_k$

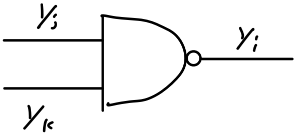
Reduction to 3-SAT

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 $\mathbf{1}$.

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee y_j \vee \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)

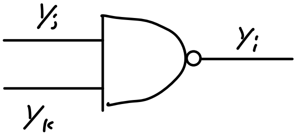
Reduction to 3-SAT

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 $\mathbf{1}$.

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = \mathbf{0}$ and $y_k = \mathbf{0}$ then $y_i = \mathbf{1}$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = \mathbf{1}$ and $y_k = \mathbf{0}$ then $y_i = \mathbf{1}$)
- $y_i \vee y_j \vee \bar{y}_k$ (if $y_j = \mathbf{0}$ and $y_k = \mathbf{1}$ then $y_i = \mathbf{1}$)
- $\bar{y}_i \vee \bar{y}_j \vee \bar{y}_k$

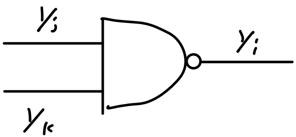
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee y_j \vee \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
- $\bar{y}_i \vee \bar{y}_j \vee \bar{y}_k$ (if $y_j = 1$ and $y_k = 1$ then $y_i = 0$)

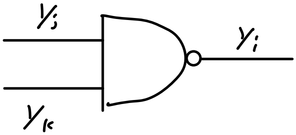
Reduction to 3-SAT

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 $y_i = NAND(y_j, y_k)$, create clauses:



- $y_i \vee y_j \vee y_k$ (if $y_j = 0$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee \bar{y}_j \vee y_k$ (if $y_j = 1$ and $y_k = 0$ then $y_i = 1$)
- $y_i \vee y_j \vee \bar{y}_k$ (if $y_j = 0$ and $y_k = 1$ then $y_i = 1$)
- $\bar{y}_i \vee \bar{y}_j \vee \bar{y}_k$ (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

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

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

- $y_i = x_i$ if $i \leq n$
- $y_i = g_{i-n}$ 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

$\Longrightarrow$ $\exists$ setting x of input wires so $g_m = 1$

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

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

$\Longrightarrow$ $f(C)$ YES of 3-SAT

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

$\Longrightarrow$ $\exists$ assignment y to variables so that all clauses satisfied

$\Longrightarrow$ $\exists$ setting x of input wires so $g_m = 1$:

- $x_i = y_i$
- Output of gate $g_i = y_{i+n}$ (by construction)
- So $g_m = 1$ (since (y_{m+n}) is a clause)

$\Longrightarrow$ C a YES instance of Circuit-SAT

General Methodology to Prove Q NP-Complete

1. Show Q is in 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 NP-hard problem A. Reduce *from* A *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

General Methodology to Prove Q NP-Complete

1. Show Q is in 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 NP-hard problem A. Reduce *from* A *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 *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.

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* $\mathbf{NP}$*-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* $\boldsymbol{NP}$*-complete.*

In $\boldsymbol{NP}$:

- 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 $\boldsymbol{NP}$-hard

Prove by reducing 3-SAT to CLIQUE

- For arbitrary $\boldsymbol{A} \in \boldsymbol{NP}$, would have $\boldsymbol{A} \leq_{\boldsymbol{p}}$ Circuit-SAT $\leq_{\boldsymbol{p}}$ 3-SAT $\leq_{\boldsymbol{p}}$ CLIQUE

Clique is *NP*-hard

Prove by reducing 3-SAT to Clique

- For arbitrary $A \in NP$, would have $A \le_p$ Circuit-SAT $\le_p$ 3-SAT $\le_p$ 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

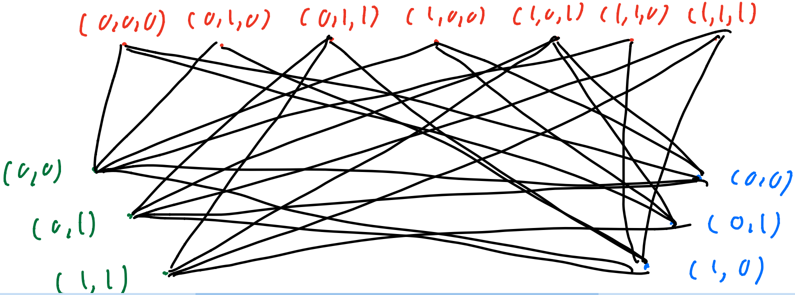
Prove by reducing 3-SAT to CLIQUE

- For arbitrary $A \in NP$, would have $A \le_p$ Circuit-SAT $\le_p$ 3-SAT $\le_p$ 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: $F = (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3)$



3-SAT to 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)

$\Longrightarrow$ (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 $\Longrightarrow$ there is an assignment that satisfies all clauses

$\Longrightarrow$ 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 $u, v \in S$

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.*

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 $\implies$ 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:

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:

$\Longrightarrow$ Clique $S \subseteq V$ of G with $|S| \geq k$

$\Longrightarrow$ S an independent set in H

If (H, k) YES of INDEPENDENT SET:

$\Longrightarrow$ Independent set $S \subseteq V$ in H with $|S| \geq k$

$\Longrightarrow$ S a clique in G

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* $\textbf{NP}$*-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 $\implies$ 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 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

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

$\implies$ G has a vertex cover S of size at most $n - k$

$\implies$ $V \setminus S$ an independent set of G of size at least k

$\implies$ (G, k) a YES instance of Independent Set