Lecture 22: NP-Completeness II

Michael Dinitz

November 11, 2021

601.433/633 Introduction to Algorithms

Introduction

Last time: Definition of **P**, **NP**, reductions, **NP**-completeness. Proof that Circuit-SAT is **NP**-complete.

Today: more NP-complete problems.

Definition

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

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

Reductions

Definition

A *Many-one* or *Karp* reduction from $\mathbf{A}$ to $\mathbf{B}$ is a function $\mathbf{f}$ which takes arbitrary instances of $\mathbf{A}$ and transforms them into instances of $\mathbf{B}$ so that

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

Definition

Problem $\mathbf{Q}$ is **NP**-*hard* if $\mathbf{Q'} \leq_{\mathbf{p}} \mathbf{Q}$ for all problems $\mathbf{Q'}$ in **NP**. Problem $\mathbf{Q}$ 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 **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)

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

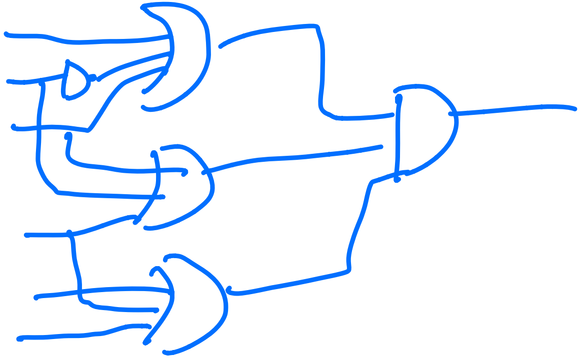
3-SAT

Boolean formula:

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

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

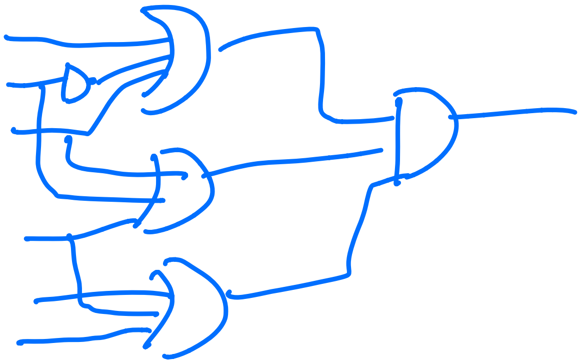
- $\mathbf{(x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee \bar{x}_6) \dots}$



3-SAT

Boolean formula:

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



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

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

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 $\leq_{\mathbf{p}}$ 3-SAT.

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 $\leq_{\mathbf{p}}$ 3-SAT.

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

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.

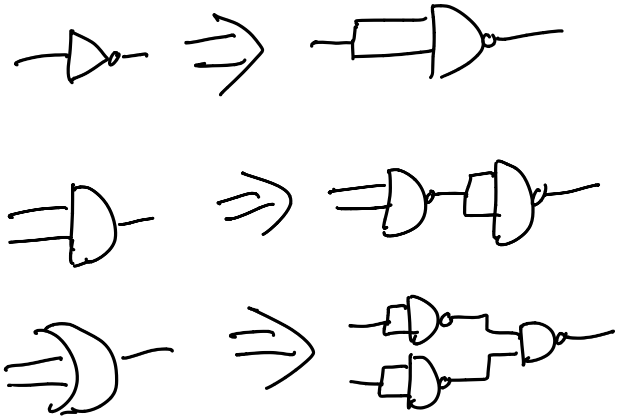
- Don't need to show that **A** $\leq_p$ 3-SAT for arbitrary **A** $\in$ **NP**: already know that **A** $\leq_p$ Circuit-SAT!

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)

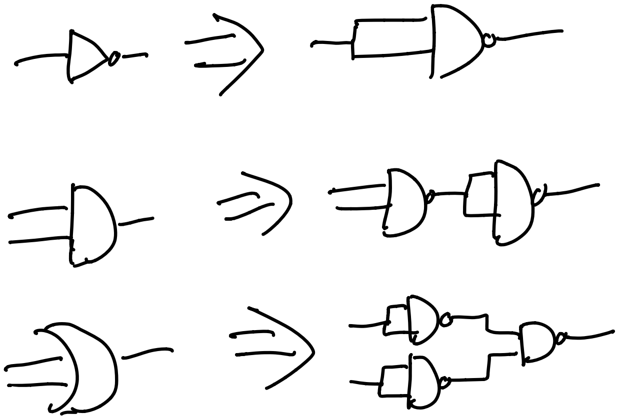
- AND/OR/NOT universal, but so is just NAND!



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" $\mathbf{x_1, x_2, \ldots, x_n}$
- **m** NAND gates: $\mathbf{g_1, \ldots, g_m}$
  - $\mathbf{g_1 = NAND(x_1, x_3)}$,
    $\mathbf{g_2 = NAND(g_1, x_4)}$, ...
- WLOG, $\mathbf{g_m}$ is the "output gate"

Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

vars for input wires

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

vars for each gate

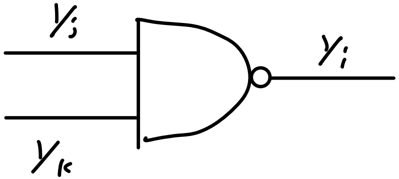
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



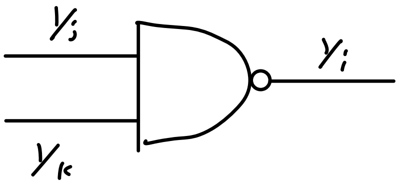
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



- $\mathbf{y_i \vee y_j \vee y_k}$

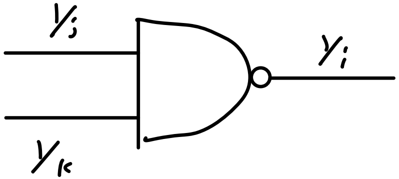
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

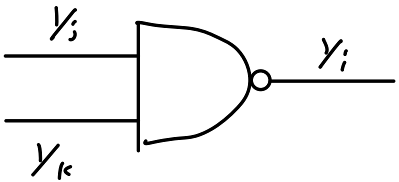
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

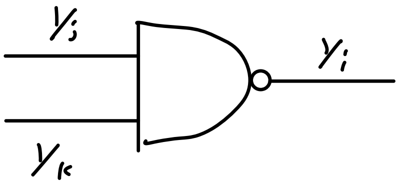
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

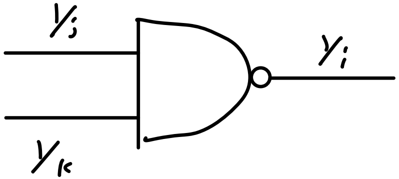
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

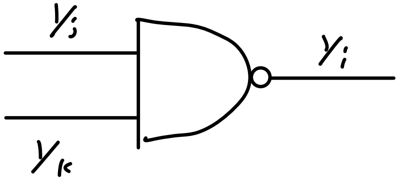
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

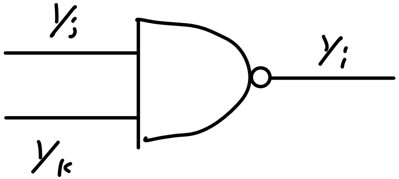
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

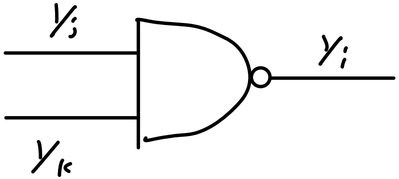
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $\mathbf{x_1, x_2, \ldots, x_n}$
- $\mathbf{m}$ NAND gates: $\mathbf{g_1, \ldots, g_m}$. Output gate $\mathbf{g_m}$

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

Variables: $\mathbf{y_1, y_2, \ldots, y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+m}}$ (one for each wire)
Clauses: For every NAND gate $\mathbf{y_i = NAND(y_j, y_k)}$, create clauses:



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

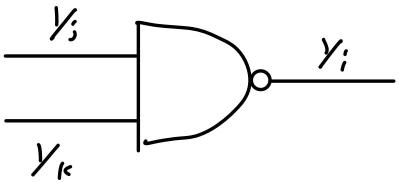
Reduction to 3-SAT

So given as input a circuit $\mathbf{C}$:

- $\mathbf{n}$ "input wires" $x_1, x_2, \ldots, x_n$
- $\mathbf{m}$ NAND gates: $g_1, \ldots, g_m$. Output gate g_m

Need to construct many-one reduction $\mathbf{f}$ to 3-SAT: in polynomial time, construct 3-CNF formula $f(C)$ such that $f(C)$ has a satisfying assignment if and only if $\mathbf{C}$ 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 $y_i = \mathrm{NAND}(y_j, y_k)$, 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 $\mathbf{1}$)

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 $\mathbf{C}$ YES of Circuit-SAT

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

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

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

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

Analysis

Theorem

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

Polytime: ✓

Suppose $\mathbf{C}$ YES of Circuit-SAT

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

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

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

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

Suppose $\mathbf{f(C)}$ YES of 3-SAT

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

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

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

$\Longrightarrow$ $\mathbf{C}$ 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 $\mathbf{f(I)}$ of **Q** (in time polynomial in $|\mathbf{I}|$)
   - **I** YES of **A** if and only if $\mathbf{f(I)}$ 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 $\mathbf{f(I)}$ of **Q** (in time polynomial in $|\mathbf{I}|$)
    - **I** YES of **A** if and only if $\mathbf{f(I)}$ 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** $\implies$ $\mathbf{f(I)}$ YES of **Q**
    - $\mathbf{f(I)}$ 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* **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** ∈ **NP**, would have **A** $\leq_p$ Circuit-SAT $\leq_p$ 3-SAT $\leq_p$ CLIQUE

CLIQUE is **NP**-hard

Prove by reducing 3-SAT to CLIQUE

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

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

- For every clause of $\mathbf{F}$, 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

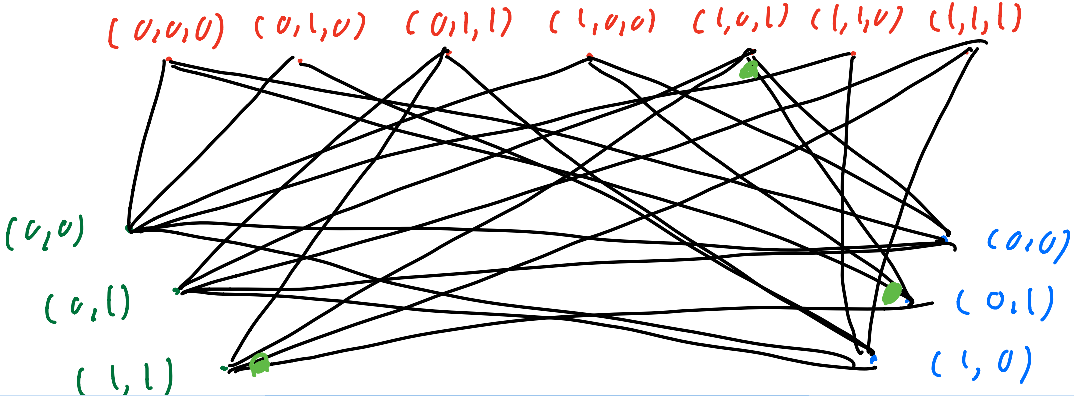
- For arbitrary $\mathbf{A} \in \mathbf{NP}$, would have $\mathbf{A} \le_p$ Circuit-SAT $\le_p$ 3-SAT $\le_p$ CLIQUE

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

- For every clause of $\mathbf{F}$, for every satisfying assignment to the clause, create vertex
- Add an edge between consistent assignments

Example: $\mathbf{F} = (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3)$

$x_1 = T$
$x_2 = F$
$x_3 = T$
$x_4 = T$



3-SAT to CLIQUE reduction analysis

Polytime: ✓

3-SAT to CLIQUE reduction analysis

Polytime: ✓

If $\mathbf{F}$ YES of 3-SAT:

- There is some satisfying assignment $\mathbf{x}$
- For every clause, choose vertex corresponding to $\mathbf{x}$. Let $\mathbf{S}$ be chosen vertices
- $|\mathbf{S}| = \mathbf{m} = \mathbf{k}$, and clique since all consistent (since all from $\mathbf{x}$)

$\implies$ $(\mathbf{G}, \mathbf{k})$ YES of CLIQUE

3-SAT to CLIQUE reduction analysis

Polytime: ✓

If $\mathbf{F}$ YES of 3-SAT:

- There is some satisfying assignment $\mathbf{x}$
- For every clause, choose vertex corresponding to $\mathbf{x}$. Let $\mathbf{S}$ be chosen vertices
- $|\mathbf{S}| = \mathbf{m} = \mathbf{k}$, and clique since all consistent (since all from $\mathbf{x}$)

$\implies$ $(\mathbf{G}, \mathbf{k})$ YES of CLIQUE

If $(\mathbf{G}, \mathbf{k})$ YES of CLIQUE:

- There is some clique $\mathbf{S}$ of size $\mathbf{k} = \mathbf{m}$
- Must contain exactly one vertex from each clause (since clique of size $\mathbf{m}$)
- Since clique, all assignments consistent $\implies$ there is an assignment that satisfies all clauses

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

Definition (Independent Set)

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

Theorem

Independent Set *is* **NP**-*complete.*

Independent Set

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

Definition (Independent Set)

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

Theorem

Independent Set *is* **NP**-*complete.*

In **NP**:

Independent Set

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

Definition (Independent Set)

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

Theorem

Independent Set *is* **NP**-*complete.*

In **NP**:

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

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:

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

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

Definition (Vertex Cover)

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

Vertex Cover

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

Definition (Vertex Cover)

Instance is graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, integer $\mathbf{k}$. YES if $\mathbf{G}$ has a vertex cover of size $\leq \mathbf{k}$, 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: $\mathbf{S} \subseteq \mathbf{V}$ is a *vertex cover* of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ if $\mathbf{S} \cap \mathbf{e} \neq \emptyset$ for all $\mathbf{e} \in \mathbf{E}$

Definition (Vertex Cover)

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

Theorem

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

In **NP**:

- Witness is $\mathbf{S} \subseteq \mathbf{V}$. Verifier checks that $|\mathbf{S}| \leq \mathbf{k}$ and every edge has at least one endpoint in $\mathbf{S}$
- If $(\mathbf{G}, \mathbf{k})$ a YES instance then such an $\mathbf{S}$ exists $\implies$ verifier returns YES on it.
- If $(\mathbf{G}, \mathbf{k})$ a NO then verifier will return NO on every $\mathbf{S}$.

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 $(\mathbf{G} = (\mathbf{V}, \mathbf{E}), \mathbf{k})$ of INDEPENDENT SET, create instance $(\mathbf{G}, \mathbf{n} - \mathbf{k})$ of VERTEX COVER (where $\mathbf{n} = |\mathbf{V}|$)

If $(\mathbf{G}, \mathbf{k})$ a YES instance of INDEPENDENT SET:

$\implies$ $\mathbf{G}$ has an independent set $\mathbf{S}$ with $|\mathbf{S}| \geq \mathbf{k}$

$\implies$ $\mathbf{V} \setminus \mathbf{S}$ a vertex cover of $\mathbf{G}$ of size $\leq \mathbf{n} - \mathbf{k}$

$\implies$ $(\mathbf{G}, \mathbf{n} - \mathbf{k})$ 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:

$\Longrightarrow$ G has an independent set S with $|S| \geq k$

$\Longrightarrow$ $V \setminus S$ a vertex cover of G of size $\leq n - k$

$\Longrightarrow$ $(G, n - k)$ a YES instance of Vertex Cover

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

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

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

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