27 March 2024 — Boolean Satisfiability (SAT)

Plan

* Recall CNF-SAT
* Announcements
* Reductions To SAT
  - ↳ IND SET $\leq_p$ CNF-SAT
  - ↳ CIRCUIT-SAT $\leq_p$ 3SAT

Boolean Satisfiability

Given: boolean formula φ in Conjunctive Normal Form

CNF $\varphi = (x_1 \vee \neg x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_7 \vee \neg x_6)$
$\wedge \ldots \wedge (\neg x_1 \vee \neg x_{100})$

Question:

Does there exist an truth assignment to $(x_1 \ldots x_n)$ that satisfies φ ?

Boolean Satisfiability

Given: boolean formula φ in Conjunctive Normal Form

CNF $\varphi = (x_1 \vee \neg x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_7 \vee \neg x_6) \wedge \dots \wedge \underbrace{(\neg x_1 \vee \neg x_{100})}_{\text{clauses}}$

literals (variable or negation)

Question:

Does there exist an <u>truth assignment</u> to $(x_1, \dots, x_n)$ that <u>satisfies</u> φ ?

$\vec{a} = (1, 0, 0, \dots, 1)$

$\varphi(\vec{a}) = T$

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

↳ SAT is in NP.

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

↳ SAT is in NP.

Poly-time verifier for SAT.

$V(\varphi, \vec{a})$.

For each clause in φ

if $\vec{a}$ does not satisfy clause

Return $\perp$

Return ✓

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

- ↳ SAT is in NP.
- ↳ SAT is NP-Hard.

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

↳ SAT is in NP.

↳ SAT is NP-Hard.

↓

Every efficiently verifiable problem reduces to SAT!

Why study SAT?

Cook-Levin Theorem. SAT is NP-Complete.

↳ SAT is in NP.

↳ SAT is NP-Hard.

↓

Every efficiently verifiable problem reduces to SAT!

Solving SAT is hard!

Solving SAT is powerful!

Announcements

* HW 7 due Thurs, 11:59 pm
  ↳ See Ed Post about update to Q3 (makes problem easier)

* Prelim Conflict Survey
  ↳ Link on Ed

* Prelim #2 Review Session
  ↳ April 9, 7-9 pm, Gates G01.

Every efficiently verifiable problem reduces to SAT!

Solving SAT is hard!

Solving SAT is powerful!

Every efficiently verifiable problem reduces to SAT!

Solving SAT is hard!

Solving SAT is powerful!

Practical Algorithm design paradigm:

- Reduce problem to SAT
- Use optimized SAT SOLVER to solve the problem

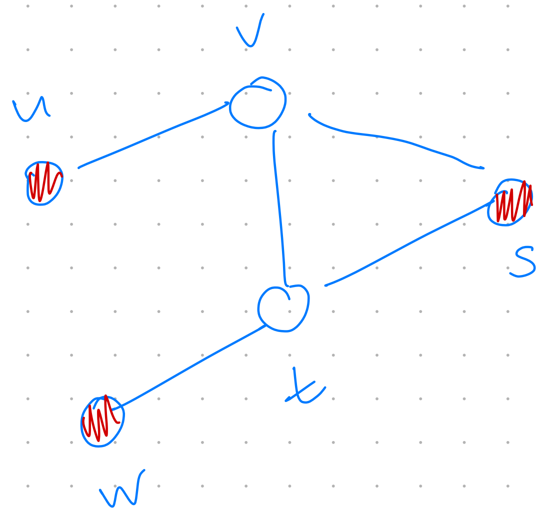
INDEPENDENT SET REDUCES TO CNF-SAT

↳ Given: Graph G, parameter k

$|S| \geq k$

Find: A subset $S \subseteq V$ of vertices such that no two vertices $u, v \in S$ share an edge $(u,v) \in E$

$G =$



$\xrightarrow{\text{reduction}}$ φ_G

φ_G satisfiable

$\Longleftrightarrow$

G has INDSET $\geq k$

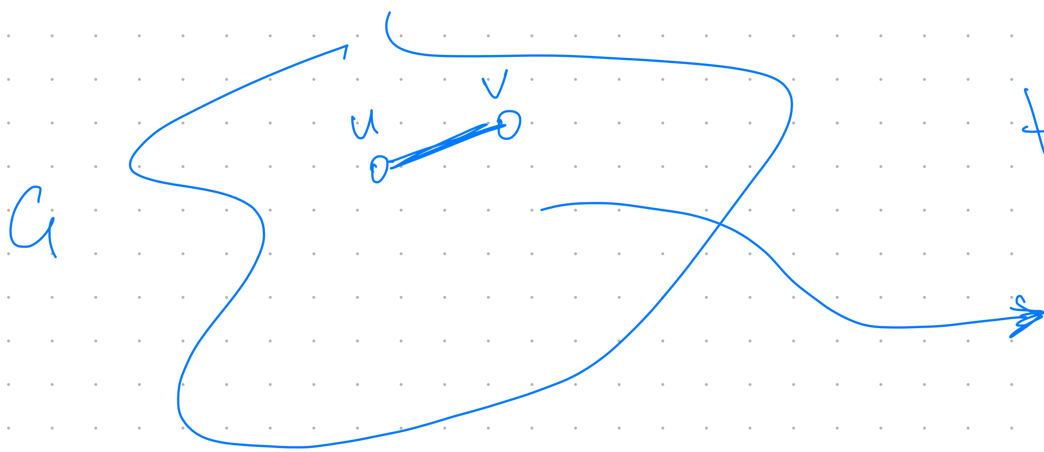
INDEPENDENT SET REDUCES TO CNF-SAT

↳ Given: Graph G, parameter k

Find: A subset $S \subseteq V$ of vertices $|S| \geq k$

such that no two vertices $u, v \in S$

share an edge $(u,v) \in E$



$\forall (u,v) \in E$

$\neg(u \in S) \vee \neg(v \in S)$

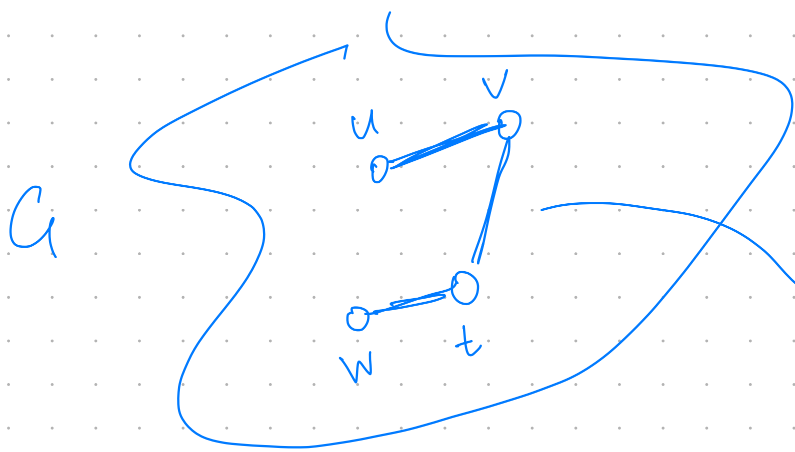
INDEPENDENT SET REDUCES TO CNF-SAT

↳ Given: Graph G, parameter k

Find: A subset $S \subseteq V$ of vertices $|S| \geq k$

such that no two vertices $u, v \in S$

share an edge $(u,v) \in E$



$\forall (u,v) \in E$

$$\neg\underbrace{(u \in S)}_{x_u} \vee \neg\underbrace{(v \in S)}_{x_v}$$

$$(\neg x_u \vee \neg x_v) \wedge (\neg x_v \vee \neg x_t) \wedge (\neg x_w \vee \neg x_t) \wedge \ldots \quad \forall e \in E$$

INDEPENDENT SET REDUCES TO CNF-SAT

↳ Given: Graph G, parameter k

Find: A subset $S \subseteq V$ of vertices $|S| \geq k$

such that no two vertices $u, v \in S$

share an edge $(u,v) \in E$

Need some mechanism to count to k!

Otherwise $S = \emptyset$, i.e. $\vec{a} = \vec{0}$ satisfies φ_G

$$(\neg x_u \vee \neg x_v) \wedge (\neg x_v \vee \neg x_t) \wedge (\neg x_w \vee \neg x_t) \wedge \ldots \quad \forall e \in E$$

$x_u^i = T \iff u$ is the i^{th} vertex in S

$$\overbrace{\boxed{x_u^1} \, \boxed{\dots} \, \boxed{x_u^i} \, \boxed{\dots} \, \boxed{x_u^k}}^{x_u}$$

Ind Set instance G, k

k boolean variable per vertex

$x_u^i = T \iff u$ is the i^{th} vertex in S

$$\overbrace{\boxed{x_u^1} \;\boxed{\cdots}\; \boxed{x_u^i} \;\boxed{\cdots}\; \boxed{x_u^k}}^{x_u}$$

k boolean variable per vertex

1

u is "in" S at most once

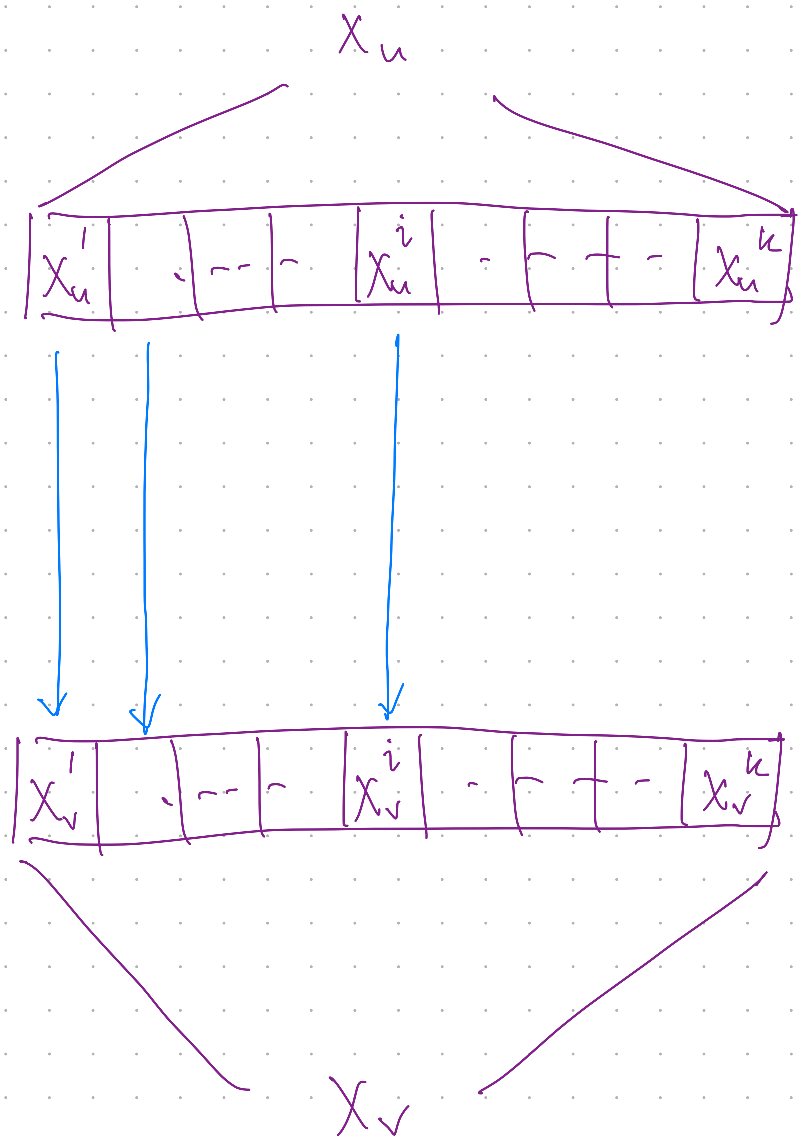
For every $u \in V$ $\quad 1 \leq i < j \leq k$

$$(\neg x_u^i \vee \neg x_u^j)$$

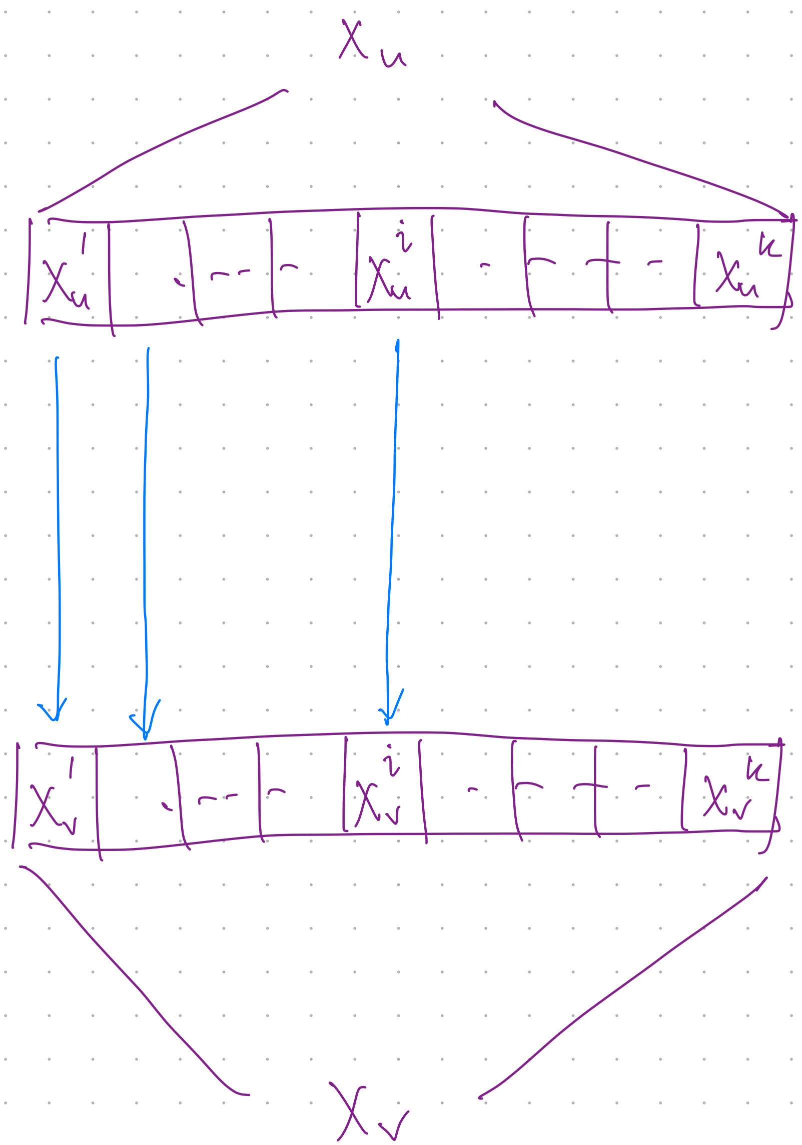
u cannot be i^{th} and j^{th} vertex of S

$x_u^i = T \iff u$ is the i^{th} vertex in S

Exactly 1 vertex must be the i^{th} vertex



$x_u^i = T \iff u$ is the i^{th} vertex in S

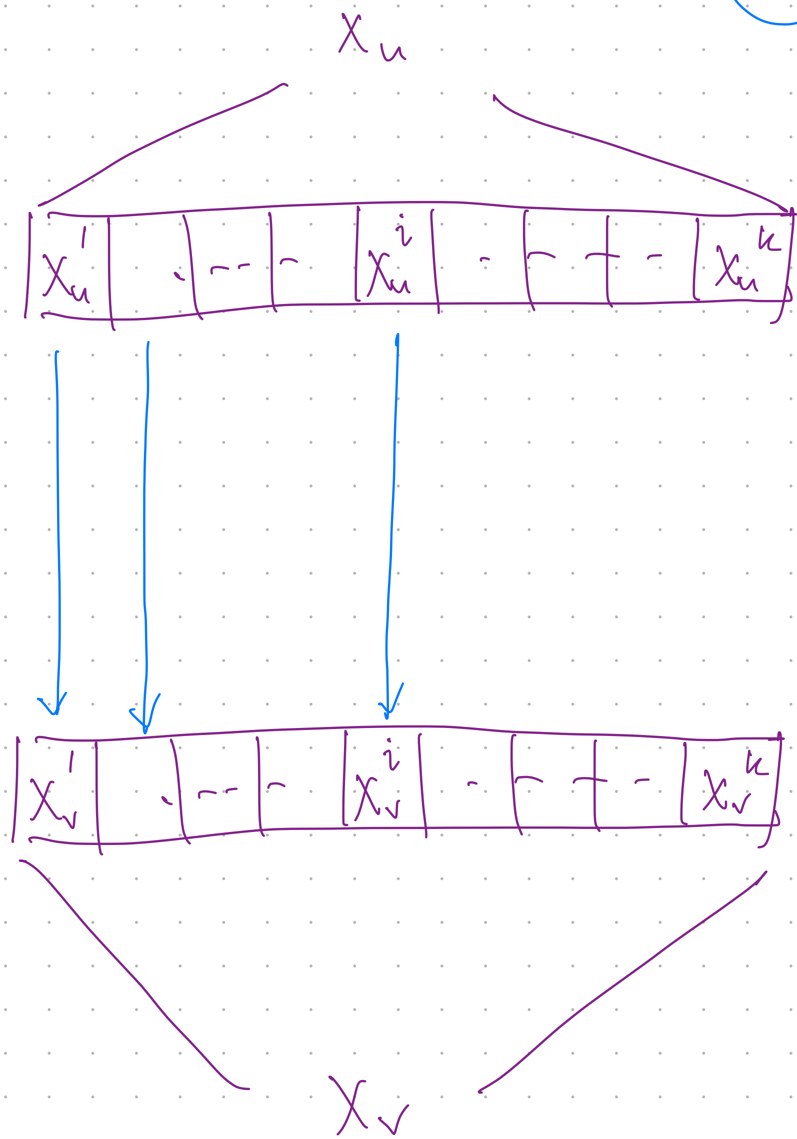


Exactly 1 vertex must be the i^{th} vertex

At least 1.

$$\left(x_u^i \vee x_v^i \vee x_w^i \vee \cdots_{\forall v \in V}\right)$$

$x_u^i = T \iff u$ is the i^{th} vertex in S



2 Exactly 1 vertex must be the i^{th} vertex

At least 1.

(2a) $\left(x_u^i \vee x_v^i \vee x_w^i \vee \dots_{\forall v \in V}\right)$

At most 1

(2b) $\forall u, v \in V$

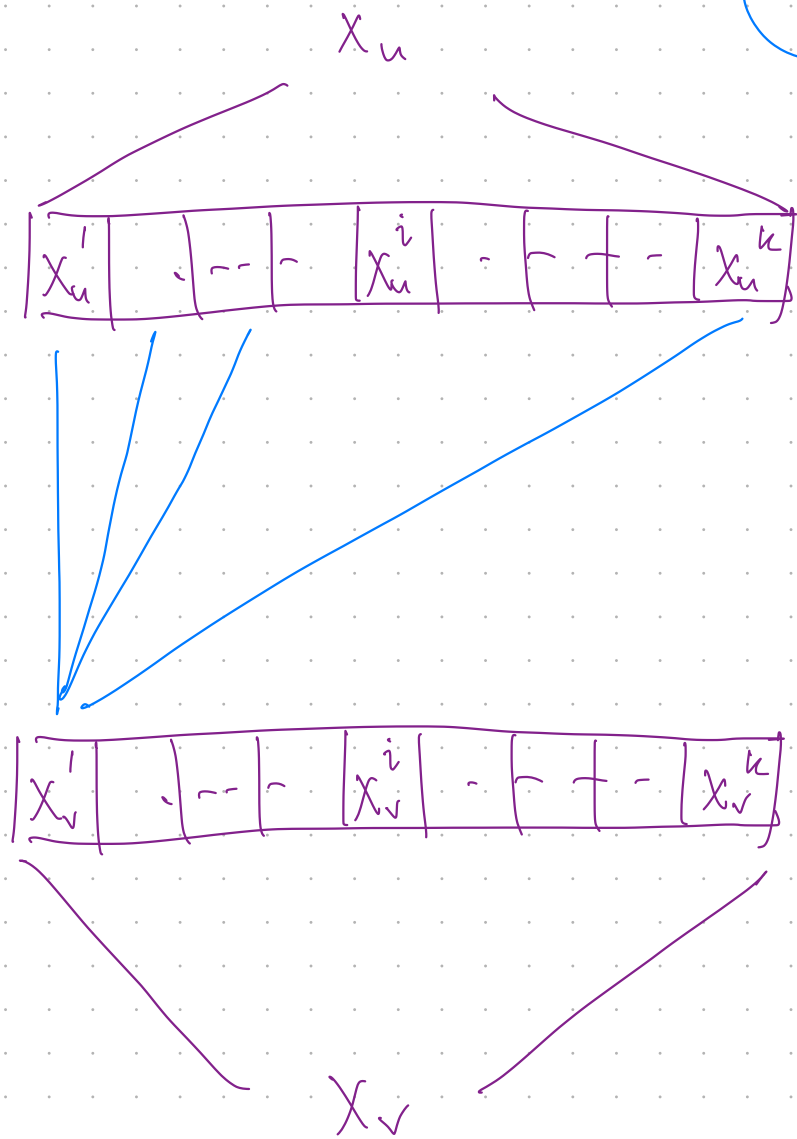
$$\left(\neg x_u^i \vee \neg x_v^i\right)$$

$x_u^i = T \iff u$ is the i^{th} vertex in S

3

IND SET constraints

No vertices in S share an edge



$\forall u, v \in E$

$\forall i, j \in \{1, \dots k\}$

$$(\neg x_u^i \vee \neg x_v^j)$$

$G = (V, E)$, k

$\downarrow$

$$\varphi_{G,k} = \bigwedge_{u \in V} (\neg x_u^1 \vee \neg x_u^2) \wedge (\neg x_u^1 \vee \neg x_u^3) \wedge \dots \wedge (\neg x_u^{k-1} \vee \neg x_u^k) \quad (1)$$

$$\wedge\ (x_u^1 \vee x_v^1 \vee \dots \vee x_w^1) \wedge (x_u^2 \vee x_v^2 \vee \dots \vee x_w^2) \wedge \dots \wedge (x_u^k \vee x_v^k \vee \dots \vee x_w^k) \quad (2a)$$

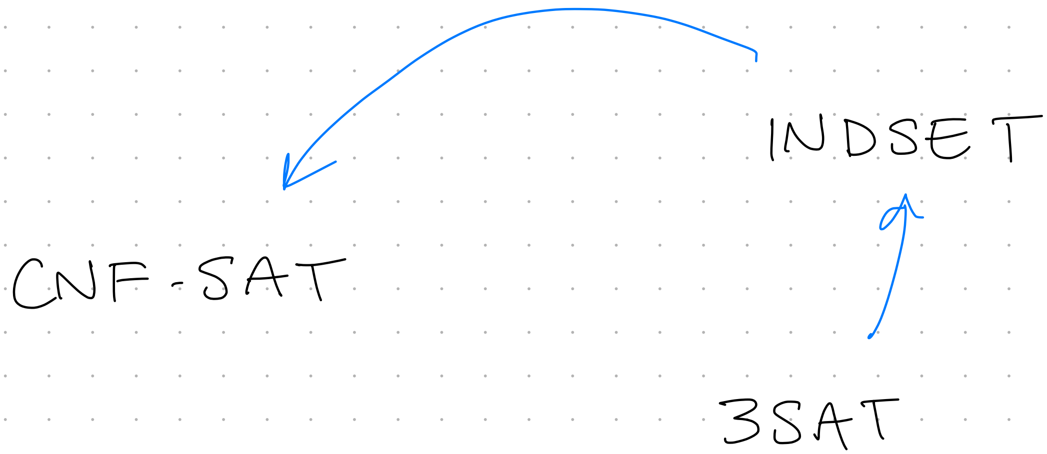
$$\wedge\ (\neg x_u^1 \vee \neg x_v^1) \wedge \dots \wedge (\neg x_v^1 \vee \neg x_w^1) \wedge \dots \wedge (\neg x_u^k \vee \neg x_v^k) \wedge \dots \wedge (\neg x_v^k \vee \neg x_w^k) \quad (2b)$$

$$\bigwedge_{(u,v) \in E} (\neg x_u^1 \vee \neg x_v^2) \wedge (\neg x_u^1 \vee \neg x_v^3) \wedge \dots \wedge (\neg x_u^{k-1} \vee \neg x_v^k) \quad (3)$$

polynomial-sized CNF

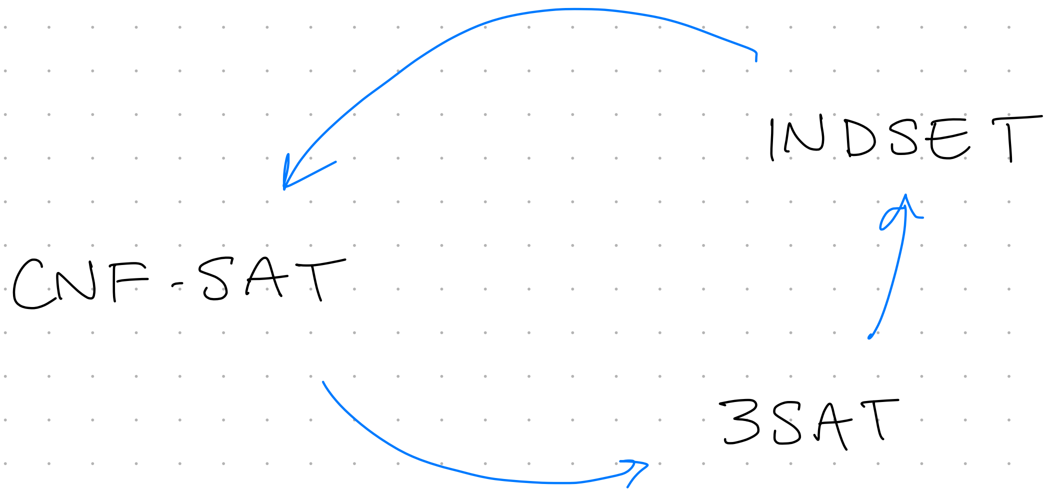
INDSET $\leq_p$ CNF-SAT.

Previously we saw 3SAT $\leq_p$ INDSET



INDSET $\leq_p$ CNF-SAT.

Previously we saw 3SAT $\leq_p$ INDSET



NP-Complete problems form an equivalence class (under poly-time Karp reductions)

INDSET $\leq_p$ CNF-SAT.

Previously we saw 3SAT $\leq_p$ INDSET

INDSET

CNF-SAT

3SAT

SUBSET-SUM

MAX CUT

CLIQUE

VERTEX-COVER

TSP

CIRCUIT SAT

HAMCYCLE

PROTEIN-FOLDING

INTEGER PROGRAMMING

If **any** NP-Complete problem has a poly-time algorithm, they **all** do!

(i.e. P = NP)

NP-Complete problems form an equivalence class

(under poly-time Karp reductions)

CIRCUIT - SAT $\leq_p$ 3SAT

Given: Logical Circuit $C : \{0,1\}^n \longrightarrow \{0,1\}$

Question:

Does there exist $x \in \{0,1\}^n$

s.t. $C(x) = 1$?

CIRCUIT - SAT $\leq_p$ 3SAT

Given: Logical Circuit $C : \{0,1\}^n \longrightarrow \{0,1\}$

Question.

Does there exist $x \in \{0,1\}^n$

s.t. $C(x) = 1$?

Note. CIRCUIT SAT $\leq_p$ 3SAT is a key step in proof of Cook-Levin Theorem!

Intuition. Logical Circuits can implement any algorithm.

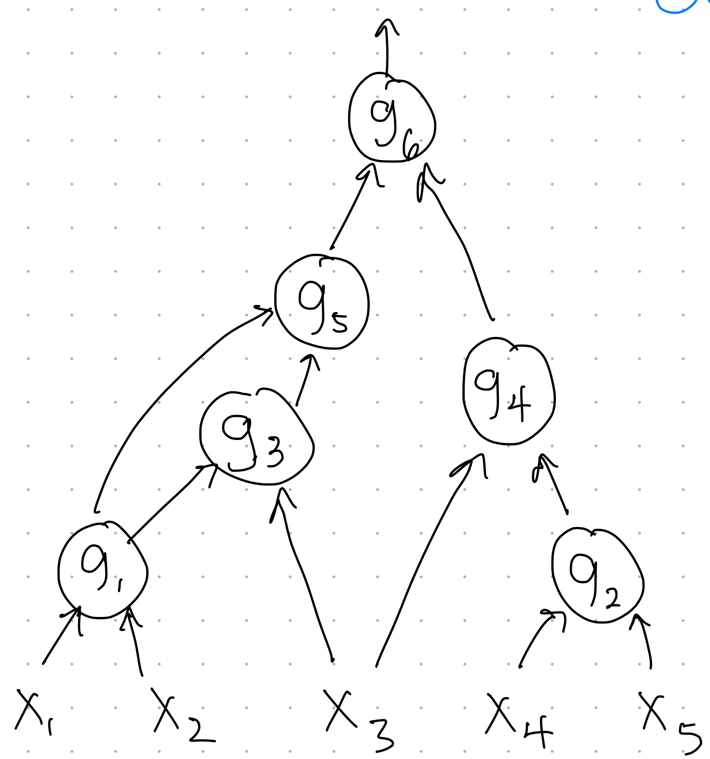
Circuits.

* Represented as a DAG
* Vertices ≡ "Gates"
* Edges ≡ "Wires"

// Each gate computes a boolean fn. on 2-variables

→ n total input wires to circuit

→ Output determined by evaluating each gate from bottom to top.



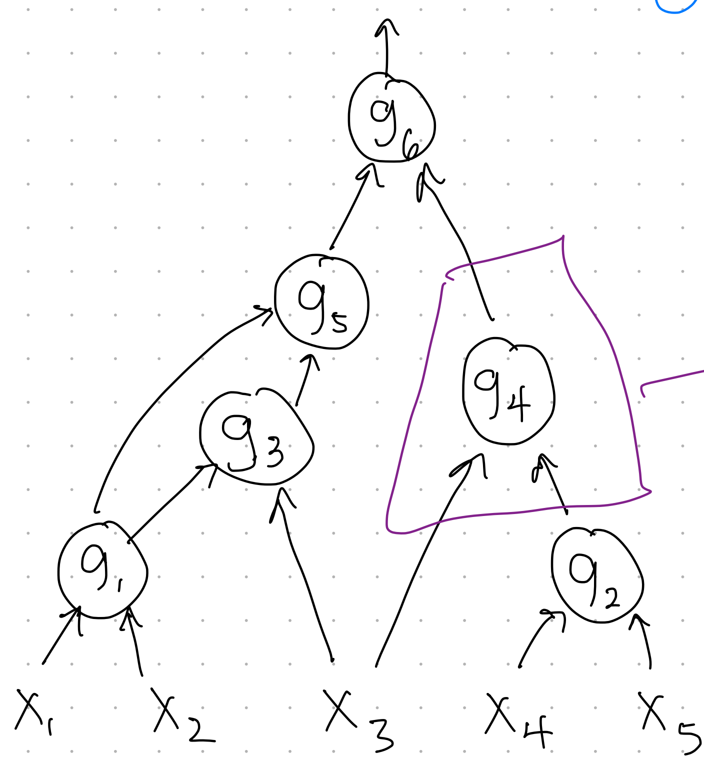
Circuits.

* Represented as a DAG
* Vertices ≡ "Gates"
* Edges ≡ "Wires"

// Each gate computes a boolean fn. on 2-variables

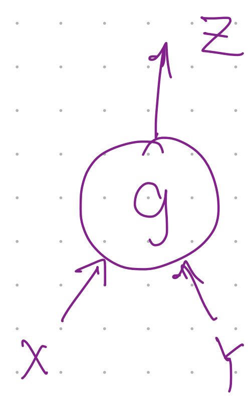
→ n total input wires to circuit

→ Output determined by evaluating each gate from bottom to top.



Reduction idea

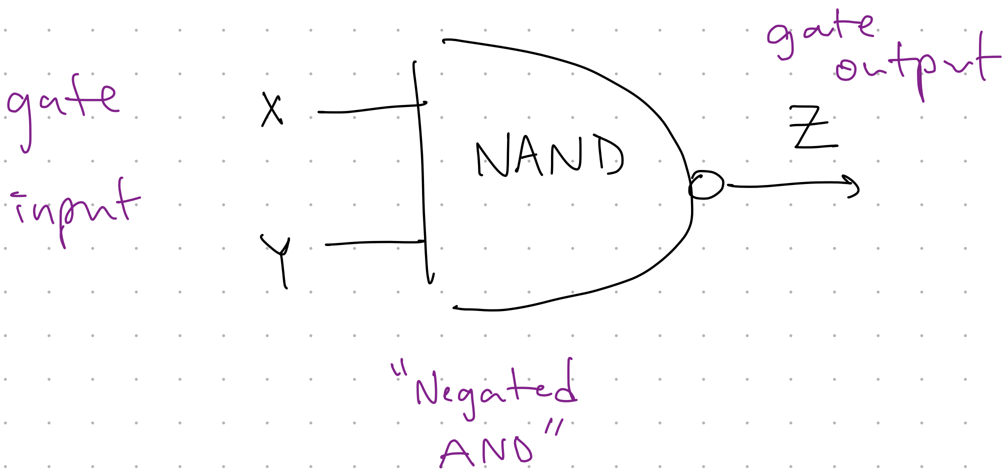
For each gate



Check correctness

$$z \longleftrightarrow g(x, y)$$

Verify each gate

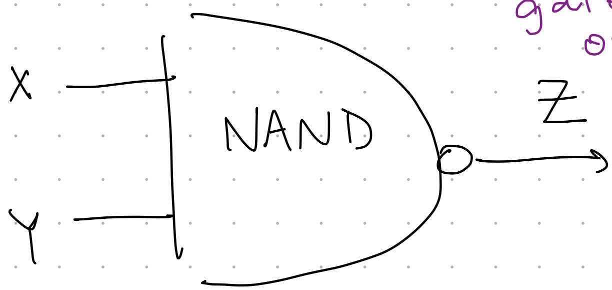


X	Y	Z = NAND(X, Y)
0	0	1
1	0	1
0	1	1
1	1	0

Note: NAND is Complete

Verify each gate

gate input



gate output

"Negated AND"

X	Y	Z = NAND(X, Y)
0	0	1
1	0	1
0	1	1
1	1	0

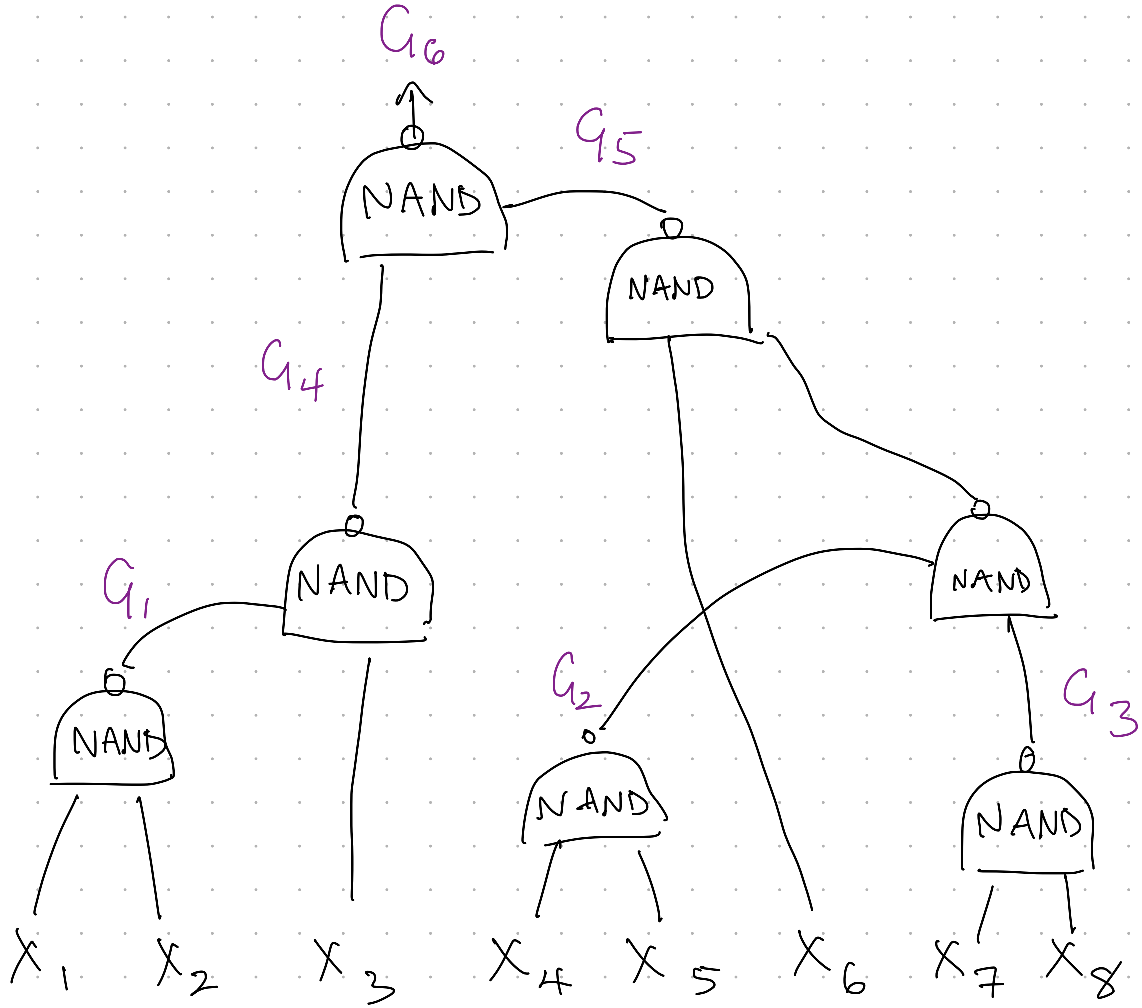
Note: NAND is Complete

GATE GADGET. $\longrightarrow$ Set of clause satisfied by any X, Y, Z s.t. Z = NAND(X, Y)

$$Z \leftrightarrow \neg(X \wedge Y)$$

$$\Leftrightarrow \quad \underbrace{Z \to \neg(X \wedge Y)}_{(\neg Z \vee \neg X \vee \neg Y)} \quad \wedge \quad \underbrace{\neg Z \to (X \wedge Y)}_{(Z \vee X) \wedge (Z \vee Y)}$$

$$(\neg Z \vee \neg X \vee \neg Y) \wedge (Z \vee X) \wedge (Z \vee Y)$$



poly-sized CNF satisfiable iff $\exists\, x \in \{0,1\}^8$ s.t. $C(x) = 1$.

$$\varphi_C = G_6 \wedge \Big[(\neg G_1 \vee \neg x_1 \vee \neg x_2) \wedge (G_1 \vee x_1) \wedge (G_1 \vee x_2) \quad \text{(gate 1 gadget)}$$

$$\wedge\ (\neg G_2 \vee \neg x_4 \vee \neg x_5) \wedge (G_2 \vee x_4) \wedge (G_2 \vee x_5) \quad \text{(gate 2 gadget)}$$

$$\vdots$$

$$\wedge\ (\neg G_6 \vee \neg G_4 \vee \neg G_5) \wedge (G_6 \vee G_4) \wedge (G_6 \vee G_5) \Big] \quad \text{(gate 6 gadget)}$$