Algorithms 2020

NP-Hardness: Reductions

Recap

- HW: oral grading next Thursday/Friday.
  - Sign up for a group and a time!
- Reading as usual

P, vs. NP, + co-NP

Consider only decision problems:

so Yes/No output

P: Set of decision problems that can be solved in polynomial time.

?||

Ex essentially any thing we've seen (except backtracking, exp time problem)

NP: Set of problems such that, if the answer is yes + you hand me ~~proof~~ certificate, I can verify/check in polynomial time.

Ex: CIRCUIT SAT

Co-NP: Can verify a "No" answer.

↳ Primality of a #.

no is easy!

Dfn: NP-Hard

X is NP-Hard

$\iff$

If X could be solved in polynomial time, then

P = NP.

Recall P & NP?

So if <u>any</u> NP-Hard problem could be solved in polynomial time, then all of NP could be.

To prove NP-Hardness of A:

Reduce a known NP-Hard problem to A.

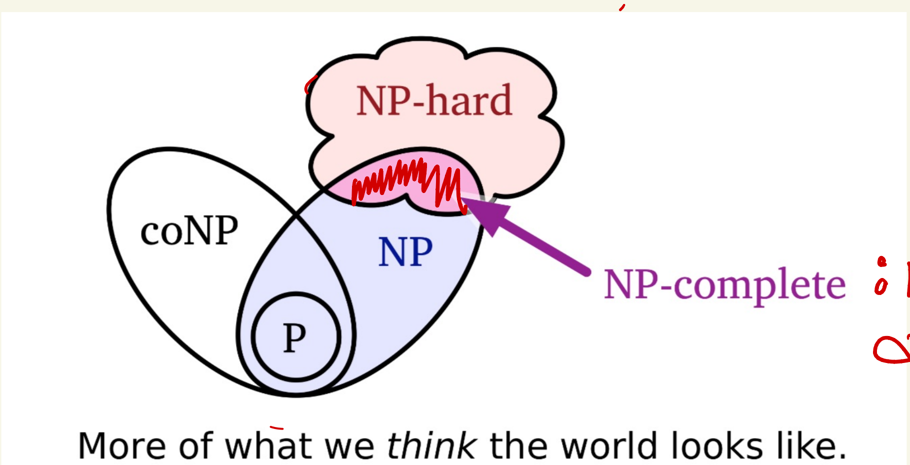
our approach

First known NP-Hard problem:

Cook-Levine Thm:

Circuit SAT is NP-Hard.

Gospel truth:



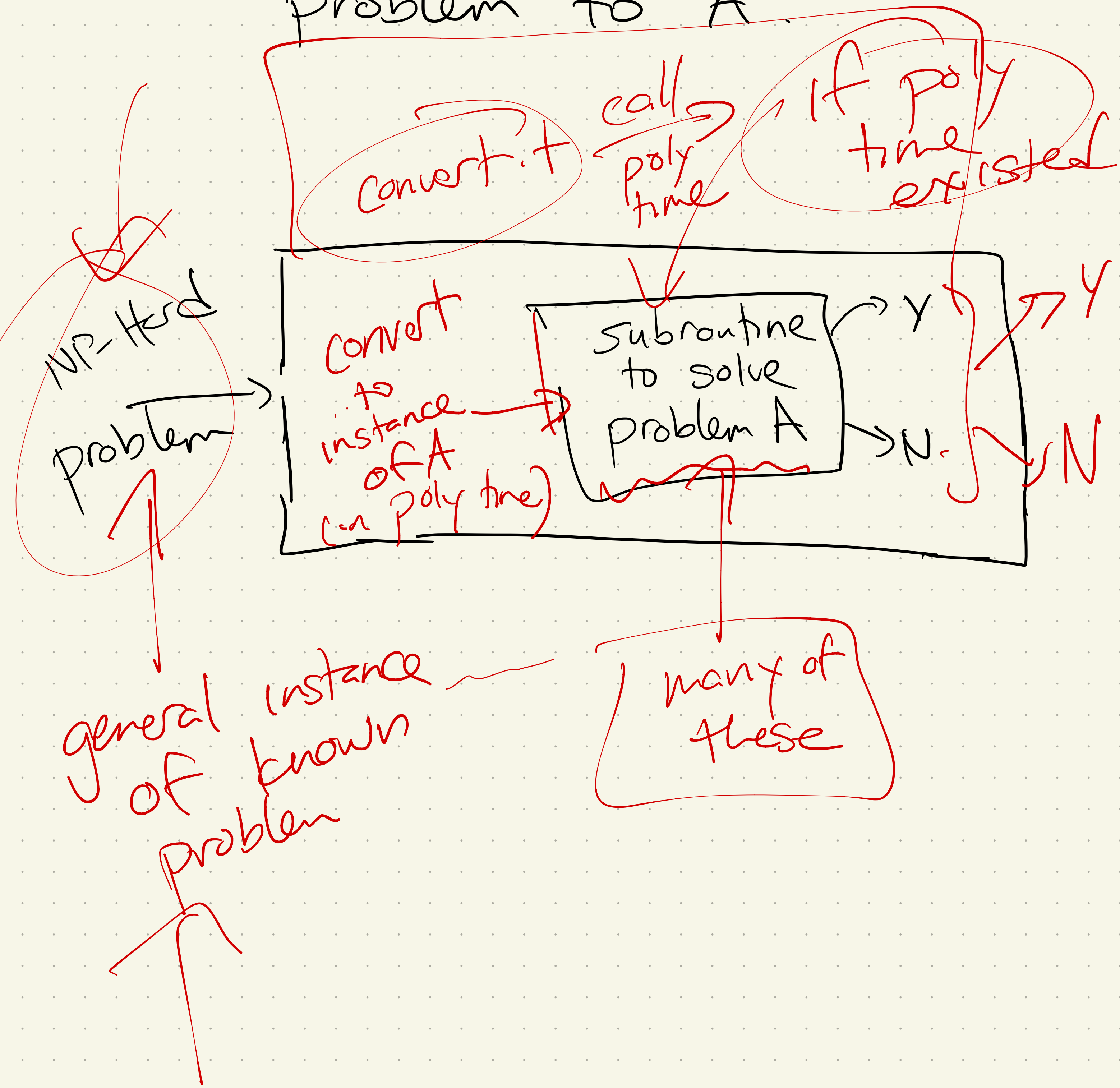
More of what we *think* the world looks like.

: in NP & NP-Hard

Proof: involves simulating any Turing machine w/ a circuit.

To prove NP-Hardness of A:

Reduce a known NP-Hard problem to A.



This will feel odd, though:

To prove a new problem is hard, we'll show how we could solve a known hard problem using new problem as a subroutine.

Why?

Well, if a poly time algorithm existed, than you'd also be able to solve the hard problem!

(Therefore, "can't" be any such solution.)

Other NP-Hard Problems:

SAT: Given a boolean formula, is there a way to assign inputs so result is 1?

and, or, not, ⇒, = + ≠, ...

Ex: $(a \vee b \vee c \vee d) \Leftrightarrow ((b \wedge \bar{c}) \vee \overline{(\bar{a} \Rightarrow d)} \vee (c \neq a \wedge b)),$

T T T D=T F

n variables, here! a..d $x_1, \ldots, x_n$

m clauses: 5 = m

In NP: we can check "yes" in poly time!

Given n T/F inputs, scan the clauses & check each in O(1)

poly!

Total: O(n+m)

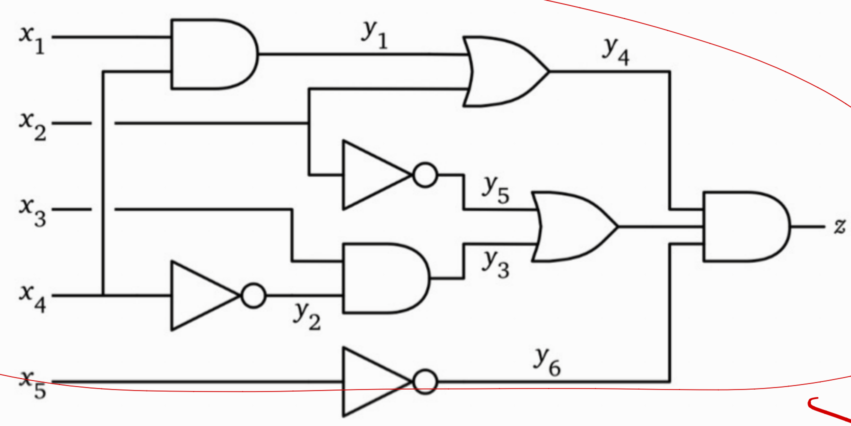
nm if clauses are O(n) long

Thm: SAT is NP-Hard.

only known NP-Hard

Pf: Reduce CIRCUIT SAT to SAT:

new one



poly

SAT

$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge$$
$$(y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (z = y_4 \wedge y_7 \wedge y_6) \wedge z$$

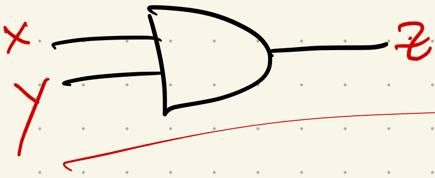
A boolean circuit with gate variables added, and an equivalent boolean formula.

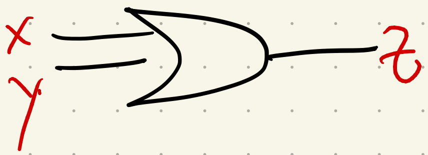
Given circuit, write equivalent boolean SAT expression

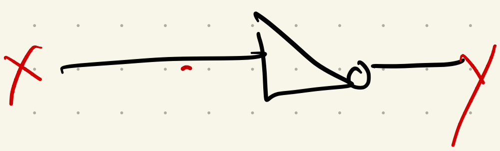
Need to use input: Circuit

More carefully: input is in form of gates

1) For any gate, can transform:

AND  : $(x \wedge y = z)$

OR  : $(x \vee y = z)$

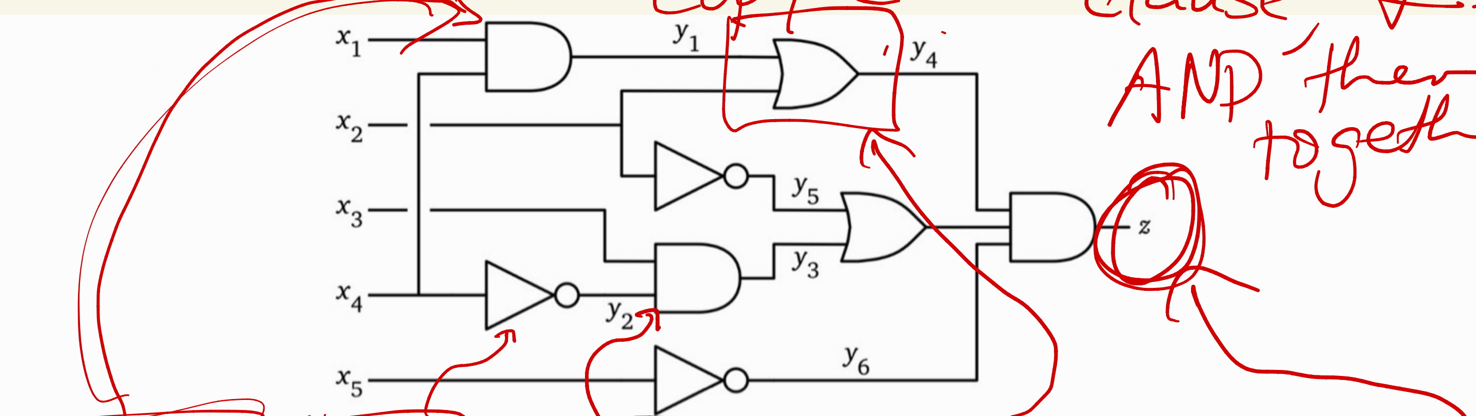
NOT  : $(\overline{x} = y)$

(written $\neg x = y$)

2) "And" these together, + want final output true:

move left to right

copy each clause, + AND them together



$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge$$
$$(y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (z = y_4 \wedge y_7 \wedge y_6) \wedge z$$

forces output wire to be correct T/F value

Note:

Some boolean expressions can't be result of this conversion

circuit

converts (poly time)

uses SAT

Is this poly-size?

Given n inputs + m gates in circuit:

Variables: 1 per input + gate

Clauses: 1 per gate

≤ 3 variables per clause

↳ result is m clauses, + $m+n$ variables in SAT expression

So our reduction:

$$\begin{tikzcd}
\text{boolean circuit} \arrow[r, "\cancel{O(n)}\ O(n+m)"] & \text{boolean formula} \arrow[d, Rightarrow, "\text{SAT}"] \\
\text{True or False} & \text{True or False} \arrow[l, "\text{trivial}"']
\end{tikzcd}$$

$$T_{\text{CSAT}}(n) \le O(n) + T_{\text{SAT}}(O(n)) \quad \Longrightarrow \quad T_{\text{SAT}}(n) \ge T_{\text{CSAT}}(\Omega(n)) - O(n)$$

Thm: 3SAT is NP-Hard ← exactly 3 variables per clause, all clauses are OR, all AND-ed together

pf: Reduce circuitSAT to 3SAT.

↑ new ↑ known

Need to show any circuit can be transformed to 3CNF form

(so last reduction fails)

OR: $\vee$

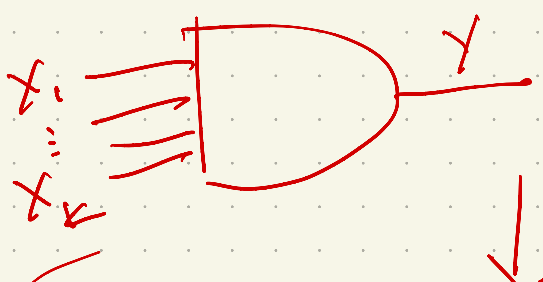
AND: $\wedge$

Ex: $(x \vee y \vee z) \wedge (a \vee b \vee c) \cdots$

OR AND

Steps:

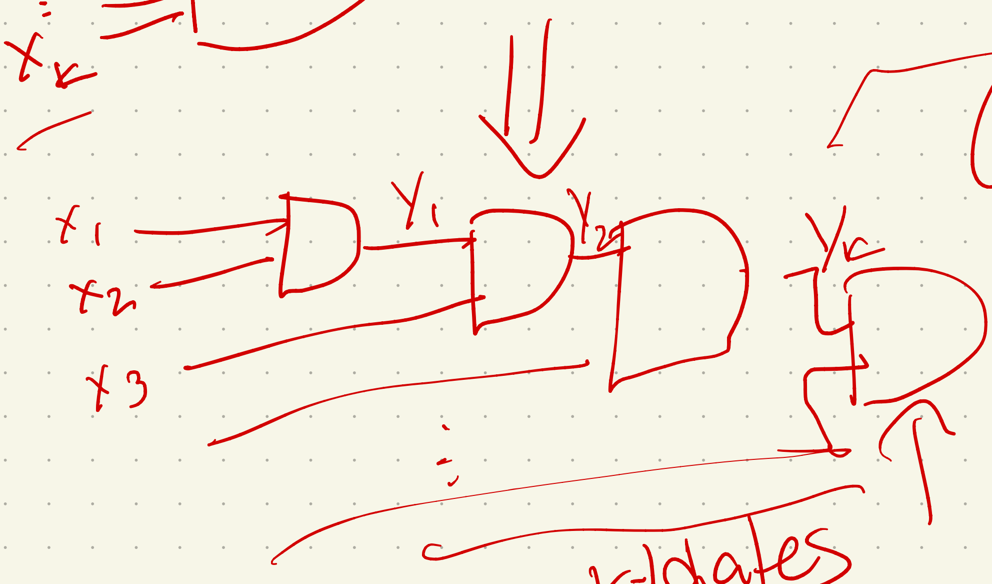
① Rewrite so each gate has 2 inputs:



$$(x_1 \vee x_2 \vee \cdots \vee x_k = y)$$

$$\Downarrow$$

$$(x_1 \vee x_2 = y_1) \wedge (y_1 \vee x_3 = y_2) \cdots$$



k-1 gates

(2) Write formula, like in SAT:

3 types

$y = a \vee b \quad \xleftrightarrow{\text{equiv.}} \quad (y \vee \bar{a} \vee b) \wedge (\bar{y} \vee b) \wedge (\bar{y} \vee a)$

$y = a \wedge b$

$y = \bar{a}$

a	b	y	
0	0	0	T
0	0	1	F
0	1	0	F
0	1	1	T
1	0	0	F
1	0	1	T
1	1	0	F
1	1	1	T

a	b	y
-	F	F

same!

(3) Now, change to CNF:

go back to truth tables

$$a = b \wedge c \longleftrightarrow \boxed{(a \vee \bar{b} \vee \bar{c})} \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c)$$
$$a = b \vee c \longleftrightarrow \boxed{(\bar{a} \vee b \vee c)} \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c})$$
$$a = \bar{b} \longleftrightarrow (a \vee b) \wedge (\bar{a} \vee \bar{b})$$

only 2

(4) Now, need 3 per clause! plus

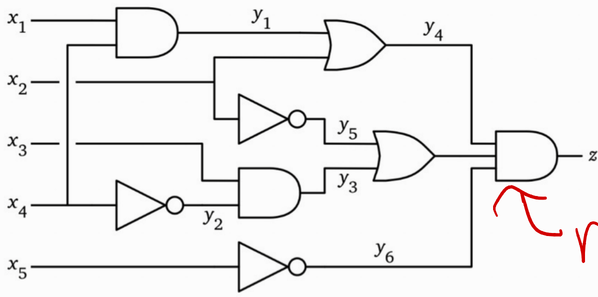
introduce dummy variables

$$a \longrightarrow (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge (a \vee x \vee \bar{y}) \wedge (a \vee \bar{x} \vee \bar{y})$$
$$a \vee b \longrightarrow (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$$

a	
T	F
F	F

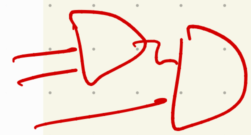
a	x	y	output
0	0	0	F
0			
0			
1			T

Note: Bigger!



Step 1

replace



$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge (y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (z = y_4 \wedge y_7 \wedge y_6) \wedge z$$

A boolean circuit with gate variables added, and an equivalent boolean formula.

rewrite, then pad w/ dummy variables (Steps 2+3)

$$\begin{aligned}
&(y_1 \vee \overline{x_1} \vee \overline{x_4}) \wedge (\overline{y_1} \vee x_1 \vee z_1) \wedge (\overline{y_1} \vee x_1 \vee \overline{z_1}) \wedge (\overline{y_1} \vee x_4 \vee z_2) \wedge (\overline{y_1} \vee x_4 \vee \overline{z_2}) \\
&\wedge (y_2 \vee x_4 \vee z_3) \wedge (y_2 \vee x_4 \vee \overline{z_3}) \wedge (\overline{y_2} \vee \overline{x_4} \vee z_4) \wedge (\overline{y_2} \vee \overline{x_4} \vee \overline{z_4}) \\
&\wedge (y_3 \vee \overline{x_3} \vee \overline{y_2}) \wedge (\overline{y_3} \vee x_3 \vee z_5) \wedge (\overline{y_3} \vee x_3 \vee \overline{z_5}) \wedge (\overline{y_3} \vee y_2 \vee z_6) \wedge (\overline{y_3} \vee y_2 \vee \overline{z_6}) \\
&\wedge (\overline{y_4} \vee y_1 \vee x_2) \wedge (y_4 \vee \overline{x_2} \vee z_7) \wedge (y_4 \vee \overline{x_2} \vee \overline{z_7}) \wedge (y_4 \vee \overline{y_1} \vee z_8) \wedge (y_4 \vee \overline{y_1} \vee \overline{z_8}) \\
&\wedge (y_5 \vee x_2 \vee z_9) \wedge (y_5 \vee x_2 \vee \overline{z_9}) \wedge (\overline{y_5} \vee \overline{x_2} \vee z_{10}) \wedge (\overline{y_5} \vee \overline{x_2} \vee \overline{z_{10}}) \\
&\wedge (y_6 \vee x_5 \vee z_{11}) \wedge (y_6 \vee x_5 \vee \overline{z_{11}}) \wedge (\overline{y_6} \vee \overline{x_5} \vee z_{12}) \wedge (\overline{y_6} \vee \overline{x_5} \vee \overline{z_{12}}) \\
&\wedge (\overline{y_7} \vee y_3 \vee y_5) \wedge (y_7 \vee \overline{y_3} \vee z_{13}) \wedge (y_7 \vee \overline{y_3} \vee \overline{z_{13}}) \wedge (y_7 \vee \overline{y_5} \vee z_{14}) \wedge (y_7 \vee \overline{y_5} \vee \overline{z_{14}}) \\
&\wedge (y_8 \vee \overline{y_4} \vee \overline{y_7}) \wedge (\overline{y_8} \vee y_4 \vee z_{15}) \wedge (\overline{y_8} \vee y_4 \vee \overline{z_{15}}) \wedge (\overline{y_8} \vee y_7 \vee z_{16}) \wedge (\overline{y_8} \vee y_7 \vee \overline{z_{16}}) \\
&\wedge (y_9 \vee \overline{y_8} \vee \overline{y_6}) \wedge (\overline{y_9} \vee y_8 \vee z_{17}) \wedge (\overline{y_9} \vee y_8 \vee \overline{z_{17}}) \wedge (\overline{y_9} \vee y_6 \vee z_{18}) \wedge (\overline{y_9} \vee y_6 \vee \overline{z_{18}}) \\
&\wedge (y_9 \vee z_{19} \vee z_{20}) \wedge (y_9 \vee \overline{z_{19}} \vee z_{20}) \wedge (y_9 \vee z_{19} \vee \overline{z_{20}}) \wedge (y_9 \vee \overline{z_{19}} \vee \overline{z_{20}})
\end{aligned}$$

!!

all z's are dummies

How big?

Step 1: 1 gate ⇒ #inputs − 1 gates

$m \longrightarrow \leq m \cdot n$

Step 2: clause ⟶ ≤ 3 clauses

Step 3: ≤ 4

O(mn clauses)

O(mn inputs)

Still polynomial:

$O(mn)$ clauses + variables

$O(mn)$

<u>So:</u>

$$
\begin{tikzcd}
\text{boolean circuit} \arrow[r, "\cancel{O(n)}"] & \text{3CNF formula} \arrow[d, Rightarrow, "\text{3SAT}"] \\
\textsc{True or False} & \textsc{True or False} \arrow[l, "\text{trivial}"']
\end{tikzcd}
$$

$$T_{\mathrm{CSAT}}(n) \le O(n) + T_{\mathrm{3SAT}}(O(n)) \quad \Longrightarrow \quad T_{\mathrm{3SAT}}(n) \ge T_{\mathrm{CSAT}}(\Omega(n)) - O(n)$$

Circuit is SATisfiable

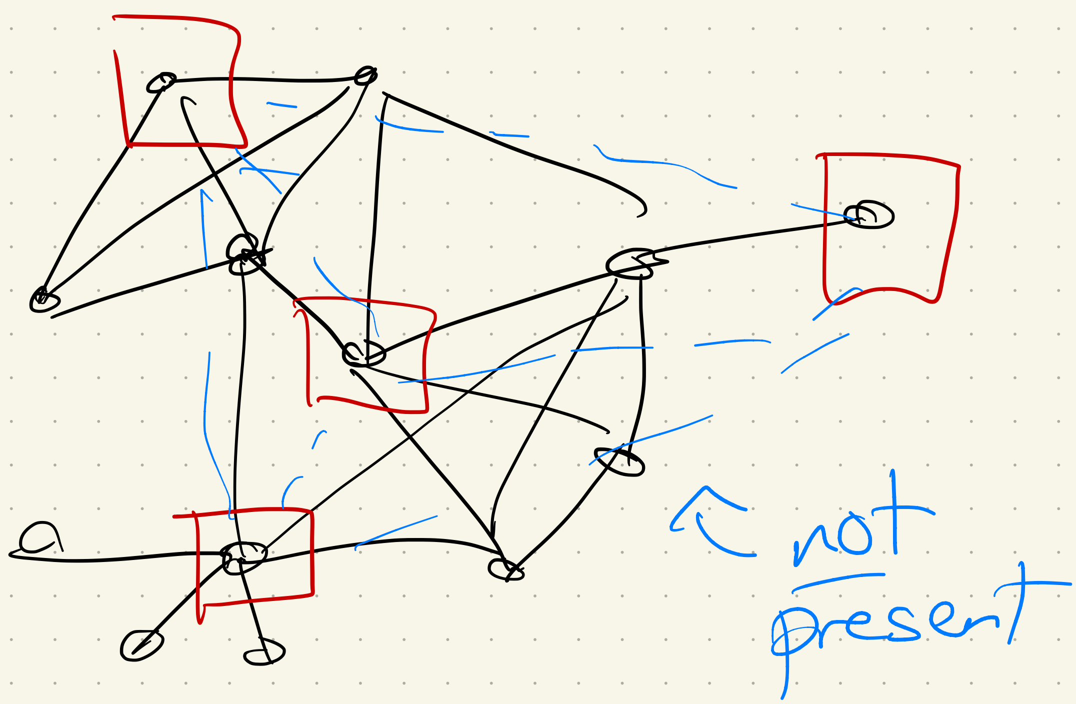
$\Updownarrow$

3SAT is Satisfiable

Next Problem: looks more useful

Independent Set:

A set of vertices in a graph with no edges between them:



decision version:

Is there a ind set of size k in G?

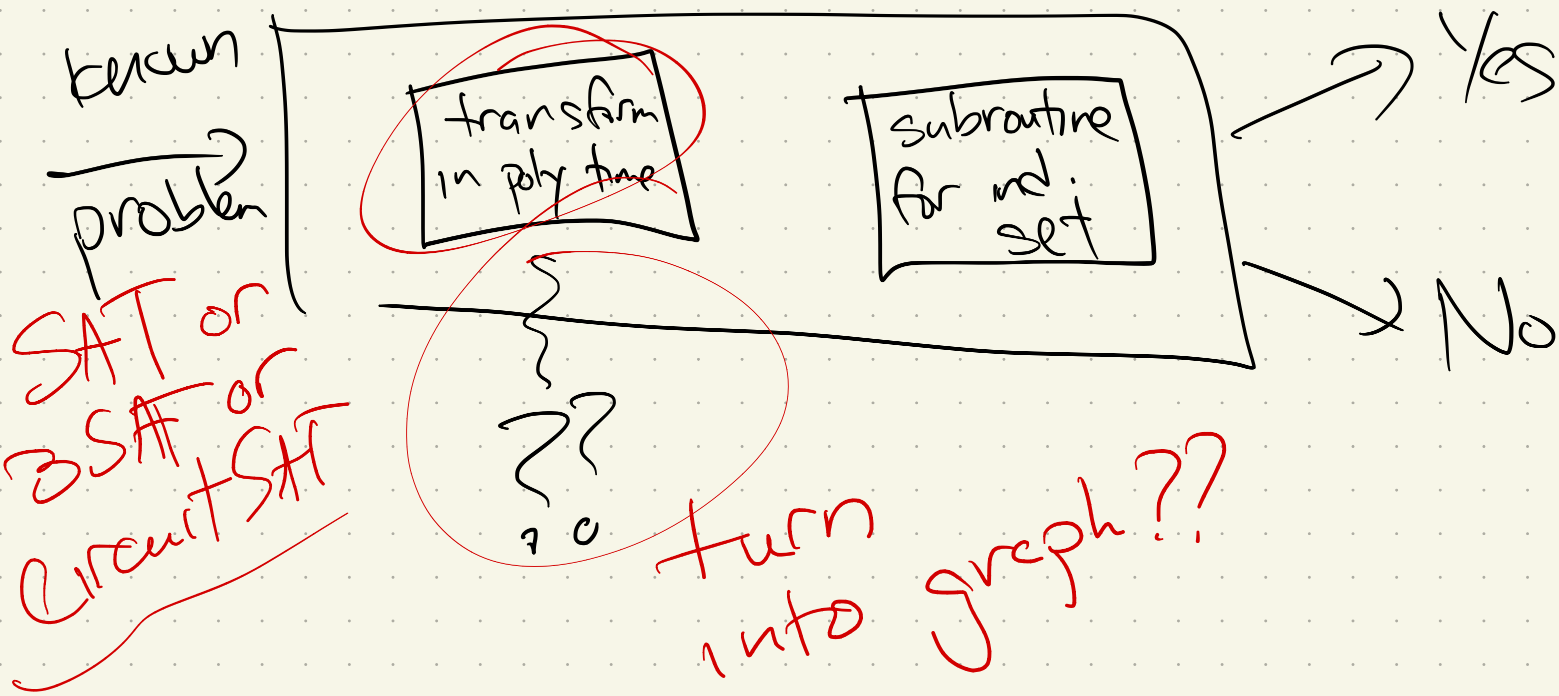
(Wait - didn't we see this already!?)

input: k & G

solves in trees ~~DAGs?~~

Challenge: No booleans!

But reduction needs to take known NP-Hard problem + build a graph!



We'll use 3SAT

(but stop and marvel a bit first...)

Reduction:

m clauses
n literals (variables)

Input is 3CNF boolean formula

AND

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

OR

1. Make a vertex for each literal in <u>each</u> clause

   ↳ 3m vertices

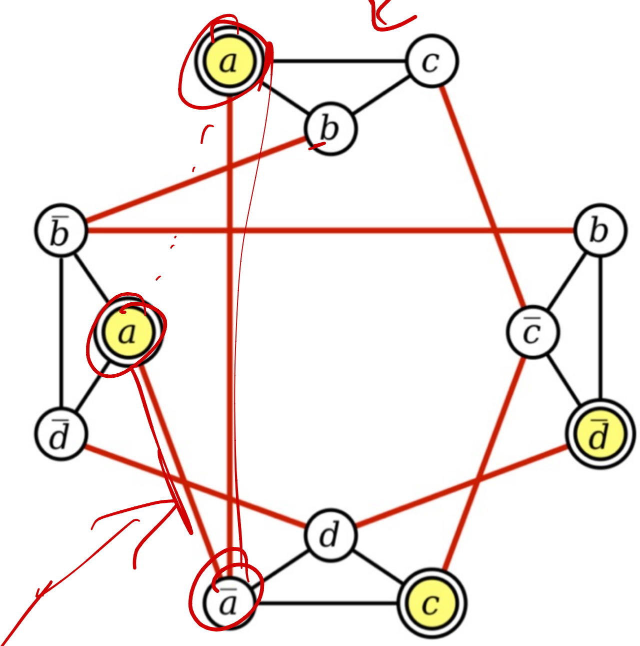
2. Connect two vertices if:

   - they are in same clause
   - they are a variable & its <u>inverse</u>

Example :

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

clause



A graph derived from a 3CNF formula, and an independent set of size 4.

go between var & its negation

Claim:

formula is satisfiable

$\Longleftrightarrow$

G has independent set of size m

Proof:

So!

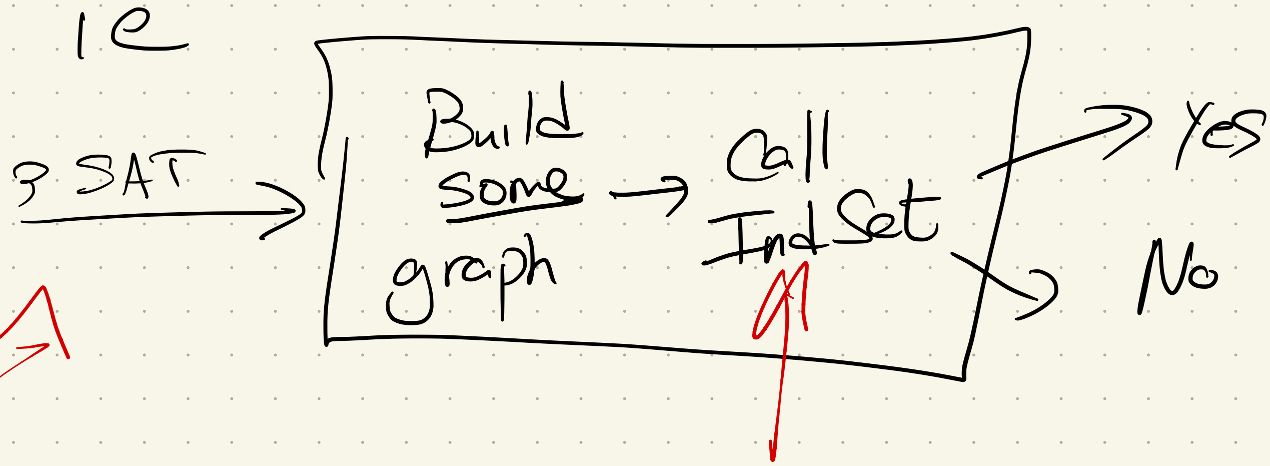
$$\begin{tikzcd}
\boxed{\text{3CNF formula with } k \text{ clauses}} \arrow[r, "O(n)"] & \boxed{\text{graph with } 3k \text{ nodes}} \arrow[d, Rightarrow, "\textsc{MaxIndSet}"] \\
\boxed{\textsc{True} \text{ or } \textsc{False}} & \boxed{\text{maximum independent set size}} \arrow[l, "O(1)"']
\end{tikzcd}$$

$$T_{3\text{SAT}}(n) \le O(n) + T_{\textsc{MaxIndSet}}(O(n)) \quad \Longrightarrow \quad T_{\textsc{MaxIndSet}}(n) \ge T_{3\text{SAT}}(\Omega(n)) - O(n)$$

The Pattern:

1) Find an NP-Hard problem, & solve it using unknown problem as a subroutine

ie



Proof:

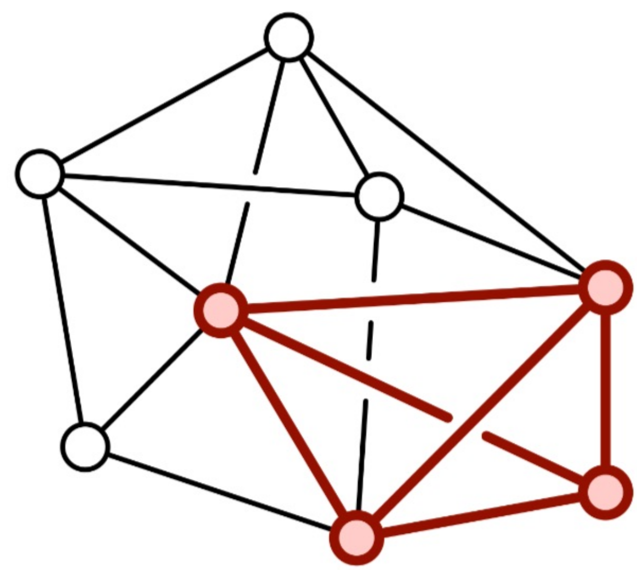
Need if & only if!

(ie might be some wierd indep. set that doesn't make a SAT)

Challenge: Finding correct NP Hard problem

Next one: Clique #

A clique in a graph is a subgraph which is complete - all possible edges are present.



A graph with maximum clique size 4.

How could we check if G has a clique of size k?

Decision version: Does G have a clique of size k?

<u>Input</u>:

<u>Output</u>:

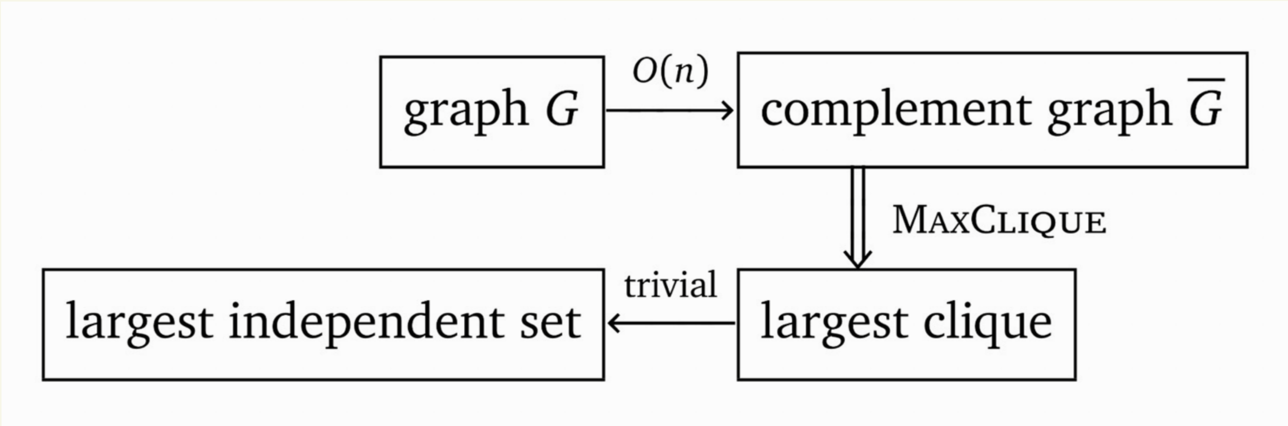
<u>This is NP-Complete</u>:

1. In NP. Why?

2 NP-Hard:

What should we reduce to k-Clique?

So:


graph G
$O(n)$
complement graph $\overline{G}$
MaxClique
largest clique
trivial
largest independent set