Complexity & Algorithms, Spring '26

Complexity:
Hardness

Recap

- Almost done w/ HW2
  - ↳ hand papers back Thurs
- HW3 up soon
  - ↳ due at end of next week
- Midterm: March 3 ←
  - ↳ covering up to approx.
    (not NP-Hard)

Quantifying Hardness:

Fundamental question:

Are there "harder" problems? (Yes)

Why should we care?

Good to know what computers can do!

How do we rank? space

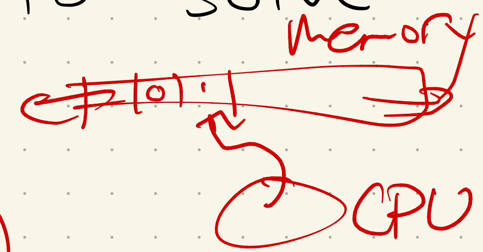
in terms of big-O runtime

polynomial = have hope for practical solutions

↳ levels of hardness

The bad news: Undecidability

Some problems are impossible to solve!



The Halting Problem: – Turing

Given a program P and input I, does P halt or run forever if given I?

Output: True/False

(Utility should be obvious!)

Note: Can't just simulate P on I.

Why? If I stop while P(I) is still going

↳ could stop in next round.

Thm [Turing 1936]:

The halting problem is undecidable.

(That is, no such algorithm can exist.)

Proof by contradiction - suppose we have such a program, H.

$$\underset{\text{code}}{H}(\underset{\text{code}}{P}, \underset{\text{input}}{I}) = \begin{cases} \text{True} & \text{if } P(I) \text{ halts} \\ \text{False} & \text{if } P(I) \text{ loops forever} \end{cases}$$

Need to find a contradiction now...

Define a program G, which uses H as a subroutine:

G(X): if H(X, X) = false
 return false
 else
 loop forever

0's + 1's

So: if X(x) halts, loop forever
 if X(x) loops, halt + return false

How to break something

↳ run G on itself

Now: What does $G(G)$ do?

If $H(G,G) = \text{false}$, then halts

↳ but if $H(G,G)$ is false, means $G(G)$ has infinite loop!

If $H(G,G) = \text{true}$, then loops forever

↳ but if $H(G,G)$ is true, means $G(G)$ halts.

Logical contradiction

↳ no such function exists.

So... what next?

Clearly, many things *are* solvable in polynomial time.

undecidable halting

Some things are impossible

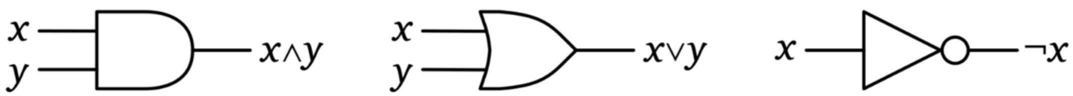
But – what is in between?

P

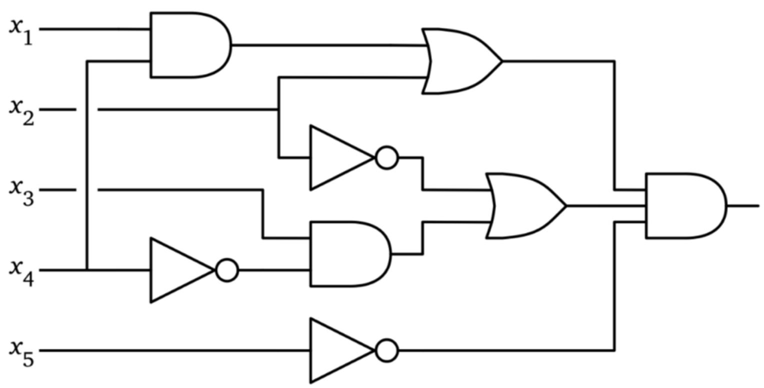
Idea:

Try to formalize a notion of "hardness", to better understand what computation can do.

The first problem found: Boolean circuits

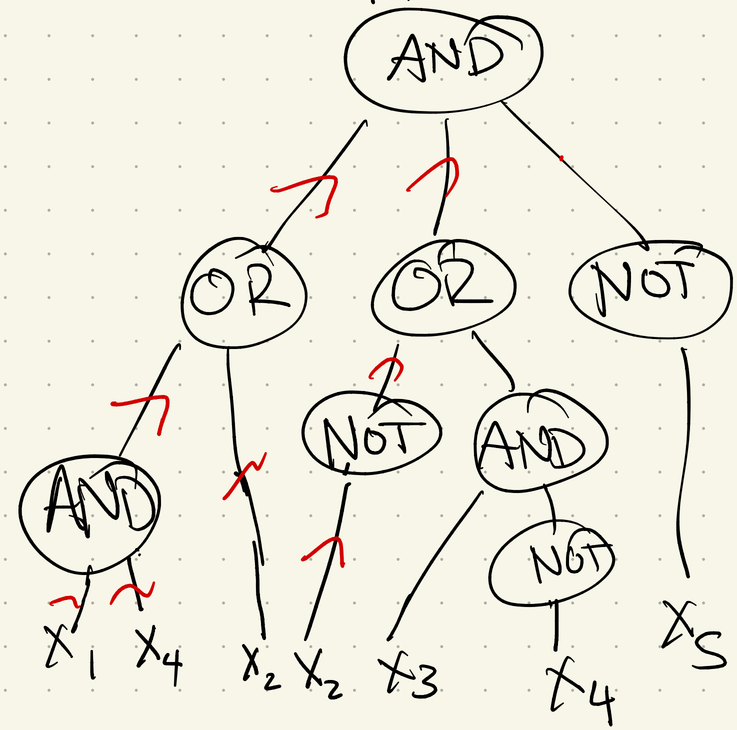


An AND gate, an OR gate, and a NOT gate.



A boolean circuit. inputs enter from the left, and the output leaves to the right.

"Flipped view":

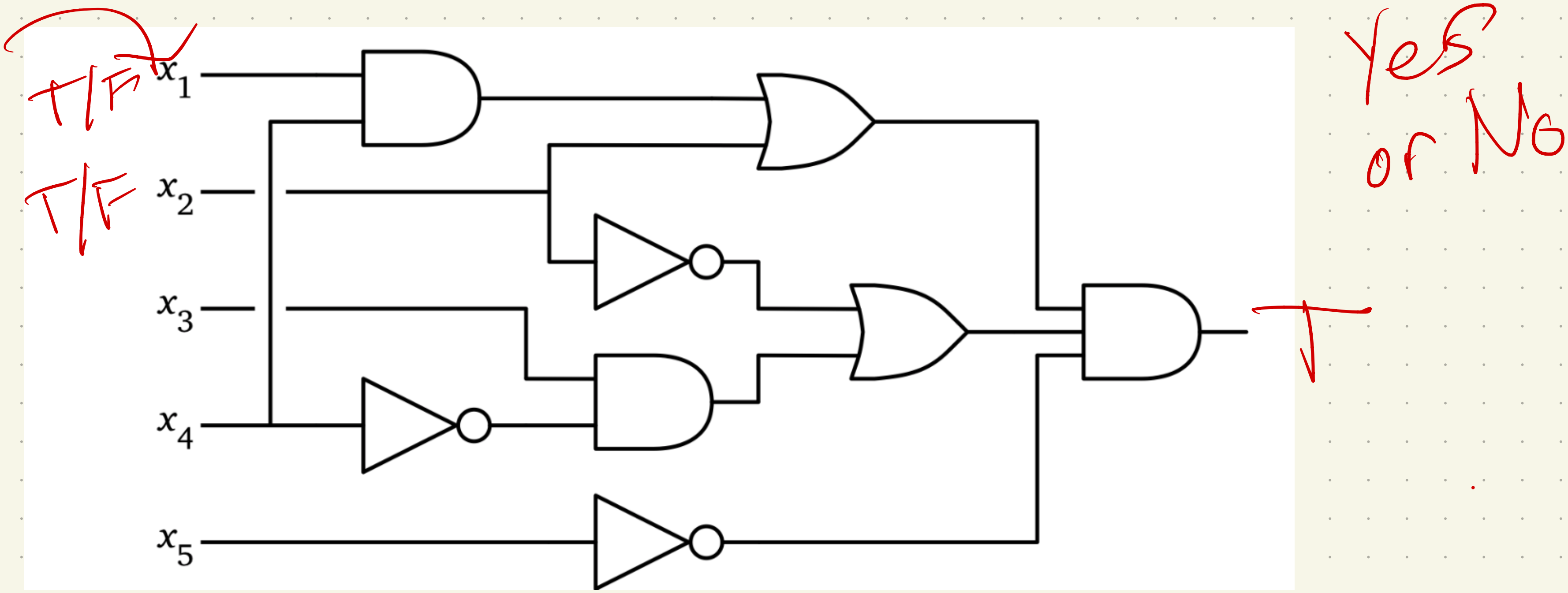


$O(m+n)$

Given a set of inputs can clearly calculate output in linear time (in # inputs + # gates).

How? "bottom" up algorithm: postorder or topological sort

Q: Given such a boolean circuit, is there a set of inputs which result in TRUE output?



Known as CIRCUIT SATISFIABILITY (or CIRCUIT SAT)

Best known algorithm: $\leftarrow 2^n$

Try all possible inputs

If one works, return true

else return false

Runtime: $2^n \cdot O(n+m)$

Note: Best known: $O(2^n)$

P, NP, + co-NP

Consider only decision problems: so Yes/No output

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

Examples: Is this list sorted? (see book)
Is x in the list?

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

Examples: CircuitSAT, Is list sorted?

$P \subseteq NP$

Co-NP: Set of problems where we can verify a "no" (in poly time)

Examples: Test if n is a prime number.

<u>Dfn</u>: NP-Hard

X is NP-Hard

$\iff$

If X could be solved in polynomial time, then <u>P=NP</u>.

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

Note: Not <u>at all</u> obvious any such problem exists!

Cook-Levine Thm:

Circuit SAT is NP-Hard.

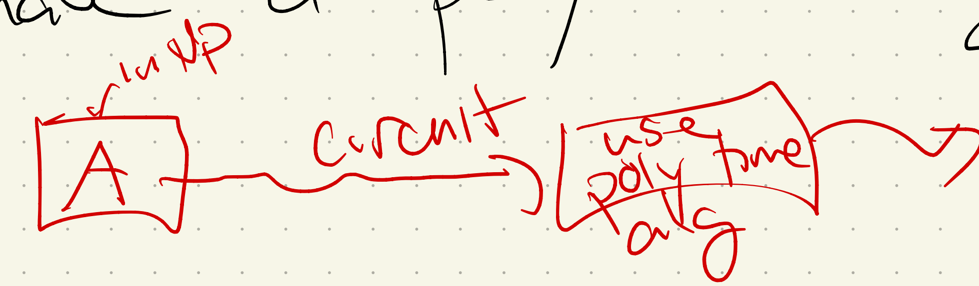
Proof (sketch):

Suppose I have an algorithm to solve CIRCUIT-SAT in poly time.

Take any problem in NP, A.

Reduce A to CIRCUIT-SAT in polynomial time: build circuit.

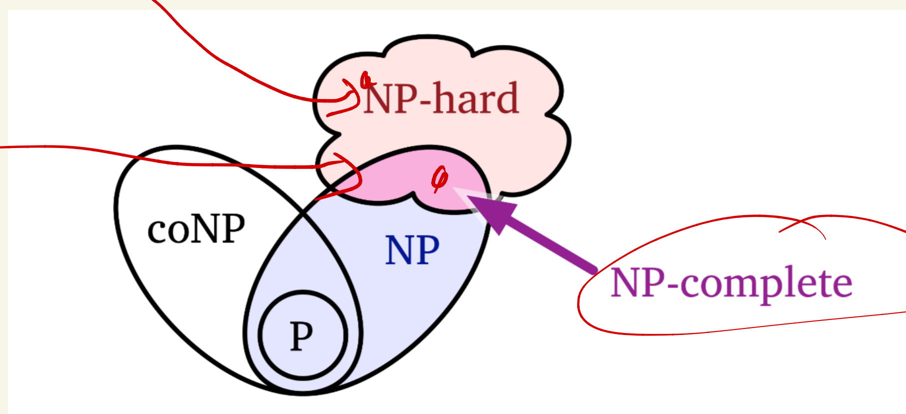
Therefore, I have a poly time alg for A.



So, there is at least one problem that is NP-Hard, & in NP, but which we don't think is in P:

Circuit-sat is · in NP · NP-Hard

min VC



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

is there a VC of size k

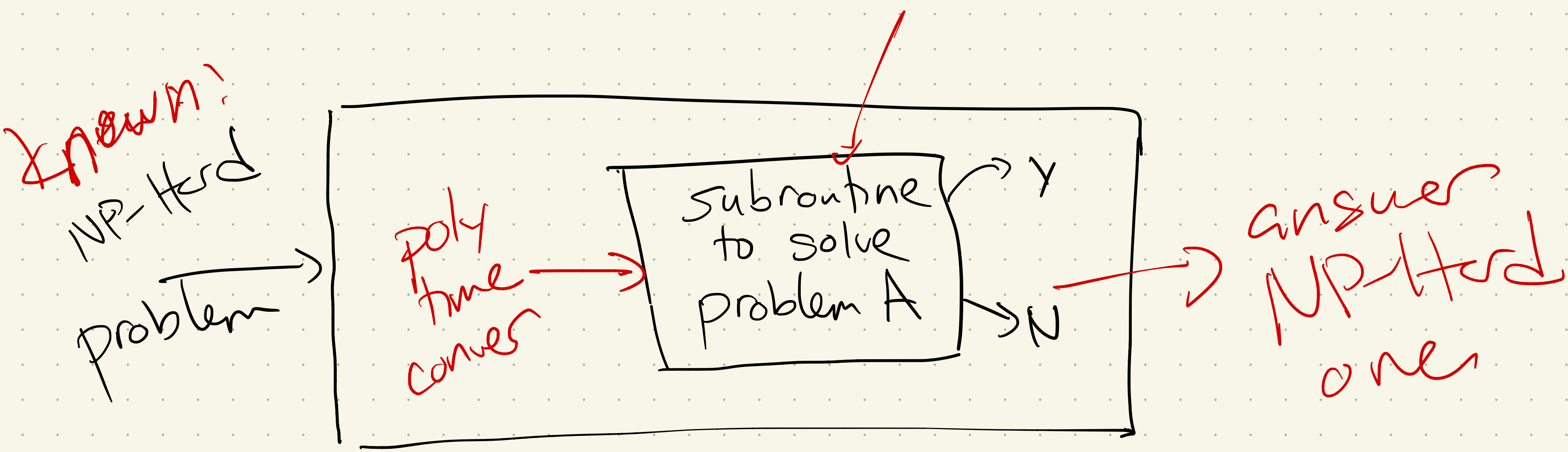
NP-Complete: in NP and NP-Hard

Is no proof that CircuitSAT is not in P.

To prove NP-Hardness of A:

Reduce a known NP-Hard problem to A.

(Alternative is to show any problem in NP can be turned into A, like Cook.)



You've used reductions before:

Fastest way to drive from A to B $\longrightarrow$

turn map into graph poly time

run Dijkstra

run alg

driving route

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? Just like halting problem!

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

(Therefore, "can't" be any such alg)

Other NP-~~Hard~~ Complete Problems:

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

T

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

n variables, m clauses

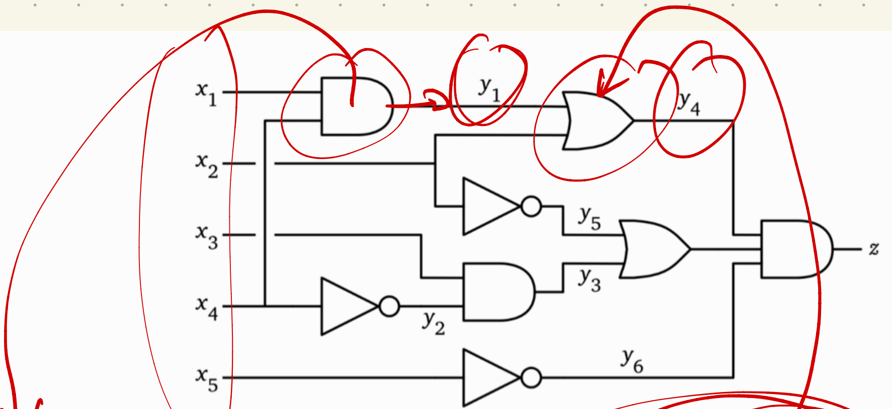
First: in NP?

given n inputs: evaluate each expression $\rightarrow$ Truth

$O(m+n)$

Thm: SAT is NP-Hard.

pf: Reduce CIRCUIT SAT to SAT:



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

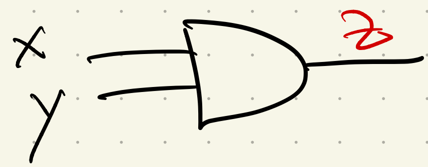
A boolean circuit with gate variables added, and an equivalent boolean formula.

Input: CIRCUIT

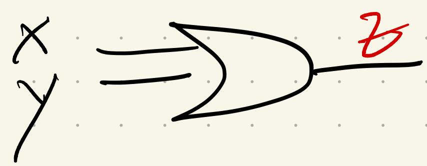
Convert in poly time to clauses:

More carefully:

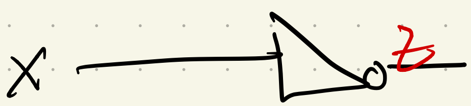
1) For any gate, can transform:



: $z = x \wedge y$



: $z = x \vee y$



: $z = \neg x$

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

Circuit can be sat.

$\Longleftrightarrow$ Formula can be sat

Is this poly-size?

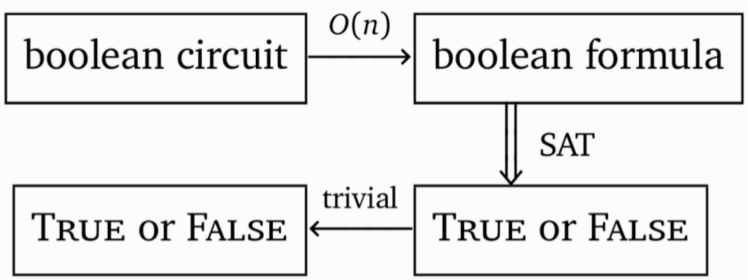
Given n inputs + m gates:

In formula

Variables: $n+m$

Clauses: $m \longrightarrow O(m+n)$

So our reduction:



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

3SAT: 3CNF formulas:

3 variables in each clause, OR-ed together

$(x \vee \bar{y} \vee z)$

↳ and the clauses

Thm: 3SAT is NP-Hard

pf: Reduce circuitSAT to 3SAT:

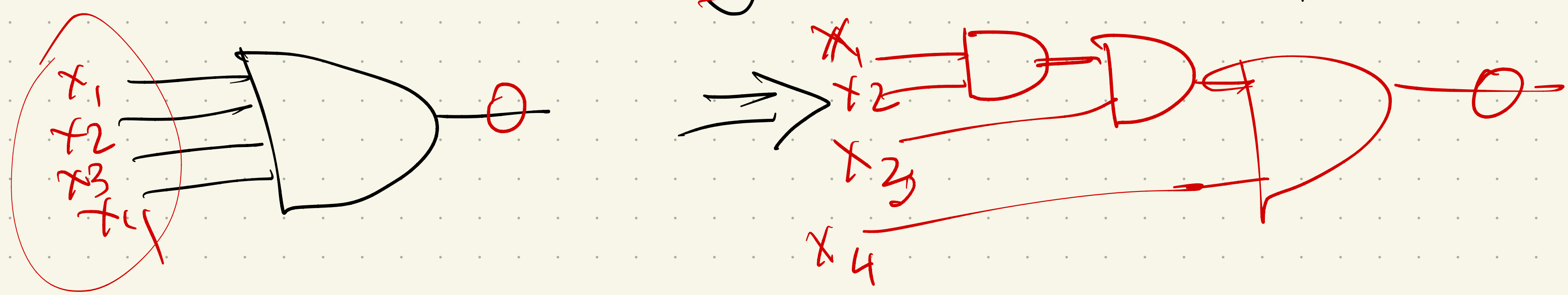
Need to show any circuit can be transformed to 3CNF form

(so last reduction fails) $Y_1 = X_1 \vee X_2$

Instead ⟶

Given a Circuit!

① Rewrite so each gate has ≤ 2 inputs:



② Write formula, like SAT. Only 3 types!

$$y = a \vee b$$
$$y = a \wedge b$$
$$y = \overline{a}$$

convert

③ Now, change to CNF:
go back to truth tables

a	b	c
T	T	T
T	T	F

$$a = b \wedge c \quad \longmapsto \quad (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c)$$

$$a = b \vee c \quad \longmapsto \quad (\bar{a} \vee b \vee c) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c})$$

$$a = \bar{b} \quad \longmapsto \quad (a \vee b) \wedge (\bar{a} \vee \bar{b})$$

logical equiv.

Not 3 variables

④ Now, need 3 per clause: insert dummy variable

$$a \quad \longmapsto \quad (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge (a \vee x \vee \bar{y}) \wedge (a \vee \bar{x} \vee \bar{y})$$

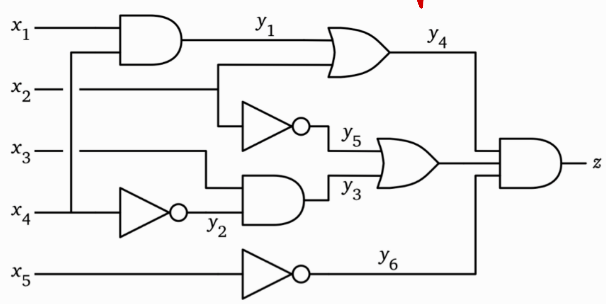
$$a \vee b \quad \longmapsto \quad (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$$

a	b	a∨b	x	$\bar{x}$
T	T	T		
T	F	T		
F	T	T		
F	F	F		

Note: Bigger!

n variables
m clauses

How much bigger?
(need polynomial)



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

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

n+m vars
+ m clauses

$\Downarrow$

$$\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}$$

12 m Clauses

$\leq 2(m+n)$ variable

So:

size? $O(n+m)$

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

$$T_{\text{CSAT}}(n) \le \cancel{O(n)} + T_{\text{3SAT}}(O(n)) \implies T_{\text{3SAT}}(n) \ge T_{\text{CSAT}}(\Omega(n)) - \cancel{O(n)}$$

$O(n+m)$ $\quad T_{3SAT} \ge T_{CSAT} - O(m+n)$

So: if could solve 3CNF, could solve CIRCUITSAT in poly time.

Historical note:

Why booleen functions?

(Think like a computer engineer for a moment...)

Next:

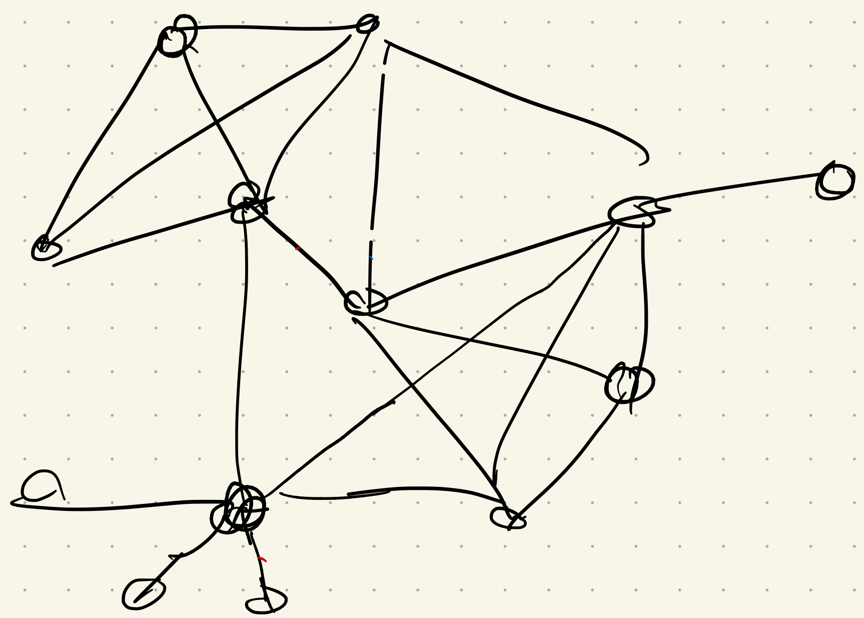
Can we do this with any <u>useful</u> problems?

(Logic is all well + good...)

Maybe → graphs?

Independent Set:

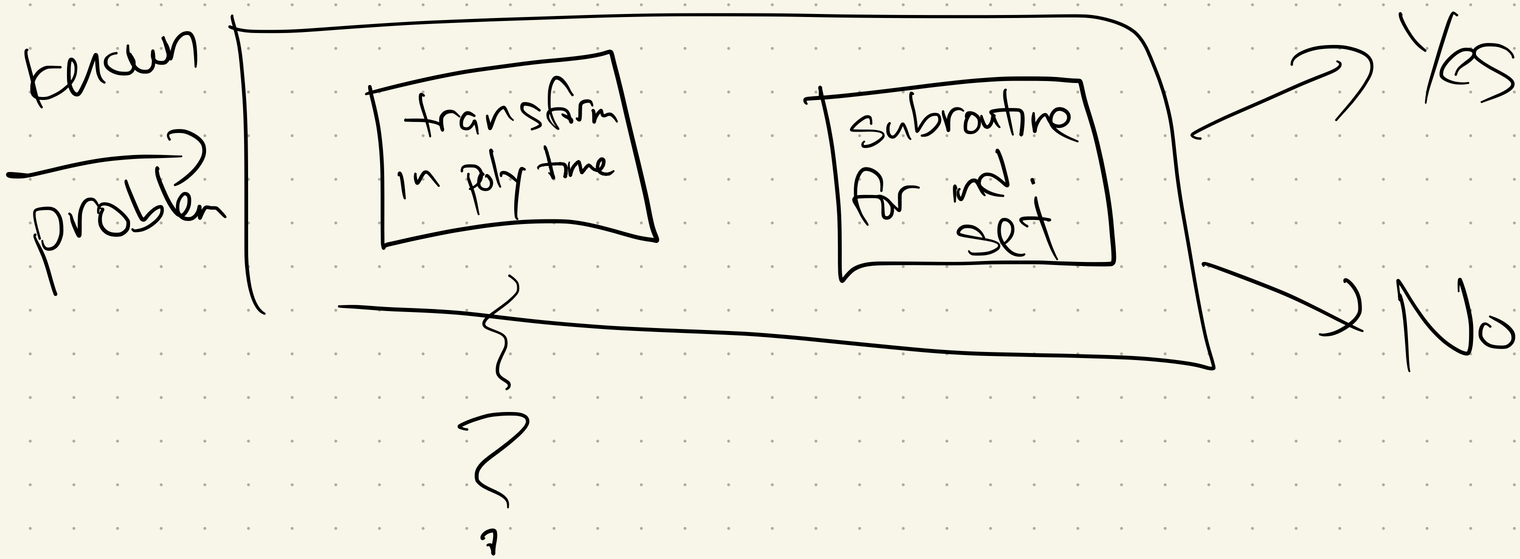
A set of vertices in a graph with no edges between them:



Decision version:

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:

Input is 3CNF boolean formula

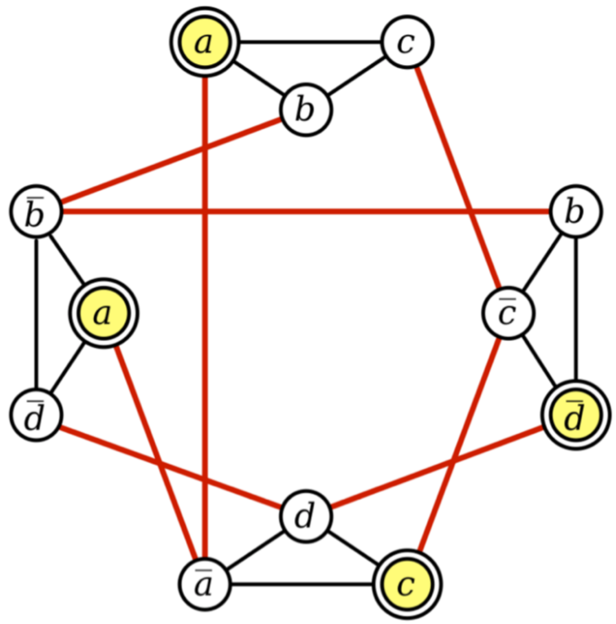
$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

1. Make a vertex for each literal in each clause

2. Connect two vertices if:
   - they are in same clause
   - they are a variable + its <u>inverse</u>

Example :

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



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

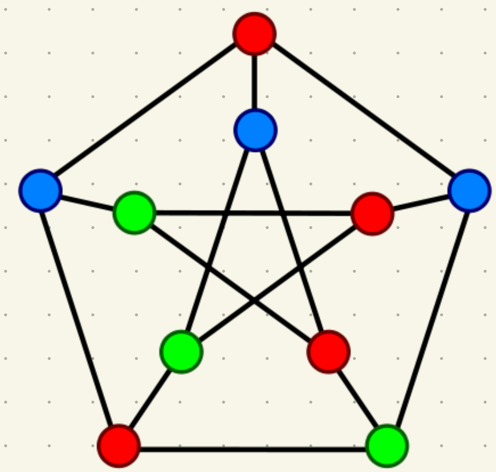
Claim:

formula is satisfiable $\Longleftrightarrow$ G has independent set of size $\geq m$

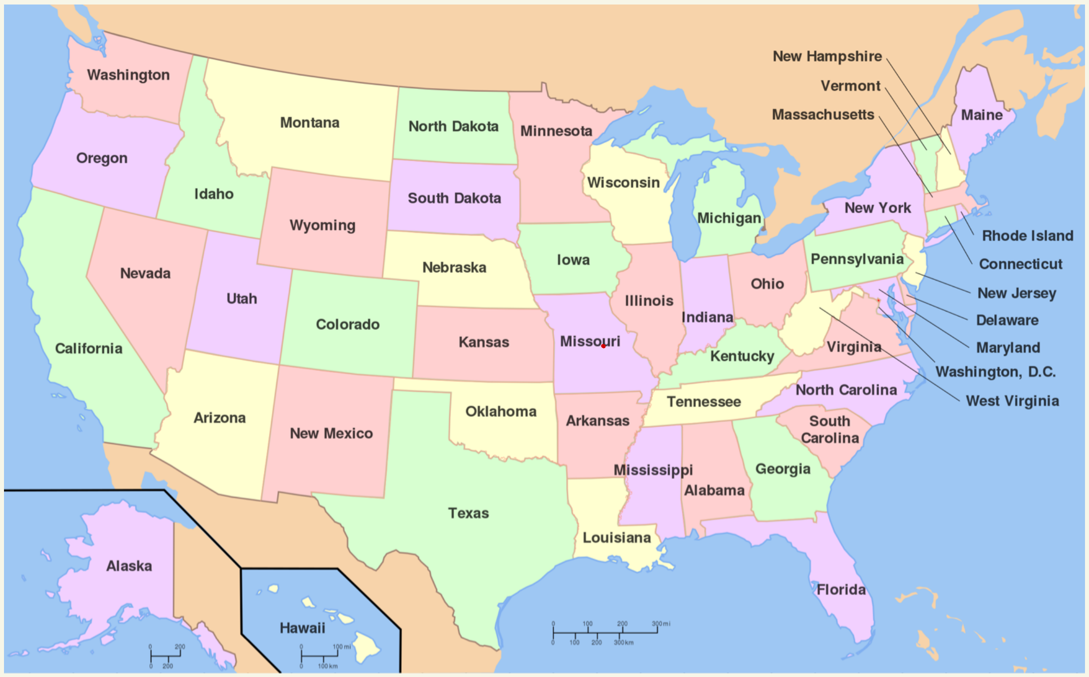
Next: Graph Coloring

A k-coloring of a graph G is a map: $c: V \to \{1, \ldots, k\}$ that assigns one of k "colors" to each vertex so that every edge has 2 different colors at its endpoints

Goal: Use few colors



Aside: this is famous!
Ever heard of map coloring?



Famous theorem:

Thm: 3-colorability is NP-Complete.

(Decision version: Given G & k,
output yes/no)

In NP:

certificate:

NP-Hard:

Reduction from 3SAT.

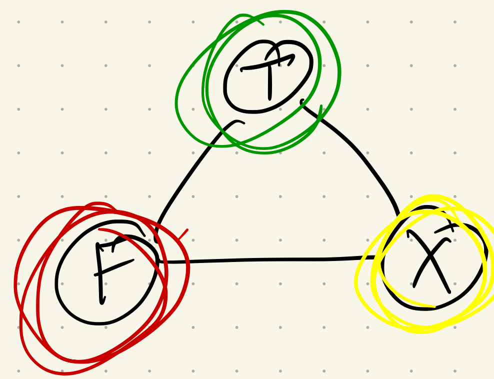
Given formula for 3SAT Φ,

we'll make a graph G_Φ.

Φ will be satisfiable

$\Longleftrightarrow G_\Phi$ can be 3-colored.

Key notion: Build "gadgets"!

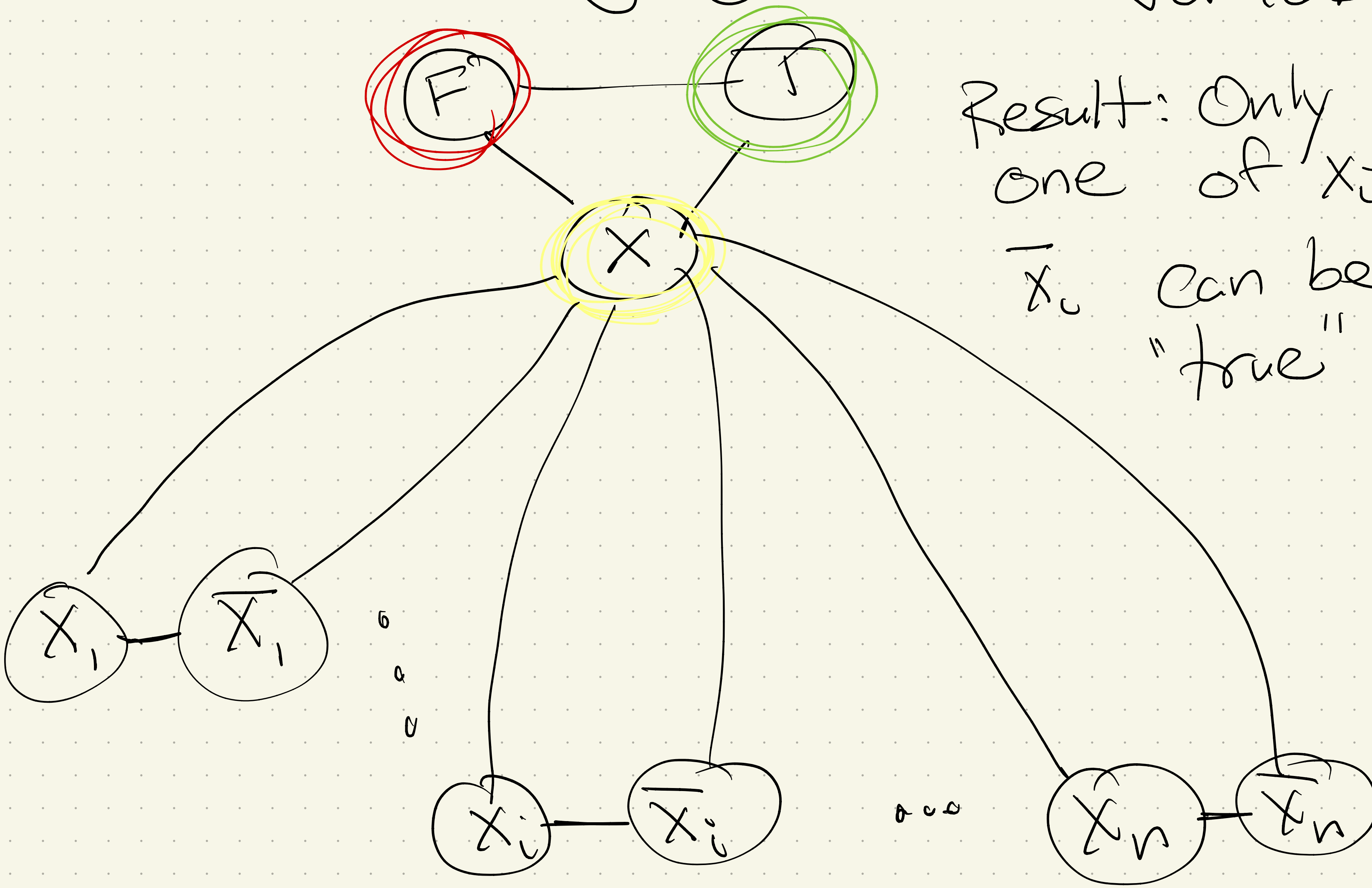


(1) Truth gadget –

Must use 3 colors –

so establishes a "true" color.

(2) Variable gadget: one per variable

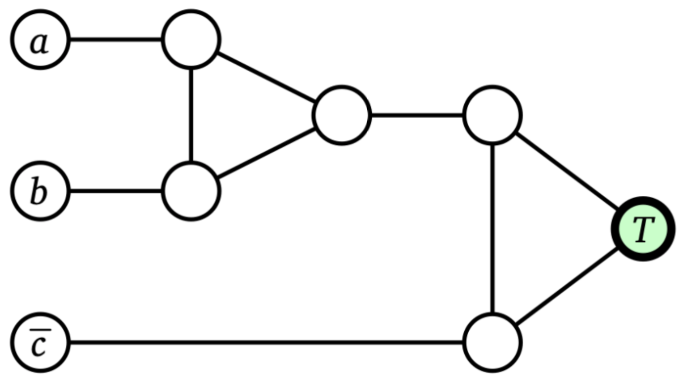


Result: Only one of x_i + $\overline{x_i}$ can be "true"

③ Clause gadget:

For each clause, join 3 of the variable vertices to the "true" vertex from the truth gadget.

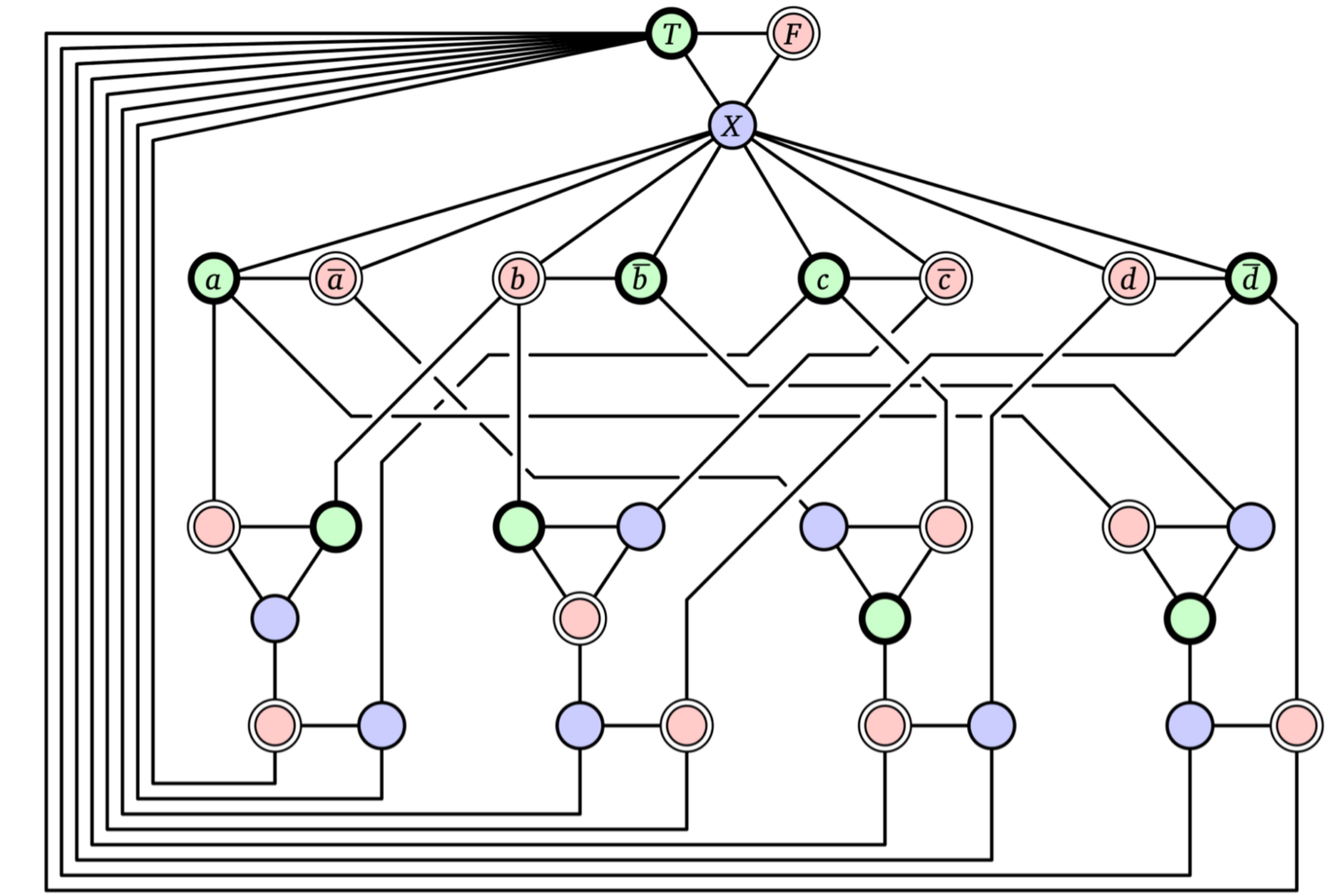
Goal: If all 3 are false, no valid 3-coloring



A clause gadget for $(a \vee b \vee \bar{c})$.

Why?? try to color all "false"

Final reduction image:



A 3-colorable graph derived from the satisfiable 3CNF formula
$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$

Now, need reduction proof:

3 colorings of G_{Φ}

$\iff \Phi$ is satisfiable

<u>Pf:</u>

$\Rightarrow$: Consider a 3-coloring of G:

$\Leftarrow$ Consider a satisfying assignment to Φ!