Algorithms 2020

NP-Hardness:
ALL the graphs!

Recap:

- 3 more reading assignments!
  - Sun, Thurs, + next Sun.
- HW - Sign up for oral grading + group
- Last HW - due on Wed. before Thanksgiving (all submitted by that weekend)

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

$\Longleftrightarrow$

If X could be solved in polynomial time, then $P = NP$.

Recall $P \overset{?}{=} NP$

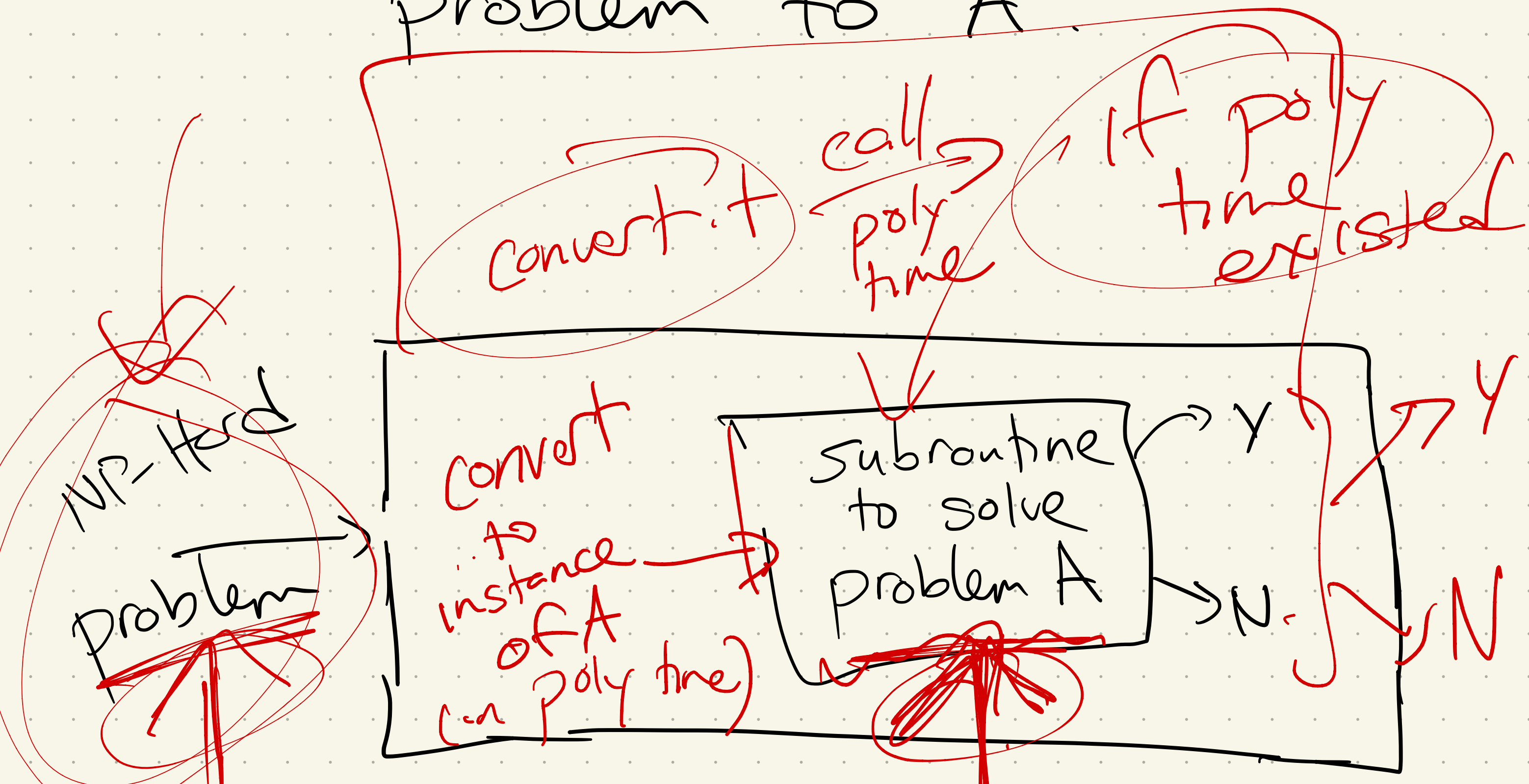
So if <u>any</u> NP-Hard problem could be solved in polynomial time, then all of NP could be.

Initial ex:

CircuitSAT

To prove NP-Hardness of A:

Reduce a known NP-Hard problem to A.



general instance of known problem

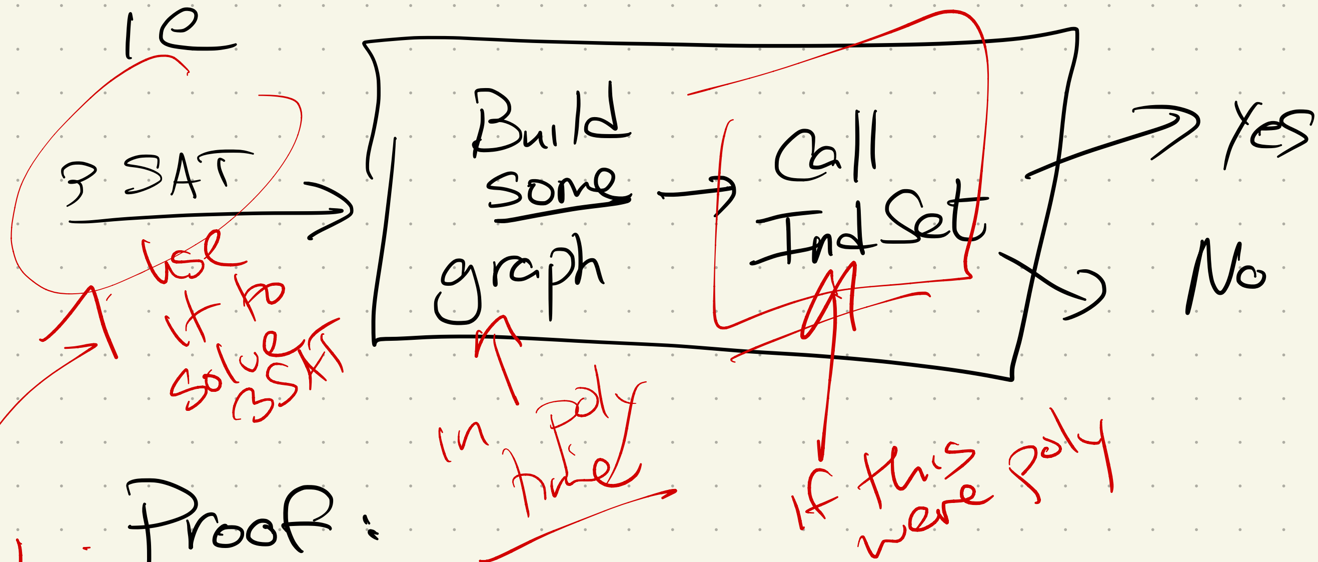
many of these

normally: to solve X, reduce to A & call subroutine

The Pattern: Reductions

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

ie



Key: **Proof:**

Need if & only if!

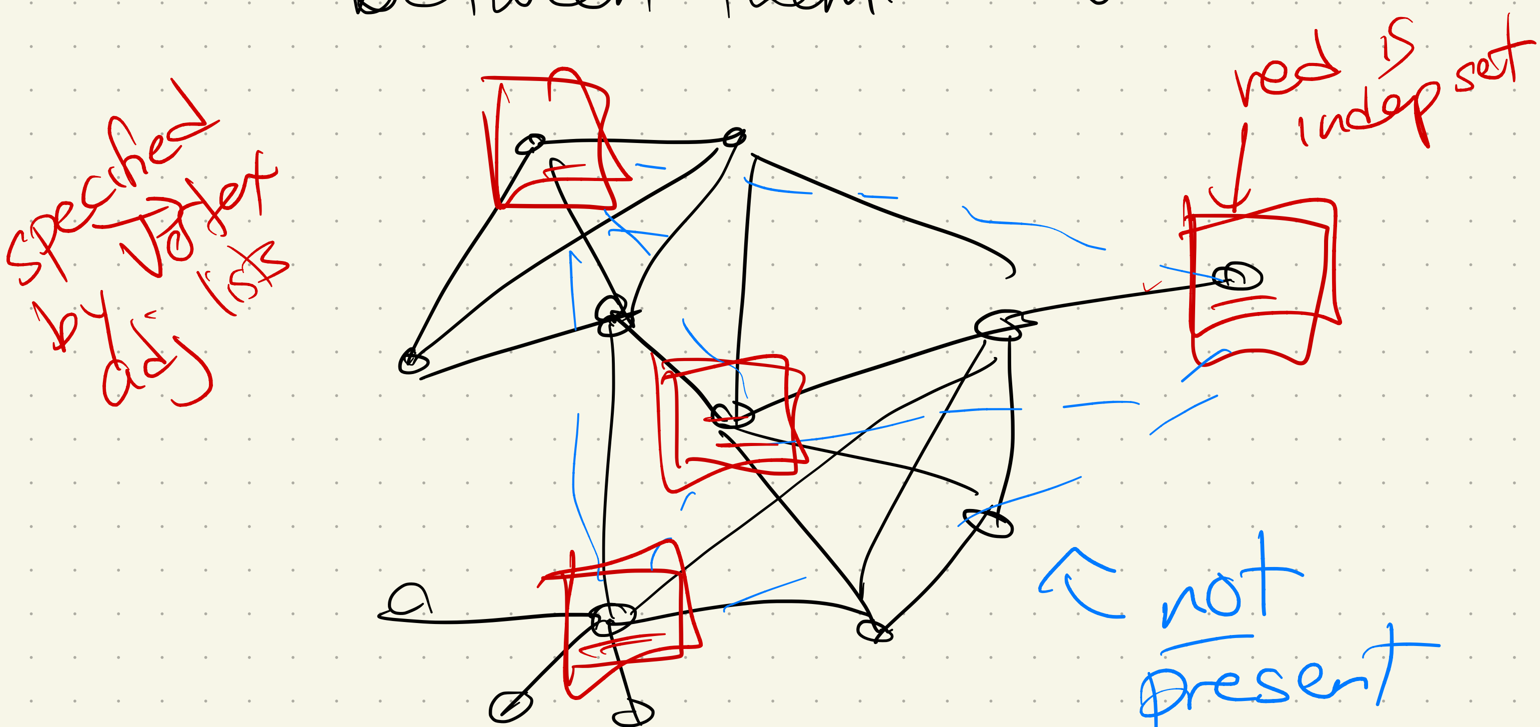
(ie might be some wierd indep. set that doesn't make a SAT)

Challenge: Finding correct NP Hard problem

Next Problem: looks more useful

Independent Set:

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



decision version:

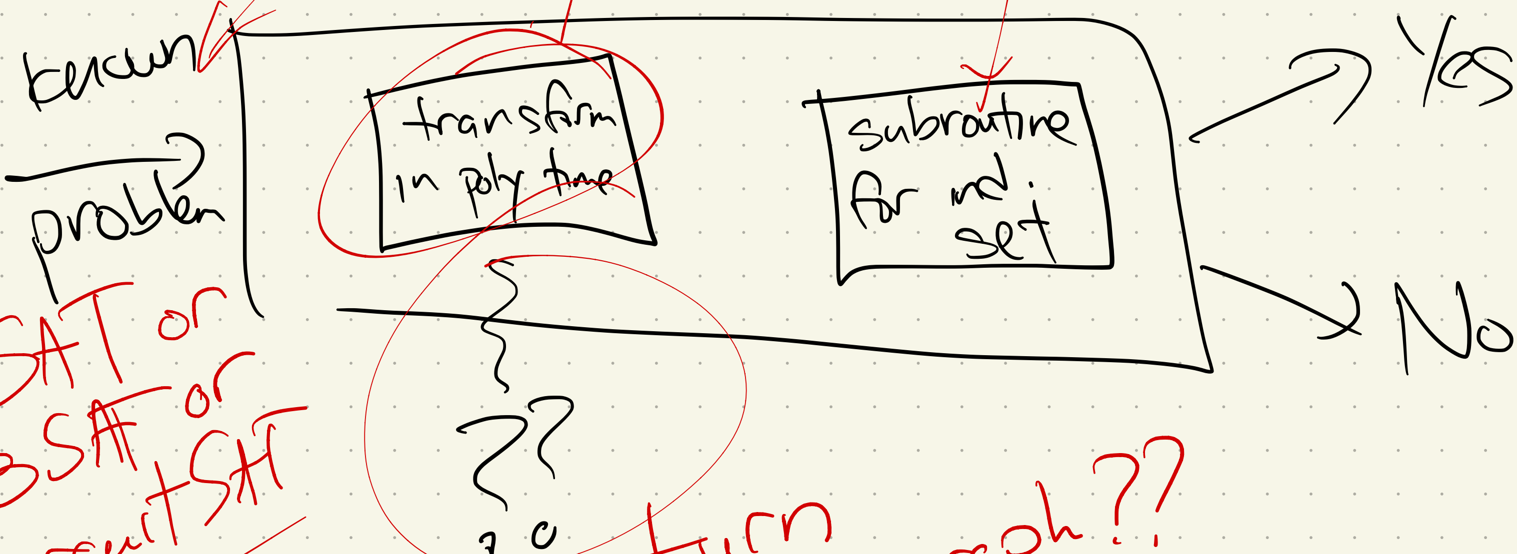
Is there a ind set of size k in G?

(Wait - didn't we see this already!?) → solves in trees ~~DAGs?~~

→ input: k & G

Challenge: No booleans!

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



SAT or 3SAT or CircuitSAT

turn into graph??

We'll use 3SAT

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

First look at "gadgets"

pieces in transformations

Reduction:

m clauses

n literals (variables)

Input is 3CNF boolean formula

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

AND

OR

① Make a vertex for each literal in each clause

really vertex lists

↳ 3m vertices

② Connect two vertices if:

- they are in same clause → 2 edges per vertex } ≤ 6m
- they are a variable + its inverse

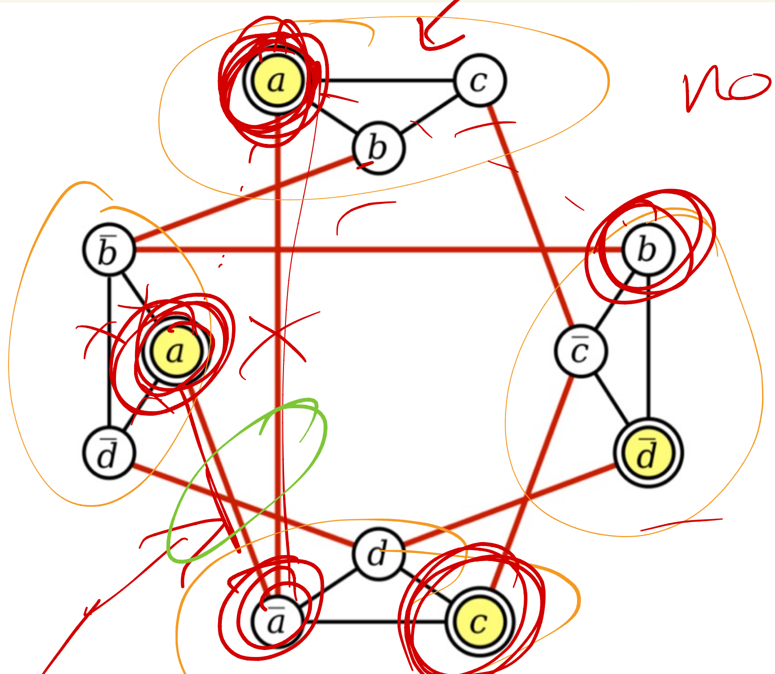
m clauses, so might add ≤3m edges per vertex

Example:

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

clause

no edges



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

go between var & its negation

Reduction: spent $O(m+n)$ time building graph

↳ poly!

__Claim:__ original 3SAT formula is Satisfiable

$\Longleftrightarrow$

G has independent set of size m

__Proof:__

__$\Rightarrow$:__ Assume formula is SAT.

Then $\exists$ assignment of input variables $x_1 \ldots x_n$, so that formula = true.

Build ind. set in G:

for each clause $\triangle \Leftrightarrow (x_i \vee x_j \vee x_k)$

must have something true.

Grab <u>one</u> of the true vars + add corr. vertex to set.

independent since:

- one vertex per $\triangle$
- no var + its negation can be true

$\Leftarrow$: Take ind set of size m:
(in G)

$\hookrightarrow$ must have one vertex per  gadget

That corresponds to a variable!

Set var = True.

(Set other variables however you want)

$$\underbrace{(x_i \vee x_j \vee x_k)}_{T} \wedge (\ \vee\ \vee\ ) \wedge \cdots \wedge (\_ \vee \_ \vee \_)$$

ORs, so T as long as 1 value is true

So!

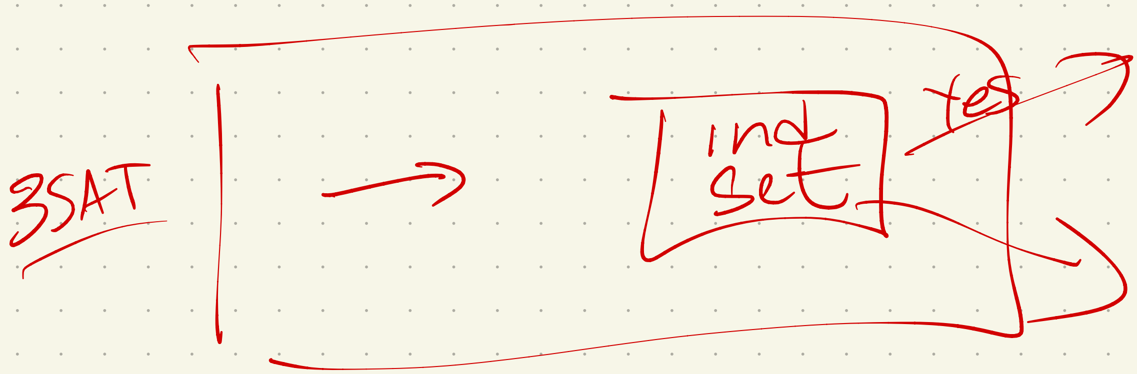
$O(n+k)$ $+ \le 9k$ edges

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

Y or N

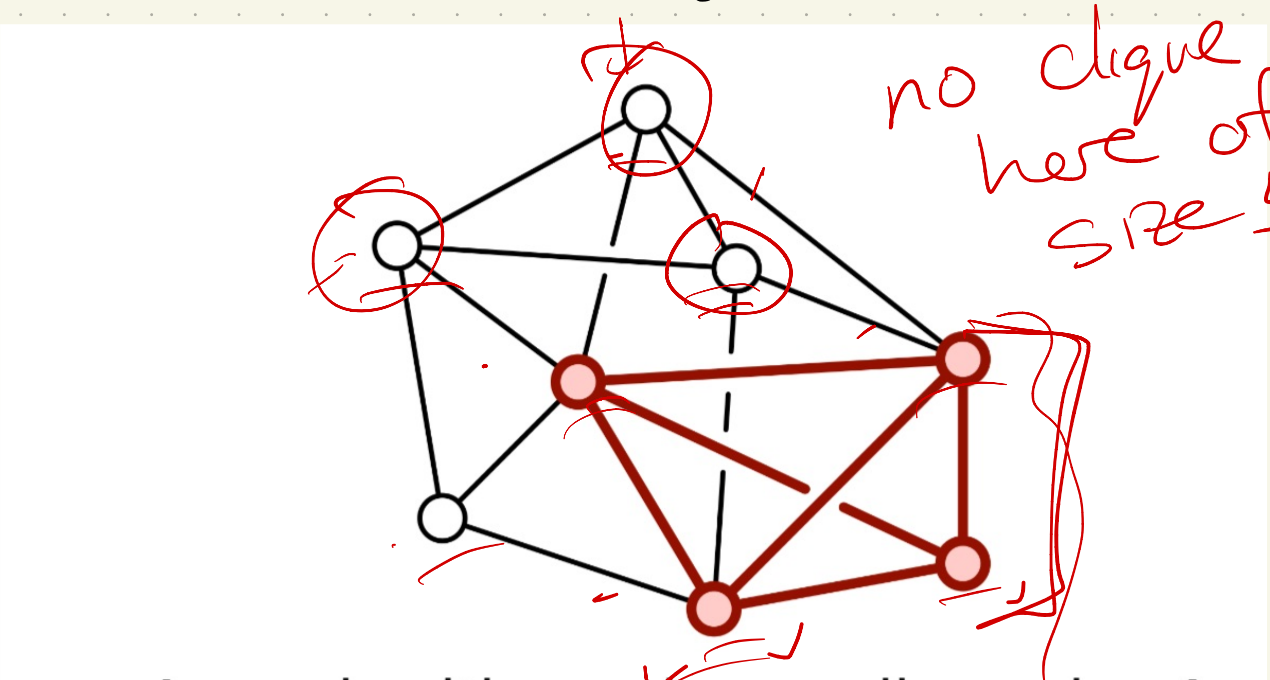
$$T_{3\text{SAT}}(n) \le O(n) + T_{\text{MaxIndSet}}(O(n)) \implies T_{\text{MaxIndSet}}(n) \ge T_{3\text{SAT}}(\Omega(n)) - O(n)$$

3SAT depended on CircuitSAT



Next one: Clique size

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?

Brute force: Look at all subsets of k vertices: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

input

exponential in k $\approx O(n^k)$

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

Input: G + k

Output: Y or N

This is NP-Complete:

1. In NP. Why?

Poly time checking if yes:

If set of size k vertices is handed to me, I can check if all $\binom{k}{2} \approx O(k^2)$ edges in clique are in G.

② NP-Hard:

What should we reduce to k-clique?

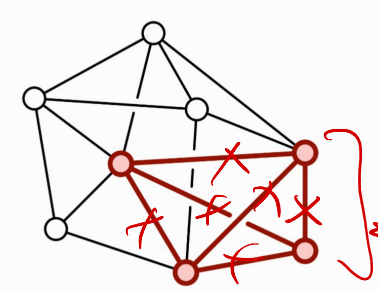
Options: CircuitSAT
3SAT
Ind Set!

Input: G + k, question is does G have ind set of size k?

in poly time

change G ⇝ G'
so that G' has clique
⟺ G has ind set.

G'



A graph with maximum clique size 4.

want G' have an edge whenever G doesn't.

So:

$O(V+E+V^2)$ all possible edges

$$\begin{tikzcd}
\text{graph } G \arrow[r, "\cancel{O(n)}"] & \text{complement graph } \overline{G} \arrow[d, Rightarrow, "\textsc{MaxClique}"] \\
\text{largest independent set} & \text{largest clique} \arrow[l, "\text{trivial}"']
\end{tikzcd}$$

Complement $\overline{G}$ (not G'):

vertices of $\overline{G}$ = vertices of G

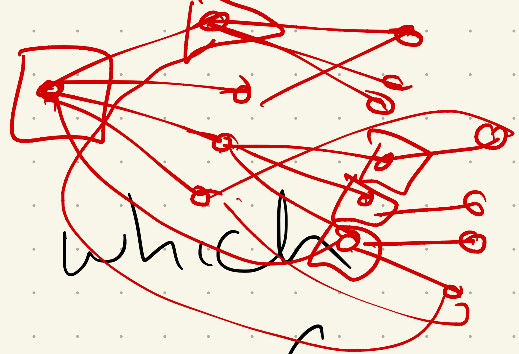
look at every pair in V:

- if G has edge u,v, $\overline{G}$ will not
- if G does not have uv, $\overline{G}$ will have it
  - add it to adj. lists

End: if poly time alg. for clique, get one for ind set.

Next: Vertex Cover:

A set of vertices which touches every edge in G.



min cover

we solved in a tree!

k-Vertex cover (decision version):

Given G, + k,
answer Y/N if ∃ vertex cover of size k.

In NP: (verifying "yes" in poly time)

Given k vertices, check if they are a cover.

For each e:
check if ≥ endpoint is in k-set of vertices.

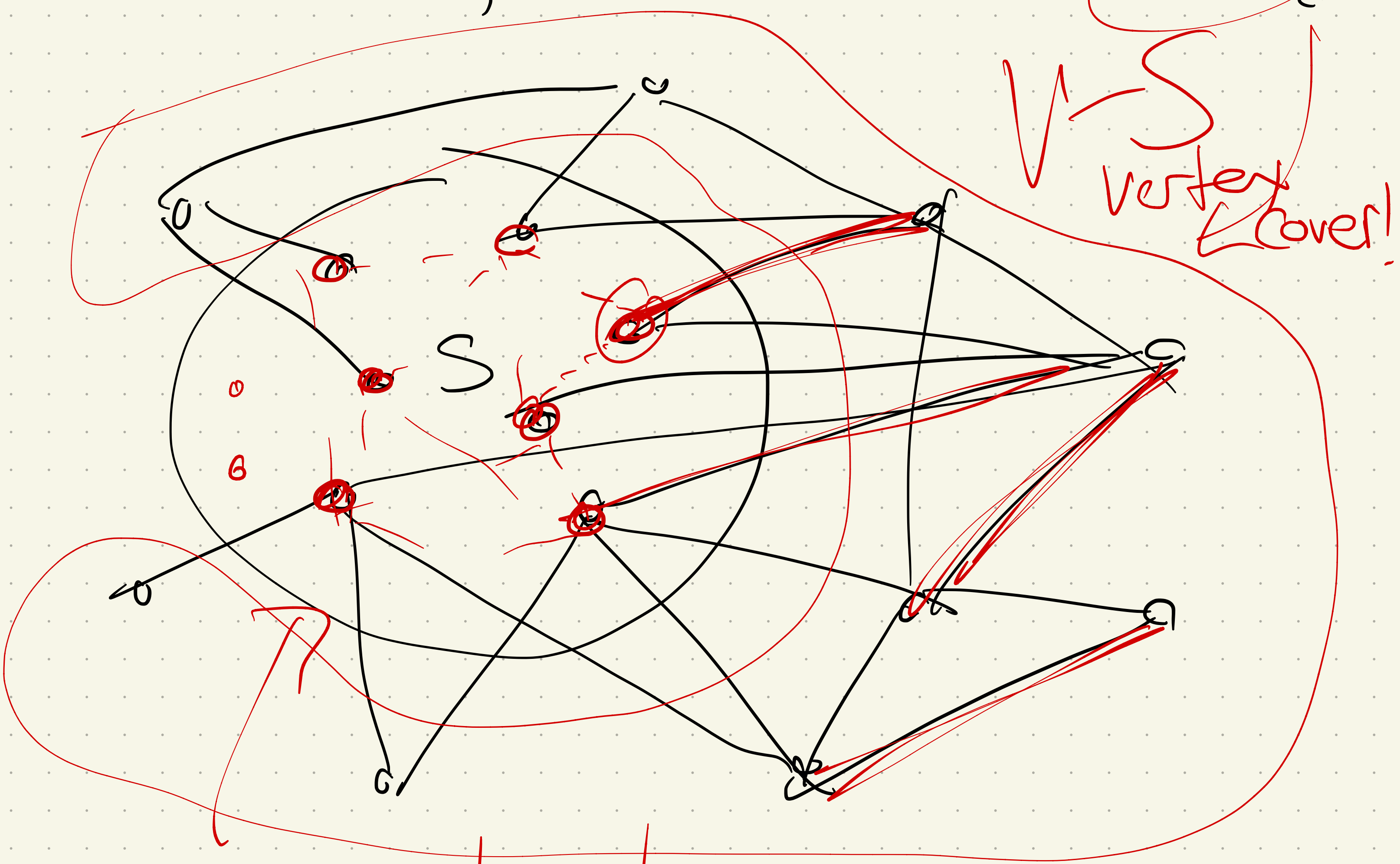
$O(V+E)$

NP-Hardness: reduce what?

(propably clique or ind set!)

?

Key: If S is independent set, what is $V-S$?

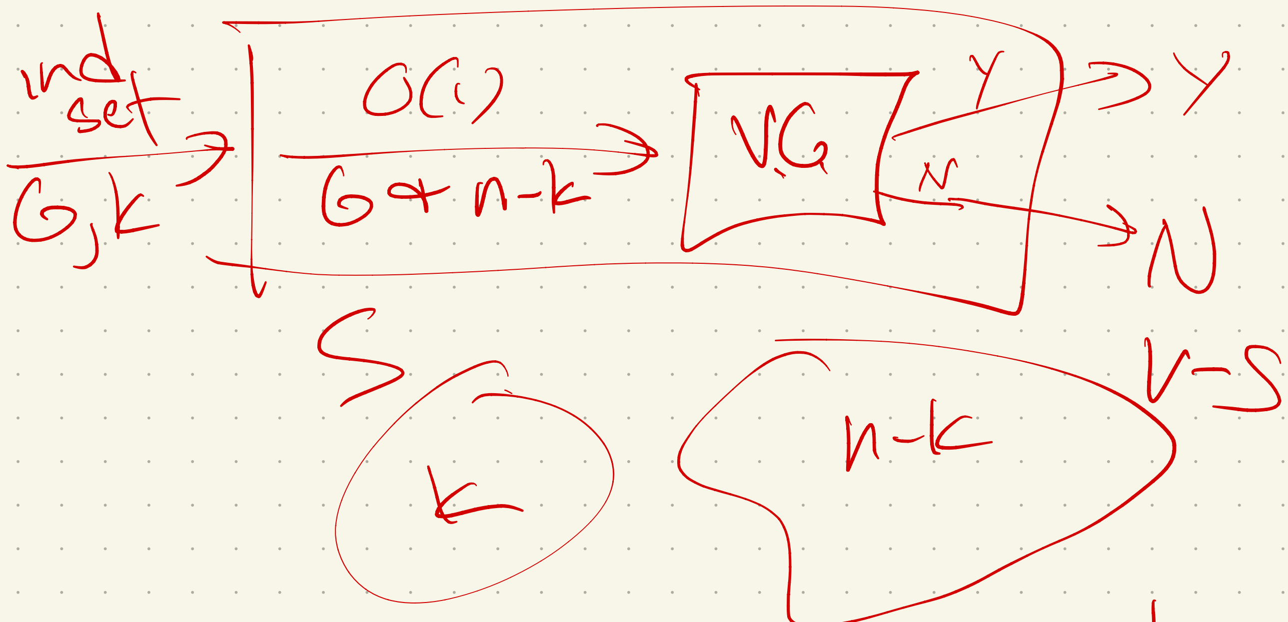


S is indpend, b/c no edges b/t 2 vertices in S

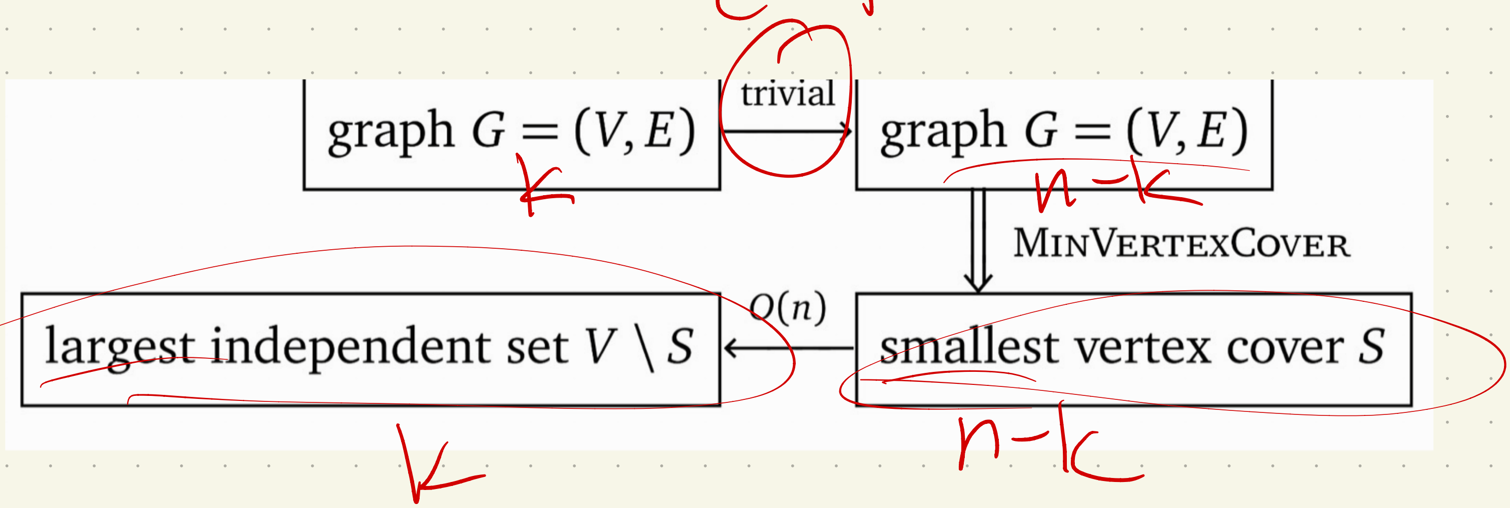
all edges use $\geq$ 1 endpoint outside of $S \Rightarrow V-S$ is V.C.

So simple reduction!

Given G & k to indep. set, ask if $\exists$ vertex cover of size $n-k$.



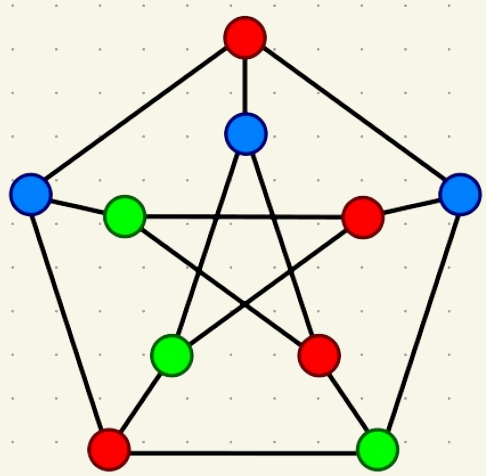
compute $n-k$



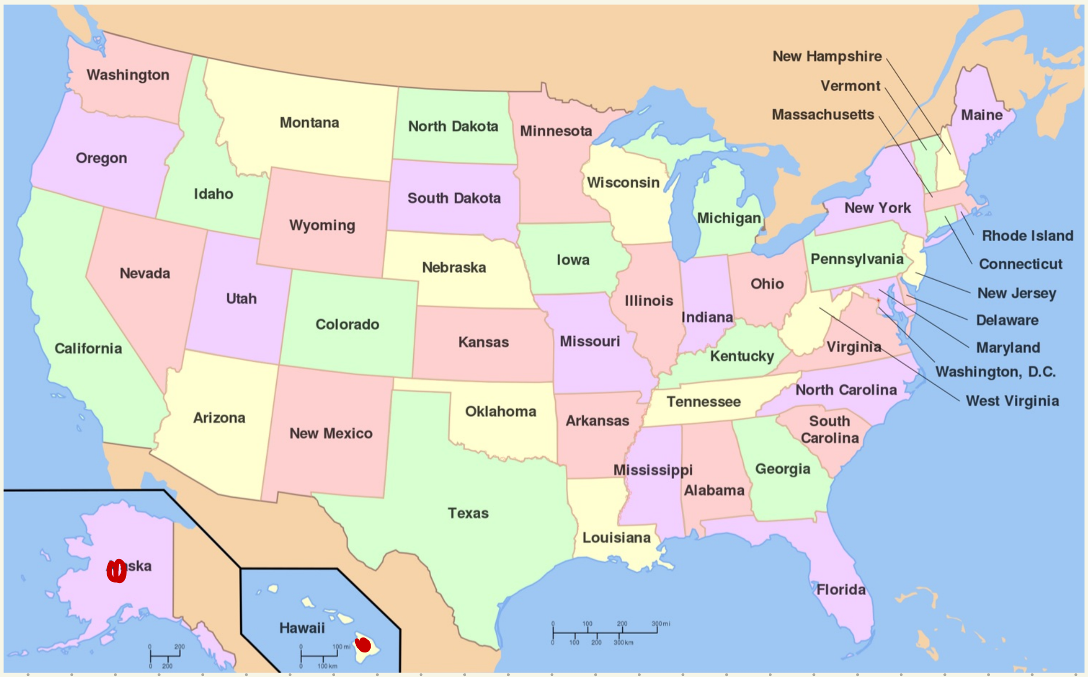
Next: Graph Coloring

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, output yes/no)

In NP:

NP-Hard:

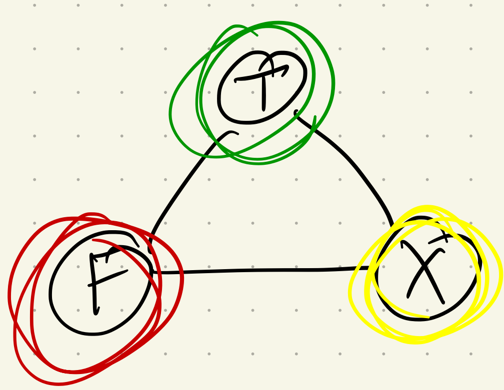
Reduction from 3SAT.

Given formula for 3SAT Φ, we'll make a graph G_Φ.

Φ will be satisfiable $\iff$ G_Φ can be 3-colored.

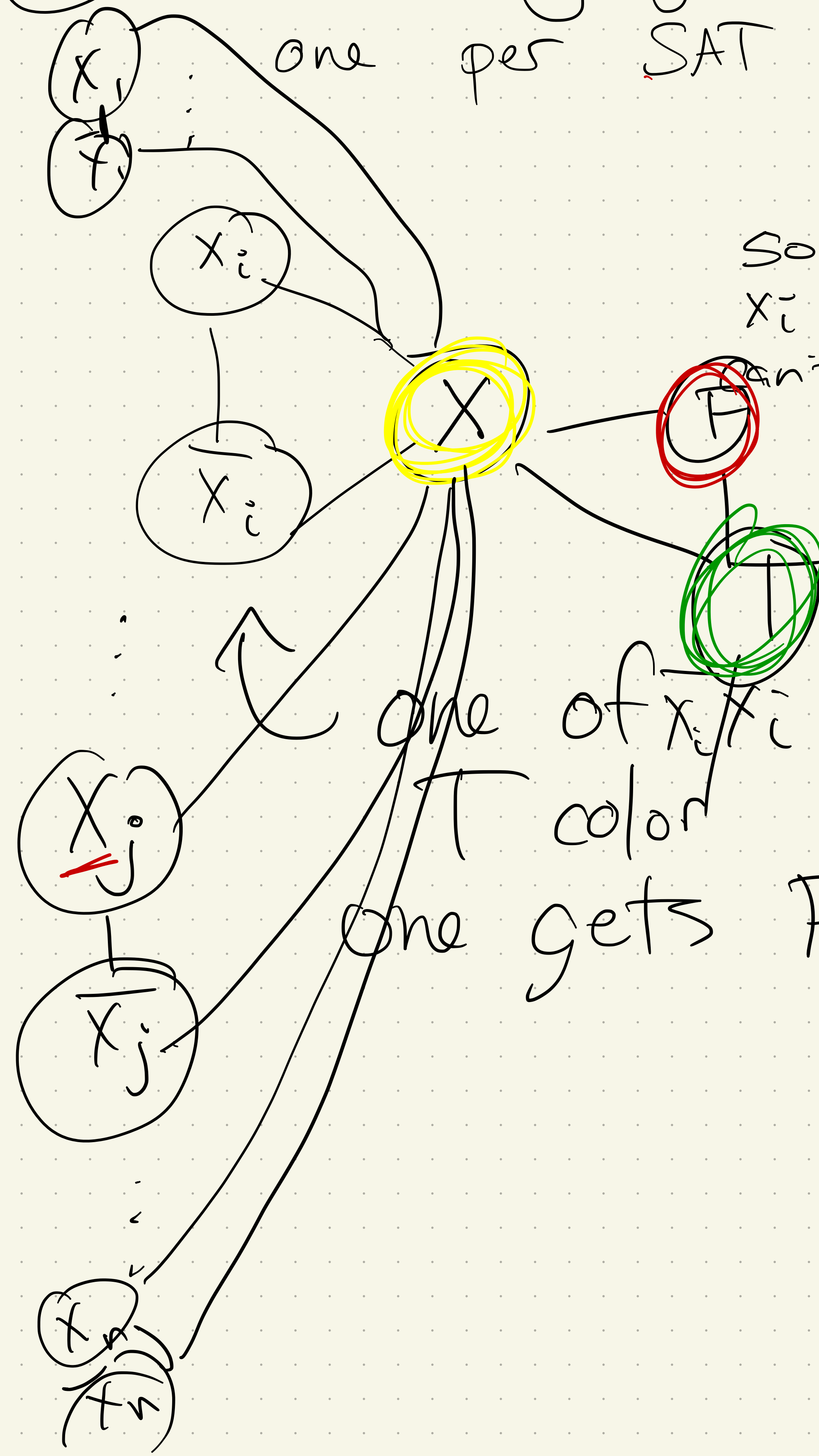
Key notion: Build "gadgets"!

① Truth gadget – one



Must use 3 colors – establishes a "true" color.

② Variable gadget — one per SAT variable



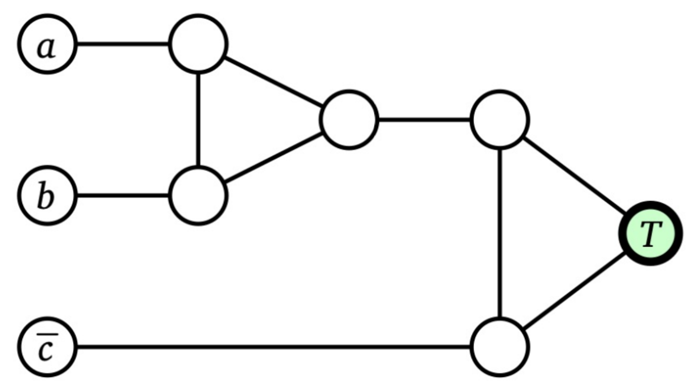
so both x_i + $\bar{x}_i$ can't be true

one of x_i/$\bar{x}_i$ gets T color
one gets F color

③ Clause gadget:

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

<u>Goal</u>: If all 3 are false, no valid 3-coloring



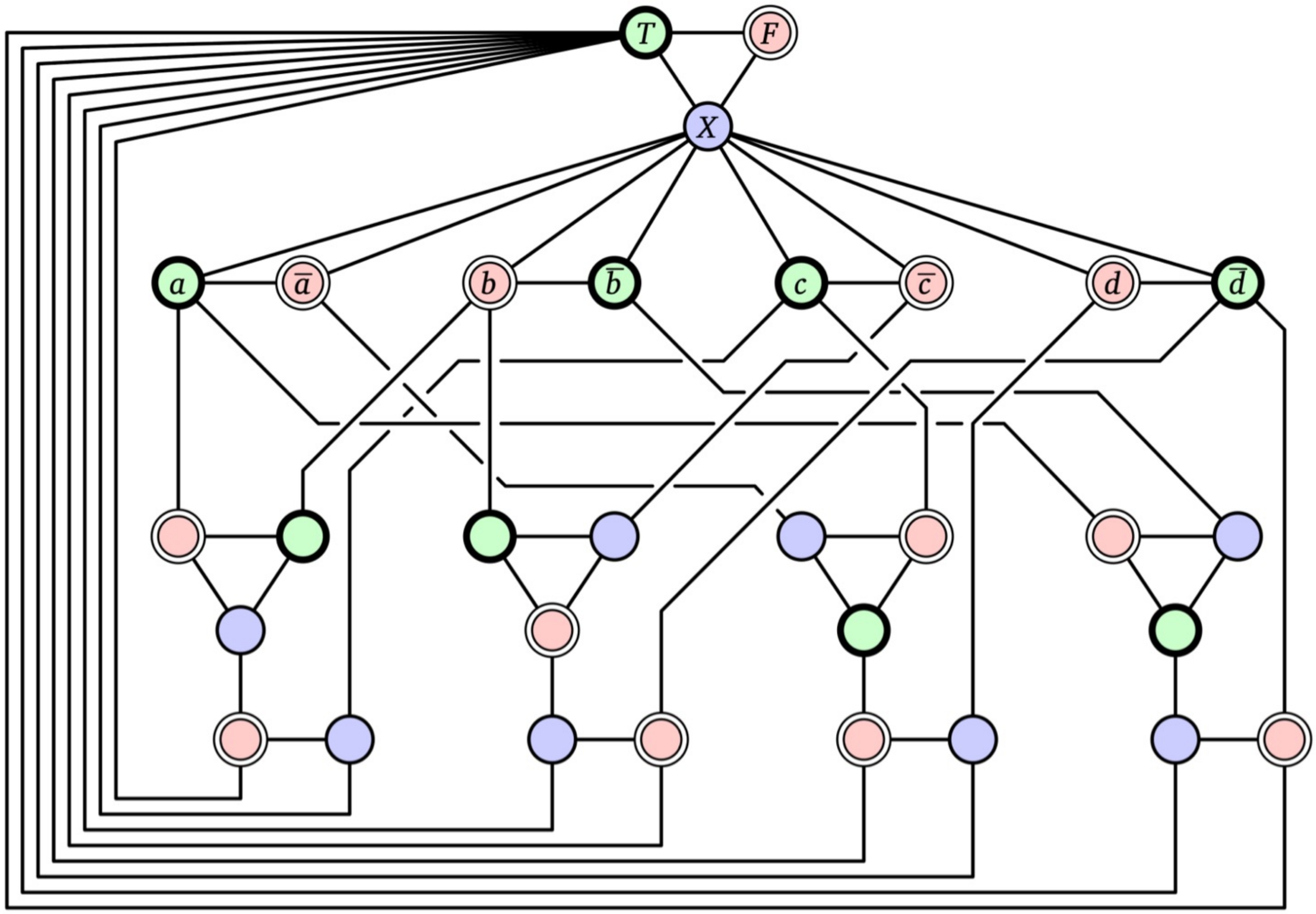
A clause gadget for $(a \vee b \vee \bar{c})$.

<u>Idea</u>: If all inputs are colored False, can't 3-color:

3 colorings of G_Φ $\Longleftrightarrow$ Φ is satisfiable

Pf:

Final reduction image:



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

Time to build G_Φ:
(remember, need polynomial in formula size, $n + m$)

<u>So:</u>

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