3SAT (3CNF-SAT)

<u>literal</u>: a boolean variable <u>or</u> its negation $(a, \bar{a})$ $(\neg a)$

<u>clause</u>: a disjunction of one or more literals $(b \vee \bar{c} \vee \bar{d})$
not

<u>Conjunctive normal form</u> (CNF):
the conjunction (and) of clauses

$$\overbrace{(a \vee b \vee c \vee d)}^{\text{clause}} \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b})$$

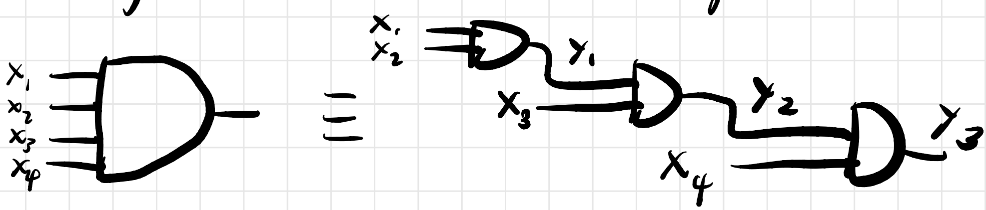
3CNF: CNF formula with <u>exactly</u> three literals per clause

3SAT: Given a 3CNF formula, is there an assignment to variable so formula evaluates to True?

Reduction from Circuit SAT:

Given a boolean circuit:

1) Change circuit so all AND + OR gates have two inputs:



2) $(y_1 = x_1 \wedge x_2) \wedge (y_2 = y_1 \wedge x_3) \wedge$

$(y_3 = y_2 \wedge x_4)$

Write an equation for each gate like in SAT.

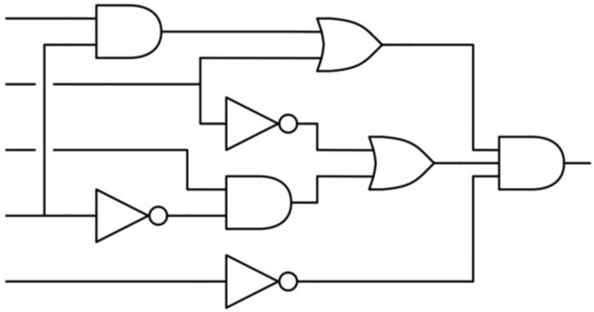
3) Change each gate equation into a conjuction of clauses.

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

(now the circuit's formula is in CNF)

4) Change each two or one literal clause into a couple three literal clauses.

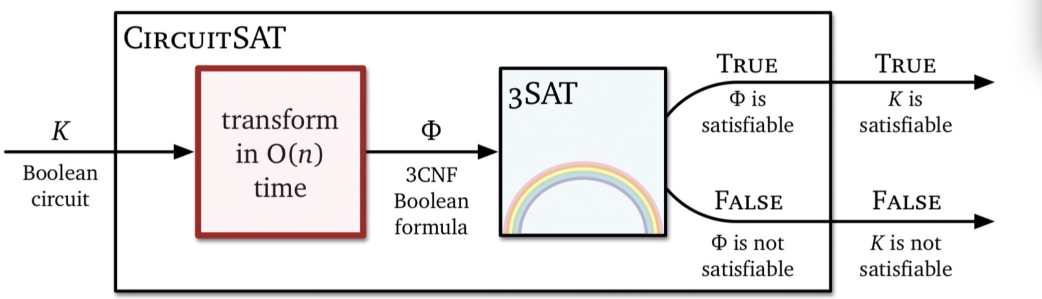
$$a \vee b \longmapsto (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$$
$$a \longmapsto (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge (a \vee x \vee \bar{y}) \wedge (a \vee \bar{x} \vee \bar{y})$$



$$
\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_6 \vee z_{18}) \wedge (\overline{y_9} \vee y_6 \vee \overline{z_{18}}) \wedge (\overline{y_9} \vee y_8 \vee \overline{z_{17}}) \\
&\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}
$$

(circuit to formula example)

gross but linear time!



So 3SAT is NP-hard.

3SAT $\in$ NP. (Proof of yes answer is how is set variables.)

$\implies$ 3SAT is NP-complete.

Given simple, unweighted graph $G = (V, E)$.

Independent set: $S \subseteq V$. No pair $u, v \in S$ share an edge.

Maximum Independent Set (MaxIndSet)

Find largest ind. set in G.

MaxIndSet is NP-hard!

(from 3SAT)

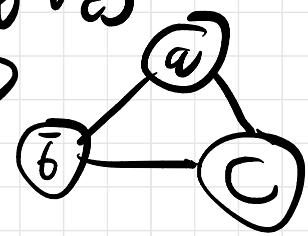
Given 3CNF formula Φ.

$k \leftarrow$ number of clauses in Φ

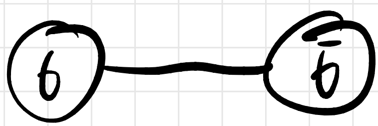
Make a graph G with $3k$ vertices, one for each literal of Φ (one per literal-clause pair)

Any two literals in same clause share an edge. Call these "triangle" edges.

$(a \vee \bar{b} \vee c)$ $\Rightarrow$



Any pair of literals a & $\bar{a}$ share a "negation" edge.

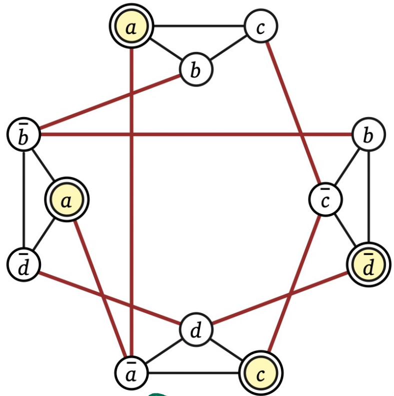


a

b

c

$\bar{d}$



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

Claim: G contains an ind. set of size k iff Φ is sat.

Proof:

Suppose Φ is sat. Fix a sat. assignment. Each clause has ≥ 1 true literal. Take the corresponding vertex of one true literal in each clause to make a set $S \subseteq V$. No triangle edge has both end points in S. No negation edge either (one of a or $\bar{a}$ is false)

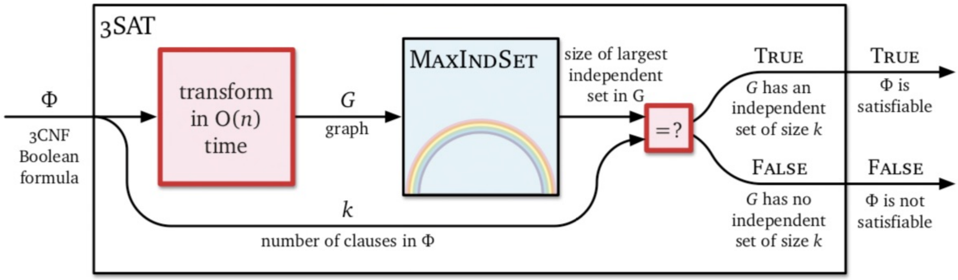
S is an ind. set of size k!

Suppose $\exists$ ind. set S of size k. Use any assignment that makes S's literals True.

Triangle edges limit S to one literal per clause $\Rightarrow$ it must use exactly one literal per clause.

Negation edges guarantee at most one of a or $\bar{a}$ need be set to True.

We sat. every clause!



We reduced known NP-hard problem 3SAT to Max Ind Set in poly time ⟹ Max Ind Set is NP-hard.

This was the optimization version of the problem Max Ind Set.

The decision version: Given G & an integer k, is there an independent set of size k?

Still hard, but now in NP ⟹ NP-complete.

The General Pattern:

To prove B is NP-hard, take known NP-hard problem A & reduce A to B.

1) Show a poly time algorithm to reduce a arbitrary instance a of A to a special instance b of B.

2) Prove that if a is "good" then b is "good".

3) Prove that if b is "good" then a is "good".

May help to imagine each stage as an algorithm!

The "proofs" are often called <u>certificates</u>.

1) The algorithm to turn a into b.

2) Alg to turn a certificate for a into one for b.

3) Alg to turn a certificate for b into a certificate for a.

Still have $G=(V,E)$.

clique: a complete subgraph

Max Clique: Find the largest clique in G.

vertex cover: a subset of vertices touching every edge at least once

Min Vertex Cover: Find a smallest vertex cover.

Both are NP-hard!

Max Clique from Max Ind Set.

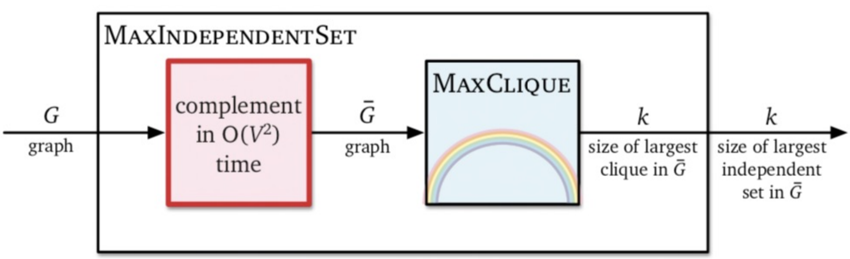
Given G for which we want an ind. set.

edge-complement: $\bar{G}$ of G:

opposite set of edges

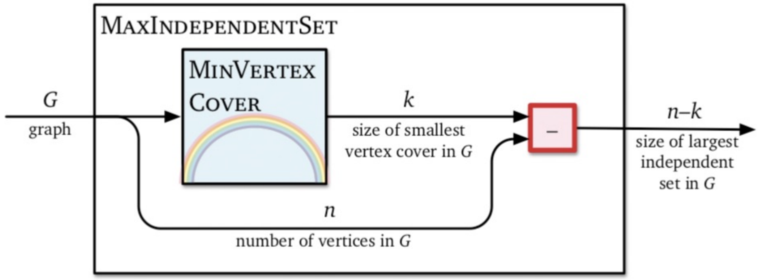
e in $\bar{G}$ iff e not in G.

$S \subseteq V$ is independent in G iff S is a clique in $\bar{G}$.



Min Vertex Cover from Max Ind Set:

Obs: $S \subseteq V$ is independent iff $V \setminus S$ is a vertex cover.



Decision versions (given G & k, is there a ____ of size k?) are NP-hard also.

And in NP

=> NP-complete.