Algorithms

NP-Hardness + Complexity: Reductions

Recap

- HW8: due next Monday.

  Sorry! $\frown$

  (Predictably, my computer crashed...)

P, NP, + co-NP $P \subseteq NP$

Consider only decision problems:
so Yes/No output

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

Ex: - Is x in the ~~list~~ list? $O(n)$ or $O(\log n)$

- Is there a cut in G of size 100? ⟶ F-F: $O(VE)$

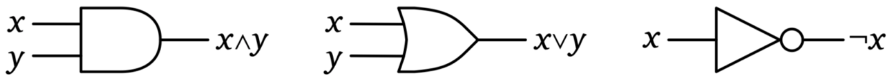
Non-determinate poly time

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

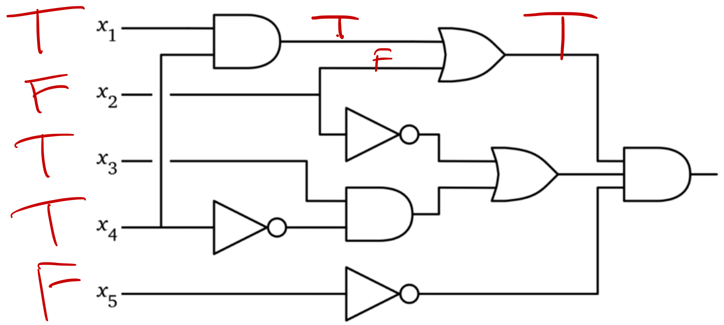
Ex: CircuitSAT: hand me inputs I can check in $O(n+m)$ time

Co-NP: If answer is no, I can check that in poly time.

The first problem found:
Boolean circuits



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

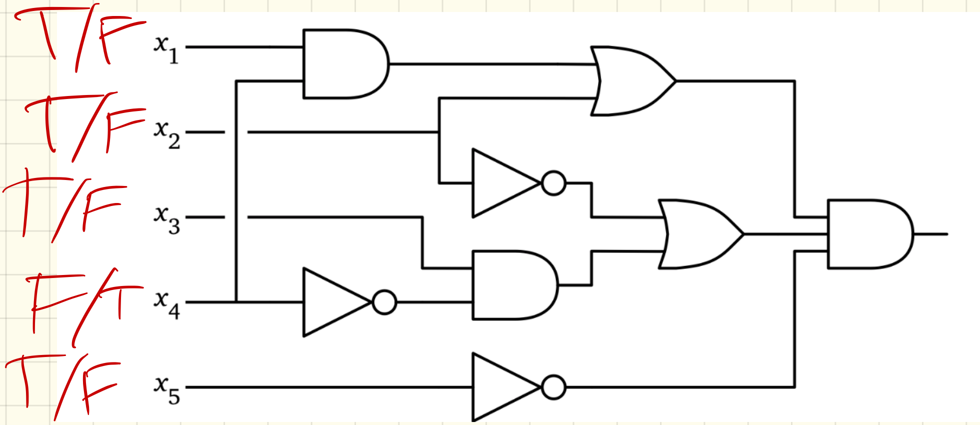


A boolean circuit. inputs enter from the left, and the output leaves to the right.

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

How? Circuit is a tree → trace T/F values through gates starting from "leaves"

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:

Try all 2^n inputs.

Track through gates & check if T/F in output

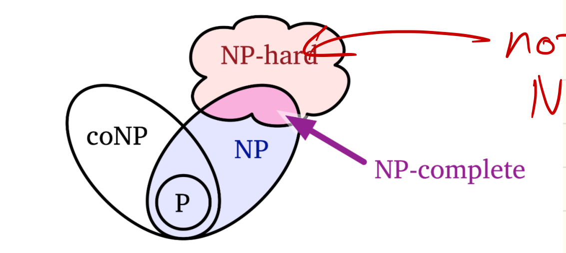
Running time:

$$2^n(n+m)$$

Note: Might be a better way!

Cook-Levine Thm:

Circuit SAT is NP-Hard.



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

not in NP

NP-Complete:

- in NP
- and NP-Hard

Why?

They mimic any Turing Machine using a circuit.

Just trust me. :)

Dfn: NP-Hard

X is NP-Hard

$\Longleftrightarrow$

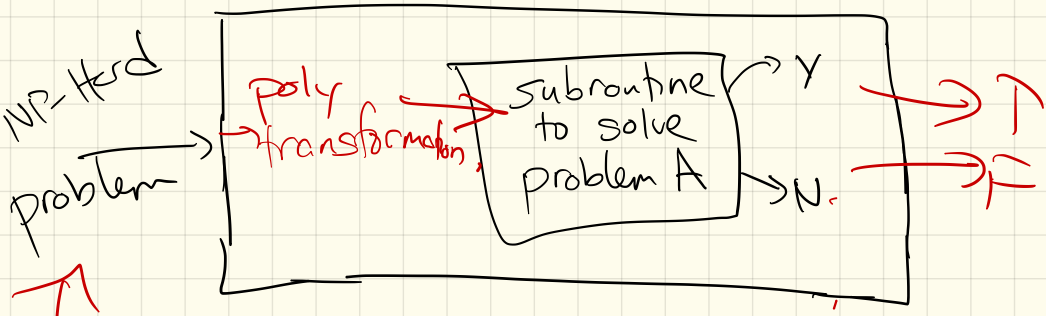
If X could be solved in polynomial time, then

$P = NP$.

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.

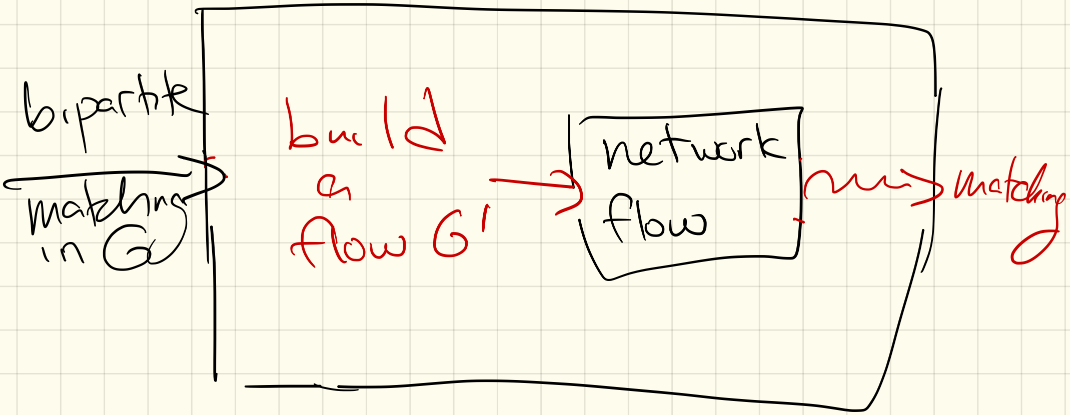


circuitSAT

If tranformation + subroutine for A is poly time, then could solve circuitSAT in that time.

We've seen reductions!

Remember flows + graphs?



Equivalent pseudocode:

BipartiteMatching(G): ← O(VE)

O(V+E) → Modify G to a flow network G'

O(VE) FF(G')

O(V+E) → Turn flow into the matching

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!

<u>Why?</u>

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 a way to assign inputs so result is 1?

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

n variables,
m clauses

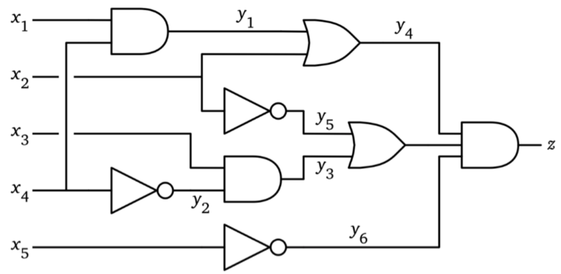
In NP:

If you give me n T/F values, I can go left to right & evaluate boolean.

$O(m+n)$

Thm: SAT is NP-Hard.

Pf: Reduce CIRCUIT SAT ← Known NP-Hard
to SAT: ← unknown



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

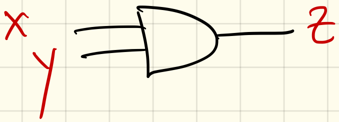
A boolean circuit with gate variables added, and an equivalent boolean formula.

Reduction:

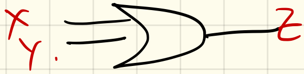
Given a circuit, want to write (in poly-time) an equivalent clause.
Then call subroutine

More carefully:

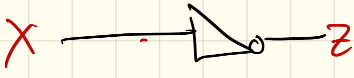
1) For any gate, can transform:



: $(z = x \wedge y)$

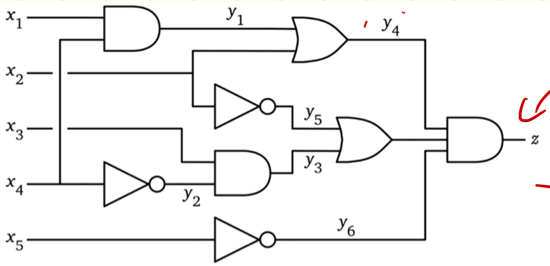


: $(z = x \vee y)$



: $(z = \overline{x})$

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



this is true

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

Is this poly-size?

to circuitSAT

Given n inputs + m gates:

in SAT

Variables: $n+m$ variables

Clauses: 1 per gate $\Rightarrow O(m)$

End reduction:

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

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

(Here, "n" is total input size)

Thm: 3SAT is NP-Hard

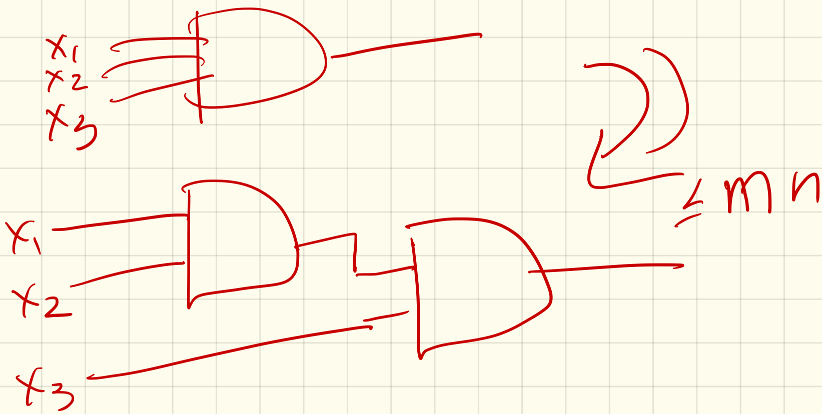
pf: Reduce circuitSAT to 3SAT.

Need to show any circuit can be transformed to CNF form

(so last reduction fails)

Steps:

1. Rewrite so each gate has 2 inputs:



(2) Write formula, like in SAT:

3 types: (Same as SAT)

$y = a \vee b$

$y = a \wedge b$ $\quad O(n+m)$

$y = \bar{a}$

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

CNF: conj. normal form

3 clauses per old gate

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

(not quite 3SAT yet)

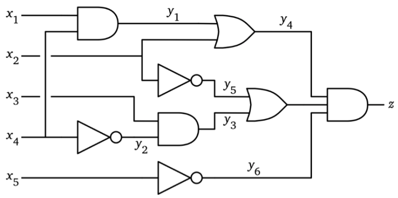
(4) Now, need 3 per clause:

x4 of # clauses

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

$\subset$ introducing dummy vars

Note: Bigger!



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

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

$$\Downarrow$$

$$\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}$$

How big? Polynomial!

$O(mn)$ or $O(m+n)$

Still polynomial:

So:

$O(n+m)$

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

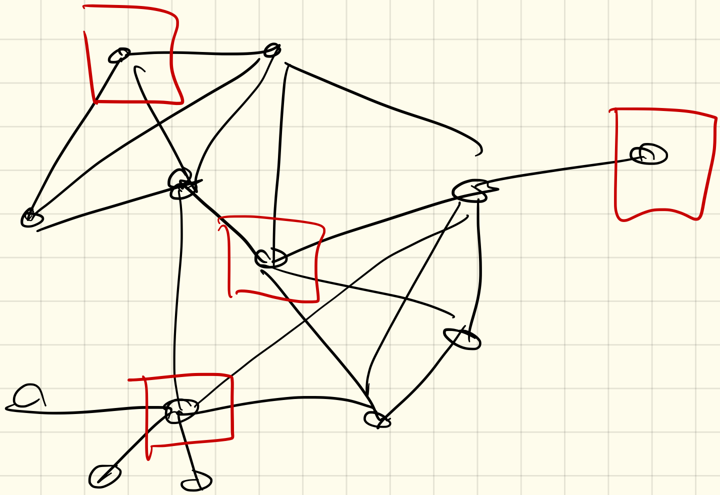
$$T_{\text{CSAT}}(n) \leq O(n) + T_{\text{3SAT}}(O(n)) \implies T_{\text{3SAT}}(n) \geq T_{\text{CSAT}}(\Omega(n)) - O(n)$$

(go check reading)

Next Problem:

Independent Set:

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



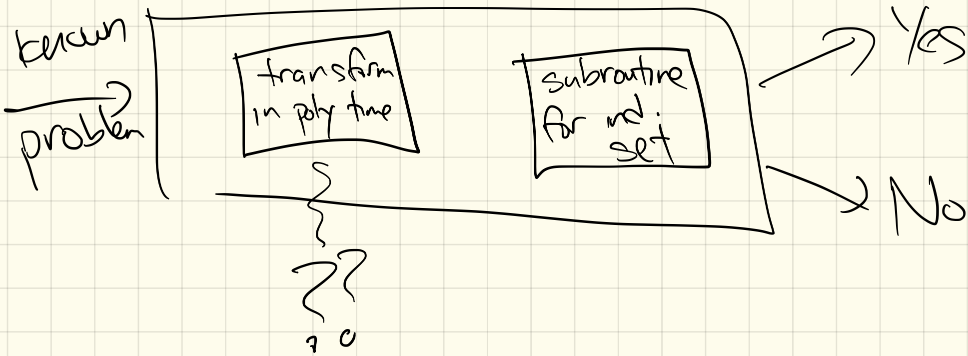
decision version:

Does G have ind set of size k?

(Wait - didn't we see this already!?)
solved in paths or trees

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

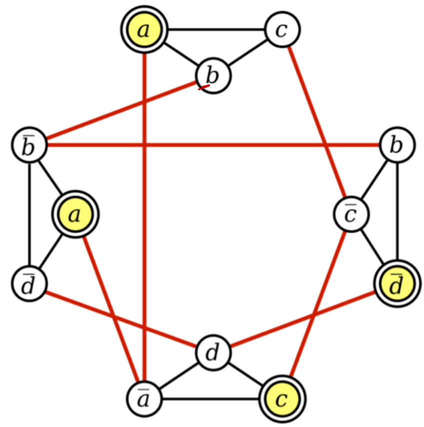
$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

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 inverse

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 n (= # input ??)

Pf (cont)

So!

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

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