Algorithms - Spring '25

- End of flows
- Complexity

Recap

HW due Monday

Readings Mon + Wed

No class Friday

Max flow/Min Cut Algorithms:

Residual-graph based:

- FF: $O((V+E)f^*)$
- Edmonds-Karp: $O(E^2 \log E \log f^*)$
- BFS based: $O(VE^2)$

find a path in G_f & improve

FOR HW

Just cite $O(VE)$ if unsure

$O(VE)$

Technique	Direct	With dynamic trees	Source(s)
Blocking flow	$O(V^2E)$	$O(VE\log V)$	[Dinitz; Karzanov; Even and Itai; Sleator and Tarjan]
Network simplex	$O(V^2E)$	$O(VE\log V)$	[Dantzig; Goldfarb and Hao; Goldberg, Grigoriadis, and Tarjan]
Push-relabel (generic)	$O(V^2E)$	—	[Goldberg and Tarjan]
Push-relabel (FIFO)	$O(V^3)$	$O(VE\log(V^2/E))$	[Goldberg and Tarjan]
Push-relabel (highest label)	$O(V^2\sqrt{E})$	—	[Cheriyan and Maheshwari; Tunçel]
Push-relabel-add games	—	$O(VE\log_{E/(V\log V)} V)$	[Cheriyan and Hagerup; King, Rao, and Tarjan]
Pseudoflow	$O(V^2E)$	$O(VE\log V)$	[Hochbaum]
Pseudoflow (highest label)	$O(V^3)$	$O(VE\log(V^2/E))$	[Hochbaum and Orlin]
Incremental BFS	$O(V^2E)$	$O(VE\log(V^2/E))$	[Goldberg, Held, Kaplan, Tarjan, and Werneck]
Compact networks	—	$O(VE)$	[Orlin]

Figure 10.10. Several purely combinatorial maximum-flow algorithms and their running times.

Another: Exam scheduling

Input: n classes, r classrooms
t time slots, p proctors

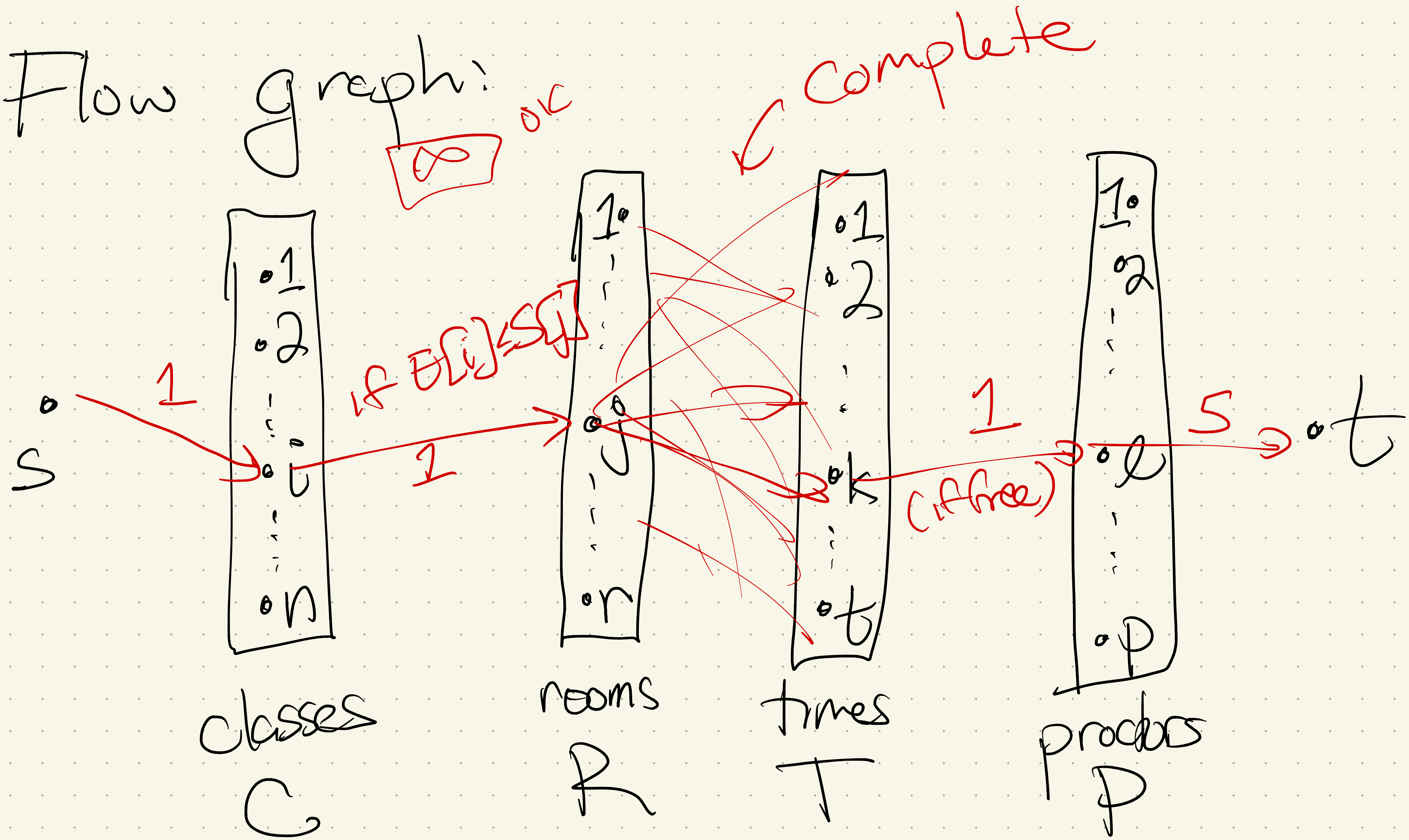
$E[1..n]$: # of students in each class

$S[1..r]$: capacity of each classroom

$A[1..t, 1..p]$: $A[k, \ell]$ is true if ℓ^{th} proctor is free at time slot k

+ each proctor gets ≤ 5 classes.

Flow graph:



Edges: $\forall v \in C$, $s \to v$ with cap $= 1$,

so flow paths "assign" 1 class to valid room, time & proctor

Then $C \to R$ edges:

If $E[i] \leq S[j]$: class will fit in room.

So add edge $i \to j$ [for $i \in C$ + $j \in R$].

capacity = ~~2~~ 1

Then $R \to T$ edges:

add all edges $j \to k$ with capacity = 1, since each room is open to start at every time

Next: $T \to P$ edges

If $A[k, \ell]$ is true, then proctor ℓ is open at time k.

↳ add edge of capacity $= 1$
(so can't be assigned 2)

Finally: $P \to t$

Add all $\ell \to t$ edges, for $\ell \in P$

Capacity $= \cancel{2}$ 5

Then: find max flow.

If $= n$, done! If $< n$: problem.

r.t nodes

room1

Find flow paths:

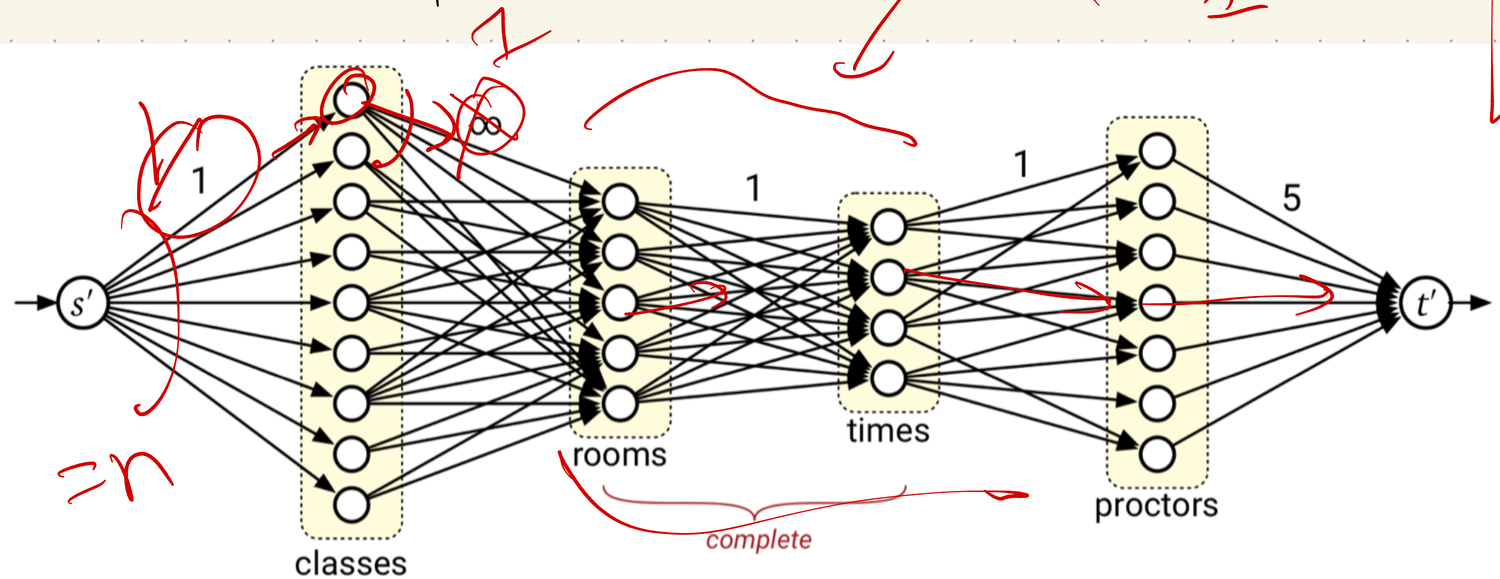


Figure 11.5. A flow network for the exam scheduling problem.

Must go $s \to i \to j \to k \to l \to t$.

So: if $\exists$ flow of value n,
can find assignment of exams.

Other way:
If can assign rooms, classes, times, & proctors, can also use each assignment to build a flow path of value 1 in G. So, assignment of k classes $\Rightarrow$ flow of value k.

Runtime:

$$V = \underbrace{2}_{s+t} + n + r + t + p$$

$$E \leq \underbrace{n}_{s \to n} + n \cdot r + r \cdot t + tp + \underbrace{p}_{\text{proctors} \to t}$$

$$N = \max\{n, r, t, p\}$$
$$= n + r + t + p$$

+ Orlin $= O(VE) =$

$$O((n+r+t+p)(nr + rt + tp))$$

$$= O(N^3)$$

Quantifying Hardness:

Fundamental question:

Are there "harder" problems? (Yes)

Why should we care?

Because sometimes we can't solve exactly!

How do we rank?

Runtime: $O(n) \ll O(n^2) \ll O(n^3)$

$\ll O(2^n) \ldots$

The bad news: Undecidabily

Some problems are impossible to solve!

The Halting Problem:

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

Output: True/False ←

(Utility should be obvious!)

Note: Cant just simulate P on I.

Why? if doesn't halt, don't know when to stop

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.

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

Need to find a contradiction now...

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

$G(X)$: if $H(X,X) = \text{false}$
    return false
  else
    loop forever

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

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 <u>are</u> solvable in polynomial time.

Some things are impossible

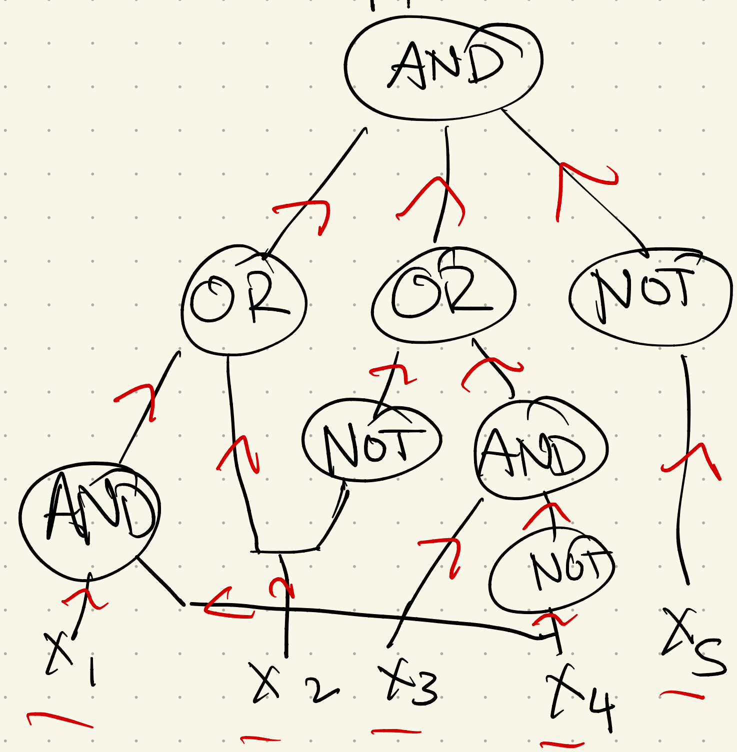
But - what is in between?

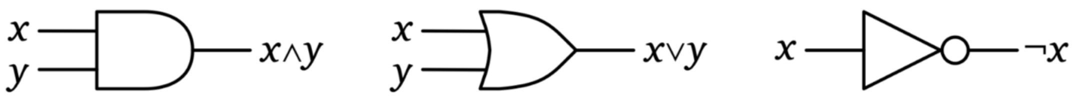
<u>Idea:</u>

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

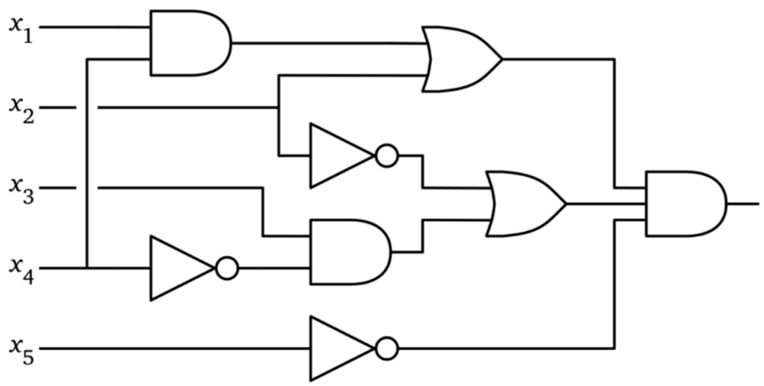
The first problem found:
Boolean circuits

"Flipped view":





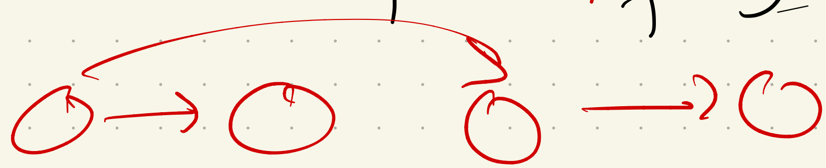
An AND gate, an OR gate, and a NOT gate.



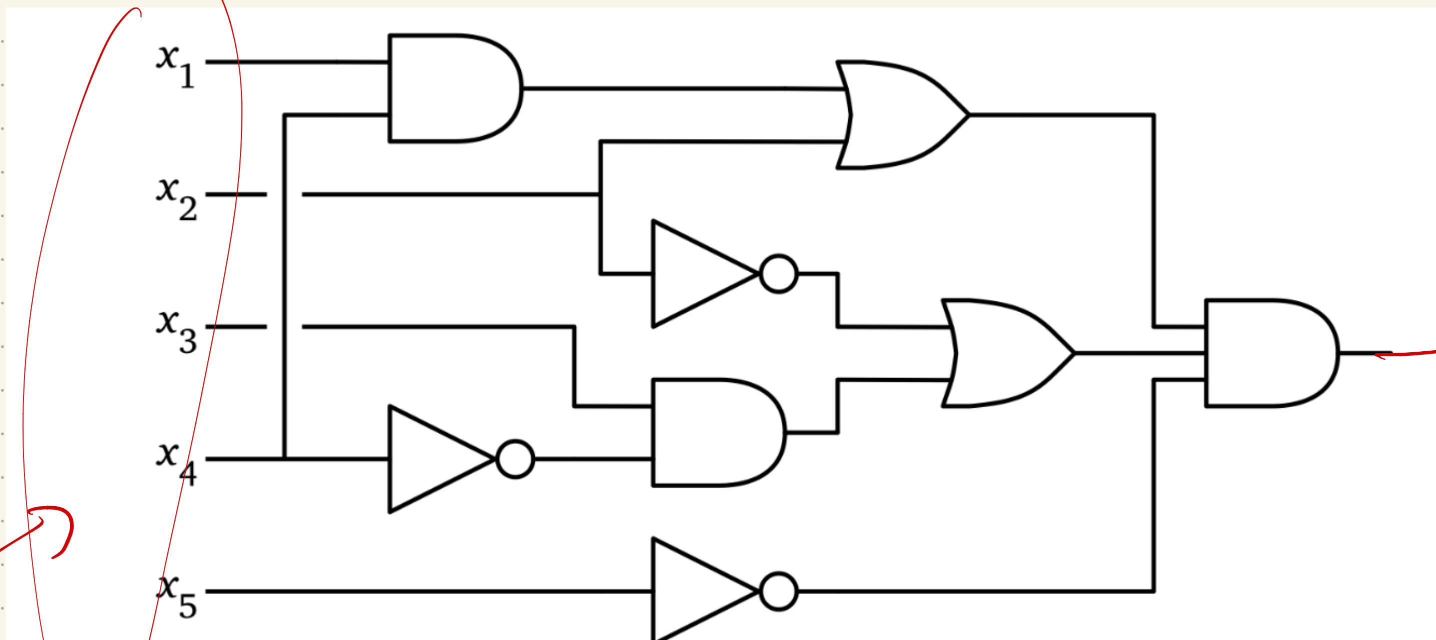
A boolean circuit. inputs enter from the left, and the output leaves to the right.

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

How? Top sort



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



→ True

one set of T/F foreach input

Known as CIRCUIT SATISFIABILITY
(or CIRCUIT SAT)

Best known algorithm:

Try all possible inputs

If one works, return true

else return false

top sort

Runtime: n T/F values

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

Note: Best known approach

(No lower bound) $\Omega(2^n)$

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 x in list? (whole book)
Is flow in G = n?

$P \subseteq NP$

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

Examples: CIRCUIT SAT (use top sort)
everything in P!

Co-NP: Set of problems where we can verify a "no".

Examples: Is number n prime?

$\underline{\text{Dfn}}$: 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.

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.

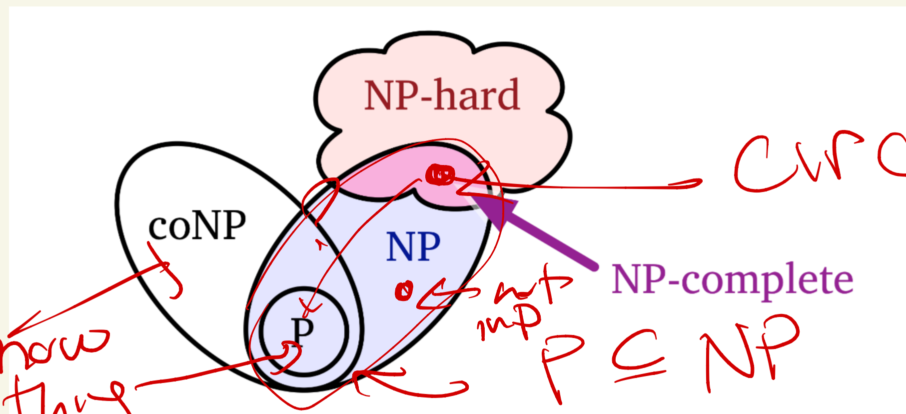
Reduce A to CIRCUIT-SAT in polynomial time: build circuit.

Therefore, I have a poly time alg for A.

$A \overset{}{\underset{\text{convert}}{\Longrightarrow}}$ circuit ∎

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

is $P = NP$?



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

circuit SAT

not in P

$P \subseteq NP$

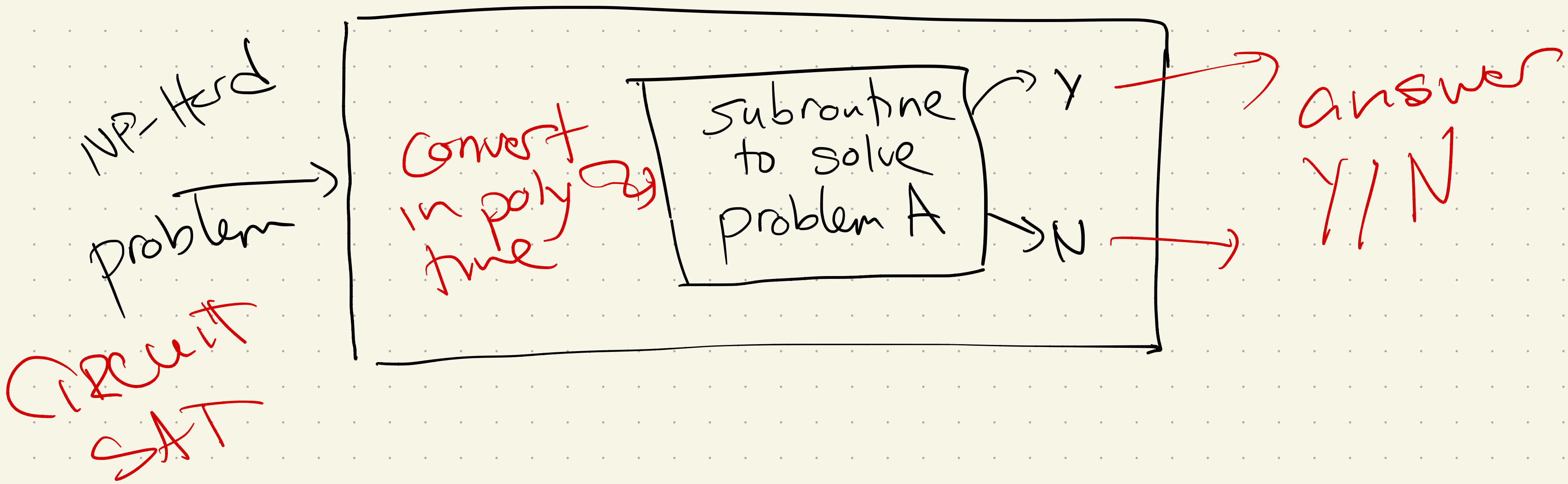
don't know of anything here

NP-Complete: NP-Hard & in NP

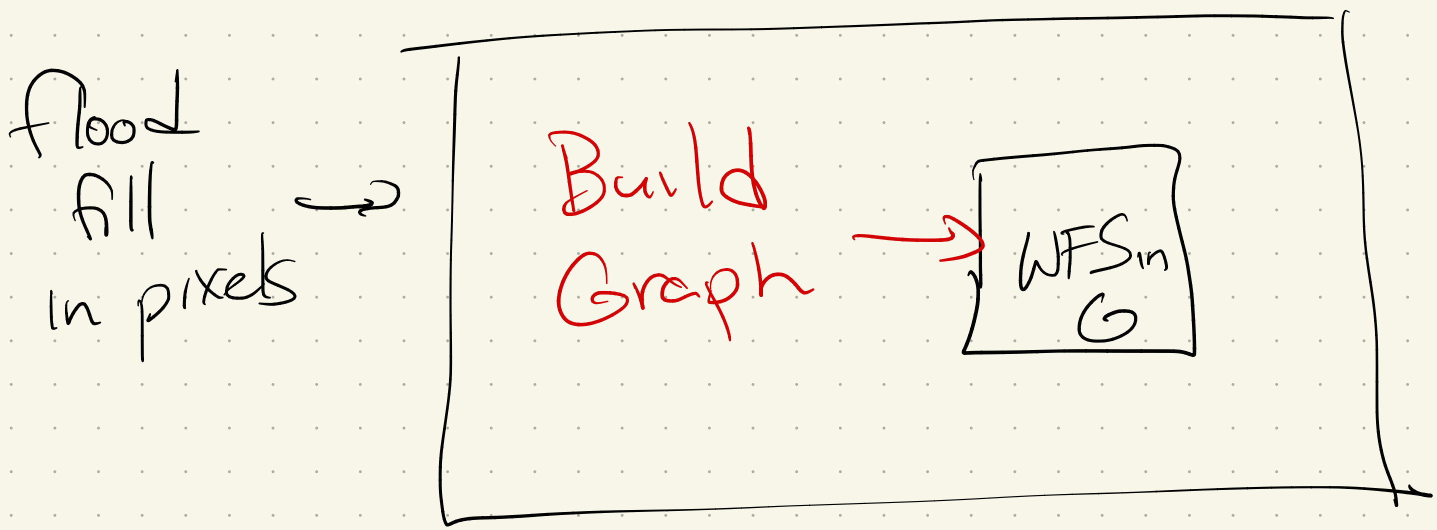
To prove NP-Hardness of A:

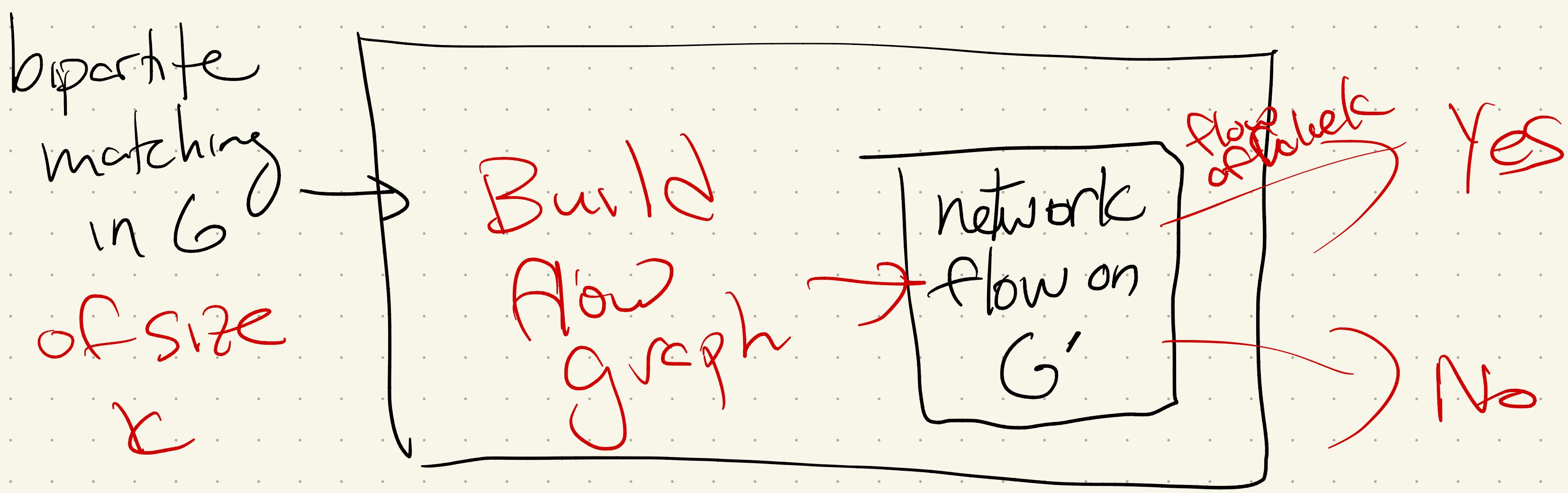
Reduce a known NP-Hard problem to A.

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



We've seen reductions!
But used them to solve problems!





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>? 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 Problems:

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

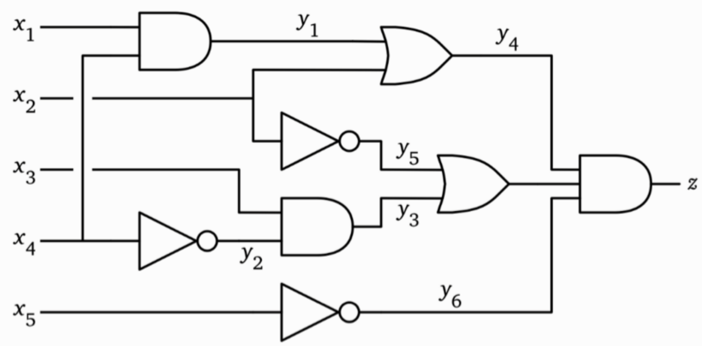
Ex: $(a \vee b \vee c \vee \bar{d}) \Leftrightarrow ((b \wedge \bar{c}) \vee \overline{(\bar{a} \Rightarrow d)} \vee (c \neq a \wedge b)),$

n variables, m clauses

First: in NP?

Thm: SAT is NP-Hard.

pf: Reduce CIRCUIT SAT to SAT:



Input: CIRCUIT

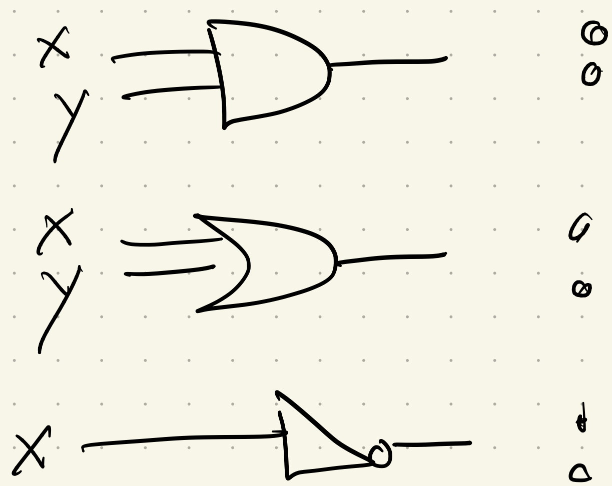
$$(y_1 = x_1 \wedge x_4) \wedge (y_2 = \overline{x_4}) \wedge (y_3 = x_3 \wedge y_2) \wedge (y_4 = y_1 \vee x_2) \wedge$$
$$(y_5 = \overline{x_2}) \wedge (y_6 = \overline{x_5}) \wedge (y_7 = y_3 \vee y_5) \wedge (z = y_4 \wedge y_7 \wedge y_6) \wedge z$$

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

Convert in poly time to clauses:

More carefully:

1) For any gate, can transform:



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

Is this poly-size?

Given n inputs + m gates:

Variables:

Clauses:

So our reduction:

$$
\begin{tikzcd}
\boxed{\text{boolean circuit}} \arrow[r, "O(n)"] & \boxed{\text{boolean formula}} \arrow[d, Rightarrow, "\text{SAT}"] \\
\boxed{\text{TRUE or FALSE}} & \boxed{\text{TRUE or FALSE}} \arrow[l, "\text{trivial}"']
\end{tikzcd}
$$

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

3SAT: 3CNF formulas!

Thm: 3SAT is NP-Hard

pf: Reduce circuitSAT to 3SAT:

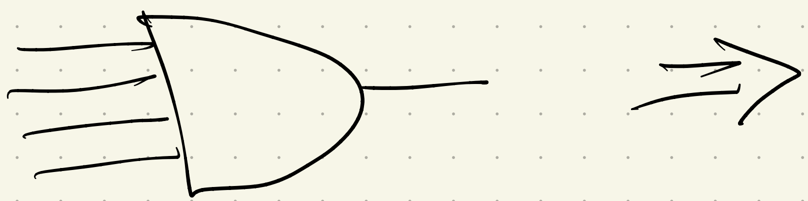
Need to show any circuit can be transformed to 3CNF form

(so last reduction fails)

Instead $\longrightarrow$

Given a Circuit!

1. Rewrite so each gate has ≤ 2 inputs:



2. Write formula, like SAT. Only 3 types!

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

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

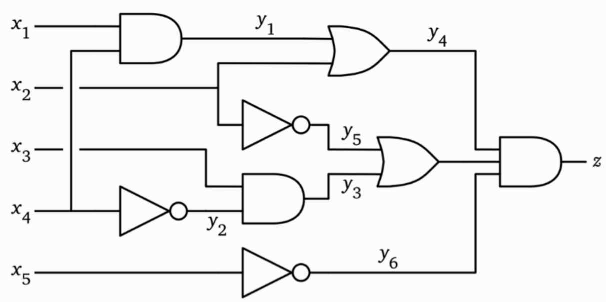
$$a = b \wedge c \longmapsto (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c)$$
$$a = b \vee c \longmapsto (\bar{a} \vee b \vee c) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c})$$
$$a = \bar{b} \longmapsto (a \vee b) \wedge (\bar{a} \vee \bar{b})$$

4 Now, need 3 per clause!

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

Note: Bigger!

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.

$\Downarrow$

$$
\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}
$$

So:

size?

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

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

$O(\quad)$ $\qquad$ $-$

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

Next time:

Can we do this with any useful problems?

(Logic is all well + good...)

Maybe → graphs?