CSCI 3100

Hardness &

Undecidability

Today:

- HW due
- Office hours today:
  12-2:30

Fundamental question:

Are there "harder" problems?

How do we rank?

- Polynomial
- Unsolvable?

Undecidabily:

Some problems are <u>impossible</u> 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: Can't just simulate P on I. Why?

We'd never output False!

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:

$$h(P, I) = \begin{cases} 1 & \text{if } P \text{ halts on input } I \\ 0 & \text{otherwise} \end{cases}$$

$$h(X, X)$$

Now define a program g that uses h:

$$g(X) := \text{ if } h(X,X)=0 \text{ return } 0 \text{ else } (h(X,X)=1) \text{ loop forever}$$

The contraction: What does $g(g)$ do?

Calls $h(g,g)$:

if $h(g,g)=1$, that means g halts on input g.
But then $g(g)$ should run forever!

If $h(g,g)=0$, then g on input g runs forever.
But by dfn of g, should return 0 and halt.
Either way, impossible behavior.

$\blacksquare$

So... what next?

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

Some things are impossible.

But – what is in between?

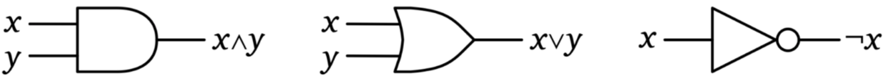
Idea:

- Some things require exponential time.
- Subexponential (but super polynomial)

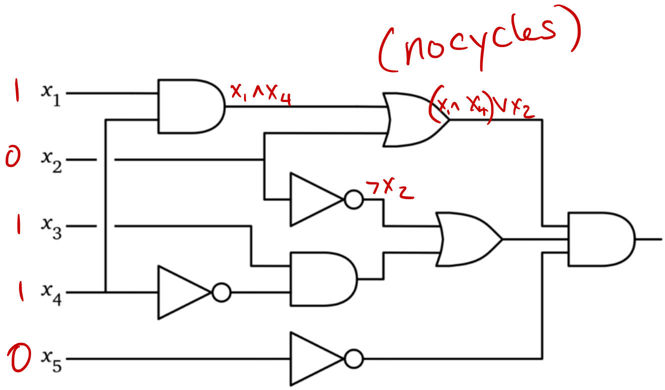
  eg. $2^{\sqrt{\lg n}}$

  ↳ factoring

The first problem found: Boolean circuits



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



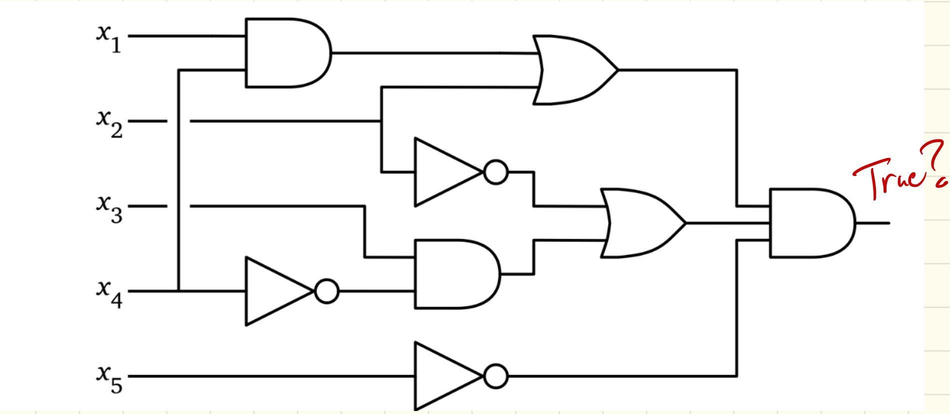
A boolean circuit. inputs enter from the left, and the output leaves to the right.

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

How? "reverse" BFS

$O(m+n)$

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 possible inputs.

Running time:

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

$$\hookrightarrow O(2^n)$$

Note:

P, NP, + co-NP

Consider only decision problems:
so Yes/No output

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

Ex:
- Is list sorted?
- Is x is list?
- Is LIS of length k?
- Is there a flow of value k in G?

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

Ex:
- CircuitSAT

Co-NP: can check "No" answers

Dfn: NP-Hard

X is NP-Hard

$\Longleftrightarrow$

If X could be solved in polynomial time, then

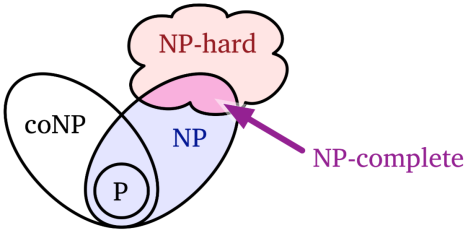
$P = NP$.

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

(Paths story in reading...)

Cook-Levine Thm:

Circuit SAT is NP-Hard.



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

Polynomial heirarchy

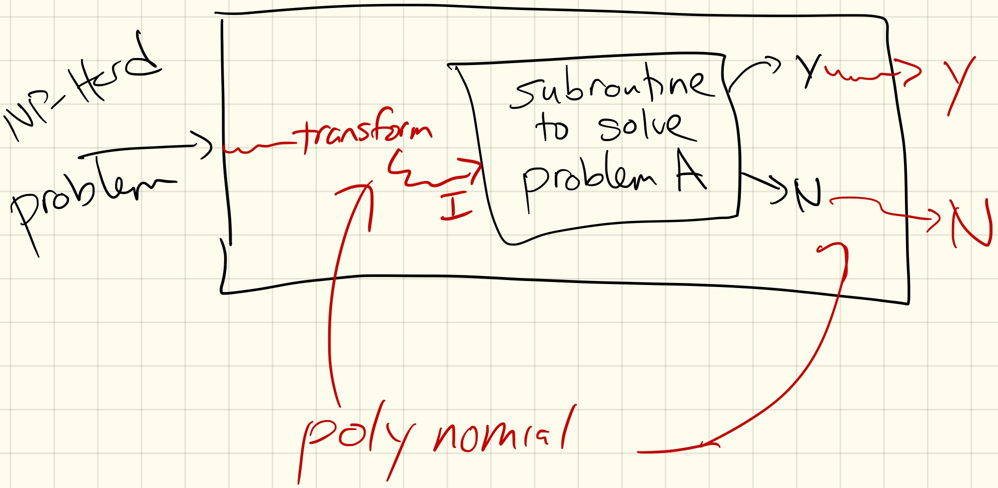
NP-Complete:

- In NP
- And NP-Hard

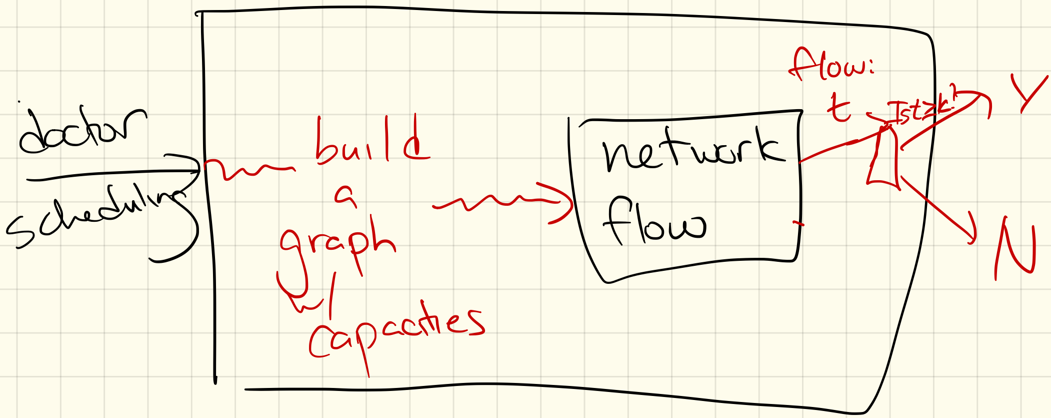
To prove NP-Hardness of A:

Reduction:

Reduce a known NP-Hard problem to A.



We've seen reductions!



$$O(\text{doctor scheduling}) \leq O(\text{network flow})$$

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?

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 ~~m~~ variables,

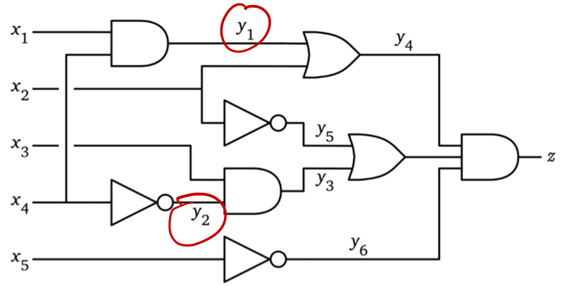
m ~~n~~ clauses

In NP:

Given assignment, can check in poly time.

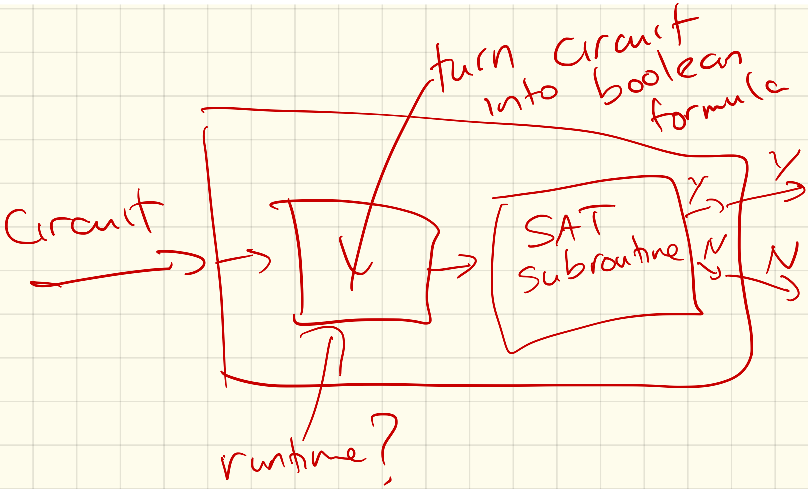
Thm: SAT is NP-Hard.

pf: Reduce CIRCUITSAT 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.



So our reduction:

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

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

Tomorrow: More reductions!