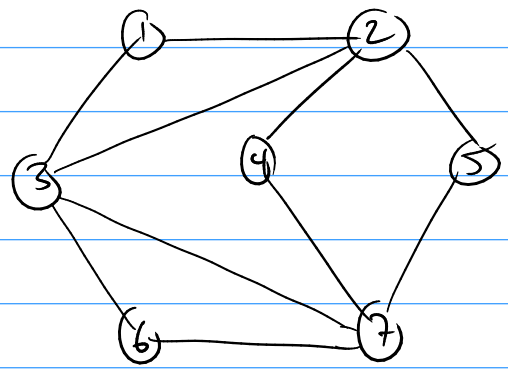
Independent set.

Def'n An <u>independent set</u> in an undirected graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that no two vertices in S are ajacent.

e.g.



$\{1, 4, 5, 6\}$

$\{6\}$

$\emptyset$

$\{2, 7\}$

Interesting Q: given a graph G, what is <u>biggest</u> independent set?

Brute force:

- Try adding in all possible ways? $O(n!)$?
- List all subsets, check which are independent? $O(v^2 \cdot 2^v)$?

We don't know any better algorithms than exponential!

Decision problem version: Given graph G and a number k, does G have an ind. set. of size $\geq k$?

Satisfiability

Def'n A <u>term</u> is either a variable x_i or its logical negation, $\overline{x_i}$ $(\neg x_i)$

Def'n A <u>clause</u> is a disjunction (or) of one or more terms.

e.g. $x_1 \vee \overline{x_3} \vee x_4$. e.g. x_5 e.g. $x_1 \vee \overline{x_1}$

Def'n A truth assignment is a function assigning a T/F value to each variable. A truth assignment satisfies a clause if it makes the clause true.

e.g. $\{x_1 \mapsto T, x_3 \mapsto T, x_4 \mapsto F\}$ satisfies

Def'n A collection of clauses $C_1, C_2, \ldots, C_k$ is satisfied by a truth assignment if it satisfies all of them, i.e. $C_1 \wedge C_2 \wedge \ldots \wedge C_k$.

e.g. $x_1 \vee \overline{x_2}$, $\overline{x_1} \vee \overline{x_3}$, $x_2 \vee \overline{x_3}$
is satisfied by $\{x_1 \mapsto ?, x_2 \mapsto F, x_3 \mapsto F\}$.

e.g.
x_1, $x_3 \vee \overline{x_1}$, $\overline{x_3} \vee \overline{x_1}$ is not satisfiable!

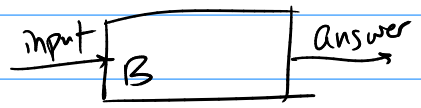
Q (SAT): Given a collection of clauses, is it satisfiable?

Brute force: $O(2^n)$, try all truth assignments.

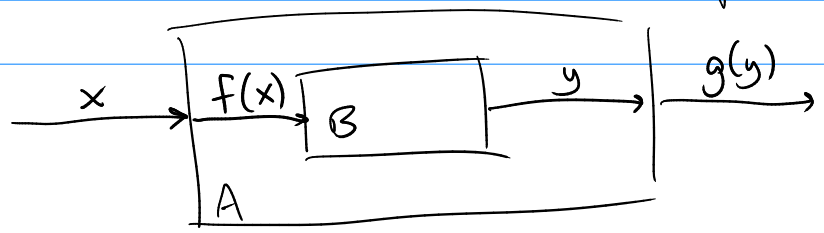
Q (3-SAT): Given a collection of clauses, each of which has exactly 3 terms, is it satisfiable?

Reducibility

If we have a black box to solve problem B:



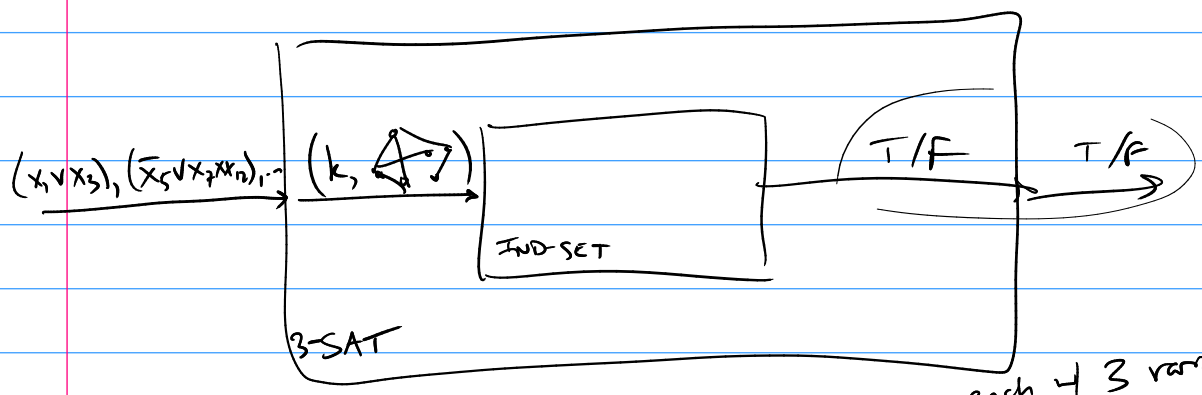
We can use it to solve other problems:



If this correctly solves problem A whenever the inner black box correctly solves problem B, then we say "A is reducible to B" and write

$$A \leq B.$$

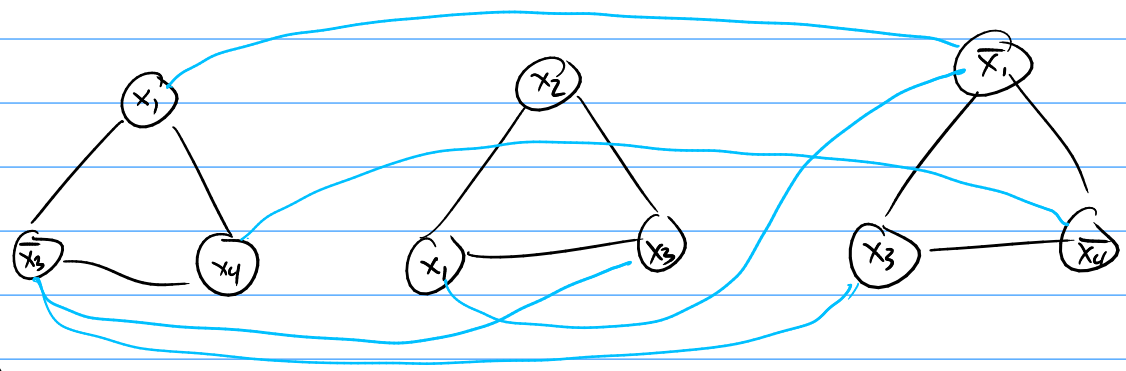
Claim: 3-SAT $\leq$ IND-SET. (Fact: SAT $\leq$ 3-SAT).



Given a set of clauses $C_1, \ldots, C_n$ (each w/ 3 variables): goal is to translate into a graph G such that G has an independent set of size n iff $C_1, \ldots, C_n$ are satisfiable.

Step 1: Create $3n$ triangles, one per clause, w/ vertices labelled by the terms of each clause.

e.g. $(x_1 \vee \bar{x}_3 \vee x_4) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$



Step 2: Connect any x_i and any $\bar{x}_i$ with an edge.

The resulting graph has independent set of size n iff clauses can be satisfied — make each vertex in the ind. set true.

<u>Def'n</u> If $A \leq B$ and the translation function f takes polynomial time to compute, then we say A is <u>polynomial-time reducible</u> to B, and write

$$A \leq_p B$$

"Polynomial time" means $O(n^c)$ for some constant c.
e.g. $O(n^2)$, $n^2 \lg n$, $n^{3.27}$, ...

<u>Thm</u> If B is solvable in polynomial time and $A \leq_p B$, then A is also solvable in polynomial time.

<u>Cor.</u> If $A \leq_p B$ and A is <u>not</u> solvable in poly. time, then neither is B.

<u>P and NP</u>

- The input to a problem will be encoded as a binary string.
- The <u>size</u> of the input = # of bits.
- A <u>decision problem</u> can be identified w/ the set of binary inputs for which the answer is true.

- An algorithm A <u>solves</u> decision problem D iff for all binary strings s, $A(s) = T$ iff $s \in D$.

- If there is a polynomial $p(n)$ such that $A(s)$ always terminates in at most $p(|s|)$ steps, we say A is a <u>polynomial-time algorithm.</u>

<u>Def'n</u> P is the set of decision problems D which are solvable in polynomial time. i.e. for which there exists a poly-time algorithm that solves them

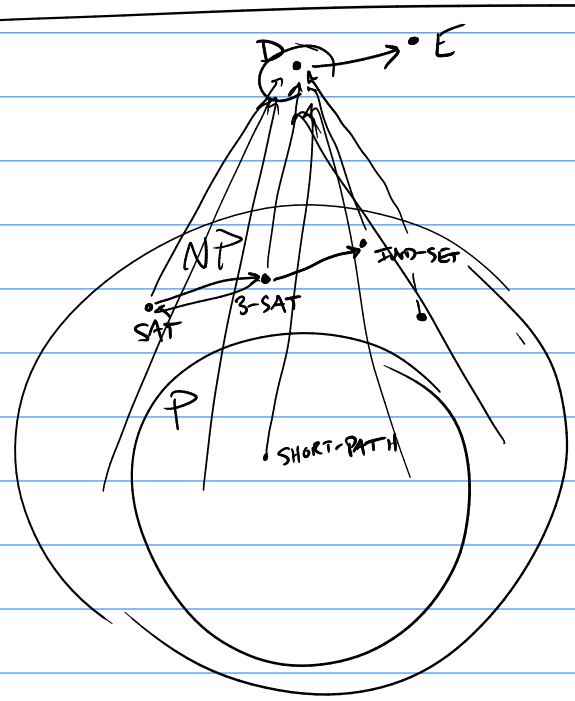
Def'n An algorithm B is a <u>polynomial-time certifier</u> for a decision problem D if:

input — evidence or "certificate"

- B is a poly-time algorithm taking inputs s and t
- there is a polynomial p such that for all bitstrings s, we have $s \in D$ iff there exists some bitstring t (a "certificate") with $|t| \le p(|s|)$ and $B(s,t) = T$.

Def'n NP is the set of decision problems with a poly-time certifier.

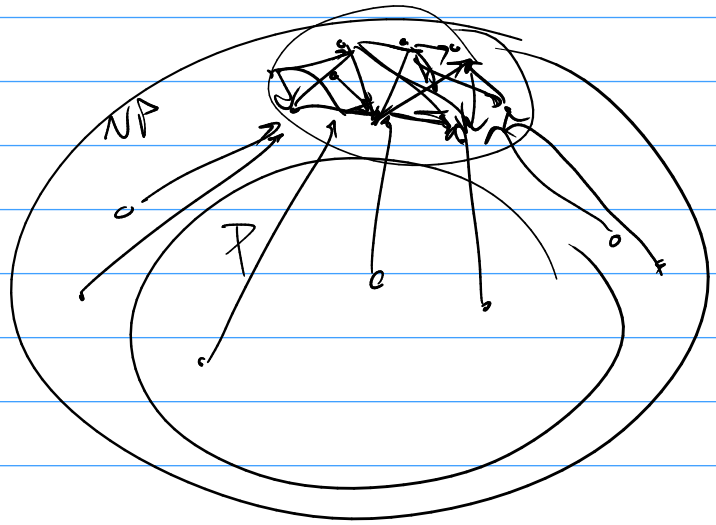
$P \subseteq NP$ since B could just ignore t and solve the problem.
But is $NP \subseteq P$? \$1 million.



Def'n A decision problem D is <u>NP-hard</u> if for all $X \in NP$, $X \le_p D$.

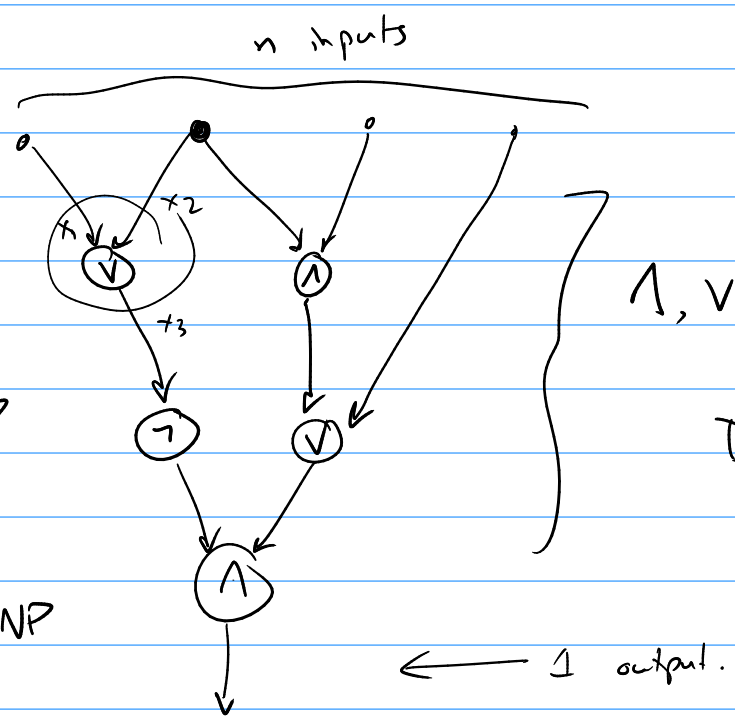
Lemma. If D is NP-hard and $D \le_p E$, then E is also NP-hard (by transitivity of $\le_p$).

Def'n If D is NP-hard and $D \in NP$, then we say D is <u>NP-Complete</u>.



<u>The first NP-Complete problem!</u> (Cook, Levin (1971))

<u>CIRCUIT-SAT</u>



Q: Is a given graph satisfiable?

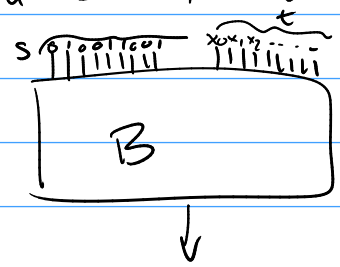
CIRCUIT-SAT $\in$ NP

Claim: CIRCUIT-SAT is NP-hard!

<u>Proof</u>: Let X be an arbitrary NP problem. Then X must have a polynomial-time certifier $B(s,t)$.

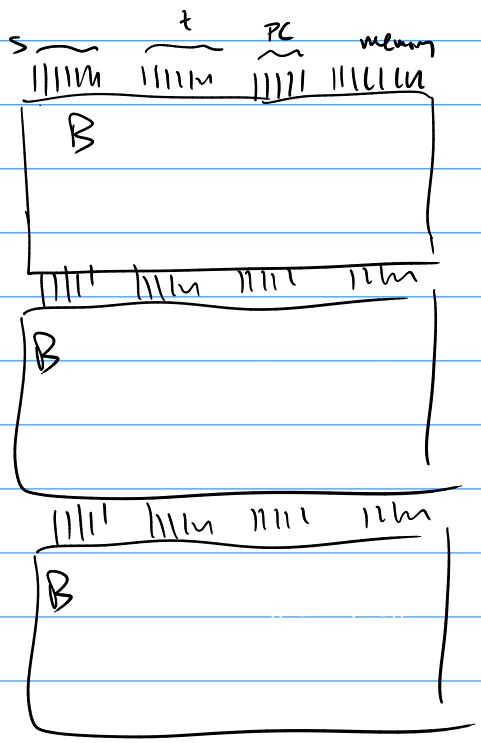
Given an input s, we must build a circuit which is satisfiable iff the answer for input s is supposed to be true for problem X.

Idea: build a circuit that simulates algorithm B!



← satisfiable iff there is a certificate t which makes B say yes.

Problem: what about cycles? Answer: just chain a bunch of copies!



$\leq_p$ HAMILTONIAN CIRCUIT $\leq_p$ TRAVELLING SALESMAN

Claim: CIRCUIT-SAT $\leq_p$ SAT $\leq_p$ 3-SAT $\leq_p$ IND-SET.

Hence all these are NP-complete!

<u>Prop.</u> If decision problem D is NP-Complete, D is solvable in poly. time iff P = NP.

<u>Cor.</u> If you find a poly time algorithm to solve <u>any</u> NP-complete problem, you win $1 million.