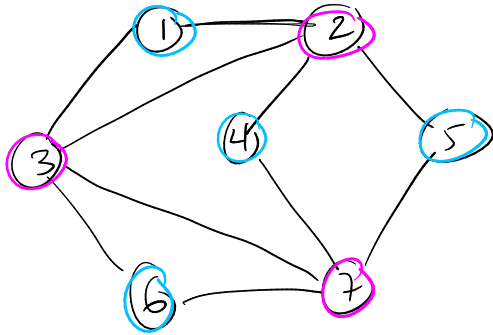


Hard (?) problems

Def: An independent set in an undirected graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ such that no two vertices in S are adjacent.

e.g.



Independent sets:

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

$\{3, 4, 5\}$

$\{2, 6\}$

$\{4\}$

$\{1, 4, 6\}$

$\emptyset$

Interesting Q's about independent sets?

- Largest ind. set? $\leftarrow$

- Total # of ind. sets? $\leftarrow$

Brute force?

- List every subset, check if each is independent. keep track of biggest.

$O(2^V \cdot V^2)$. — fundamentally, we don't know better algorithms.

Example 2: Satisfiability (SAT)

Def: A term is either a variable or the negation of a variable

e.g. $x_1, x_3, \neg x_5$
 $(\overline{x_5})$

Def: A clause is one or more terms combined with OR.

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

x_2

$x_1 \vee x_2 \vee x_3 \vee x_5 \vee \overline{x_7} \vee x_{10}$

Def'n A truth assignment assigns T or F to each variable.

e.g. $\{x_1 \mapsto T, x_2 \mapsto F, x_3 \mapsto T\}.$

A truth assignment satisfies a clause if it makes the clause true.

e.g. the above satisfies

$$\underline{x_2} \vee \underline{x_1} \vee \underline{\overline{x_3}}.$$
$$F \vee T \vee F \equiv T.$$

but not

$$x_2 \vee \overline{x_3} \vee \overline{x_1}$$

Def'n A truth assignment satisfies a collection of clauses if it satisfies all of them i.e. $C_1 \wedge C_2 \wedge \dots \wedge C_n.$

e.g. $(x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (x_2 \vee \overline{x_3})$

$\{x_1 \mapsto T, x_2 \mapsto T, x_3 \mapsto F\}$ satisfies above clauses.

e.g. $x_1, (x_3 \vee \overline{x_1}), (\overline{x_1} \vee \overline{x_3})$

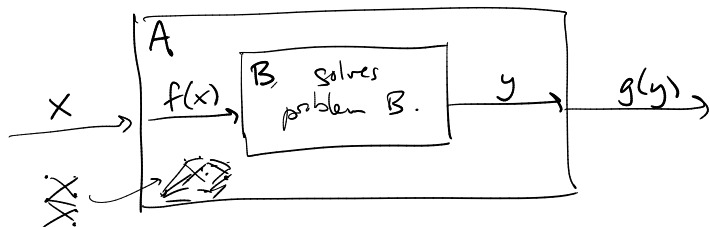
is not satisfiable, i.e. no truth assignment satisfies it.

Q (SAT) : Given a collection of clauses $C_1, \dots, C_k$ over the variables $\{x_1, \dots, x_n\}$, are the clauses satisfiable?

Brute force : try every truth assignment, $\Omega(2^n).$

Variant : Same q, but where all clauses have exactly 3 terms (3-SAT).

How do we know if one problem is "harder than" another?



If A can be solved in this way using a black box that solves B , we say " A is reducible to B " and write $A \leq B$.

eg. $\text{Bipartite matching} \leq \text{Max flow}$
 $\text{SAT} \leq \text{3-SAT}$
 $\text{3-SAT} \leq \text{INDEPENDENT SET.}$

P vs NP

Intuitively:

P = problems that can be solved in polynomial time

NP = problems whose solutions can be verified in polynomial time.

Decision problems

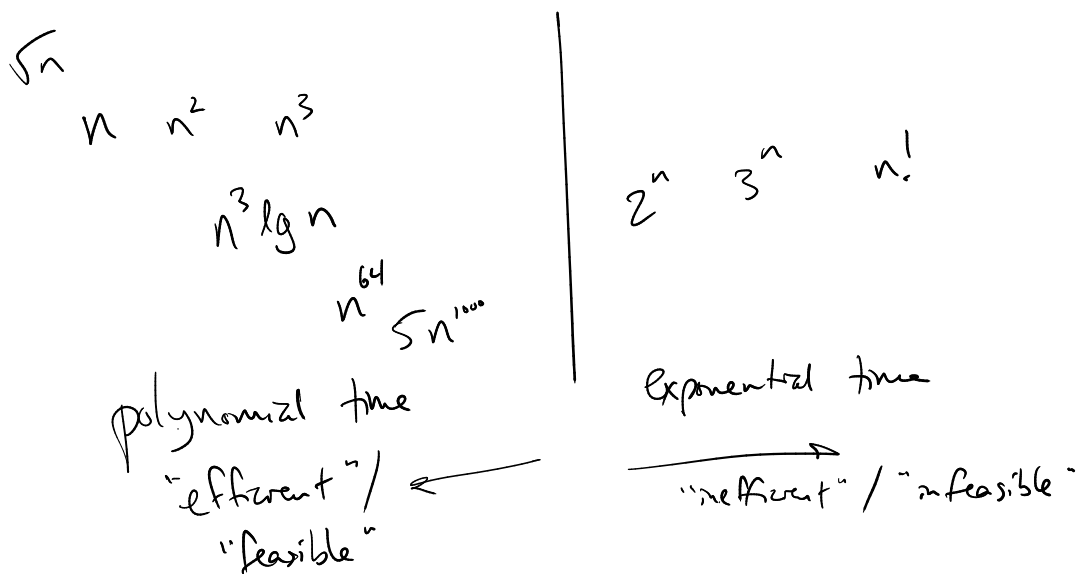
A decision problem is a problem with a Y/N answer.

eg. SAT: satisfiable or not?

IND-SET: Given (G, k) , is there ind. set in G of size k ?

- we will consider inputs to problems as bit strings.
- The size of an input = # of bits. = $|s|$

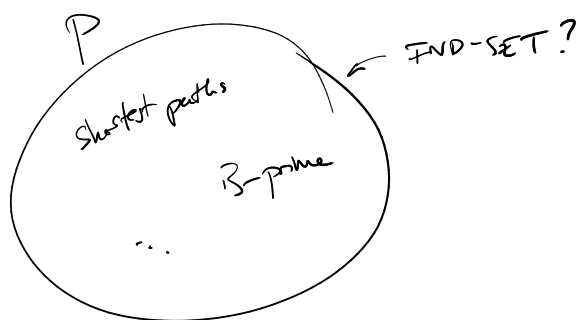
Def'n A polynomial-time algorithm is an algorithm A such that for every input s , $A(s)$ finishes in $\leq p(|s|)$ steps where $p(n)$ is a polynomial.



We can identify a decision problem D with the set of inputs for which the answer should be yes.

Def'n An algorithm A solves decision problem D if for any s ,
 $A(s) = T$ iff $s \in D$.

Def'n P is the set of all decision problems which are solved by some polynomial-time algorithm.



Def'n Given a decision problem D , a polynomial-time certifier (verifier) is an algorithm $B(\underline{s}, \underline{t})$ such that:

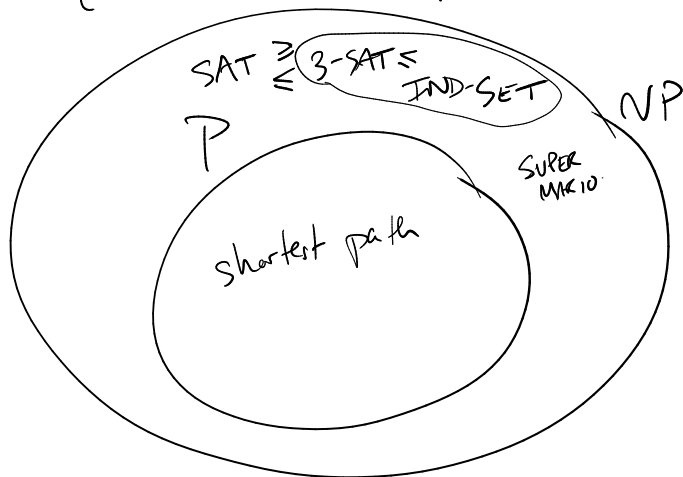
input $\swarrow$ $\nwarrow$ certificate

- There exists t such that $B(s, t) = T$ iff $s \in D$.
- There is a polynomial P such that $|t| \leq P(|s|)$
- B takes polynomial time.

Defn NP is the set of decision problems which have a poly-time certifier.

Claim: $P \subseteq NP$.

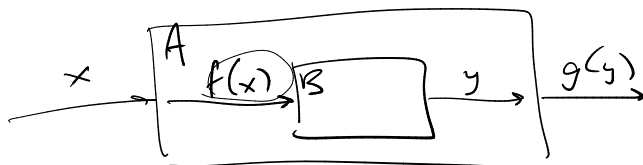
If $D \in P$, it has a poly-time solution A . We must show $D \in NP$ by giving a poly-time certifier for D . Easy: $B(s, t)$ ignores t and runs $A(s)$.



Question (\$1 million):

does $P = NP$?

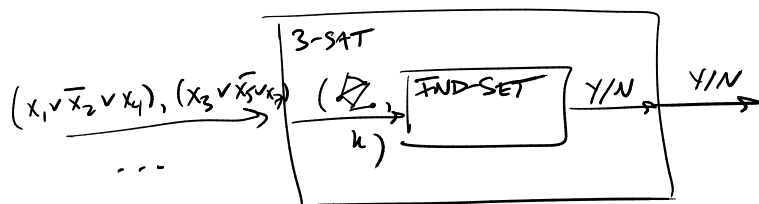
Defn $A \leq_p B$ (" A is poly-time reducible to B ") if $A \in B$ and the translation functions f and g take poly time.



Lemma If $A \leq_p B$ and $B \in P$, then $A \in P$. $\leftarrow$

Cor: If $A \leq_p B$ and $A \notin P$, then $B \notin P$. $\leftarrow$

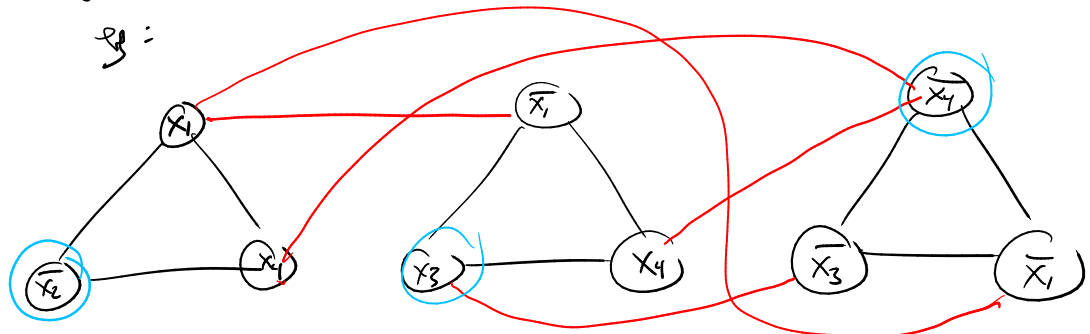
Thm: $3\text{-SAT} \leq_p \text{IND-SET}$.



Proof. Given a set of clauses $C_1, \dots, C_n$. Example:

$(x_1, \bar{x}_2, x_4)$ $(\bar{x}_1, x_3, x_4)$ $(\bar{x}_4, \bar{x}_5, \bar{x}_1)$

Create n triangles, with vertices of each triangle labelled w/ the terms of clauses. eg:

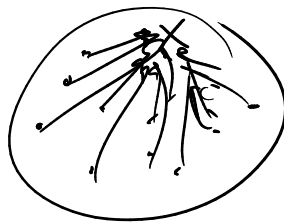


Add an edge between every pair of vertices labelled $x_i, \bar{x}_i$. The resulting graph will have an ind. set of size n iff the clauses are satisfiable.

Def'n A decision problem X is NP-hard if $D \leq_p X$ for all $D \in NP$.



Def'n A decision problem X is NP-Complete if X is NP-hard and $X \in NP$.



Lemma. If X is NP-hard and $X \leq_p Y$ then Y is NP-hard.

Lemma If X is NP-complete, then $X \in P$ iff $P = NP$.

Proof: If $X \notin P$, then $P \neq NP$.

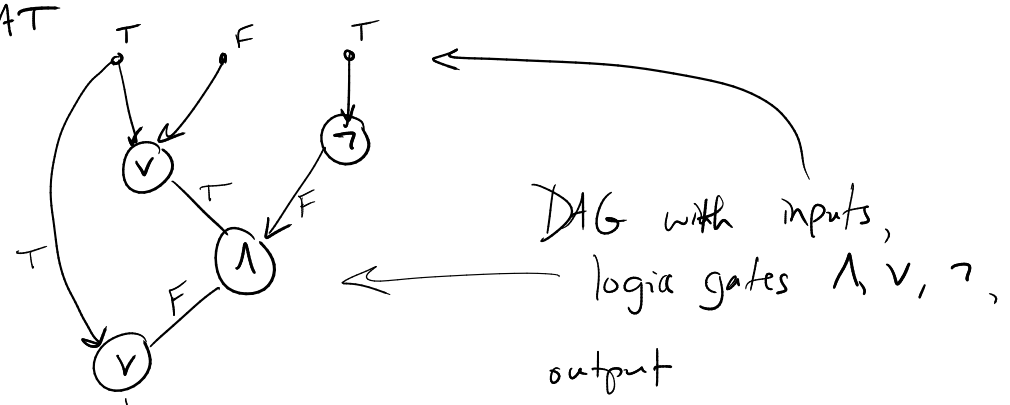


If $X \in P$, then $P = NP$ since any problem in $NP \leq_p X$ and hence $\in P$ as well.

Cor. If you solve an NP-complete problem in poly time, you win \$1 million.

The first NP-complete problem: (Cook, Levin 1971)

CIRCUIT-SAT

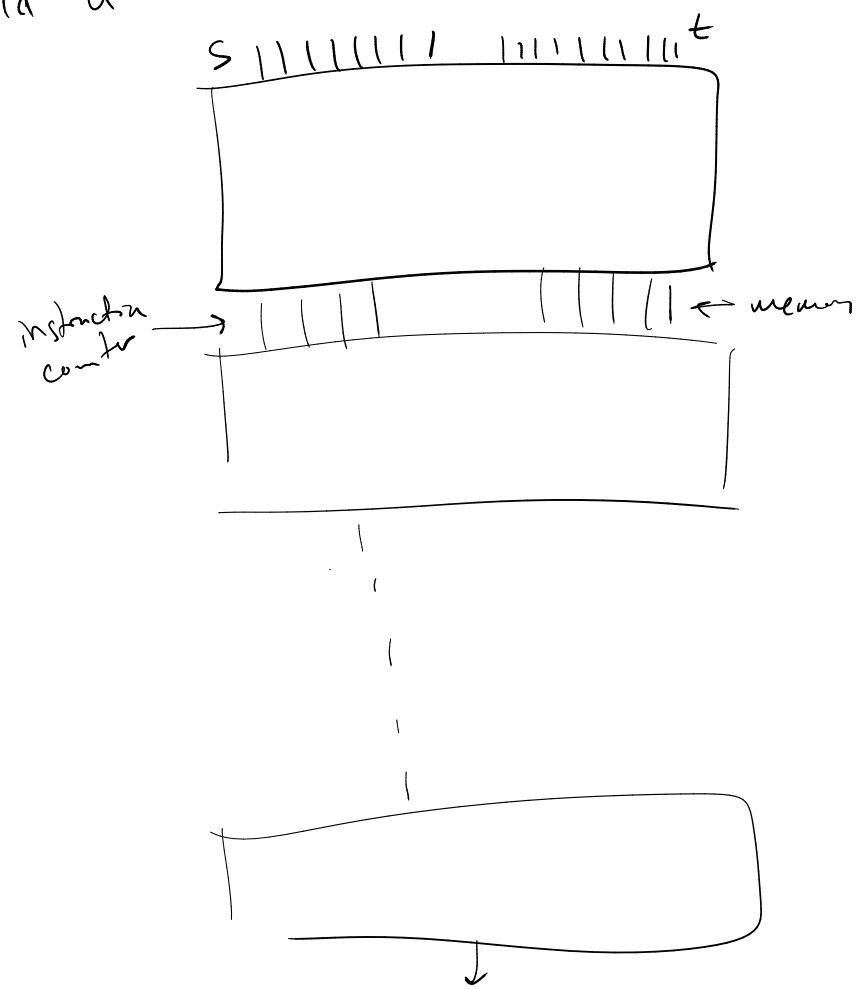


Q: is there a way to set inputs to make output true?

Thm: CIRCUIT-SAT is NP-hard.

Proof: Given $D \in NP$, D has a certifier $B(s, t)$.

Build a circuit that does B .



ask: is there input t that makes it say true?

$\boxed{\text{CIRCUIT-SAT}} \leq_P \text{SAT} \leq_P 3\text{-SAT} \leq_P \text{IND-SET}$
NP-complete ↗ ↗ ↗
 hence so are these

$\text{IND-SET} \leq_P \text{MARIO}$