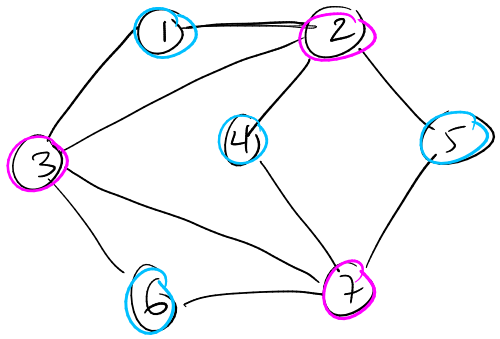
Hard (?) problems

Defn: 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? ←
- Total # of ind. sets? ←

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)

Defn: A *term* is either a variable or the negation of a variable

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

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

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

x_2

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

Defn 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

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

$$F \vee T \vee F \equiv T.$$

but not

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

Defn A truth assignment satisfies a collection of clauses if it satisfies all of them ie. $C_1 \wedge C_2 \wedge \dots \wedge C_n$.

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, $(x_3 \vee \overline{x_1})$, $(\overline{x_1} \vee \overline{x_3})$

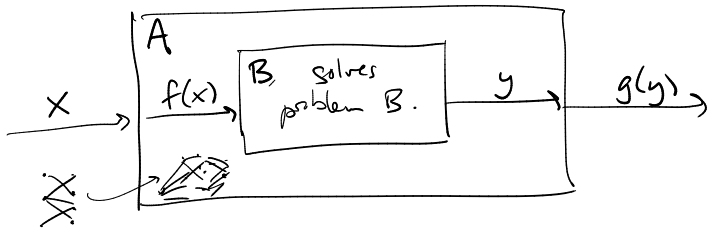
is not satisfiable, ie. no truth assignment satisfies it.

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. Bipartite matching $\leq$ Max flow

SAT $\leq$ 3-SAT

3-SAT $\leq$ 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.

$\sqrt{n}$ n n^2 n^3 $n^3 \lg n$ n^{64} $5n^{1000}$ | 2^n 3^n $n!$

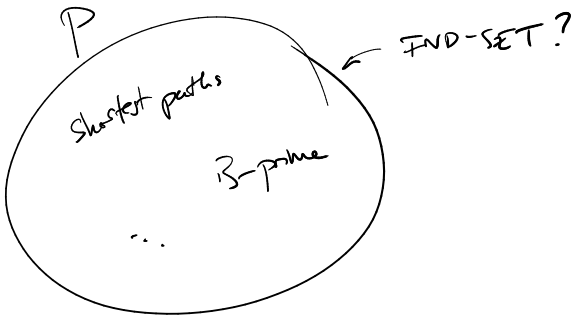
polynomial time "efficient" / "feasible" ← | → exponential time "inefficient" / "infeasible"

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 $s \in D$.

Defn: P is the set of all decision problems which are solved by some polynomial-time algorithm.



Defn: Given a decision problem D, a polynomial-time certifier (verifier) is an algorithm $B(\underline{s}, \underline{t})$ such that:

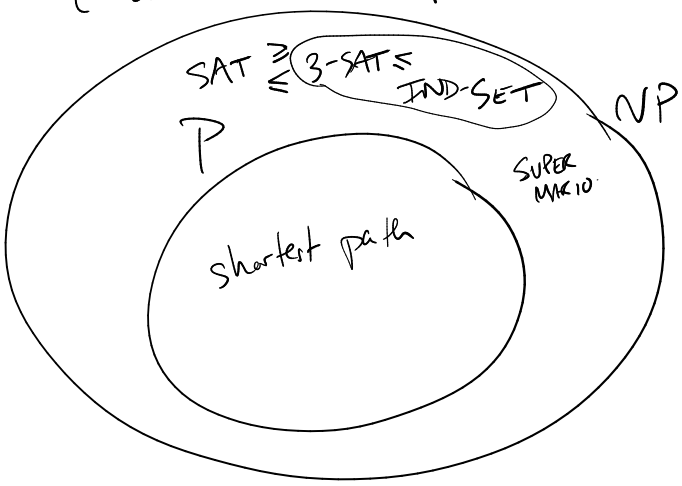
input, certificate

- 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 $P = NP$?