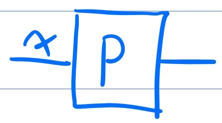
1. Computability and NP-Completeness

(Very very brief intro to computation and complexity theory)

Q: Are all problems solvable by some algorithm?

A: No! e.g. The halting problem (Turing, 1940s) is <u>uncomputable</u>.



Does program P halt on input x?

Assume we have some algorithm that solves the halting problem:

Consider following program:

```
def Z(P):
    if HALT(P,P):
        Loop Forever
    else
        Return
```

Does Z halt on input Z?

- If HALT(Z, Z)=1, it loops forever;
- If HALT(Z, Z)=0, it halts immediately.

Contradictions!

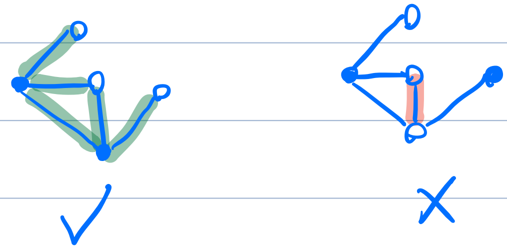
Luckily, almost all problems in real life are computable.
However, many don't (seem to) have polynomial time algorithms.

How to deal with that?

1. Heuristics
2. Simplify problem, maybe assume something about input
3. Approximate
4. Buy more hardware (only gets you that far...)

Now let's see some NP problems.

Def: A <u>vertex cover</u> of an undirected graph $G=(V,E)$ is a subset of vertices that touches all edges.



"install monitoring station for all fibers"

The vertex cover problem (VC) asks to find the smallest vertex cover.

Def: An <u>independent set</u> of an undirected graph $G=(V,E)$ is a subset of vertices that there are no edges between them.

"vertices are broadcasted messages, and edges are between messages that can be confused"

The independent set (IS) problem asks to find the largest independent set.

Best known algs for VC and IS are exponential time.

Reductions

Claim: We can reduce a VC problem to an IS problem,
we denote as $VC \leq IS$.
(Why $\leq$? Think "cannot be harder than")

Proof: Given a VC problem for $G=(V,E)$,
simply solve the IS for same G, say the result is S,
the solution to the VC problem is given by $V-S$. ※

Similarly $IS \leq VC$

Another example:

"satisfiability"
A 3-SAT formula looks like:

variable

NOT

$$(X \vee Y \vee \bar{Z}) \wedge (X \vee \bar{Y} \vee Z) \wedge (\bar{X} \vee \bar{Y} \vee \bar{W}) \wedge (Y \vee W \vee Z)$$
$$\wedge \underbrace{(\bar{X} \vee \bar{W} \vee Z)}_{\text{clause}}$$

OR AND

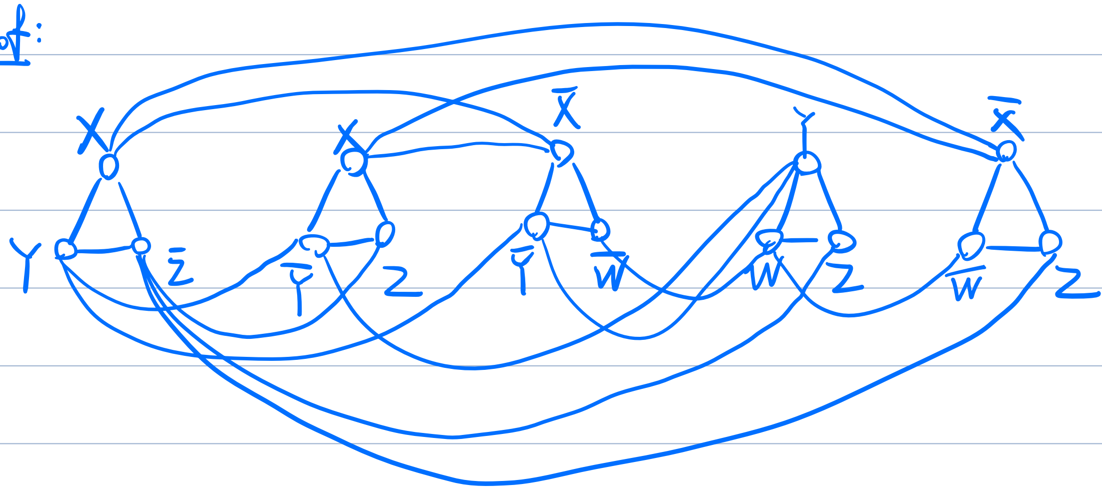
$n=4$ variable
$m=5$ clauses

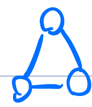
This formula is satisfied by $X=T, Y=F, W=T, Z=T$

3SAT Problem: given a 3SAT formula on n variables, is it satisfiable?
Best alg. is exp. time.

Claim: 3SAT $\leq$ IS

Proof:



Represent each clause with . For each variable X, connect all $(X, \bar{X})$ pairs.

The graph has an ind. set of size m if and only if the formula is satisfiable.

Def: The class NP (Nondeterministic Polynomial time) consists of Yes/No problems ("decisional problems") for which there is an efficient (poly time) "verifier algorithm" that can check that an instance of the problem is a YES instance when given an appropriate "witness".

e.g.
- 3SAT $^{\in NP}$: witness is a satisfying variable assignment
- VC $^{\in NP}$: the decisional problem asks if there is a vertex cover of size $\leq k$. The witness is a vertex cover itself, we check ① it is a valid vertex cover ② its size $\leq k$.
- IS $^{\in NP}$: similar to VC, of size $\geq k$.

Def: A problem $A \in NP$ is called <u>NP-Complete</u> if it is hardest in NP, i.e., $\forall B \in NP$, $B \leq A$.

Thm [Cook-Levin 1971] 3SAT is NP-complete.

Cor: VC, IS are NP-complete.

There are thousands more NP-complete problems.
If you solve any one of them, you solve all of them. This is the famous $P \stackrel{?}{=} NP$ question, one of the "millenium problems" with a \$1M prize.

Summary:

NP ("non-deterministic polynomial time"): decisional problems with witness that can be verified in poly. time. "witness checkable in polynomial time".

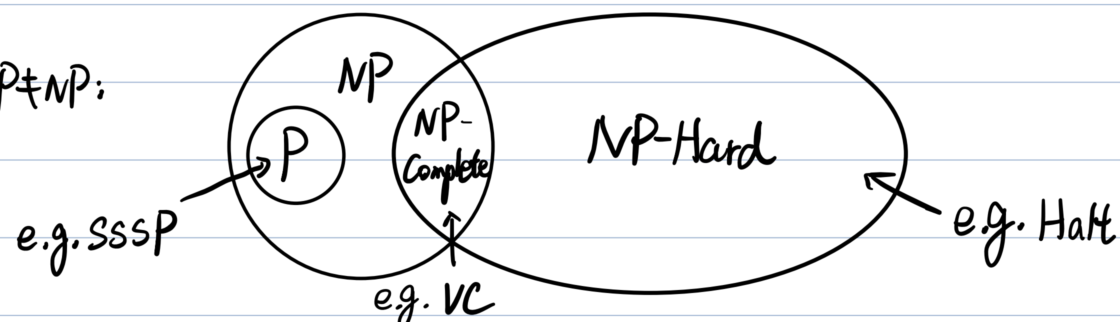
P ("polynomial time"): decisional problems solvable in poly. time.

NP-Hard: Y is NP-hard if $\forall X \in NP$, $X \leq Y$

"Y is at least as hard as any NP problem"

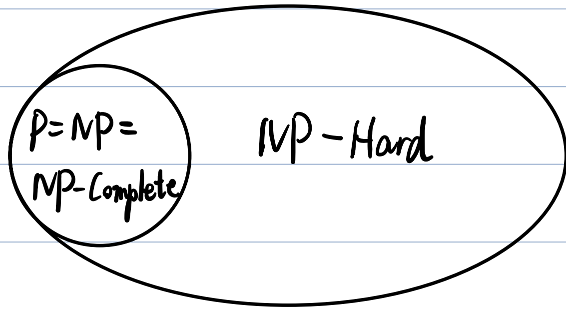
NP-Complete: $NP \cap$ NP-Hard, "hardest problems in NP"

If $P \neq NP$:



Widely believed that $P \neq NP$, but proving it is beyond reach.

If $P=NP$:



If it somehow turned out that $P=NP$, the we would have efficient algorithms for lots of important problems (but also no cryptography ☹).