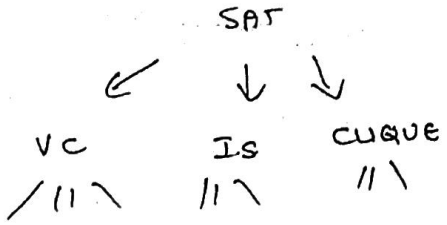
The hardest part is done!

SAT ⟶ (reduced to) Any other given problem in NP

⇩

NP COMPLETE!



and so on...

Let's talk about V.C., I.S. & clique Try to prove they are NP complete.

(reduce SAT to one of them)

Reduce SAT to Clique

Every boolean formula

$(\ldots \vee \ldots \vee \ldots) \wedge \ldots \wedge (\ldots \vee \ldots \vee \ldots)$

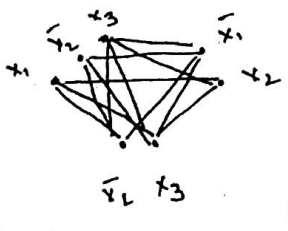
⇩ conjunctive normal form

clause

each clause should be true for satisfiability

SAT ⟶ CLIQUE

$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3)$



$x_1, \bar{x}$ ⇒ no edge

⇒ no edge within a clause

what if the graph has a clique of size $\frac{3}{m}$ $\frac{?}{o}$

$\Rightarrow$ clique contains ~~at least~~ one vertex from each clause group

clique can be > 3 $\frac{?}{o}$ $\Rightarrow$ no

still contains at least one vertex from each clause group

max-size of clique $\Rightarrow$ m (# clauses)

if size $< m$ $\Rightarrow$ not satisfiable
if size $= m$ $\Rightarrow$ satisfiable

Soln $\Rightarrow$ set the vertices of the clique to 1 for satisfiability

$\Rightarrow$ clique is np-complete

$\Rightarrow$ v.c. & i.s. are also np complete.

P vs NP

To show P=NP, we just have to find a polynomial time algorithm for any np-complete problem.

$\Rightarrow$ implies RAM and NDRAM are just as powerful

Many believe P!=NP, but there's no proof.

	Accomplishment	Implication
①	A poly. time algo. for a NP-C problem	✓P=NP ∘ P!=NP ∘ $\frac{?}{o}$
②	A poly-time algo(on avg) for a NP-C problem	∘ P=NP ∘ P!=NP ✓ $\frac{?}{o}$
③	Show that most efficient algorithm for clique req. exponential time	∘ P=NP ✓P!=NP ∘ $\frac{?}{o}$
④	Show v.c. has exponential no. of solns.	∘ P=NP ∘ P!=NP ✓ $\frac{?}{o}$
⑤	Reduce clique to shortest path	✓P=NP ∘ P!=NP ∘ $\frac{?}{o}$
⑥	Reduce shortest path to clique	∘ P=NP ∘ P!=NP ✓ $\frac{?}{o}$

Reasons we believe P != NP

→ Non-determinism (or) the power of guessing seems too powerful to be achieved on deterministic RAM in polynomial time.

→ There are problems that are believed to be much harder than NP complete problems and P=NP would imply that these problems are also solvable in polynomial time

Proving P != NP is harder than P=NP.

Should you try to solve P vs NP?

→ 41 proofs ⇒ P = NP
→ 46 proofs ⇒ P != NP
→ none of them hold up against rigorous scrutiny
→ people refuse to look at your proof.

What does this mean for NP complete problems, many of which have practical significance?

Stay tuned

QUIZ

- Some NP complete problems can't be transformed into SAT in polynomial time — FALSE
- Non deterministic RAM may give different results for same input — FALSE
- A problem with exponential solns can only be in P if P = NP — FALSE
- Every program that takes exponential time on RAM can be run in polynomial time on ND-RAM — FALSE

A Polynomial algorithm uses a Simulation time if-better x no. of times

x	Constant	Polynomial	Linear	Exponential
$O(1)$		✓		
$O(\log n)$		✓		
$O(n)$				✓
every 3rd instruction				✓

Given a boolean formula F, what do we need to verify if it's satisfiable in polynomial time

→ a satisfying assignment

→ access to calls to if-better