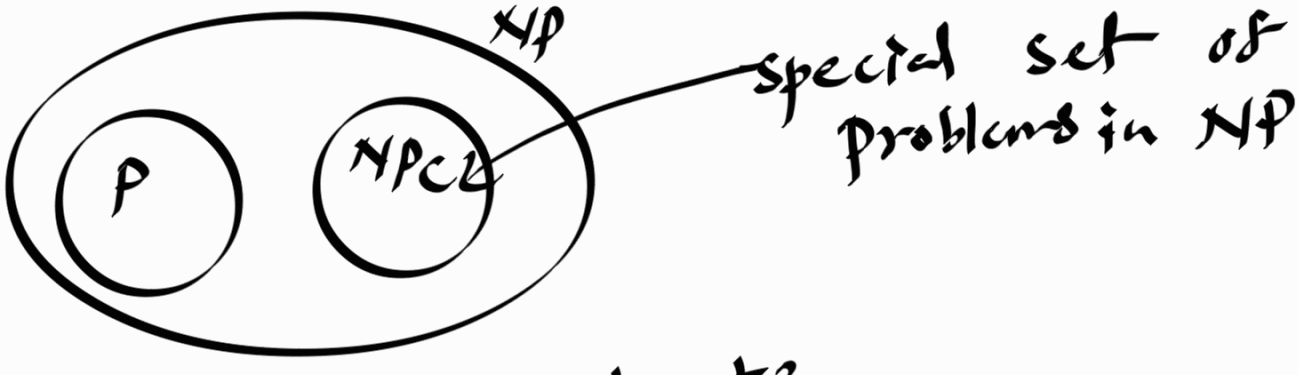
In 1970, Stephen Cook and Leonid Levin discovered that certain problems in NP whose individual complexities is related to that of the entire class.

If a poly-time solution exists for any of these problems, all problems in NP is polynomially solvable
These set of problems are called NP-complete (NPC) problems.




Why is this important?

- If you are trying to prove P=NP, then find a poly-time algo for one of the NPC problems.
- Proving a problem is NPC is a strong indication that the problem does not have a poly-time algorithm.

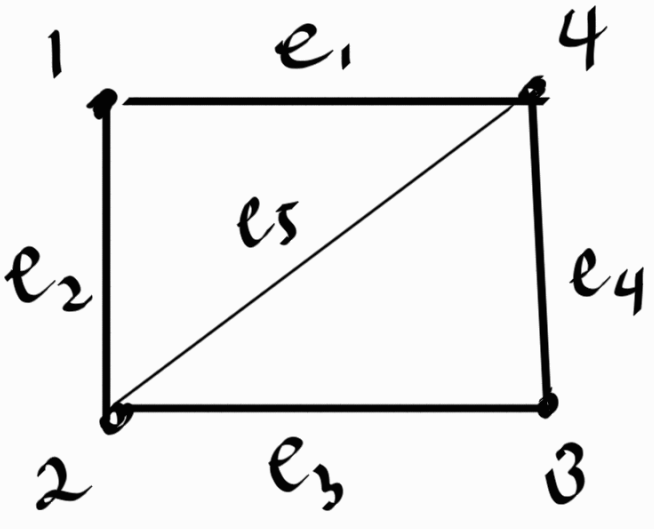
When we talk about Problems in NP, we only talk about decision problems.

- The decision version of a problem is equal to the corresponding optimization problem in terms of poly time solvability.

Example: Min-vertex cover

Given a graph $G = (V, E)$, find the smallest vertex cover

$VC \quad V' \subseteq V : \forall (u,v) \in E : (u \in V') \vee (v \in V')$




G

$\{2, 4\}$

$\{1, 2, 3\}$

I can rephrase the question.

Given $G=(V,E)$ and $k \in \mathbb{N}$, is there a VC of size at most $k$?

$\langle G, 1 \rangle$ no

$\langle G, 2 \rangle$ yes

$\langle G, 3 \rangle$ yes

$\langle G, n \rangle$ yes

$\langle G, n-1 \rangle$ yes

$\langle G, n-2 \rangle$ yes

$\vdots$

$\langle G, k^* \rangle$ yes

$\langle G, k^*-1 \rangle$ no

# Polynomial reductions

- These are map reductions that takes polynomial time.

**Definition 7.28**

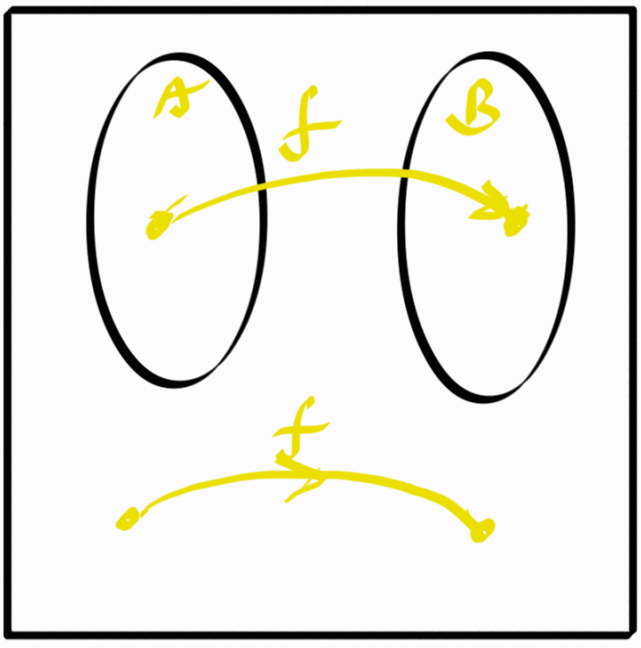
A function $f: \Sigma^* \longrightarrow \Sigma^*$ is a Polynomial time computable function if some polynomial time TM $M$ exists that halts with just $f(w)$ on its tape, when started on any input $w$.

**Definition 7.29**

A language $A$ is polynomial time mapping reducible, or polynomial time reducible, to language $B$, written $A \leq_p B$, if polynomial time computable function $f: \Sigma^* \longrightarrow \Sigma^*$ exists, where every $w$,

$$w \in A \iff f(w) \in B$$

The function $f$ is called the polynomial time reduction of $A$ to $B$.




**Theorem 7.31**

If $A \leq_p B$ and $B \in P$, then $A \in P$

we will do polynomial reduction examples later.

A language B is NP-complete if it satisfies the following two conditions.

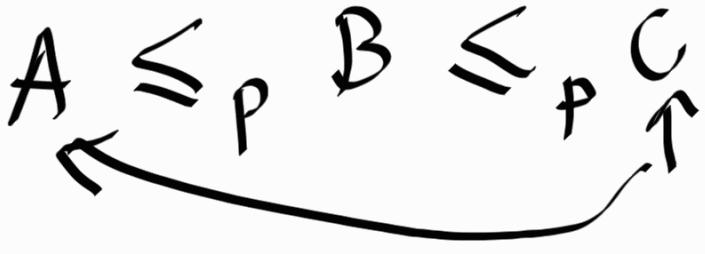
1. $B \in NP$
2. Every problem in NP is polynomial time reducible to B.

**Theorem 7.35**

If B is NP-complete and $B \in P$, then $P = NP$

**Theorem 7.36**

If B is NP-complete and $B \leq_p C$ and $C \in NP$, then C is NP-complete.



# The cook-levin theorem

## Theorem 7.37

SAT is NP-complete

$SAT = \{\langle \phi \rangle \mid \phi$ is satisfiable boolean formula$\}$

$\phi = x_1 \vee x_2 \wedge x_3 \wedge \bar{x}_4 \vee x_5 \cdots$

$\phi = x_1 \vee x_2 \wedge \bar{x}_3$

$T \vee T \wedge T$

$x_1 - T$

$x_2 - T$

$x_3 - F$

This proof is very complicated.

First, they prove that SAT is in NP.

Then, using a clever arguement, they show that any problem in NP can be reduced to a SAT instance in polynomial time.

You are given that SAT is NP-complete.

Every problem in NP is Polynomial time reducible to SAT

$$SAT \leq_p A \quad \text{and} \quad A \in NP$$

$$A \text{ is } NP\text{-complete}$$