NP algorithms -

- Make a guess (certificate) (non-deterministic step) : size of certificate is polynomial in n

- Verify in polynomial time

Any problem (decision problem) for which we can solve the "YES" instance using an NP algorithm belong to the class NP

However, the complement of the problem, i.e., we flip the YES/No may not have an NP algorithm
Note: Normal polynomial time algorithms belong to the class \( \text{NP} \): (No guess is required)

This is the class \( \text{P} \). For problems in class \( \text{P} \), the complement is also in \( \text{P} \)

\[
\begin{array}{c}
\text{instance} \\
\downarrow \\
\text{complement}
\end{array}
\]

Decision problem can be thought of as a set of strings (encodes the input instance) for which the answer is yes.

Example Hamiltonian cycle problem:

\[
\text{all graphs for the complete graph for the complement of } \chi G
\]
The problems for which the complement is in NP are called co-NP-class.

\[ \text{NP} \subseteq \text{co-NP} \]

We know \( P \subseteq \text{NP} \).

If \( P = \text{NP} \):

\[ \implies \text{NP} = \text{co-NP} \]

Either \( P = \text{NP} \), \( P \neq \text{NP} \).

Consider the "hardest" problems in NP and design a polynomial-time algorithm for it.

NP complete problems

Show some non-polynomial lower bound for some problem in NP.
NP complete problems (NPC)

- The problem is in NP
  (given & verify algorithm in polynomial time)

- All problems in NP can be reduced to this problem in polynomial time.

Sorting vs Selection

Selection is reducible to sorting.

Given an instance of a selection problem
(a set of elements and a rank k < n)
we can create an instance of the sorting
(a set of elements)

\[
I_1, \quad f \quad I_2
\]

So that by solving the sorting problem \( I_2 \),
we can find the soln for \( I_1 \),
The reduction function (mapping) $f$ should be efficiently computable (polynomial time), so that solving $I_1$ using the algorithm of $I_2$ is an efficient procedure.

The reduction relation is denoted by $I_1 \leq_{t(n)} I_2$ — time for reduction.

This reduction may not be symmetric.

Can we reduce sorting to selection?

**Note:** The definition of "reduction" does not allow us to call the algorithm for $I_2$ more than once (many-to-reduction).

- When we allowed repeated calls: Turing reduction.
Polynomial time reduction:

If the mapping function is computable in polynomial time, we say that problem \( \Pi_1 \) is polynomial-time reducible to \( \Pi_2 \).

Defn: If all problems \( \Pi \in \text{NP} \) are polynomial-time reducible to some problem \( \Pi' \in \text{NP} \), then \( \Pi' \) is called \( \text{NP} \) complete.

Claim: If \( \Pi' \) is \( \text{NPC} \) and \( \Pi' \) can be solved in polynomial time, then \( P = \text{NP} \).

Do we have \( \text{NPC} \) problems? How do we establish some problem in \( \text{NPC} \)?
Cook-Levin theorem

The boolean satisfiability problem is NP complete

Given a boolean formula

\[ x_1 \lor (x_2 \land x_3) \lor x_4. \]

where \( x_i \) can be assigned True/False values. Can the formula be set to true by some assignment

Brute force: try all possible assignments \( 2^n \)

The proof constructs an instance of the boolean satisfiability problem which is satisfiable iff the original instance had a "Yes" answer