**Name:- Rutuja Varpe**
**Roll No:-71**
**PRN:-12420166**
**Class:- CSE-AI&ML B**
**Batch:- B3**

---

# TUTORIAL NO :- 10

**Aim:**- Problem solving based on complexity classes, NP-completeness.

## Theory:-

### 1. Complexity Classes

In computational theory, complexity classes group problems based on the time or resources required to solve them.

2. Steps to Prove a Problem is NP-Complete

To prove a decision problem PP is NP-complete:

1. Show PP is in NP:
   → Verify a given solution in polynomial time.
2. Choose a known NP-complete problem QQ.
3. Reduce QQ to PP in polynomial time:
   Q≤pPQ≤pP

If both conditions hold, then PP is NP-complete.

3. Example Problems & Solutions

### Example 1: Subset Sum Problem

Problem:
Given a set of integers S={a1,a2,...,an}S={a1,a2,...,an} and an integer KK, does there exist a subset whose sum is exactly KK?

Step 1 – In NP:

- If a subset is given, we can check if the sum = KK in O(n)O(n).
-

  Step 2 – Reduce Known NP-Complete Problem:

- 3-SAT or Partition can be reduced to Subset Sum in polynomial time.
- Hence, Subset Sum is NP-Complete.

# Real-World Importance

- Cryptography: Relies on NP-hardness for security (e.g., RSA).
- AI & Optimization: Many AI search problems are NP-complete.
- Logistics & Scheduling: TSP and scheduling problems appear in supply chains, manufacturing, etc.

## Class P (Polynomial Time)

- Definition: The set of decision problems that can be solved in polynomial time by a deterministic Turing machine.
- Polynomial time means the runtime is bounded by O(nk)O(nk) for some constant kk.

**Examples:**

- Searching (Linear or Binary)
- Sorting algorithms (Merge Sort, Quick Sort)
- BFS and DFS in graphs

Interpretation:
Problems in P are considered "efficiently solvable".

## Class NP (Nondeterministic Polynomial Time)

- Definition: The set of decision problems where a proposed solution can be verified in polynomial time.
- It is not necessary that we can find the solution in polynomial time — only that we can check it quickly if someone gives it to us.

**Examples:**

- Boolean Satisfiability (SAT)
- Subset Sum
- Hamiltonian Cycle

Key Point:
Every problem in P is also in NP (because if you can solve it quickly, you can definitely verify it quickly).
So,

$P \subseteq NPP \subseteq NP$

## NP-hard

- Definition: A problem is NP-hard if every problem in NP can be reduced to it in polynomial time.
- These are at least as hard as NP problems. They may or may not be in NP themselves.

Examples:

- Halting Problem
- Optimization version of Travelling Salesman Problem
- Scheduling Problems

Key Point:
NP-hard problems are not necessarily decision problems.

## NP-complete

- Definition: A problem is NP-complete if:
    1. It is in NP.
    2. Every problem in NP can be polynomially reduced to it.

Examples:

- 3-SAT
- Vertex Cover
- Clique Problem
- Hamiltonian Cycle

Key Insight:
NP-complete problems are the “hardest problems in NP”. If any NP-complete problem is solved in polynomial time, then:

$P=NPP=NP$

This is one of the biggest unsolved questions in computer science.