Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

TUTORIAL SESSION 22:

Class P, class NP, NP-Hard, NP-Complete

Concept Building

The concepts of **P** and **NP** are fundamental in computational complexity theory, which studies the resources required to solve computational problems. Here's a breakdown of these classes along with examples for better understanding.

What is P?

- **Definition**: The class **P** consists of decision problems (problems with a yes/no answer) that can be solved by a deterministic Turing machine in polynomial time. This means that there exists an algorithm that can solve the problem in time that can be expressed as a polynomial function of the input size nn.
- **Examples**:
    1. **Sorting**: Sorting a list of numbers can be done in O(nlogn)O(nlogn) time using efficient algorithms like Merge Sort or Quick Sort.
    2. **Finding the Greatest Common Divisor (GCD)**: The Euclidean algorithm can compute the GCD of two numbers in O(log(min(a,b)))O(log(min(a,b))) time.
    3. **Graph Traversal**: Problems like finding the shortest path in a graph (using Dijkstra's or Bellman-Ford algorithms) can be solved in polynomial time.

What is NP?

- **Definition**: The class **NP** consists of decision problems for which a proposed solution can be verified in polynomial time by a deterministic Turing machine. In other words, if you are given a "certificate" (a proposed solution), you can check whether it is correct in polynomial time.
- **Examples**:
    1. **Subset Sum Problem**: Given a set of integers, is there a subset whose sum equals a given target? If someone provides a subset, you can easily sum its elements and check against the target.
    2. **Hamiltonian Cycle Problem**: Given a graph, does there exist a cycle that visits each vertex exactly once? If you are given a specific cycle, you can verify by checking if it visits all vertices and returns to the starting point.
    3. **Graph Coloring**: Given a graph, can you color the vertices with kk colors such that no two adjacent vertices share the same color? If a coloring is provided, you can check it in polynomial time.

Relationship Between P and NP

- **P vs NP**: The major question in computer science is whether P = NP. This means asking if every problem for which a solution can be verified quickly (in polynomial time) can also be solved quickly (in polynomial time).
- **NP-Complete**: A problem is NP-complete if it is in NP and as hard as any problem in NP. If any NP-complete problem can be solved in polynomial time, then every problem in NP can be solved in polynomial time (thus P = NP). Examples include:
  - **Traveling Salesman Problem** (decision version)
  - **Knapsack Problem**
  - **3-SAT Problem**

Pre-Tutorial (To be completed by student before attending tutorial session)

1. Define the complexity classes P, NP, and NP-complete? Draw Venn diagram to show their relationship.

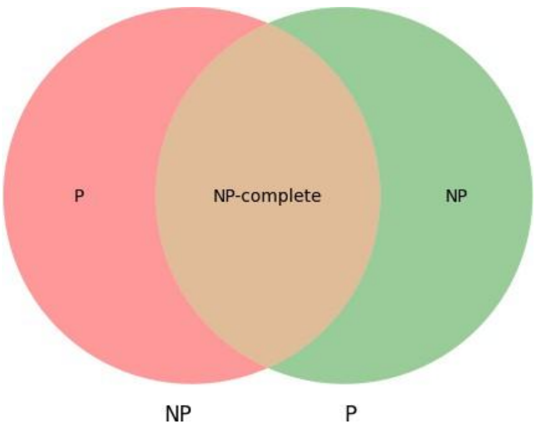
Solution:

Complexity Classes:

- **P**: Solvable in polynomial time (e.g., sorting).
- **NP**: Verifiable in polynomial time (e.g., SAT).
- **NP-complete**: Hardest problems in NP (e.g., SAT, TSP).

Venn Diagram:

- **P ⊆ NP**
- **NP-complete ⊆ NP** but not necessarily in **P**.



2. Discuss the difference between polynomial time and non-deterministic polynomial time.?

Solution:

- **Polynomial Time (P)**: Solvable in O(n^k) time.
- **Non-deterministic Polynomial Time (NP)**: Solutions verifiable in polynomial time, not always solvable.

3. Mention the difference between P and NP problems?

Solution:

- **P Problems**: Solvable in polynomial time.
- **NP Problems**: Solutions verifiable in polynomial time, not necessarily solvable in polynomial time.

IN-TUTORIAL (To be carried out in presence of faculty in classroom)

1. Explain the concept of polynomial time complexity and its significance in the context of the P versus NP problem. How does the ability to solve problems in polynomial time differ between P and NP?

Solution:

- **Polynomial Time Complexity**: Solving in O(n^k) time.
- **P Problems**: Solvable in polynomial time.
- **NP Problems**: Verifiable in polynomial time, not necessarily solvable in polynomial time.
- **P vs NP**: Asks if every problem verifiable in polynomial time (NP) can also be solved in polynomial time (P). If P = NP, all NP problems are solvable efficiently.

2. What does it mean if a problem is NP-hard?

Solution:

NP-hard: A problem as hard as the hardest NP problems. Solving it efficiently would solve all NP problems efficiently.

3. when a NP-Complete problem be solved in polynomial time?

Solution:

An NP-Complete problem can be solved in polynomial time if **P = NP**. If P equals NP, all NP-Complete problems can be solved efficiently in polynomial time.

4. Can we say Halting problem is NP-Complete?

Solution:

The **Halting Problem** is undecidable, not NP-Complete. NP-Complete problems are decidable and verifiable in polynomial time.

Post-Tutorial (To be carried out by student after attending tutorial session)

1. Define the characteristics of NP-Complete problems.

Solution:

Characteristics of NP-Complete Problems:

- **In NP**: Solutions can be verified in polynomial time.
- **NP-hard**: At least as hard as the hardest problems in NP.
- **Polynomial-time reduction**: Every NP problem can be reduced to it in polynomial time.
- **Decidable**: The problem has a solution, but finding it may take exponential time unless P = NP.

2. Describe the implications of proving P = NP. How would this breakthrough affect various fields, such as cryptography, optimization, and artificial intelligence?

Solution:

If **P = NP**:

- **Cryptography**: Current encryption methods would be **compromised**.
- **Optimization**: Complex problems could be solved efficiently.
- **AI**: Improved efficiency in decision-making and learning tasks.

3. Show that Kruskal's algorithm is in class P .?

Solution:

Kruskal's algorithm is in **P** because it runs in polynomial time:

- **Sorting edges**: $O(E \log E)$
- **Union-Find operations**: $O(\alpha(V))$

Thus, Kruskal's algorithm has a time complexity of $O(E \log E)$, which is polynomial, placing it in class **P**.

Viva - Questions

1. Explain the class NP and how it differs from class P.

Solution:

NP: Problems whose solutions can be verified in polynomial time.

P: Problems solvable in polynomial time.

2. What are NP-complete problems, and why are they significant?

Solution:

NP-Complete problems are the hardest in NP; solving one in polynomial time solves all.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: out of **50** Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: