NATIONAL INSTITUTE OF TECHNOLOGY

DELHI

ASSIGNMENT (TASK-5)

DESIGN & ANALYSIS OF

ALGORITHMS (CSB -252)

NAME-**YOGESH** BRANCH - **B.Tech 2nd Year**

ROLLNO- **181210063** SUBMITTED TO-**Dr. CHANDRESH**

KUMAR MAURYA

INTRODUCTION

N V/S NP PROBLEM

- The P versus NP problem is to determine whether every language accepted by some non-deterministic algorithm in polynomial time is also accepted by some deterministic algorithm in polynomial time.

- P (stands for Deterministic Polynomial Time), is the class of problems for which we have an efficient algorithm that is capable of producing a solution in polynomial time.

  Things like searching a list, sorting a list, multiplication of matrices, balancing trees, etc are such type of problems, for which we have an efficient algorithm.

All such problems can be solved by a deterministic Turing machine within a polynomial amount of computation time.

- When we know each and every statement of the algorithm, clearly and their working, we call it deterministic.
  For instance, multiplication of two numbers, we can verify the result and solve it easily.

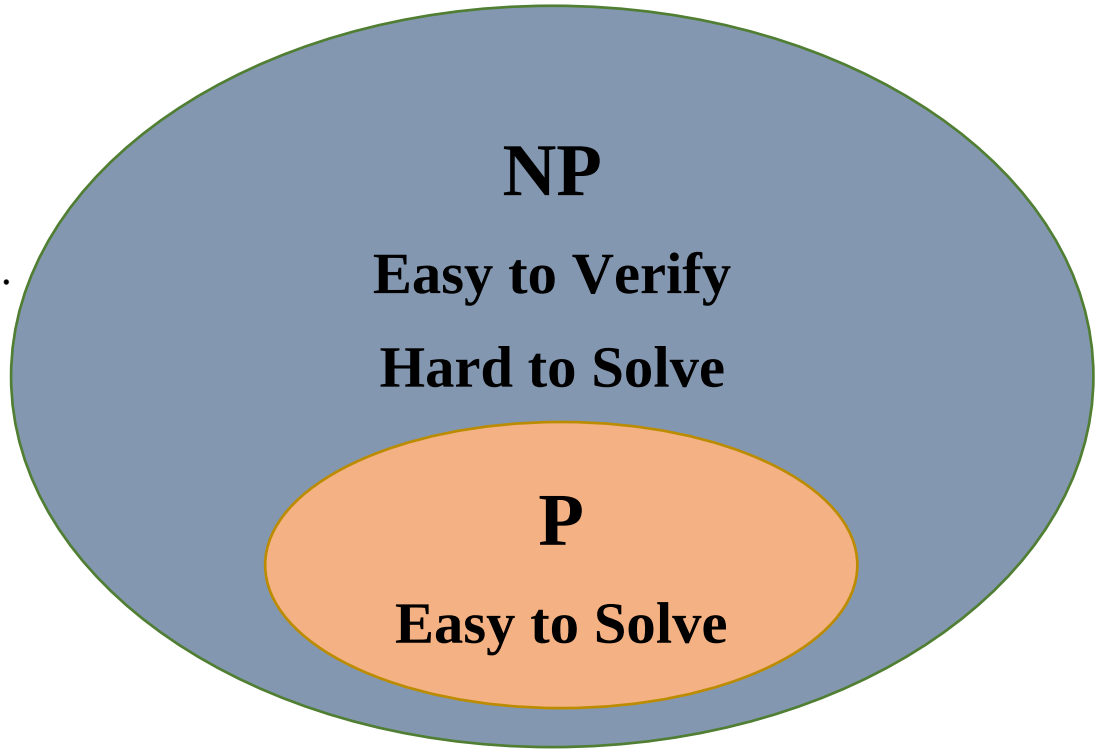
➢ **NP**

- NP (stands for Non-Deterministic Polynomial Time), is the set of decision problems that are solvable in polynomial time by a Non-deterministic Turing Machine.

- NP problems do not have a know algorithm that can produce a result in polynomial time. If we are given a solution to an NP problem, verifying that it is correct is easy and can be done in polynomial time or less. They are easy to check but hard to solve, they generally have an exponential time complexity such as $O(n^n)$.
  - ❖ **Example**:
- Solving a Sudoku Puzzle
- Unlocking a Smartphone
- Sending messages over apps, **etc.**
  - ❖ **Problems like**: -
    - 0/1 Knapsack Problem
    - Travelling Salesman Problem
    - Sum of subsets Problem
    - Graph Colouring Problem
    - Hamiltonian Cycle Problem
    - Finding Prime Factors of a large number take exponential time to solve.

❖ P is a subset of NP problems as shown below

.



IMPORTANT CASE IF P=NP?

This is why P = NP matters. We can't know for certain, but there is every reason to believe that the answer to that question runs right through NP-complete. First, any algorithm that returns a solution to an NP-complete problem in polynomial time can be modified to solve every single NP-complete problem in polynomial time, since they are all the same problem at their core.

Not just that, but part of the definition of an NP-complete problem is every problem in NP is reducible to every NP-complete problem, which means that an algorithm that solves NP-complete in polynomial time will also solve all NP problems in polynomial time as well; in other words, solving an NP-complete problem in polynomial time proves P = NP by default and effectively solves nearly all of the hardest computational problems in the real world literally overnight.

So this essentially makes the algorithm that solves an NP-complete problem in polynomial time an algorithm for everything. This is why P = NP, this weird, opaque sounding equation, holds so much promise if it can be proven true and the only real way to do that is to solve an NP-complete problem in polynomial time. That algorithm would become an algorithm that could unlock an entirely different world in an instant. There is a lot of research going on this, though whether or not P=NP remains to be seen.

DIFFERENCE BETWEEN P & NP

S No.	P Problems	NP Problems
1.	These can be solved in polynomial time by deterministic algorithms.	These can be solved in non-deterministic polynomial time.
2.	Such problems can be solved and verified in polynomial time.	NP problems solution cannot be obtained in polynomial time but if solution is given it can be verified in polynomial time.
3.	P problems are subset of NP problems.	NP problems are a superset of P problems.
4.	Eg. Searching, Sorting, Addition, Multiplication,etc.	Eg. Sudoku Puzzle ,Travelling Salesman Problem,etc.

NP HARD AND NP COMPLETE PROBLEMS:

- **NP HARD:**

If every problem in NP can be polynomial time reducible to a problem 'A' then 'A' is called a NP hard problem.

- If 'A' could be solved in polynomial time then every problem in NP can be solved in polynomial time so

P = NP

- **NP COMPLETE:**

If a problem lies in NP and also a NP hard problem then this problem is called NP complete problem.

NOTE:

1. If NP hard or NP complete problem is solved in polynomial time then NP = P.
2. If NP or NP complete problem is proven to be not solvable in polynomial time then P NP.
3. If 'A' is a NP hard problem and 'A' can be reduced in polynomial time in 'B' then be is also called a NP hard problem. If this 'B' is a NP problem then 'B' will be NP complete problem.

- **POLYNOMIAL TIME REDUCTION ALGORITHM**

To prove P = NP we have to prove that every problem which lies in NP can be solved in polynomial time

There are millions of NP problems we can't solve each problem to prove this, here comes reduction.

A problem 'A' is said to be polynomial time reducible to a problem 'B' if :-

1. Every instance 'a' of 'A' can be transformed to some instance 'b' of 'B' in polynomial time.
2. Answer of 'a' is 'YES' if and only if answer of 'b' is 'YES'.

So ,

If A is reduced to B in polynomial time then :-

- If B is easy then A is also easy.
- If B is in P then A is also in P.
- If this is proven that A can't be solved in polynomial time then B is also can't be solved in polynomial time.

If A is not in P then B is also not in P.

EXAMPLES OF NP COMPLETE PROBLEMS

1. Circuit satisfiability problem
2. Satisfiability(SAT) problem
3. 3-CNF satisfiability problem
4. Subset sum problem
5. Clique problem
6. Vertex cover problem
7. Hamiltonian cycle problem
8. Travelling salesman problem

➢ END OF ASSIGNMENT ……..