**NATIONAL INSTITUTE OF TECHNOLOGY, DELHI**

**ASSIGNMENT**

**DESIGN AND ANALYSIS OF ALGORITHMS**

**P and NP PROBLEMS**



Submitted By : ASHISH AGGARWAL

Roll No. : - 181210012

BRANCH : CSE 2nd year(group-I)

Search problems

*A problem is a search problem if there's an algorithmic way to verify the answer.*

For example finding a solution to a system of linear equations, finding a solution to a system of linear inequalities,finding a binary solution to the system of boolean equations,finding a non trivial factor of an integer.

That is, whenever we have a solution we can verify it by substituting the value/values.

**INTRODUCTION:**

In Computer Science, many problems are solved where the objective is to maximize or minimize some values, whereas in other problems we try to find whether there is a solution or not. Hence, the problems can be categorized as follows –

**OPTIMISATION PROBLEM:** Optimization problems are those for which the objective is to maximize or minimize some values. For example:

* Finding the minimum number of colors needed to color a given graph.
* Finding the shortest path between two vertices in a graph.

**DECISION PROBLEM:** There are many problems for which the answer is a Yes or a No. These types of problems are known as decision problems. For example:

* Whether a given graph can be colored by only 4-colors.
* Finding Hamiltonian cycle in a graph is not a decision problem, whereas checking a graph is Hamiltonian or not is a decision problem.

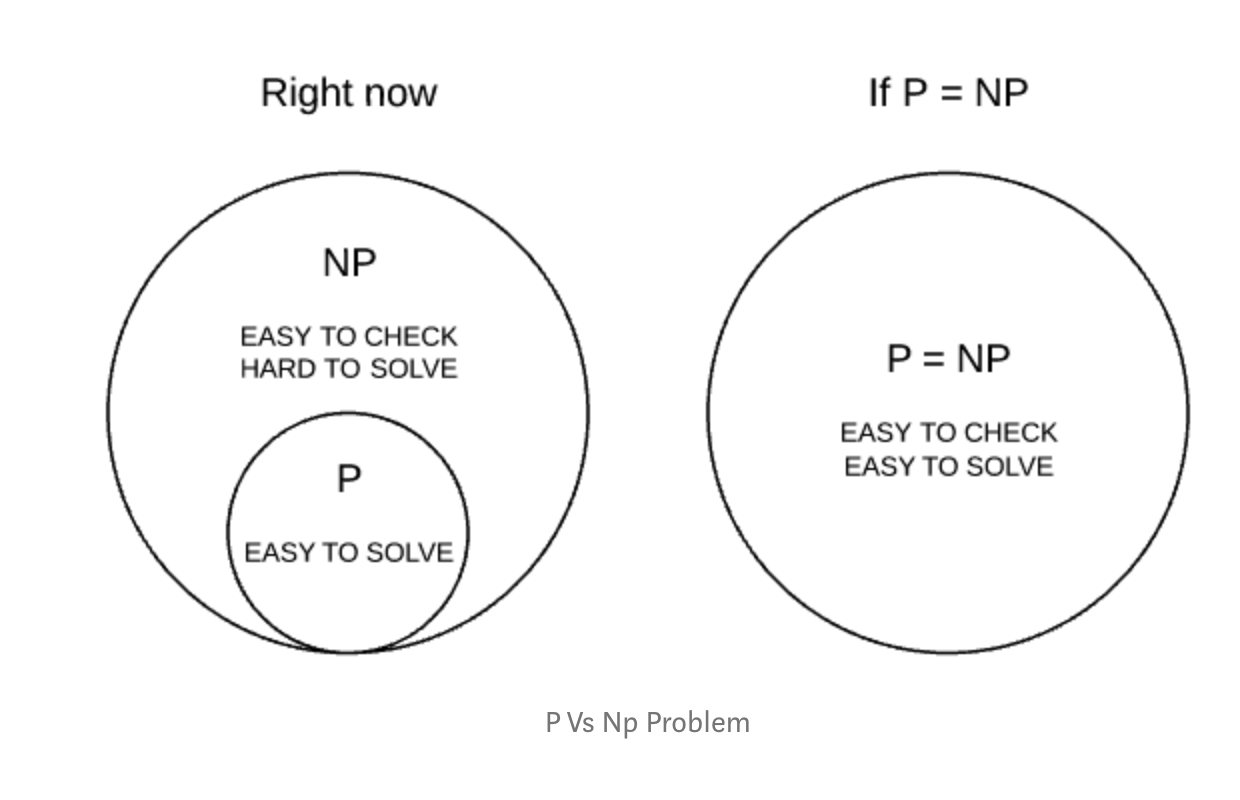
**Deterministic Polynomial Time (P):**

This is the class of problems which contains all the problems that can be solved by a deterministic Turing machine within a polynomial computation time. Examples for such problems are searching algorithms, sorting algorithms, etc. which can all be done in polynomial time.

**Non-Deterministic Polynomial Time (NP):**

This is the class of problems which can be solved by a Non-Deterministic Turing machine in a polynomial computation time. Examples for this class of problems are Sudoku problem, Knapsack problem, Travelling Salesperson Problem, etc.

* P is a subset of NP.
* Both classes of problems can be at least verified/checked in polynomial time.
* For P, computing the correct solution to the given problem can be done in polynomial time. Whereas for NP, there is no algorithm that can produce the solution in polynomial time.
* Given a problem which belongs to NP and a possible solution to the said problem, it can be easily verified if that is a correct solution or not within reasonable polynomial time.



**NP – Hard:**

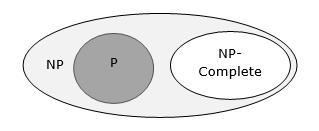
This is the class of problems which are at least as hard as the hardest problems in NP. Problems belonging to this class may or may not be part of NP, that is, the hardest problems of NP belong to the intersection of NP and NP-Hard. Problems in NP-Hard may not even be decision problems.

Example of a problem which is NP-Hard but not NP is the problem of identifying a chess move in any given board state that is the best possible move to make.

**NP – Complete:**

This is the class problems which contains the set of all the hardest problems in NP. Every problem in NP-Complete must belong to NP and NP-Hard, which is not true for NP-Hard. NP-Complete is the intersection of NP and NP-Hard.

Example of a problem which is NP-Complete is the clique graph problem, where, in an undirected graph, the largest complete sub-graph is to be found.



A problem L, is NP-Complete if:

1. L is NP-Hard
2. L belongs to NP

The diagram below (focus on the left-hand side) should make things clearer.

