**NATIONAL INSTITUTE OF TECHNOLOGY DELHI**

**ASSIGNMENT5**

**DESIGN AND ANALYSIS OF ALGORITHMS**

**P and NP PROBLEMS**

VIVEK KUMAR

181210060

CSE 2nd year(group-II)

**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):**

The class P consists of those problems that are solvable in polynomial time, i.e. these problems can be solved in time ***O(nk)*** in worst-case, where **k** is constant.

These problems are called **tractable**, while others are called **intractable or superpolynomial**.

Formally, an algorithm is polynomial time algorithm, if there exists a polynomial ***p(n)*** such that the algorithm can solve any instance of size **n** in a time ***O(p(n))***.

Problem requiring ***Ω(n50)*** time to solve are essentially intractable for large ***n***. Most known polynomial time algorithm run in time ***O(nk)*** for fairly low value of ***k***.

The advantages in considering the class of polynomial-time algorithms is that all reasonable **deterministic single processor model of computation** can be simulated on each other.

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

The class NP consists of those problems that are verifiable in polynomial time. NP is the class of decision problems for which it is easy to check the correctness of a claimed answer, with the aid of a little extra information. Hence, we aren’t asking for a way to find a solution, but only to verify that an alleged solution really is correct.

Every problem in this class can be solved in exponential time using exhaustive search.

**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.

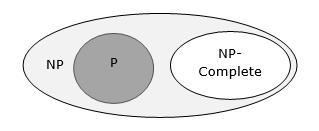
Example of a problem which is NP-Hard but not NP is the problem of identifying a chess move in any given board state that is the best possible move to make.

**NP – Complete:**

A language **B** is ***NP-complete*** if it satisfies two conditions:

* **B** is in NP
* Every **A** in NP is polynomial time reducible to **B**.

If a language satisfies the second property, but not necessarily the first one, the language **B** is known as **NP-Hard**. Informally, a search problem **B** is **NP-Hard** if there exists some **NP-Complete** problem **A** that Turing reduces to **B**.



**P vs NP Problem**

“Can every problem whose solution can be quickly verified by a computer also be quickly solved by a computer?”

Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.

All problems in P can be solved with polynomial time algorithms, whereas all problems in *NP - P* are intractable.

It is not known whether ***P = NP***. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If ***P ≠ NP***, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class **P** if it’s easy to find a solution for the problem. The problem belongs to **NP**, if it’s easy to check a solution that may have been very tedious to find.

