# National Institute of Technology – Delhi

# Name: Venkata Sai Akhil Penugonda Roll: 181210057

# Branch: C.S.E – 2nd year Subject: Design and Analysis of Algorithms

# ---------------------------------------------------------------

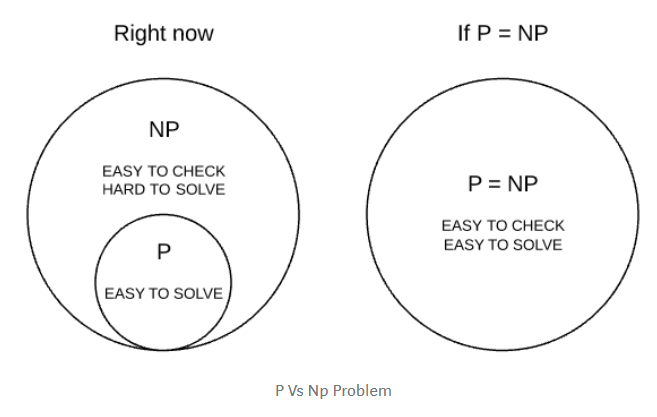
**Deterministic Polynomial Time (P):**

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.



By the term “deterministic”, we mean that we know the working of every step and statement of the algorithm used clearly.

**NP – Hard:**

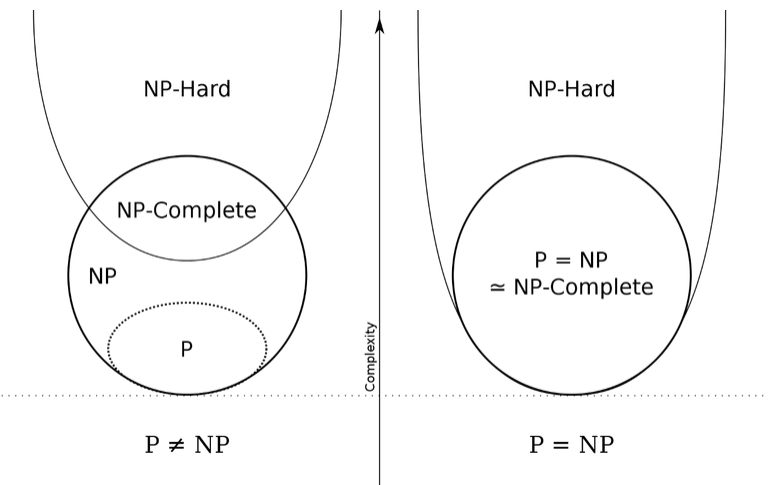
This is the class of problems which are at least as hard as the hardest problems in NP. Problems belonging to this class may or may not be part of NP, that is, the hardest problems of NP belong to the intersection of NP and NP-Hard. Problems in NP-Hard may not even be decision problems.

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.



**P vs NP Problem**

The P vs NP problem is a major unsolved problem in the field of computer science. It states that

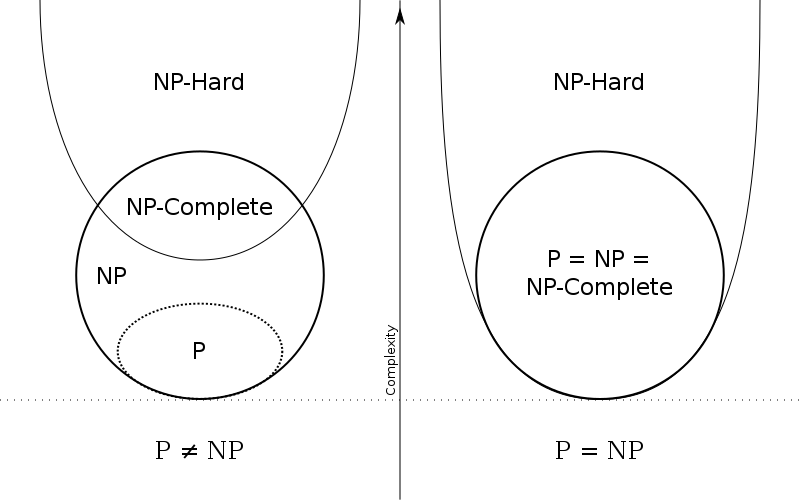
“If the solution to a problem is easy to check for correctness, must the problem be easy to solve?”.

This can be stated in other words as “If a problem can be verified by a computer quickly, can it also be solved by a computer in a reasonable time?”. This refers to proving P=NP for all problems.

Firstly, any algorithm that returns a solution to an NP-Complete problem in polynomial time can be modified to solve all NP-Complete problems in polynomial time, since they are similar in the basic functioning.

Now, since NP-Complete problems are the hardest NP problems, the same algorithm can be modified to solve all NP problems in polynomial time, which would give us P=NP.

So, essentially, making an algorithm that takes and solves an NP-Complete problem in polynomial time is the key to proving P=NP.



**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 polynomial time by non-deterministic algorithms. |
| 2. | Such problems can be solved and verified in polynomial time | These problems cannot be solved in polynomial time, but a given possible solution can be verified in polynomial time. |
| 3. | P is a subset of NP | NP is the superset of P |
| Ex. | Searching, Sorting, Addition, Multiplication, etc. | Sudoku, Travelling Salesperson, Knapsack Problem |