DESIGN AND ANALYSIS OF ALGORITHM (CSB 252)

ASSIGNMENT - 5

INTRODUCTION TO P VS. NP PROBLEMS

NATIONAL INSTITUTE OF TECHNOLOGY, DELHI

SUBMITTED BY : SUMITRA SIVAKUMAR

ROLL NUMBER : 181210053

BRANCH : COMPUTER SCIENCE AND ENGINERING (CSE)

SUBMITTED TO : CHANDRESH KUMAR MAURYA, PH.D.
ASSISTANT PROFESSOR

INTRODUCTION

COMPLEXITY CLASSES :

A complexity class contains a set of problems that are all solvable in polynomial time and with an exponential space with respect to input size.

- They contain a set of problems that tell us about **how much time and space** they require to **solve problems and verify solutions**.
- Few common Complexity Classes are :
    1. P Class
    2. NP Class
        - NP-Hard Class
        - NP-Complete Class

FEW IMPORTANT TERMS :

DECISION PROBLEMS :

- Decision Problems are those problems whose return values are either **YES or NO.**

POLYNOMIAL TIME :

- An algorithm is said to be solvable in polynomial time if the number of steps required to complete the algorithm for a given input is **$O(n^k)$** .
- Polynomial-time algorithms are said to be "fast."

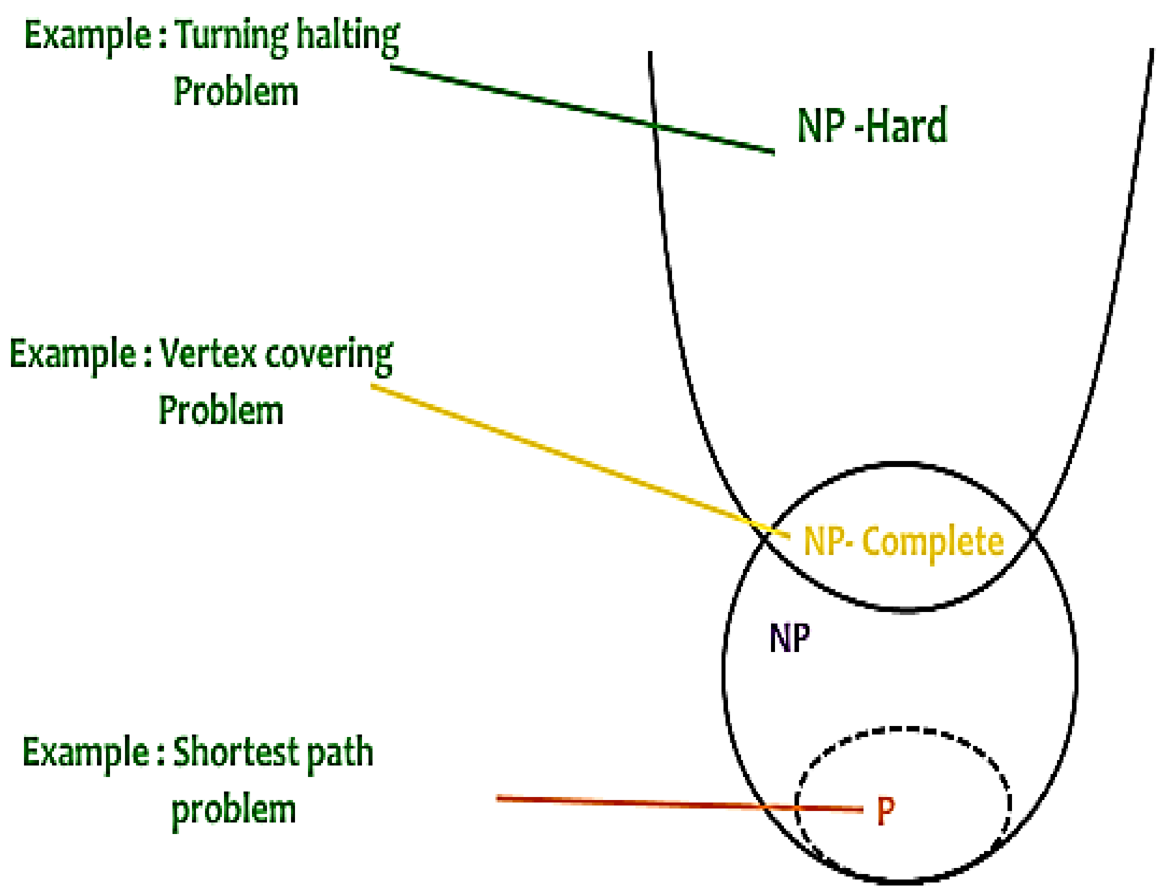
NON-DETERMINISTIC TURING MACHINE :

- It is a branching machine that has a set of rules that prescribes **more than one action for a given situation**.

REDUCTION :

- Reduction is an **algorithm for transforming** one problem into another problem
- Intuitively, problem A is **reducible** to problem B if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.
- It is a **reflexive** and **transitive relation**.
- Let there be two problems X and Y. Problem X is reducible to problem Y 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'.

  Therefore, we can conclude that if A is reduced to B in polynomial time then :
    - If B is hard then A is also hard.
    - If B is in P then A is also in P and vice-versa.
    - If this is proven that A can't be solved in polynomial time then B is also can't be solved in polynomial time.



The figure above represents the following Complexity Classes : P Class, NP Class, NP-Hard Problems and NP-Complete Problems

P CLASS

P Class contains a set of decision problems that are solvable in polynomial time.

- These problems can be solved in time ***$O(n^k)$*** in worst-case, where **k** is constant.
- The problem belongs to class **P** if it's easy to find a solution for the problem.
- These problems are called as **tractable**.
- Can be solved on a deterministic sequential machine like, Turing Machine.
- Few examples of P Class problems are :
  - To check if a given string is a palindrome or not
  - To find the maximum element in an array
  - Simple Mathematical problems –Addition,Subtraction,Multiplication, etc
  - Shortest Path problems, etc

NP CLASS

NP Class contains a set of problems that can not be obtained in polynomial time but can be verifiable in polynomial time.

- The problem belongs to **NP**, if it's easy to check a solution that may have been very tedious to find.
- Also called as **intractable** or **superpolynomial**.
- Can be solved by a non-deterministic Turing Machine.
- If we are able to solve a problem in polynomial time, we will surely be able to verify in polynomial time**, so every P problem will also be a NP problem.** Therefore, we can say that

P ⊆ NP i.e., P is a subset of NP

- Few examples of NP Class problems are :
    - Travelling Salesman Problem
    - Sudoku Puzzle
    - To check if a number is composite or not
    - The Vertex Cover Problem

P VS. NP – DIFFERENCE

P CLASS	NP CLASS
❖ Can be obtained in polynomial time	❖ Can be verified in polynomial time
❖ Easy to find the solution	❖ Easy to check the solution
❖ Tractable	❖ Intractable or Superpolynomial
❖ Can be solved using Determiistic Turing Machine (DTMs)	❖ Can be solved using Non-Deterministic Turning Machines (NDTMs)
❖ P is subest of NP	❖ NP is a superset of P

NP- HARD PROBLEMS

A problem X is NP-hard if for every problem Y in NP, there is a polynomial-time reduction from Y to X.

- More generally, if a problem X is reducible to problem Y if an algorithm for solving problem Y could also be used to solve problem X.
- An NP-hard problem **is at least as hard** as every problem in NP, and it might be much harder.
- NP−hard problems **do not** have to be in NP.
- For example :
  - The Halting Problem
  - The Vertex Cover Problem
  - The Travelling Salesman Problem

NP- COMPLETE PROBLEMS

NP_Complete problems are those problems which are both in NP and NP-hard.

- A problem **Y** is **NP-complete** if it satisfies two conditions :
  - **Y** is in NP
  - Every **X** in NP is polynomial time reducible to **Y**.
- NP-Complete problems are among the hardest problems in NP set of problems.
- Any problem in NP can be reduced to a NP-Complete problem in polynomial time.
- Few examples of NP-Complete problems are :
  - To Determine whether a graph has a Hamiltonian Cycle/Path
  - To Determine whether a graph has a Euler Cycle/Path
  - The Travelling Salesman problem
  - Graph Colouring problem