Ch: 9 Introduction to NP Completeness

⊛ The class P and NP

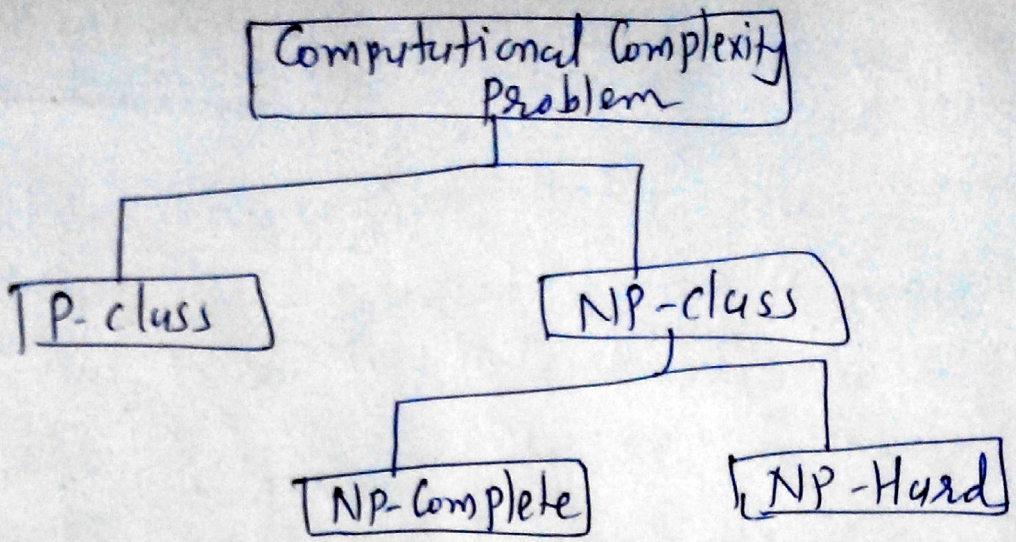
↳ These are two groups in which a Problem can be classified.

① **P-class** : "The Problems that can be Solved in Polynomial time."
("P" stands for Polynomial)

Ex:- - Searching of an element from the list.
- Sorting of elements.
- All Pair Shortest Path
- Kruskal's Algo.

② **NP_class** "The Problems that can be Solved in non-deterministic Polynomial time."
(NP Stand for non deterministic Polynomial time.)

Ex:- - Travelling Sulesman Problem → NP complete → Path Possible / Not Possible
- Graph Colouring Problem
- Hamiltonial Circuit Problems
- Knapsack problem.



↳ A Problem D is Called NP-Complete if (a) It belongs to class NP.
(b) Every problem in NP can also be solved in polynomial time.

↳ If an NP-hard problem can be solved in polynomial time then all NP-Complete problems can also be solved in Polynomial time.

↳ All NP-Complete problems are NP hard but all NP-hard problems cannot be NP-Complete.

↳ NP-class problem are the dicision problem (answer is yes or no) that can be solved by non-deterministic Polynomial algosithm.

↳ Dicision problem → NP Complete

↳ optimization problem → NP hard
(optimal cost - min or max)

⇒ Deterministic Algosithm :-

An algosithm is called deterministic when for given input the same output gets generated for a function.

- That means in deterministic algosithms the next state to be followed is fixed.
- The polynomial time algosithm (P-class) are deterministic.
- (Every operation is uniquely defined)

⇒ Non-deterministic algosithm :-

"An algosithm is said to be non-deterministic when there are more than one paths that the algosithm can follow."

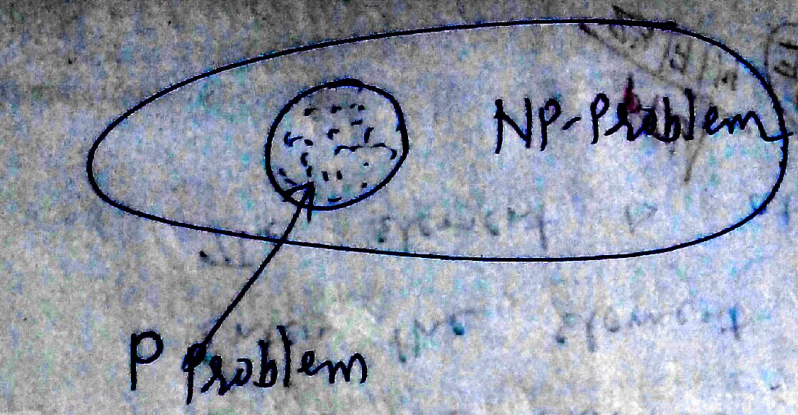
- Dye to this one cannot determine which path is to be followed after a particular stage.
- All the NP class problems are basically non deterministic.

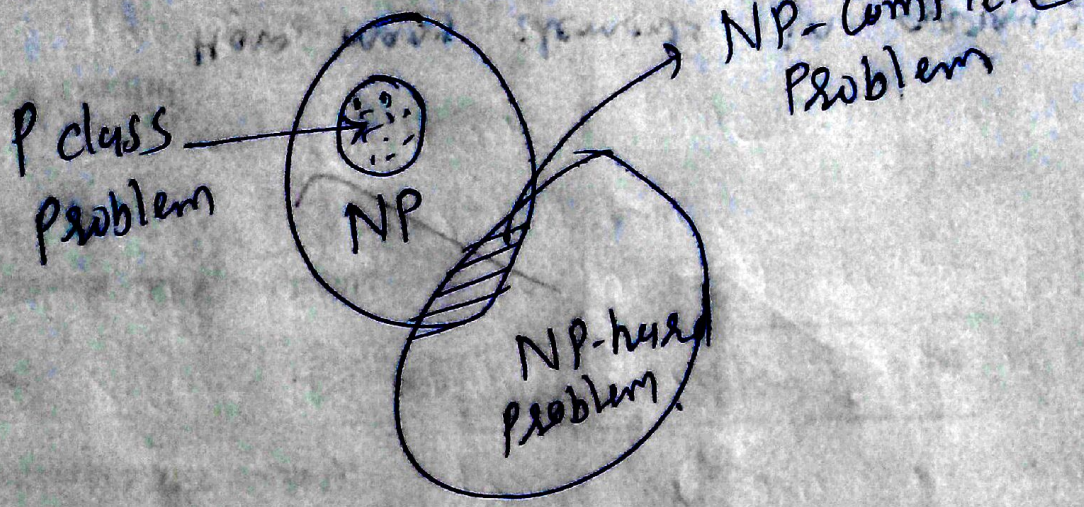
(Every operation may not have unique result).

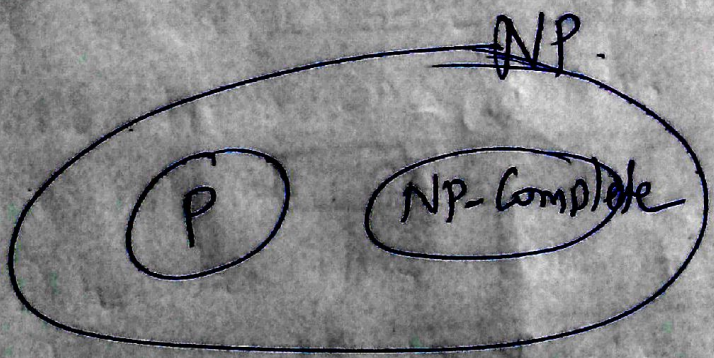
⇝ We say that

$P \subseteq NP$

but $P = NP$ is the most famous outstanding Problem in the Computer Science.







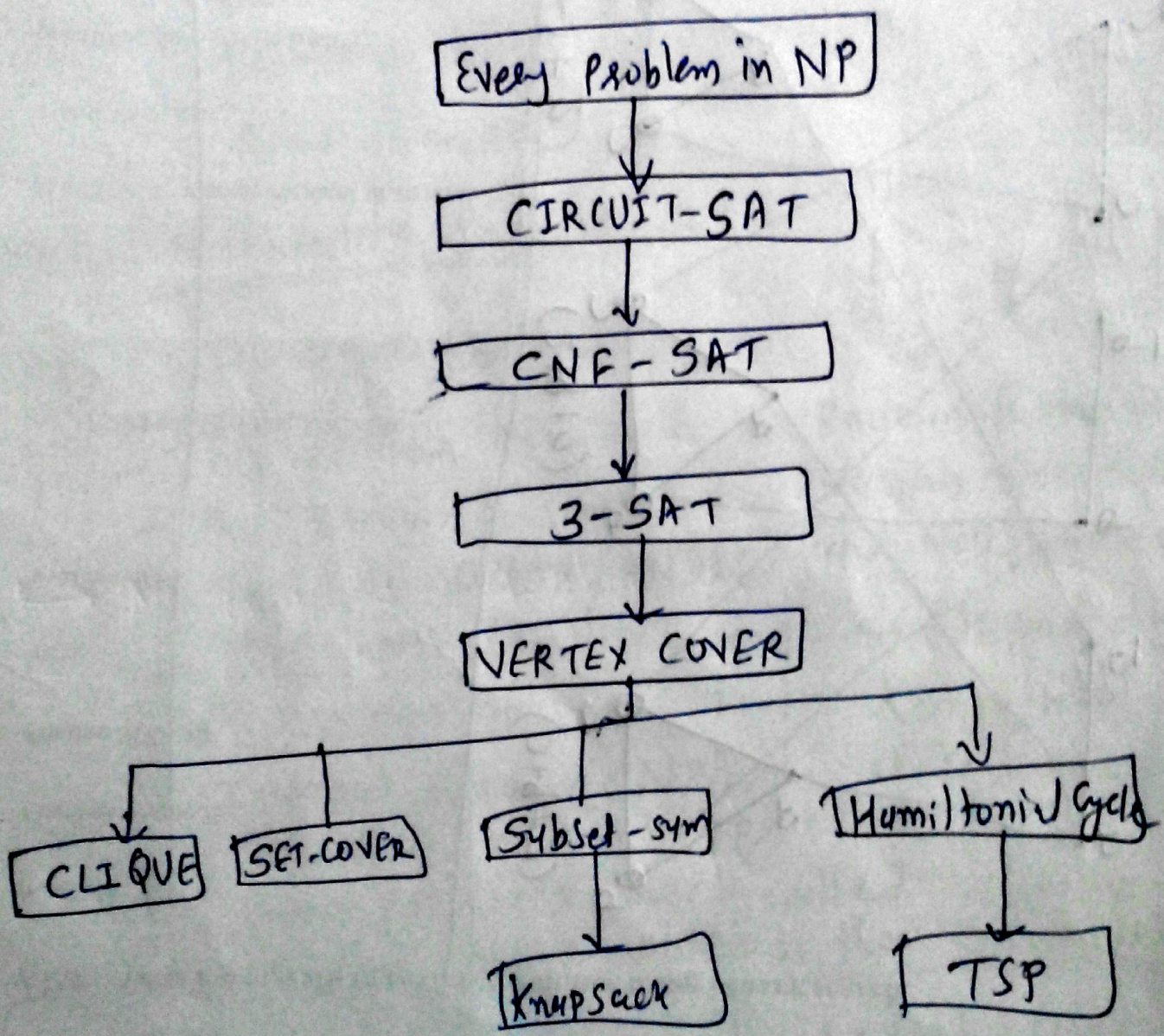
→ NP Complete problem is the Intersection of the NP and NP-hard classes.

⊛ Polynomial Reduction

↳ To Prove Whether Particular Problem is NP Complete or not we use polynomial time ~~reducti~~ reducibility.

→ That means

$A \xrightarrow{Poly} B$ & $B \xrightarrow{Poly} C$ then $A \xrightarrow{Poly} C$.



⊛ NP - Completeness Problem

↳ In this section we will discuss various Problems that can be proved to be NP-Complete.

- This Proofs of NP Completeness is Based on reduction technique.

↳ That means these are some Problems which are already Proved as NP Complete Problems. Using these Problems we can Prove the desired Problems as NP Complete.

⊛ ⟹ Satisfiability Problem

literals.

$(a + \bar{b} + c)$ ← clauses

① CNF - SAT Problem :-

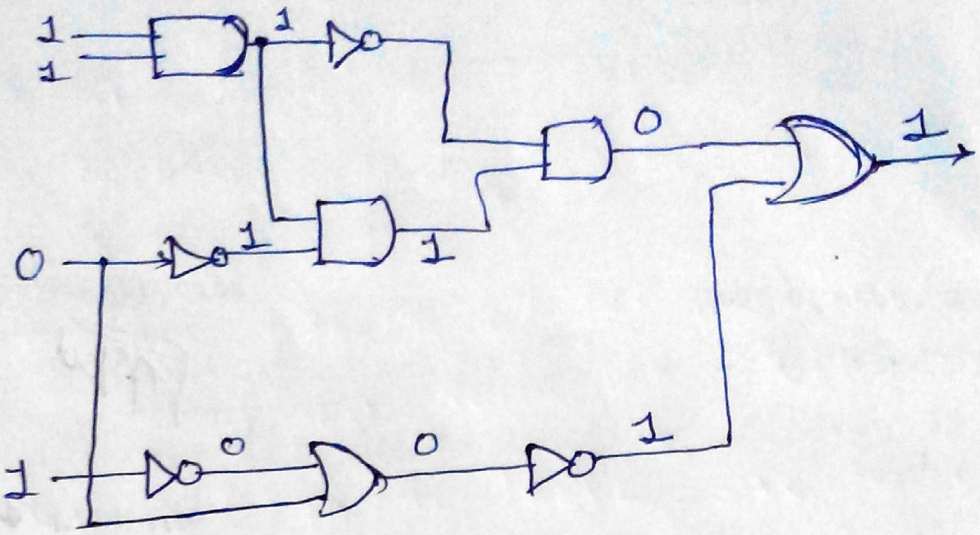
- This Problem is based on Boolean formula.
- The Boolean formula has various Boolean Opⁿ, such as OR(+), AND(·) and NOT. and also used ⟶ (implies) and ⟺ (if and only if)
- A Boolean formula is in conjuctive normal form (CNF) if it is formed as Collection of subexpressions. These subexpressions are called clauses.

Ex: prove ⟹ (All clauses evaluate = 1 then CNF-SAT is NP complete)

$(\bar{a} + b + d + \bar{g})(c + \bar{e})(b + d + \bar{f} + h)(a + c + e + \bar{h})$

↳ CNF SAT is a Problem which takes Boolean formula in CNF form and checks whether any assignment is these to Boolean values so that formula evaluates to 1.

(2) (*) CIRCUIT-SAT



(3) (*) 3-SAT - Problem

↳ A 3SAT problem is a problem which takes a Boolean formula S in CNF form with each clause having exactly three literals, and check whether S is satisfied or not. and check whether S is satisfied or not.

Ex: $(\bar{a}+b+\bar{g})(c+\bar{e}+f)(\bar{b}+d+\bar{f})(a+e+\bar{h})$

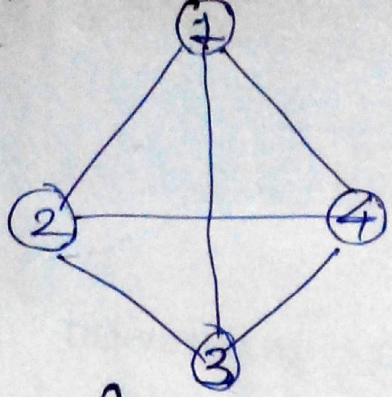
Proof ↳ S is Boolean formula having 3 literals in each clause and S is evaluated as 1 then S is satisfied. And we can prove that 3SAT is NP-Complete.

(4) (*) Clique Problem :-

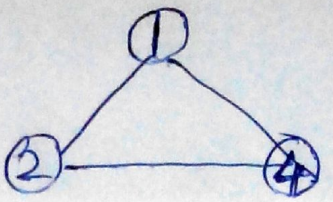
Clique is nothing but a maximum Complete subgraph of a graph G.

- The Graph G is a set of (V, E) where V is a set of vertices and E is a set of edges, the size of clique is numbers of vertices (node) present in it.

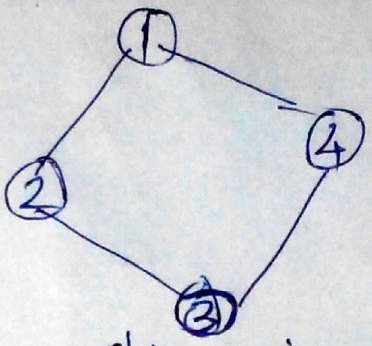
Ex



Graph G



clique size = 3



clique size = 4.

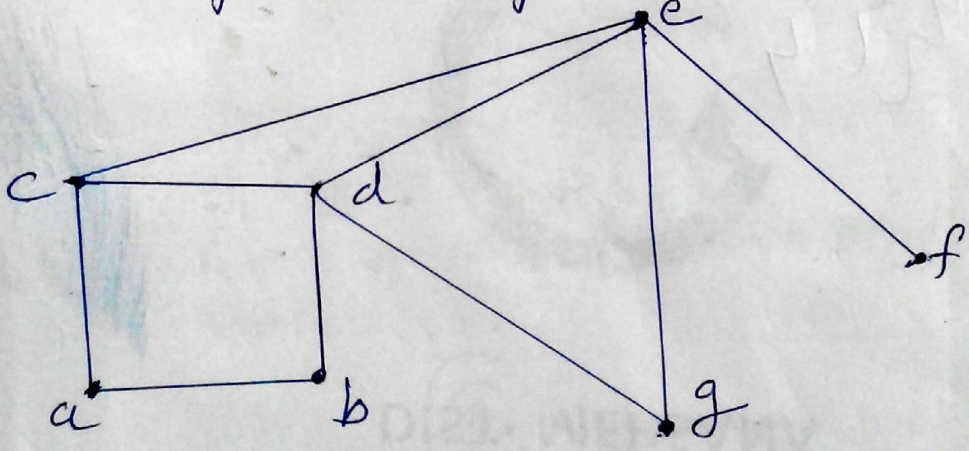
→ Max clique Problem :- It is optimization problem in which the largest size clique is to be obtained.

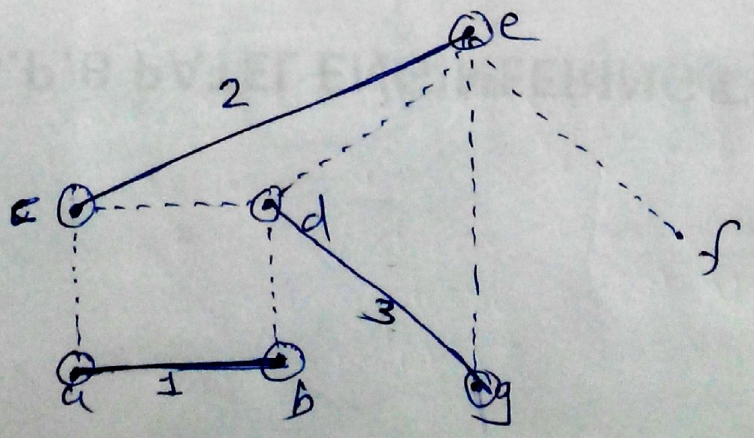
(*) Vertex Cover

The Vertex Cover Problem is to find Vertex Cover of minimum size in a given graph. The word Vertex Cover means each vertex covers its incident edges.

- Thus by Vertex Cover we expect to choose the set of vertices which covers all the edges in a graph.

Ex.





⇒ Obtain Vertex Covers as {a, b, c, d, e, g}

(4) Approximation Algorithms.

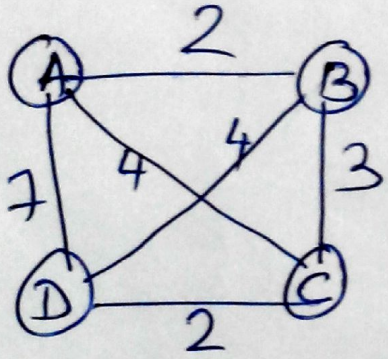
→ find approximate solution

→ "find good solution fast."

→ Approximation Algorithms for TSP.

Approximation algorithm for TSP ⇒
- → Nearest neighbor algorithm
- → Twice around the tree algorithm.

(1) Nearest Neighbour algo.



Nearest neighbour = A − B − C − D − A

= 2 + 3 + 2 + 7

= 14.

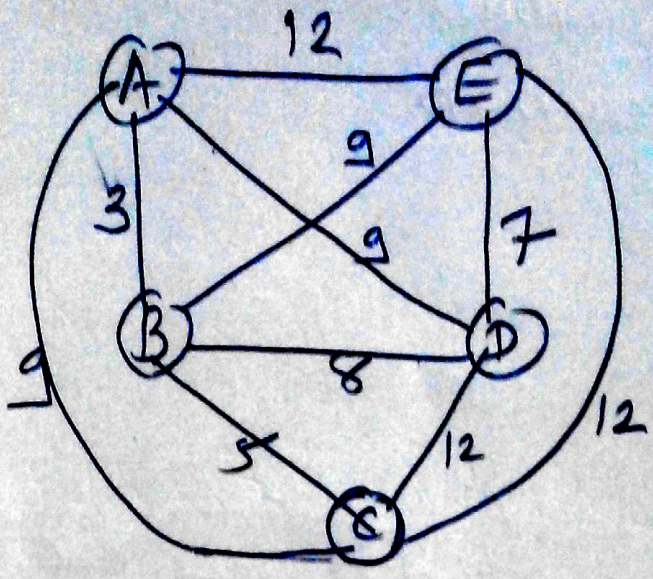
Optimal = 2 + 4 + 2 + 4

= 12

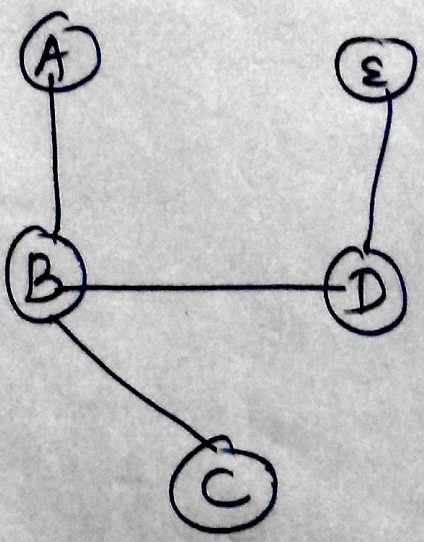
→ Accuracy Ratio = $r(S_1) = \frac{14}{12}$

= 1.16

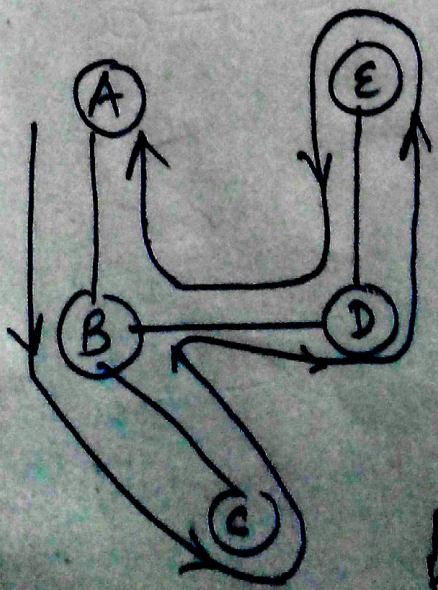
(2) twice Around the tree



(1) Obtain Minimum spanning tree



(2) Start from A and DFS walk around the tree



A-B-C-B-D-E-D-B-A.

Eliminate Duplicate

A-B-C-D-E-A Hamiltonian cycle