PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

UNIT-VI

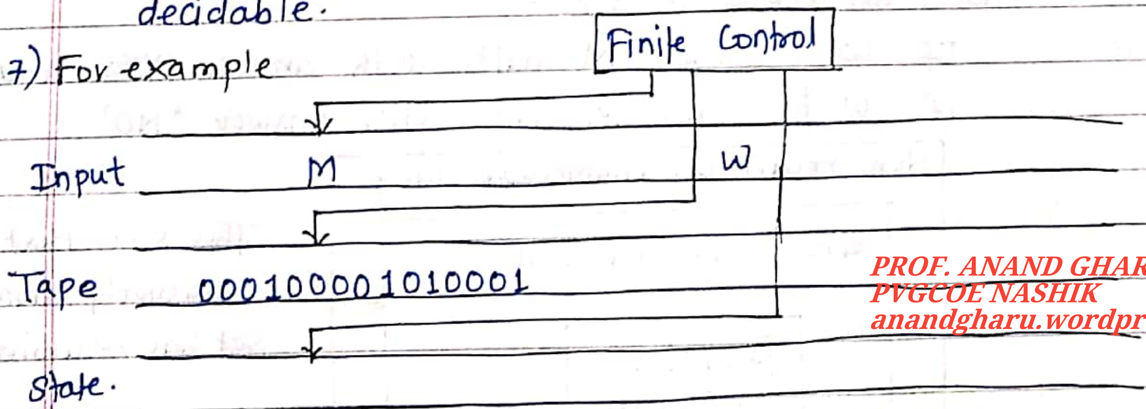
Name :- Prof. Gharu A.N
Institute : PVGCOE, Nashik
Subject :- TOC
MB NO : 8087777708
Date :- 25/11/2019

Undecidability & Intractable Problem.

* Recursively Enumerable and Recursive Language :-

1) A language is called recursively enumerable if and only if the language is accepted by some turing Machine M.
2) In other word, L is said to be L = L(M) for Some TM M. There are certain language is not recursively enumerable.
3) Consider language consisting a pair (M, w) where, 1) M, a Turing Machine with i/p set (0, 1)
   2) w, a binary string consisting of 0's and 1's.
   3) M accept w.
4) Following statements are equivalent
   - The lang. L is turing acceptable
   - The lang L is recursively enumerable.
   - The lang L is recursive.
   - The lang L is Turing Decidable.

5) Every turing decidable lang is Turing acceptable.
6) Every turing acceptable lang need not be turing decidable.
7) For example



8) - The universal lang can be represented by (M, w) where M is TM accepted lang, & w is binary string $(0+1)^*$.

9) -The universal Turing m/c ~~is~~ U accepts the TM M.
-The transition of M are stored initially on first ~~step~~.
tape along with string w.

10) on the 2nd ~~tape~~, the simulated tape of M is placed.
Here the tape symbol X_i of M will be represented by 0'
and tape symbol are seperated by single 1's.

* Turing acceptable language :-

→ **NOTE** :- you can write same answer as above question
Just add definition of turing acceptable.

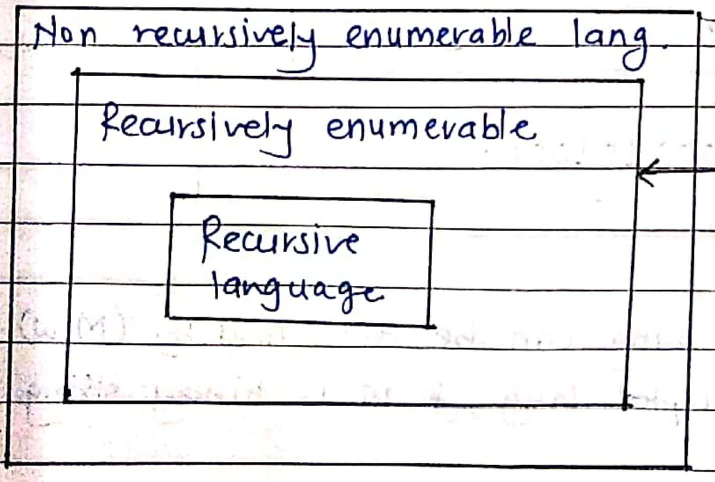
" A language $L \subseteq \Sigma^*$ is said to be turing being acceptable lang. if there is turing Machine M. which halts on every $w \in L$ with an answer 'YES' However, if $w \notin L$, then M may not halt.

* Turing decidable / Solvable :-

→ NOTE :- Same answer as above.
Just add definition of it.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

" The Language $L \subseteq \Sigma^*$ is said to be turing being decidable if there is turing machine M which always. halts on every $w \in \Sigma^*$.
If $w \in L$ then M halts with answer 'YES' and
if $w \notin L$ then M halts with answer 'NO'



This shows that every recursively enumerable set has recursive Subset.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* Show that for two recursive languages L_1 and L_2 each of the following is recursive.

i) $L_1 \cup L_2$ ii) $L_1 \cap L_2$ iii) L_1'

→

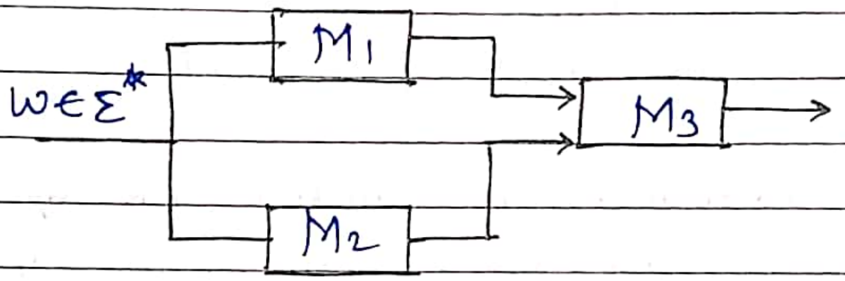
1) $L_1 \cup L_2$ is recursive :-

→ Let the TM M_1 decides L_1 and M_2 decides L_2.

- If a word $w \in L_1$ then M_1 returns 'Yes' else it returns 'No'

Similarly if word $w \in L_2$ then M_2 return 'Yes' else 'No'

- Let us construct TM M_3 as shown in figure



A Turing Machine for $L_1 \cup L_2$

- output of machine M_1 is written on the tape of M_3
- output of m/c M_2 is written on the tape of M_3
- The m/c M_3 returns 'Yes' as o/p, if atleast one of the o/p of M_1 or M_2 is 'Yes'.

- It should be clear that M_3 decides $L_1 \cup L_2$.
As both L_1 & L_2 are turing decidable. after finite time both M_1 & M_2 will halt with answer 'Yes' or 'No'
- The machine M_3 gets activated after M_1 or M_2 is halted. (stopped)
- The machine M_3 halts with answer 'Yes' if $w \in L_1$ or $w \in L_2$ else M_3 halts with o/p 'No'

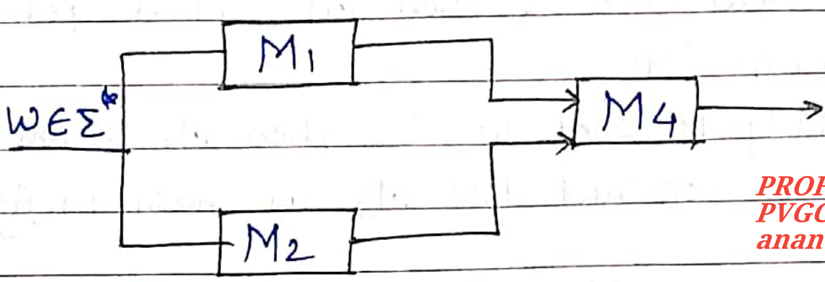
Thus, $L_1 \cup L_2$ is turing decidable or Recursive.

ii) $L_1 \cap L_2$ is recursive :-

→ - Let the turing machine M_1 decides L_1 and M_2 decides L_2. If word $w \in L$ then M_1 return 'Yes' else it returns 'No'

Similarly if $w \in L_2$ then M_2 returns 'Yes' else 'No'

- Let's consider TM M_4 as shown in fig.



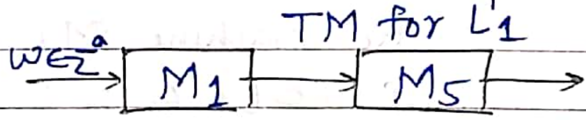
A Turing Machine for $L_1 \cap L_2$

- o/p of machine M_1 is written on the tape of M_4.
- o/p of m/c M_2 is written on the tape of M_4.
- The m/c M_4 returns 'Yes' as o/p, If both M_1 and M_2 are "Yes" otherwise M_4 returns 'No'
- It should be clear that M_4 decides $L_1 \cap L_2$. As both L_1 & L_2 are turing decidable, after finite time both M_1 & M_2 ~~are~~ will halt with answer 'Yes' / 'No'.
- The m/c M_4 is activated after M_1 & M_2 are halted. The m/c M_4 halts with answer 'Yes' if $w \in L_1$ & $w \in L_2$ else M_4 halts with answer 'No'.

iii) L_1' is recursive :-

→ Let the TM ~~decides~~ M_1 decides L_1

- Let's construct TM M_5



- o/p of machine m_1 is written on the tape of M_5.
- The m/c M_5 returns "Yes" as o/p if the o/p of M_1 is "No" otherwise. M_5 returns "No".
- It should be clear that M_5 decides L_1'. As L_1 is turing decidable after finite time M_1 will halt with answer Yes / No.
- The M/C M_5 is activated after M_1 halts.

* Undecidability :- (Unsolvable)

→ " A Problem is said to be decidable if there exist a Turing Machine that gives correct answer for every statement. in the domain of problem
otherwise, the class of problem is said to be un-decidable"

- These two statements are equivalent.
  1) A class of problem is Un-decidable
  2) A class of problem is Un-solvable

- A language can be proved to be un-decidable through a method of reduction.
- As we have already seen that halting Problem is Undecidable
- Some standard Un-decidable problem.
  1) Halting problem of Turing M/C.
  2) Diagonalization lang.
  3) The Post Correspondance Problem.
  4) The Universal lang.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

1) Halting Problem :-

→ NOTE :- 1) Already written answer in notes of Turing Machine.
2) write above theory & explain any 1 standard.

* Un-decidability of Post Correspondence :-

→ " Let A and B be two non-empty list of string over Σ.
A and B are given below.

$$A = \{x_1, x_2, x_3 \ldots x_n\}$$

$$B = \{y_1, y_2, y_3 \ldots y_n\}$$

We say, there is post correspondence between A & B.
if there is a sequence of one/more integer
$i, j, k \ldots m$ such that
The string $x_i, x_j \ldots x_m$ is equal to $y_i y_j \ldots y_m$

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

Post Correspondence Problem :

example :-

$$A = \{\underset{1}{a}, \underset{2}{aba^3}, \underset{3}{ab}\} \text{ and } B = \{\underset{1}{a^3}, \underset{2}{ab}, \underset{3}{b}\}$$

We will have to find a sequence using which when the elements of A and B are listed, it will produce identical string.

The required sequence is (2, 1, 1, 3)

$$A_2\ A_1\ A_1\ A_3 = \{aba^3\ a\ a\ ab\} = aba^6b$$

$$B_2\ B_1\ B_1\ B_3 = \{ab\ a^3\ a^3\ b\} = aba^6b.$$

We are accepting the Un-decidability of post Correspondence Problem.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* The classes P and NP.

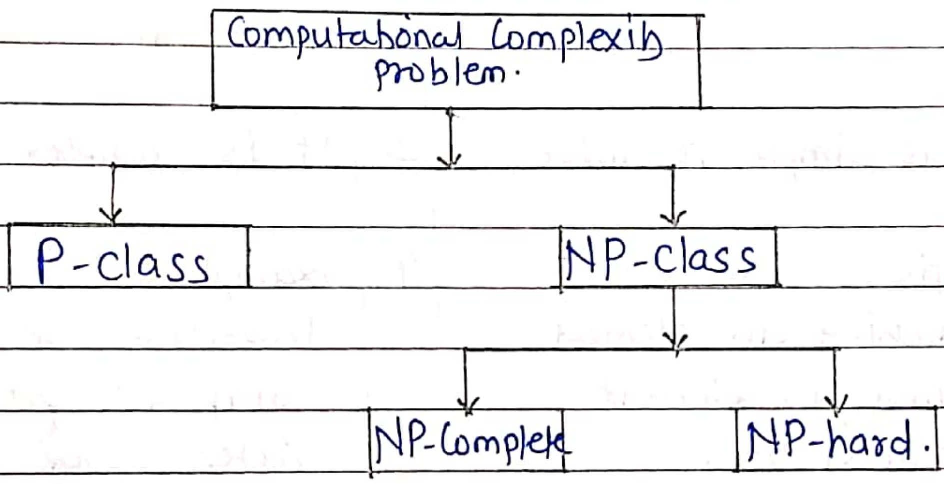
P → "The class of problem is denoted by P are solvable by a deterministic Turing m/c in polynomial time."

NP → "The class of problem denoted by NP are solvable by Non-deterministic Turing m/c polynomial time."

- P Problem are feasible or theoretically not difficult to solve by Computational means.
- The distinguish features of problem is that for each instance of any of these problem, there exists deterministic turing m/c that solves the Problem having time complexity as polynomial function of the size the Problem.
- P stands for polynomial
- NP stand for Non-deterministic Polynomial.

Scanned by CamScanner

- There are two group in which problem can be classified. the 1st problem that can be solved in polynomial time.
  e.g. Searching element
  Sorting element.
- The 2nd problem that can be solved in non-deterministic polynomial time
  e.g. Knapsack problem
  Travelling salesman Problem.



* P Problem Example :-
1) Searching element from list
2) Sorting element
3) Minimal Spanning tree.
4) Prims Algorithm
5) Kruskal Algorithm

* NP-Complete Problem Example :-
1) Traveling Salesman Problem (TSP)
2) Vertex Cover Problem (VCP)
3) Hamiltonia Circuit Problem (HCP)
4) Satisfiability Problem (SAT)
5) Exact Cover Problem

* Difference bet^n P class & NP class problem

Sr No	P Class	Sr No	NP-Class problem
1.	P stands for Polynomial time (deterministic)	1.	NP stands for Non-deterministic Polynomial.
2.	Problem that can be solved in polynomial time are called as P-class	2.	Problem that can be solved in non-deterministic Polynomial time are called as NP-Class.
3.	It is simple to solve.	3.	It is complex to solve.
4.	Example:- Searching an element, Sorting an element. Spanning tree.	4.	Example:- Travelling Salesman Problem. Knapsack problem. Vertex cover Problem.
5	P class is tractable	5	NP class is Intractable

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* Polynomial time Reduction :-

→ "A polynomial time reduction is a Polynomial-time algo. which construct the instance of a problem P2 from the instance of some other Problem P1."

- A Problem P_1 equivalently represents a lang L_1.
- we says that Problem P_2 can be solved in polynomial time if we can reduce, another problem P_1, which is known not be in P to P_2.

- Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be language, A polynomial time Computable fu^n $f: \Sigma_1^* \to \Sigma_2^*$ is called Polynomial-time reduction from L_1 to L_2 if and only if. for each $x \in L_1$, $f(x) \in L_2$.

* Polynomial time Reduction :

- To Prove whether Particular Problem is NP complete or not, we use Polynomial time reducibility. that means if.

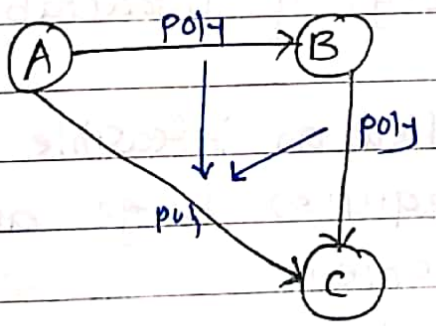
$$A \xrightarrow{poly} B \text{ and } B \xrightarrow{poly} C \text{ then } A \xrightarrow{poly} C$$

- The reduction is imp task in NP Completeness Proof.
- Various times of reduction :

1) Local Replacement :
In this reduction, $A \rightarrow B$ is dividing i/p to A in the form of components and then these component can be converted to Components of B.

2) Component design :
In this reduction. $A \rightarrow B$ by building special Component for input B that enforce Properties required by A.
The reduction can be denoted by $A \le_T^P B$.

Example :



If $f(A \rightarrow B)$ and $f(B \rightarrow C)$ then $(A \rightarrow C)$
this is Polynomial time Reduction.

* Tractable and Intractable :-

→ "Tractable Problems are the class of problem that can be solved within reasonable time & space"

For example :-

- Searching of key from list
- Sorting of list.

- These algo takes $O(n\log n)$, $O(n)$, $O(n)^2$ time complexity.
- We normally expect that tractable problem to be solved in polynomial time.
- It is also called as P-Problem.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

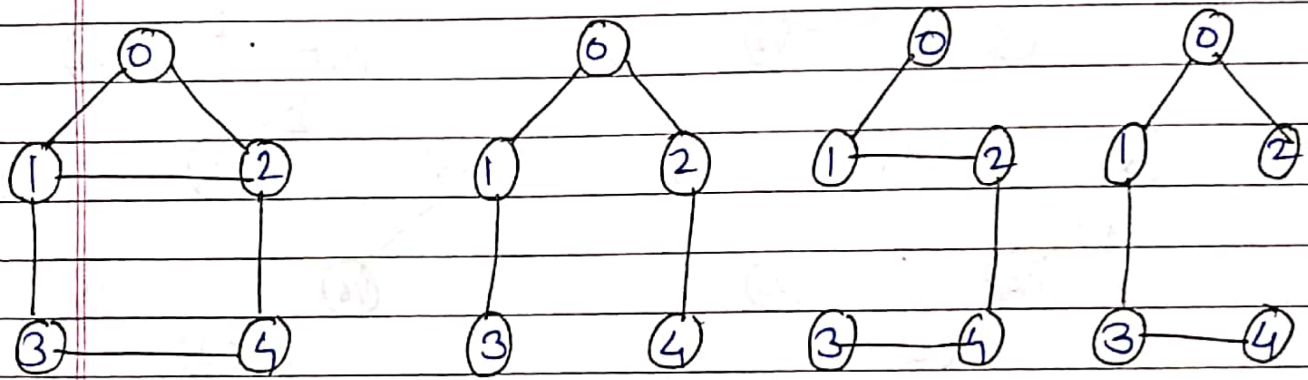
"Intractable Problem are the class of problem that can be solved within polynomial time."

- This leads to two classes of solving Problem - P class and NP- class.
- The lower bound of these algo. take exponential time complexities
- For example,
  Tower of Honoi is e.g. of Intractable Problem.

- Intractable is also called as infeasible Problem.
- Intractable Problem requires large amount of resources to solve Problem.
  Hence, It is infeasible.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* Example of Polynomial time :
* Kruskal Algorithm :
→ A spanning tree of a graph $G=(V,E)$ is a connected subgraph of G having all vertices of G and no cycle in it.
If graph G is not connected then there is no spanning tree of G.
A graph may have multiple spanning tree.



sample connected graph.

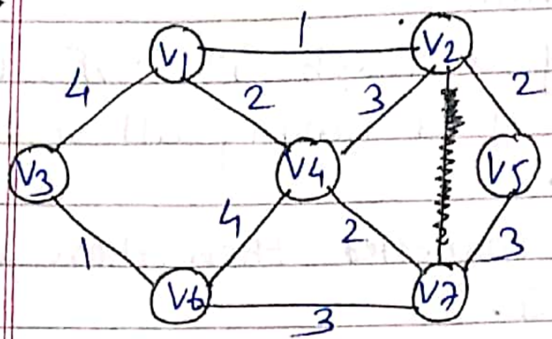
Spanning tree of graph.

- There are two types of spanning tree
1) Prims algo.
2) Kruskal's algo.

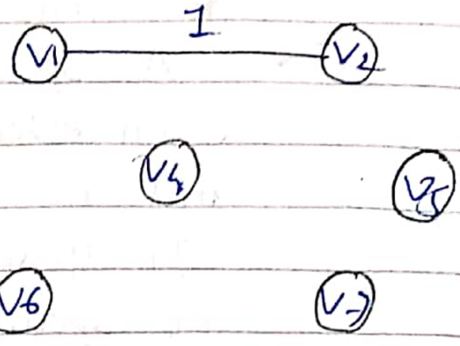
* Kruskals algorithm :
- It is one of the method for finding the minimum cost of spanning tree of the given graph.
- In kruskal algo., edes are added to spanning tree in increasing order of cost.
- If any selected edge forms cycle in spanning tree, then it is discarded.

Algo : 1) Arrang the edges of Graph G in ascending order of weight
2) let $G=(V,E)$ has n vertices. construct min spanning tree
$G_T=(V_T,E_T)$ $\therefore V_T=V$ and $E_T=\{\ \}$
3) For every edges e_i in $(e_1, e_2 \ldots e_k)$
if e_i does not form a cycle in G_T then $E_T=E_T \cup \{e_i\}$

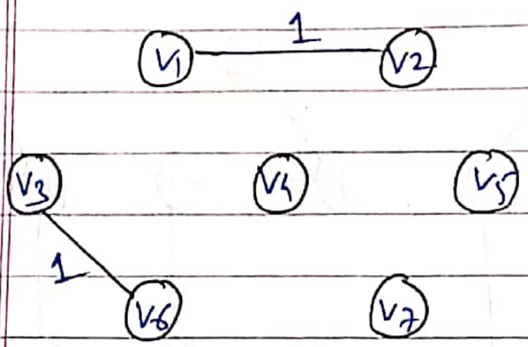
Example of Kruskal algo :-



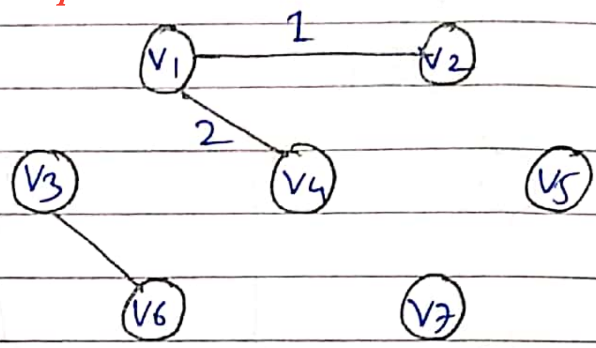
(a) Given Graph G



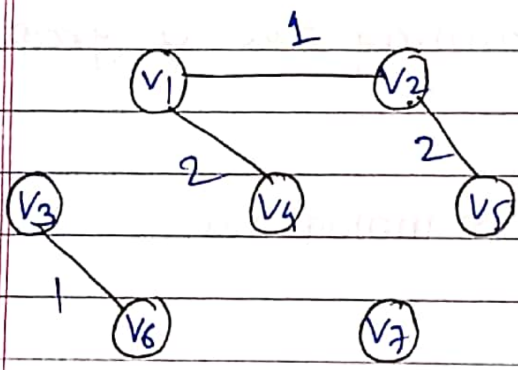
(b)



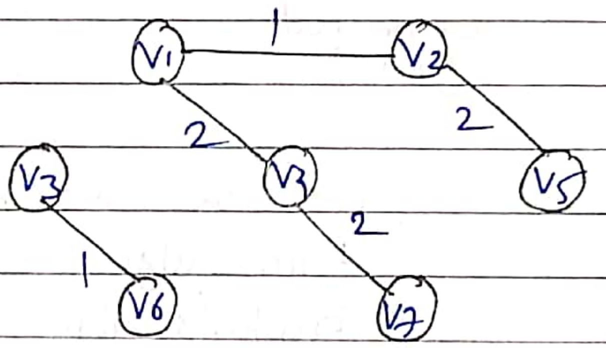
(c)



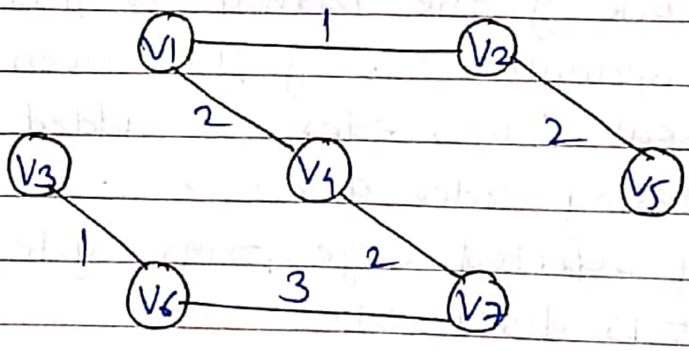
(d)



(e)



(f)



(g)

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* Kruskal algo. using Turing Machine (TM):

→ Kruskal algo is implemented using multitape TM.
- Edes of minimum weight is selected to connect two component.
- Initially, every node is in it's a Component by itself.

1) One tape of TM can be used to store every node with it's current component

2) A tape can be used for finding least edge weight. among the edges in which have not been used in the spanning tree.

3) when an edges is selected, it's two vertices are. copied on the tape. then we look for the Component of the two vertices.

4) If two component (i and j) found in the Previous. step are not the same component. then they can be merged into single component with help of another tape.

Using above algo, we can find minimum spanning tree in n round.
Thus, multiple tape TM will require $O(n)^2$ and given problem is in P-Problem.

* NP Complete Problem :

→ "A Problem P is said to be NP complete if the following two condition are satisfied.

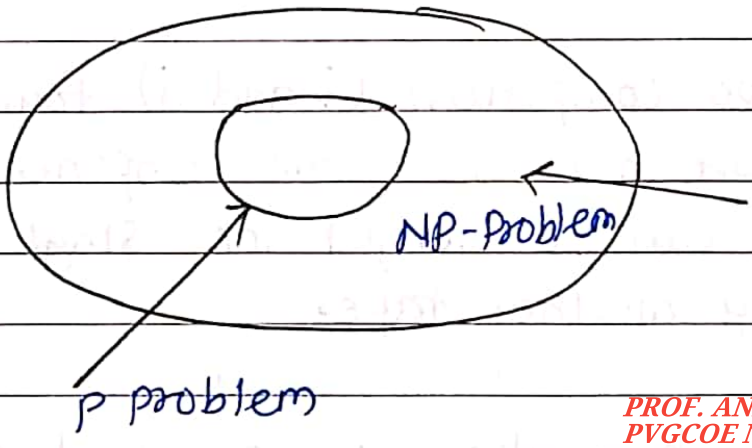
1. The Problem L2 is in the class NP.
2. For any problem L2 in NP, there is Polynomial time reduction of L1 to L2.

- A Problem is NP-Complete if it is in NP and for which no Polynomial time. Deterministic TM solution is known so far.

- NP denotes Non-deterministics Polynomial language Problem.

  Hence $P \subseteq NP$.

- NP Problem is also called as Intractable Problem.



- Example of NP Complete Problem :

1) Travelling salesman Problem (TSP)
2) Vertex Cover Problem (VCP)
3) Hamiltonian Circuit Problem (HCP)
4) Satisfiability Problem (in short SAT)
5) Exact Cover Problem.

Example of NP-Complete :

1) Satisfiability Problem (SAT)

: " The satisfiability Problem is :
" Given a Boolean expression, is it satisfiable ?

"A Boolean expression is said to be satisfiable if at least one truth assignment makes the boolean expression true."

The Boolean expr^n are created using .
1) The variable whose value can be 0 or 1.
2) The operator that can be used in expression can be ∨, ∧ . The ∨ means OR and ∧ means AND operator.
3) Unary operator ¬ stands for negation.
4) The Parenthesis are used to group the operand & operators in the expression

The highest Precedence is to ¬, then ∧ then ∨.

For example :-

The boolean expr^n is $(x \wedge y) \vee (y \wedge z)$.

If $(y \wedge z)$ and $x \wedge y$ both are true then boolean expr^n is true. if $(x \wedge y)$ and $(y \wedge z)$ both are false then boolean expr. is false.

The boolean expression $((x_1 \wedge x_2) \vee \sim x_3)$ is true.
for $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$

Therefore $((x_1 \wedge x_2) \vee \sim x_3)$ is satisfiable.

- $(x \wedge y \wedge z)$ is satisfiable when $x=1, y=1$ & $z=1$

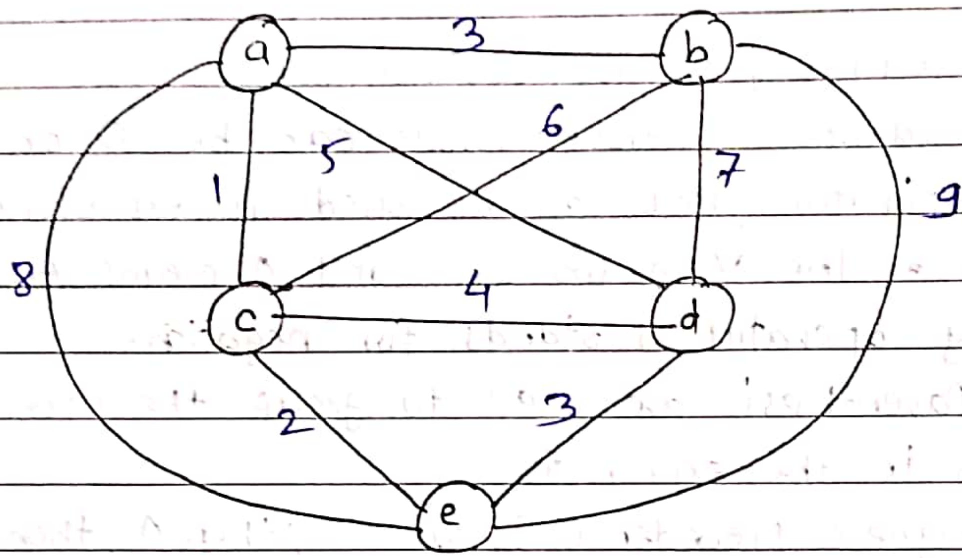
But

$x \wedge (\neg y)$ is not satisfiable.

2) Travelling Salesman Problem (TSP):

→ This Problem can be stated as
" Given set of cities and cost to travel bet^{s} each pair of cities, determine whether there is path that visit every city once and return to the first city, such that cost travelled is less."

For example,



- The tour will be a-b-d-e-c-a and total cost of tour will be 16.
- This Problem is NP Problem as there may exist some Path with least (shortest) distance between the cities.
- If you get the solution by applying certain algorithms then travelling salesman Problem is NP-Complete Problem.
- If we get no solution at all by applying an algorithm then the travelling salesman Problem belong to NP-Hard class Problem.

* Node - Cover Problem :

"A node Cover Problem is to find node cover of minimum size in a given graph.

- The word node cover means each node covers it's incident edges.
- Thus by node cover, we expect to choose the set of vertices which cover all the edges in a graph."

- A node cover undirected graph $G = (V, E)$ is a Subset V, of the vertices of graph which contain at least one of the two endpoint of each edge.

- The node cover problem is the optimization Problem of finding a node cover of minimum size in a graph.
- This Problem can be stated as <u>Decision Problem</u>.

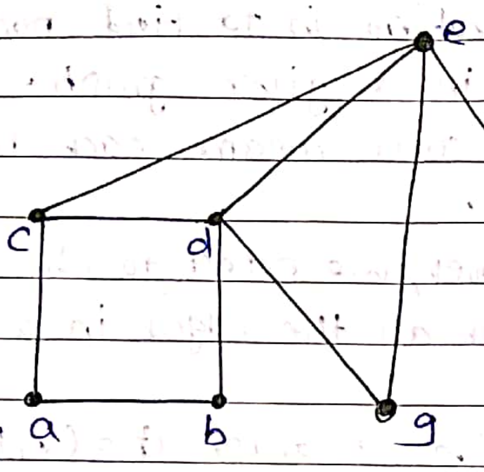
NODE-COVER = { <G, K> graph G has vertex cover of size K }.

- To show that, node cover Problem $\in NP$, for a given graph $G = (V, E)$, we take $V_1 \subseteq V$ and verifies to see if it form node cover. verification can be done by ~~using~~ checking for each edge $(u, v) \in E$, whether $v \in V_1$ or $u \in V_1$.
- This verification can be done in Polynomial time.

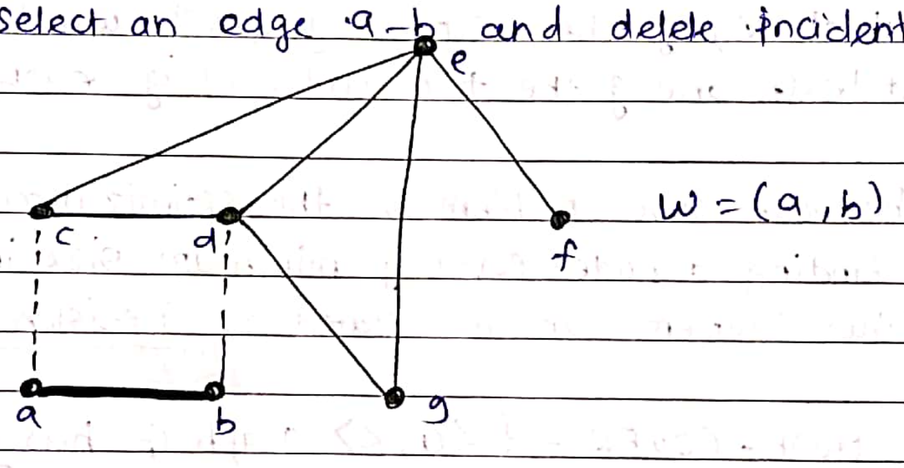
<u>Example</u> :

Consider a graph G as given below. Now we will select some arbitory edge and delete all the incident edges. repeat this process until all the identical edges get deleted.

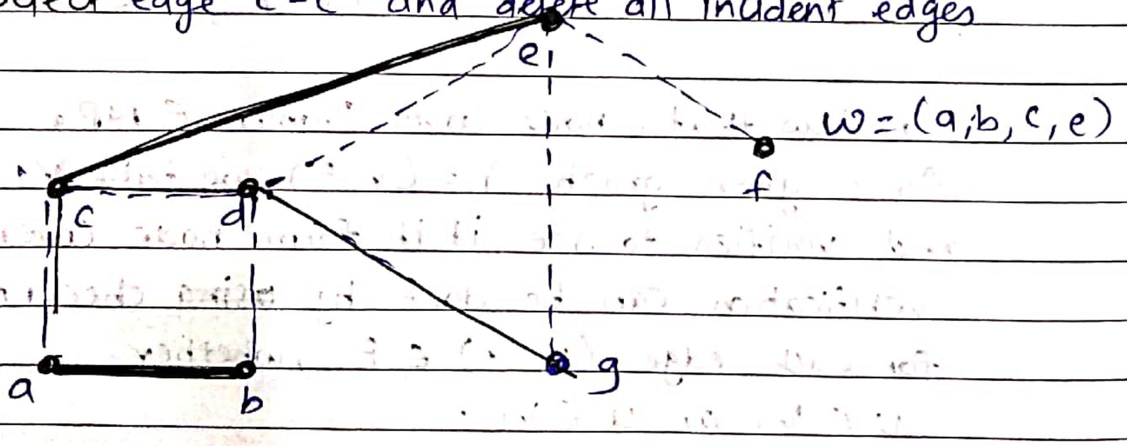
* Example Node - Cover Problem :-



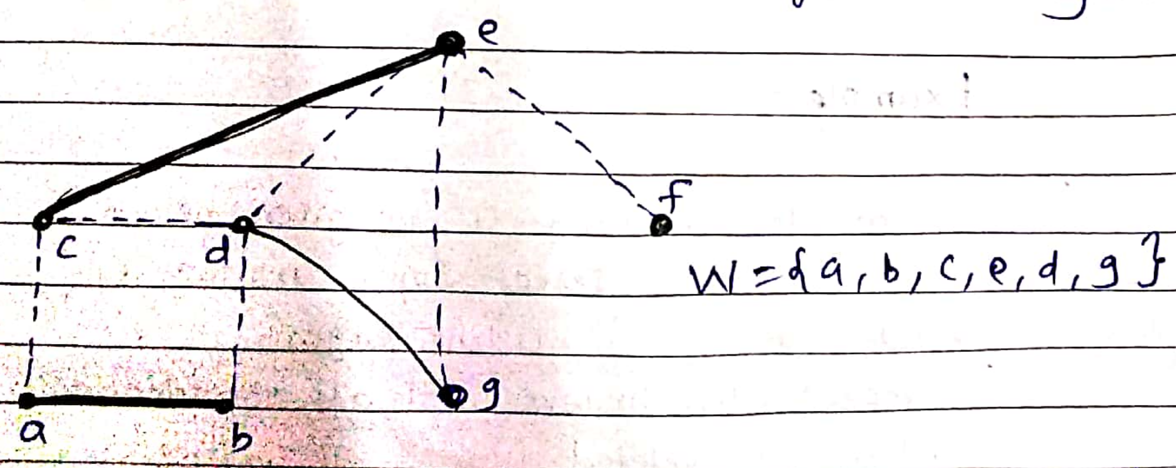
Step 1 :- select an edge a-b and delete incident edges



Step 2 :- select edge c-e and delete all incident edges



Step 3 :- select an edge d-g. All incident edges are already deleted



Thus, we obtain node cover as {a, b, c, d, e, g}.