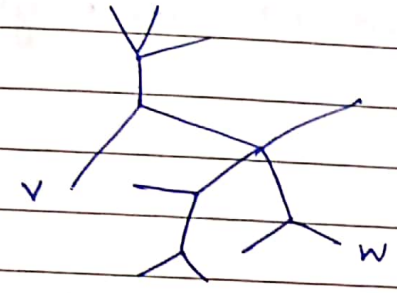
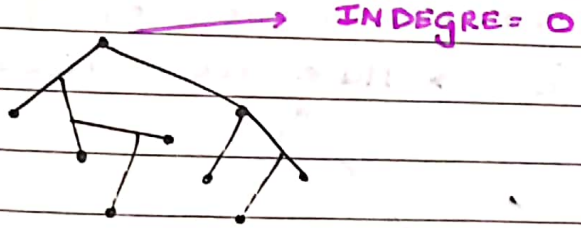
UNIT - 4 :

A simple graph such that for every pair of vertices v and w there is a unique path



* Rooted Tree.



Let T be a rooted tree :

→ The level $l(v)$ of a vertex v is the length of the simple path from v to the tree.

→ The height h of a RT T is max no of all levels numbers of its vertices

$$h = \max_{v \in V(T)} \{ l(v) \}$$

* Characterization of Trees

If T is a ~~tree~~ graph with n vertices then following are equivalent

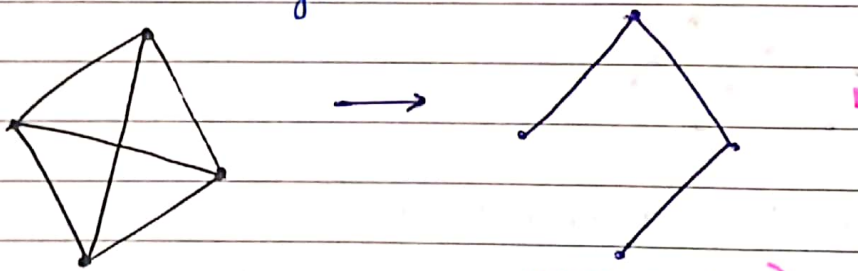
- → T is a tree
- → T is connected and acyclic → UNIQUE PATH.
- → T is connected and has n-1 edges
- → T is acyclic and has n-1 edges.

* Spanning trees

Given G is a graph & T is spanning tree of G if

- → T is subgraph of G. AND
- → T contains all the vertices of G.

→ Are Acyclic → Have all vertices → It is connected



(SUB GRAPH)

HAVE ALL VERTICES.

→ n spanning trees can be formed with n vertices.

* Searching spanning tree

- → Breadth - first search method
- BACKTRACKING → Depth - first search method

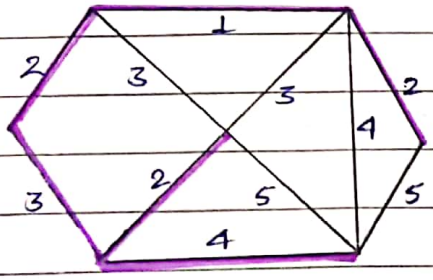
* Minimum spanning trees (MST) → EDGES HAVE WEIGHTS

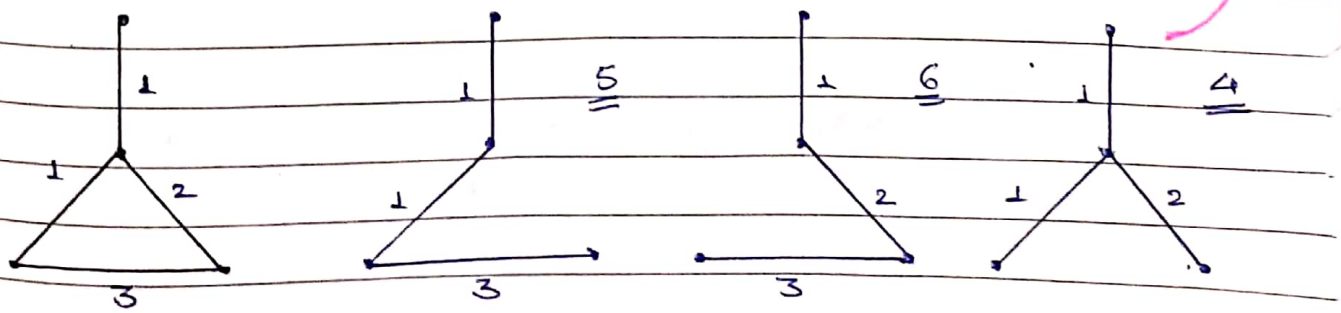
→ Is a spanning tree
→ Have minimum weight.

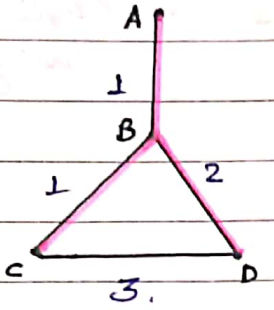
* Prim's Algorithm:

→ Pick any vertex as starting vertex. T = {a}
→ Find the edge with smallest weight incident to a. Add it to T. Also include in T the next vertex. (b).
→ Find the edge of smallest incident either to a or b. Include in T that edge and the next incident vertex. (c)
→ Repeat prev step choosing the edge of smallest weight that does not form a cycle until all vertices are in T.

The resulting subgraph T is min. sp. t. (MST)







$T = \{A\}$

$T = \{A, B\}$

$T = \{A, B, C\}$

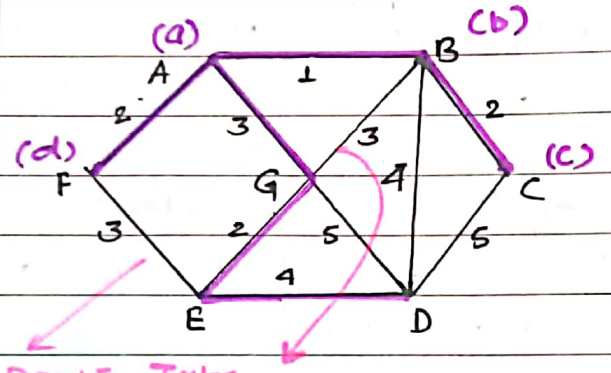
$T = \{A, B, C, D\}$

B < BC = 1 ✓
BD = 2

BD ✓

C — CD X = 3
CB

Ques



DON'T Take
→ MAKES LOOP

$T = \{A\}$

A — AF = 2 .
AB = ~~4~~ 1 ✓ .
AG = 3 ✗ .

$T = \{A, B\}$

B — BC = 2 ✓ .
BD = 4
BG = 3

$T = \{A, B, C, F\}$ ← $T = \{A, B, C\}$

F — FE = 3 C — CD = 5 AF

AG.

$T = \{A, B, C, F, G\}$

G — GE .
GD

$T = \{A, B, C, E, F, G\}$

E — ED = 4 ✓
EF = 3 (Makes loop)

n-1 EDGES

Min Weight = 1 + 2 + 2 + 3 + 2 + 4
= 14 .

Camlin | Page
Date / /

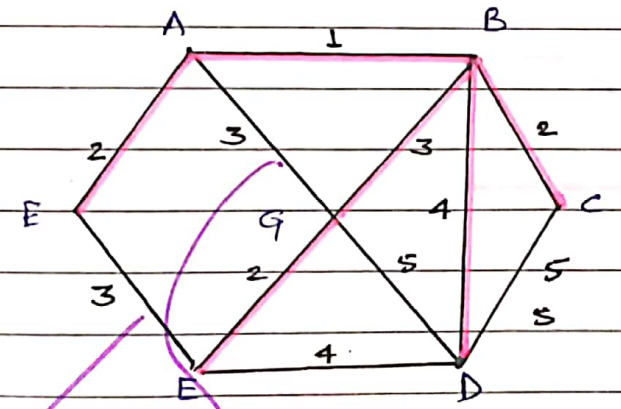
* Kruskal's Algorithm :-

→ Find the edge in the graph with smallest weight (if there is more than 1 pick @ random) Mark with any colour, say red

NO SEQUENCE

→ Find the next edge in the graph with smallest weight that doesn't close a cycle Colour that edge and the next incident vertex.

→ Repeat prev step until you reach out to every vertex of graph. The choosen edge ~~from~~ form the desired MST.



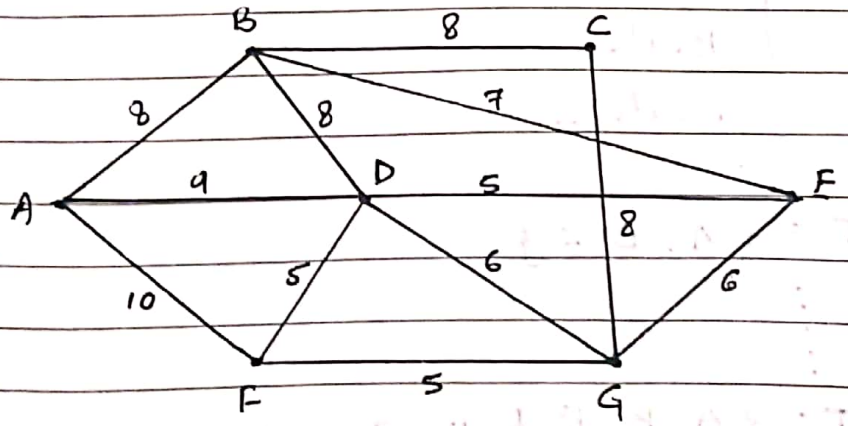
AB
BC] or ↻ Works.
AF
EG (NOT CONNETED)
BG or AG.
BD or ED.

MAKES LOOP

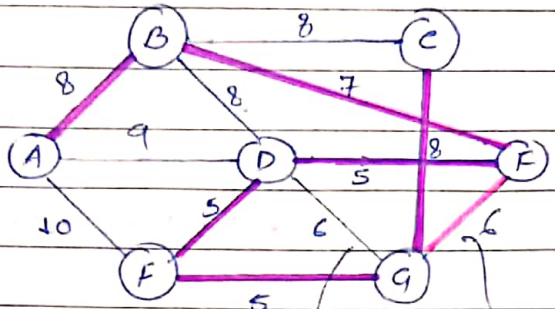
Check all but connect only

How Ques

Use Prim's Algo. to find the MST



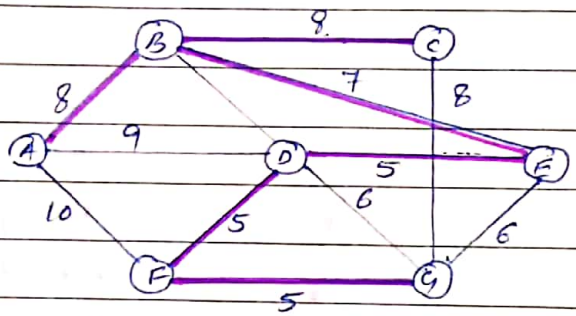
Ans Prim's Algorithm :-



T.W = 38

Making Loop ✗

Kruskal's Algorithm :-



T.W = 38

Prim's :

$T = \{A\}$

$E = \begin{cases} \bar{AB} \text{ or } (A,B) & = 8 \\ \bar{AD} & = 9 \\ \bar{AF} & = 10 \end{cases}$

$T = \{A, B\}$

$E = \begin{cases} \text{Blah} \\ \text{Blah} \\ \text{Blah} \ldots \end{cases}$

$T = \{A, B, E\}$

⋮

$T = \{A, B, E, D, F, G, C\}$

Kruskal :

Wt = 5 { DE ✓, DF ✓, FG ✓ } Wt = 6 { Blah ✗ }

Wt = 7, 8, 9, 10 { Blah.

Notes :-

→ If all the edge wts. are distinct then both the algo are garenteed to give same MSP

→ If all the edges are not distinct then both the algo may not produce the same MSP

	Prim's	Kruskal
→	The tree that we are making are always remains connected (while making)	Usually it is disconnected (Final will be connected)
→	It grows a sol^n from a random vertex to next adjecent cheapest vertex to the existing tree.	It grows the sol^n from cheapest edge to next cheapest edge to ex. tree
→	It is preferred when the graph is dense	- - - is sparse.

m-ary tree :
↓
No of BRANCHES

m = 2
→ Binary
↳ MAX 2

Camlin Page
Date / /

* Binary Trees.

A rooted tree is called m-ary tree when each internal vertex has atmost m children

i.e Order(v) ≤ m

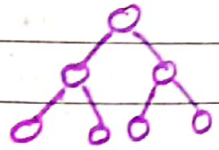
If each internal vertex to rooted tree has atmost 2 or children i.e m=2 then it is called binary tree

Properties :

→ No of vertices in a binary tree is always odd.

→ If no of vertices - P is of degree 1 then : n - P is the no of vertices of degree 3

→ Full / Complete BT = m=2 (exactly)



* Traversal :

→ Pre order — ROOT - L - R

→ In-order — L - ROOT - R

→ Post-order — L - R - ROOT

→ Pre order search aka eleft first search if you know wnit to explore the root before inspecting and leaf root node you should take pre order trav. b/w you will encounter every roots first before all the leafs

Camlin | Page
Date / /

STEPS:

1 → First visit the root (V) of the tree
2 → If V_L exist then search the left subtree T_L of the given T in pre-order
3 → Check out whether V_R exist then search the right sub tree of given tree T in pre order

Left SUB TREE
RT SUB TREE

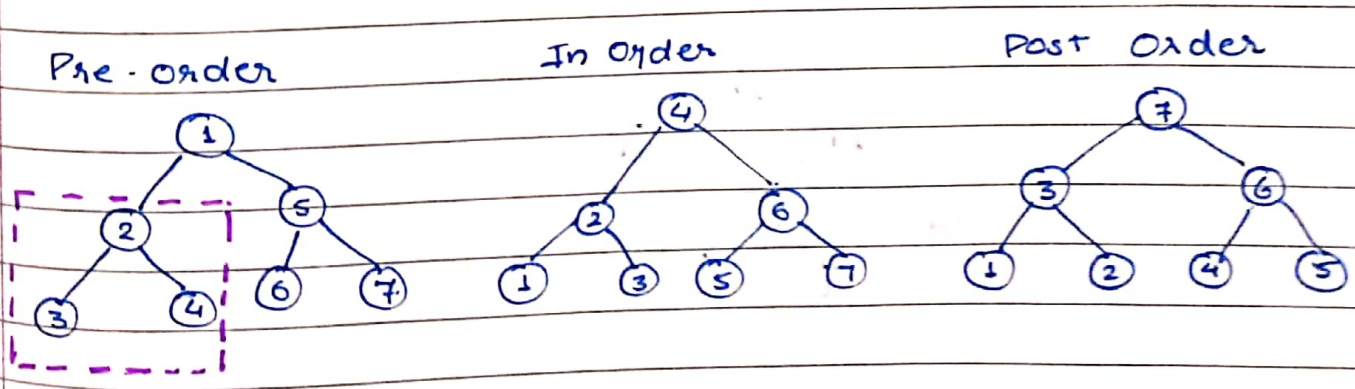
→ In-order search aka systematic order search

1 → Search the left sub tree T_L in order
2 → Visit the root of the tree
3 → Search the right sub tree T_R in order

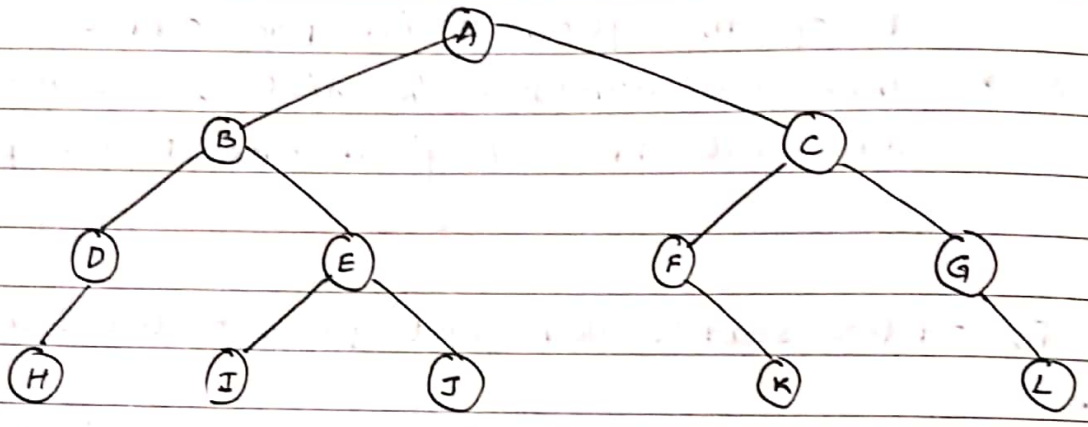
→ Post-order search

1 → Search the left sub tree T_L in post order
2 → Search the right sub tree T_R in post order
3 → Visit the root of the tree

Eg: O/P 1-2-3-4-5-6-7 in Pre, In, Post.



Ques Traverse the tree in Pre, In, Post order

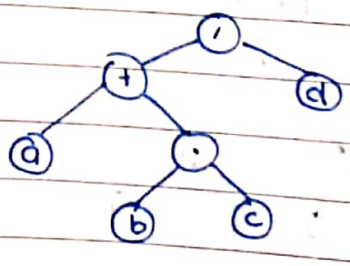


Pre : A - B - D - H - E - I - J - C - F - K - G - L

In : H - D - B - I - E - J - A - F - K - C - G - L

Post : H - D - I : J - E - B - K - F - C - G - C - A .

Ques (A + b . c) / d ⇒ draw tree.



* Transportation Network

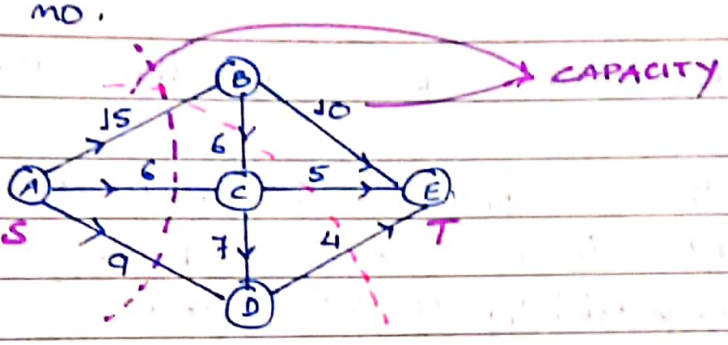
Directed weighted graph is said to be trans. net. if it satisfies 4 condns

→ Connected and no loop (SELF)

SOURCE S → ∃ Only one node whose in-deg is zero.

SINK T → ∃ Only one node whose out-deg is zero

→ Capacity of that edge which is a non-ve real no.



→ Flow

A flow in transportⁿ network is a funⁿ f that assign to each edge a number so that that no is

→ $0 \leq \left(\begin{matrix}\text{flow along}\\ \text{a edge}\end{matrix}\right) \leq \left(\begin{matrix}\text{capacity of}\\ \text{that edge}\end{matrix}\right)$

→ For each vertex V other than s and T the total flow into V is equals to the total flow out of V

i.e Total flow in V = Total flow out V

→ Cut (-----)

Divides graph in two disjoint set

→ One group (set) will have source node and other group will have sink node

→ Eg. Above

Cut 1 A | B D C E

Cut 2 A C D | B E

Capacity of cut = Only source to sink

Cut 1 = AB + AC + AD

Cut 2 = AB + CE + DE

SOURCE SINK

→ Residual graph:-

Residual cap = Capacity - Flow

It is a graph which indicate additional possible flow if there is such path from S to T then there is possibility to add flow

Rasidual cap: It is the original capity - Flow

→ Minimum cut

Minimum cut aka bottle neck cap which decide max poss. flow from S to T through an augumented path

$A \xrightarrow{15} B \xrightarrow{10} E$

Minimum of two (...) can be supplied
i.e there 10

REPRESENTATION
15, 10 ON SAME EDGE

∴ Residual cap = 15 - 10 = 5

$A \xrightarrow{5} B \xrightarrow{6} C \xrightarrow{5} E$

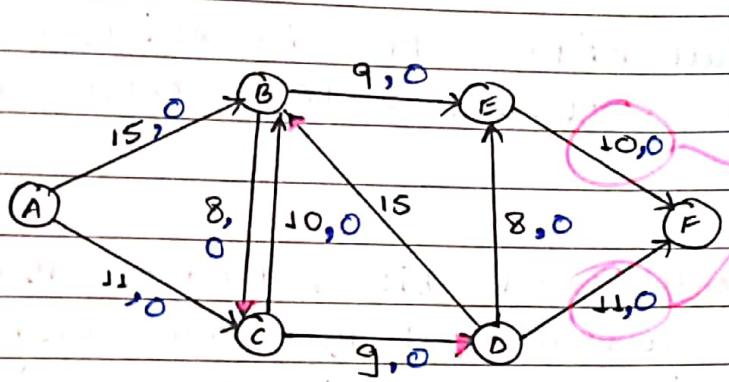
Min = 5

∴ Total flow at E (till now) = 10 + 5
= 15.

FORD FULKERSON ALGORITM.

Camlin | Page
Date / /

Ques Given a graph which represent trans. net where each edge has a capacity find out the max flow from S to T in the graph



MAX POSSIBLE FLOW IS 21

(NOT A SELF LOOP)

→ Start with initial flow as zero

→ while there is an augumenting path from S to T add two path flow by flow

→ Return flow

→ ✓

→ Path 1 : A $\xrightarrow{11}$ C $\xrightarrow{9}$ D $\xrightarrow{10}$ F ⇒ 9

RC 2 1

Flow = 9

DRAW GRAPH AGAIN AND AGAIN

Path 2 : A $\xrightarrow{2}$ C $\xrightarrow{10}$ B $\xrightarrow{9}$ E $\xrightarrow{10}$ F ⇒ 2

RC 0 8 7 8

Flow = 9 + 2 = 11

Path 3 : A $\xrightarrow{15}$ B $\xrightarrow{7}$ E $\xrightarrow{8}$ F ⇒ 7.

RC 8 0 1

Flow = 11 + 7 = 18

No more possible paths ∴ Stop.

∴ Max flow = 18

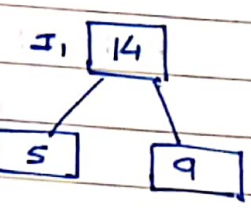
* Huffman Coding Tree (COMPRESSION ALGORITM)

Is a full binary tree in which each leaf of the tree corresponds to a letter in given alphabet

Prefix code: Code (bit) sequences that are assigned in such a way that the code assign to one character is not the prefix code assigned to other character.

Character	Frequency
a	5
b	9
c	12
d	13
e	16
f	45

a	00	0001
b	01	a, b
c	0	cc b
d	001	c d
		ERROR



After 1's iteration

c	12
d	13
I_1	14
e	16
f	45

SORTED

2nd iteration

I_1	14
e	16
I_2	25
f	45

3rd

I_2	25
I_3	30
f	45

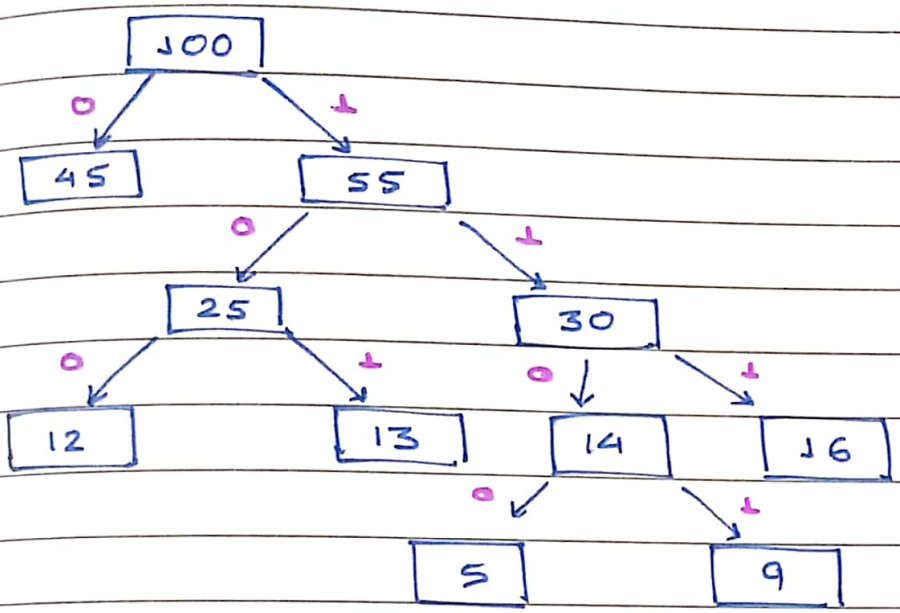
4th

f	45
I_4	55

5th

I_5	100

Camlin | Page
Date / /



a → 1100
b → 1101
⋮
f → 0
↳ In bit
(FROM BYTE
=> COMPRESSION