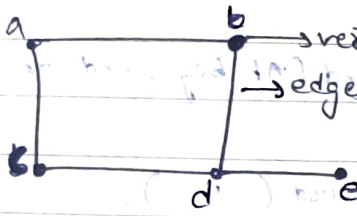
Graph

(1) → set of obj where pair of obj connected by links. Also, type of data structure like trees.

(2) → Interconnected obj are represented by points termed as vertices. & vertices are connected by links called edges.

stored data. Eg:-



⇒ G [V, E] **.

→ V → {a, b, c, d, e}

E → {ab, cc, bd, cd, de}.

(3) ∴ Graphs represented by
Arrays of vertices
& 2D Arrays of edges.

(4) Operations
→ Add Vertex, Add Edge, Display Vertex.

(5) For Traversing a graph → 2 methods

Depth first Traversal.
[Uses Stack]

Breath first Traversal.
[Uses Queue].

(6) Topological Sort.

→ An ordering of vertices in a directed acyclic graph.

⑦ Minimum Spanning Tree.

» Spanning tree is a subset of Graph.

» A spanning tree does not have cycles.

» Every undirected connected graph has atleast 1 Sp tree.

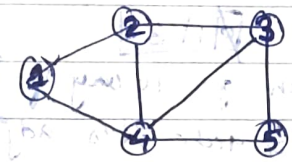
& max » n^{n-2} trees.

Unconnected graph → No trees. [n → no: of nodes].

Let's discuss Everything.

(1) Representation » Adjacent Matrix [space comp → $\Theta(n^2)$]

Adjacency list. [Scomp → $\Theta(n+2e)$.]

» 

⇒ Eg: Graph

a) Adjacent matrix; » Loops come in diagonal row.

a[i][j] = 1 if i, j are adjacent. otherwise 0.

∴ »

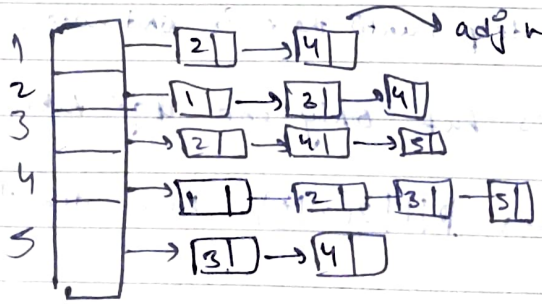
	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

5 × 5.

B/n 1 & 4 we have edge. ∴ (1,4) & (4,1) = 1. Simi for rest

b) Adjacency list. » Uses linked list

» For each vertex = 1 Link list.



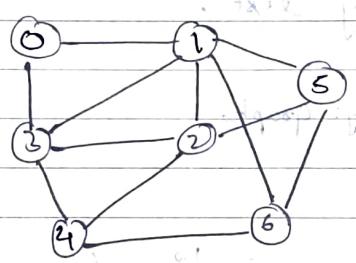
If graph is dense better to use matrix method.
↳ i.e many branches / connections.

---

(2) BFS traversal & DFS traversal.
~~(lvl order)~~

(a) → For BFS → Breath First Search
→ Done by Queue data structure.
→ Anyone node can be taken as root node unless specified.

Eg:-



Go to 0 see adj ver of 0
i.e 0 1 3 5
then go to any of its adj vert & take its adj vertices.

0 1 3 2 5 6 . . . .
Continue this process.

∴ Ans → 0, 1, 3, 2, 5, 6, 4.
↳ can be other combinations.

(b) For DFS → Depth First Search [stops when all popped out]
→ Done by Stack.
→ Anynode can be taken root node.

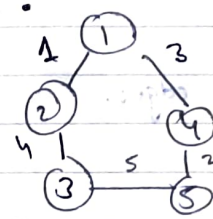
Steps → You will go deep, until there is end node.
Same as BFS but only one adj taken at a time

i.e 0 1 3 2 → 0 1 3 2 5 Not directly 0 1 3 2 5 6 ...

---

(3) Minimum S

→ WKT, a
for a



Graph

NDLO, MPn

ND: of

See more

★ From a con
edges

(3) Minimum Spanning Tree

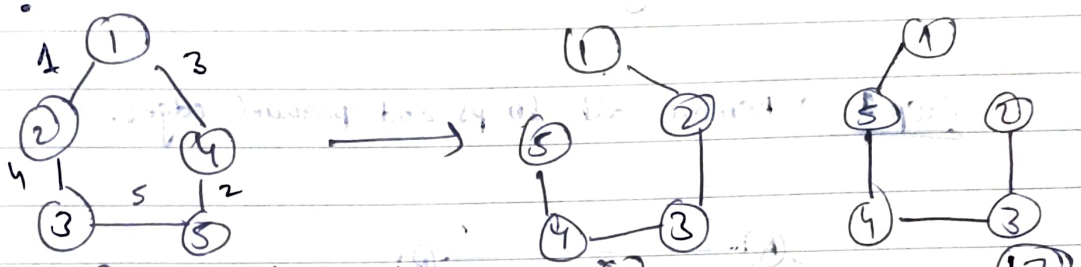
⇒ WKT, a graph ⇒ $G(V, E)$

for a spanning tree ⇒ $S(V', E')$

$V = V'$ & $E' < E$

$E' = V - 1$

[same no: of vertices & one less edge.]

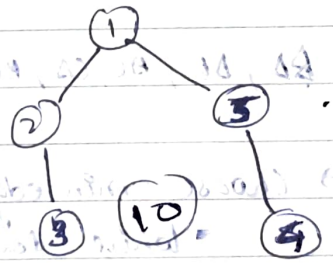


Graph (G) ∴ 14 Spanning trees.

Now, Mini Spanning tree

is ⇒ the spanning tree which has the least edge weights.

∴ for above



∴ Min spanning tree

* complete

No: of Sp trees in a Graph ⇒ (n^{n-2}).

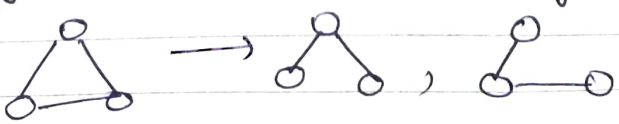
See more props in last page

---

* From a complete graph by removing max $(e - n + 1)$ edges, we can construct a spanning tree.

e - edges.
n - vertices

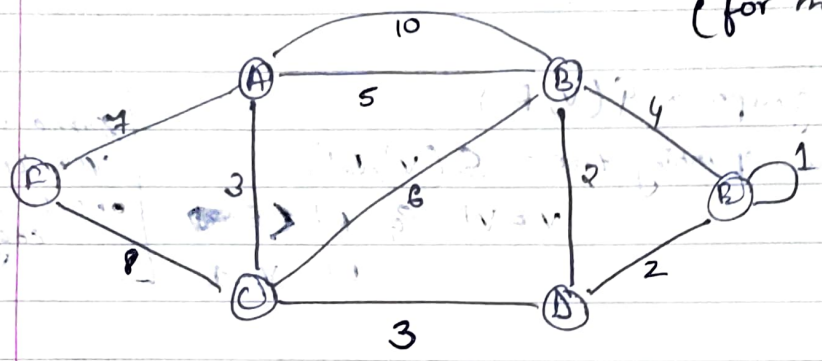
P. 7. 0.



$(3 - 3 + 1) \Rightarrow 1$ max edge.

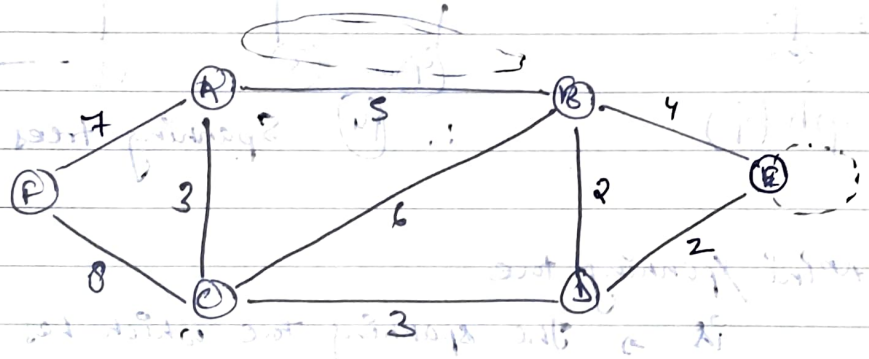
Kruskal's Algorithm

(for min spanning tree)!



Num → weight of Edge.

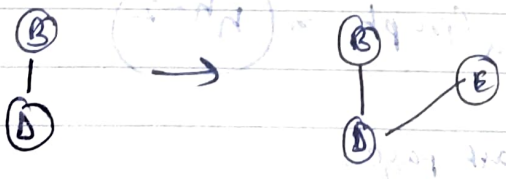
Step 1 → Remove all loops and parallel edges.



Step 2 → Write edges in ascending order (min to max).

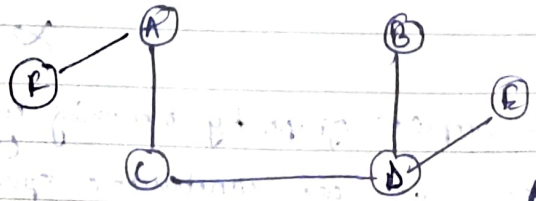
AD, DE, AC, CD, BE ..... FC.

Step 3 → Choose min edge weight & go in seq order. While drawing, DON'T DRAW CYCLES.



--- Skip BE --- Final
Skip BC
Skip FC.

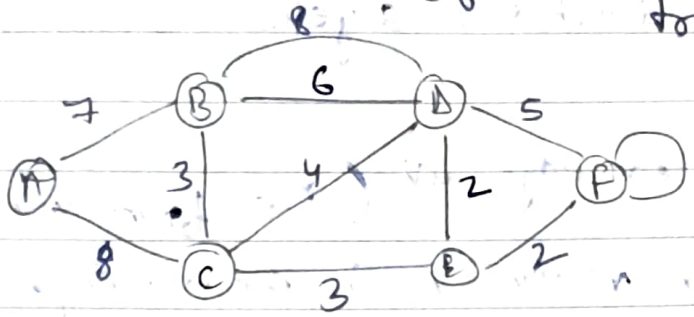
Final →



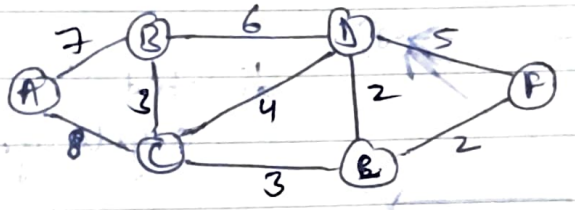
Ans

Prim's Algorithms (for min spanning tree)

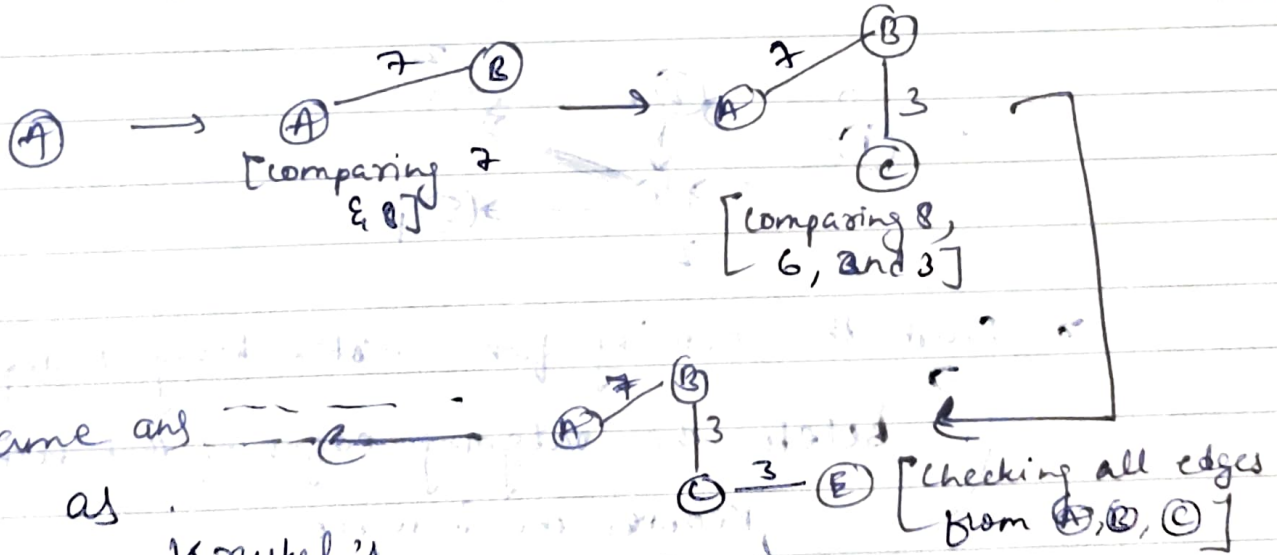
Q



Step 1 → Remove all loops and parallel edges.



Step 2 → Choose any edge and draw MST from there.
Check outgoing edges from each vertex you draw. & choose min.



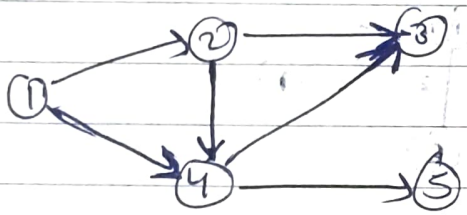
Same as Kruskal's Algo.

Topological Sorting.

(1) Graph should be directed & acyclic.
(i.e not cycles).

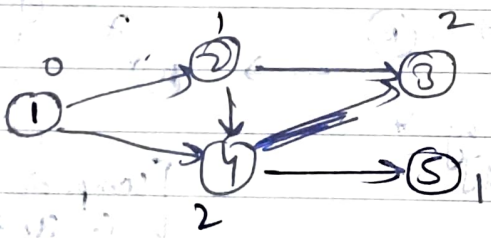
(2) for every directed edge uv (i.e from vertex u to v), u comes before vertex v in the ordering.

(3) How to find

→ 

directed & acyclic graph

→ Calculate indegree of each vertex (no: of arrows coming in).



→ Start the ordering from vertex having indegree 0.
Delete the outgoing edges of that vertex
Reorder the indegree.
Repeat.

Order → 1 2 4 3 5
1 2 4 5 3
} As both 3 & 5 have 0 indegree at last.