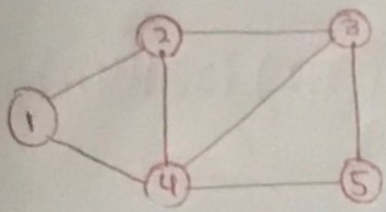
Graphs:

Two popular methods are used to represent graph is
① Adjacency matrix
② Adjacency List
Some other methods are Multi List, etc.



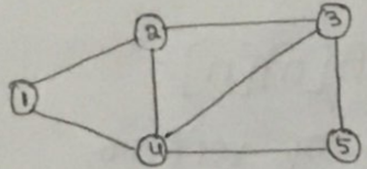
→ This is the simple graph on paper but we use different methods to implement graph.

Matrix:

No. of rows and columns $(m \times n)$ $\left[1 \underset{n}{=} \right]^m$

Adjacency Matrix:

• The matrix is used for represent graph.
• It has same order $(n \times n)$, where n = no. of vertexes.
• Example:-



→ This is undirected graph so this edge consider both like (1,2)(2,1).

→ 5 nodes, so $n = 5$, the order of matrix is 5×5.

$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} \; & \; & \; & \; & \; \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}_{5 \times 5} \end{array}$$

• First to check the nodes of loop like ① , means back to itself, it has the order same like (1,1).
• The loop only occurs in the diagonal enteries if no loop present so diagonal change to 0.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & & & & \\ 2 & & 0 & & & \\ 3 & & & 0 & & \\ 4 & & & & 0 & \\ 5 & & & & & 0 \end{array}_{5\times5}$$

→ Now we check the edge of each nodes and set to 1.

→ Edges are (1,2) (2,1) (1,4) (4,1) (2,3) (3,2) (2,4) (4,2) (3,4) (4,3) (3,5) (5,3) (4,5) (5,4)

$$\begin{array}{c|ccccc} i \downarrow \; j \rightarrow & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 1 & 1 \\ 4 & 1 & 1 & 1 & 0 & 1 \\ 5 & 0 & 0 & 1 & 1 & 0 \end{array}_{5\times5}$$

→ space complexity in this case is $\Theta(n^2)$ bcz. we have n no. of rows and col.

→ use Two dimensional Arrays to Represent Adjacency matrix.

$\boxed{\text{Define:}}$ It is a matrix $A[n][n]$ where n is the no. of vertices

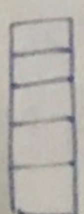
& $\begin{cases} a[i][j] = 1 & \text{According to matrix \&} \\ a[i][j] = 0 & \text{if i and j are adjacent} \end{cases}$

Adjacency List:

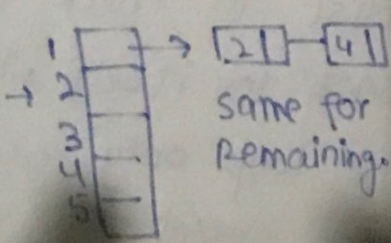
With the name we know we use linked List.

→ For each vertice we built a separate node.

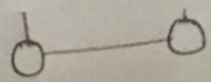
→ Like this.



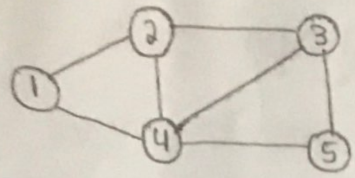
→ For further we built adjacent node of each node.



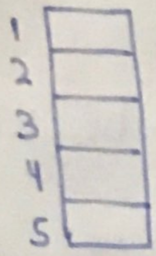
same for Remaining.



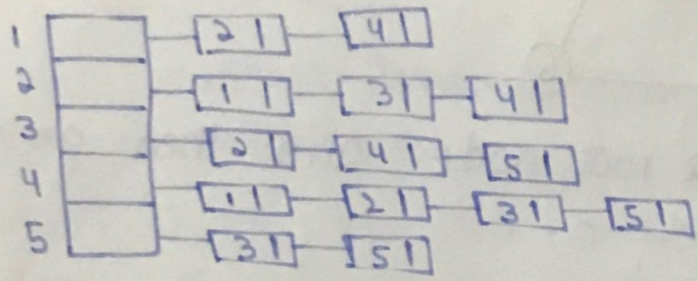
→ Use the same example for adjacency List.



→ For this we create 5 node because we have 5 vertices.



→ For further of each node we create the adjacent node of each vertix node.

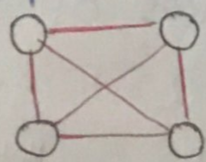


→ space complexity in this case is $\theta(n+2e)$
n is the no. of nodes and we write 2e bcz we return one edge 2 times.

→ we have 5 vertices | 5 nodes so we maintain 5 linked List.

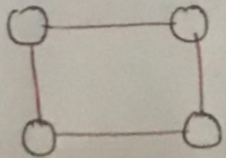
→ when the graph is $\boxed{\text{dense}}$ it is better to used adjacency matrix and when the graph is $\boxed{\text{sparse}}$ it is better to used adjaceny list.

→ Dense graph means each node connected with other nodes simple a complete graph example



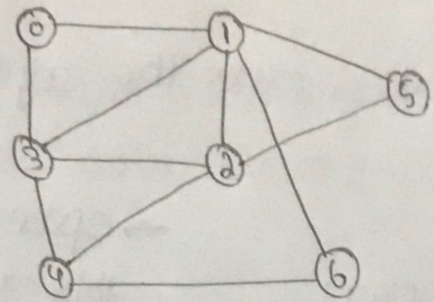
→ undirected Graph

→ sparse graph means few no. of edges of the nodes



Graph Traversals:-

- Two technique to traverse the graph
  - → BFS (Breath First Search)
  - → DFS (Depth First search).
- In the graph start the traversal from the any node of the graph.
- For BFS the QUEUE data structure will be use.



- We take 0 as a root node and then go first on the all vertices of 0.
- Queue | 0 |

Result:- 0

- First insert the root node and then all of his vertices.

| ~~0~~ | ~~1~~ | 3 |

Result:- 01

3 . Now move on the all the vertices of 1, insert only the non-visited or non-inserted. (2, 5, 6).

| ~~0~~ | ~~1~~ | ~~3~~ | 2 | 5 | 6 |

Note:- Insert in an any order.

output:- 013

- Now check for (3) → only 4.

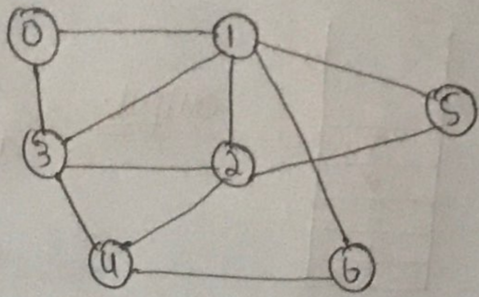
~~0~~	~~1~~	~~3~~	~~2~~	~~5~~	~~6~~	4

Result:- 0132564

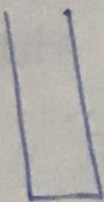
→ This is the BFS traversal of the graph. But this is not the only BFS traversal of the graph.

Now Discuss the DFS Traversal:

- DFS means depth First search.
- In DFS stack data structure will be used.



- stack.

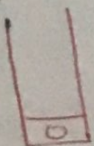


output:-

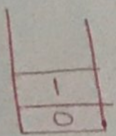
- start Traversal from any node. so we choose 0.

→ First insert 0 on the stack and print the 0.

→ After only one vertice/adjacent insert because we use depth First search go on the depth.

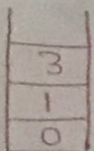
→ 

output:- 0.

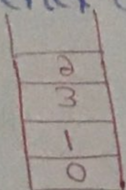
→ 

output:- 01

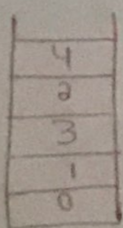
- Now Check the unvisited vertex of 1 and choose any 1.

3 

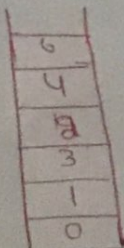
output:- 013

→ 

output:- 0 1 3 2

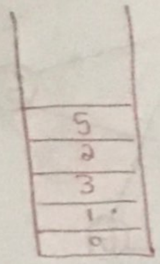


output:- 0 1 3 2 4

→ 

output:- 0, 1, 3, 2, 4, 6

- At this stage we don't have any unvisited vertex. So, used the technique of back tracking.
- Pop out 6 and check on 4 any visited vertex if yes move to that vertex if no then again pop the 4 and check again the unvisited vertex do the same step until we not found unvisited vertex.



output:- 0, 1, 3, 2, 4, 6, 5

→ Now again pop the all elements if all elements are pop then find the (DFS) means stop.

→ Again this is not the only DFS traversal.

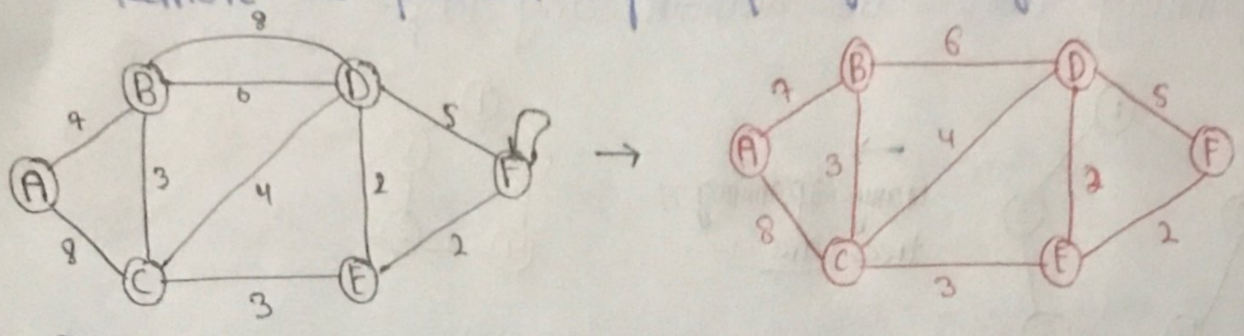
Scanned with CamScanner

Prim's Algorithim:-

- It is used to find the minimum spanning Tree.

Step 1:- To remove all the loops.

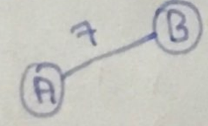
Step 2:- Remove the parallel edge of high weight.



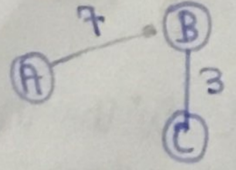
Step 3:- Choose any node as a root node.

I Choose. A

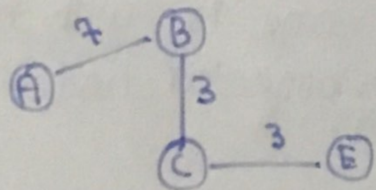
- After go on the both child of A and select with min. weight.



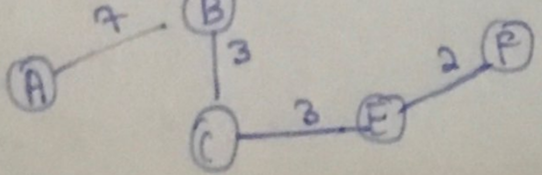
- Now check the edge of B with minimum weight.



- Now check the same steps for Ⓒ.



- Now Check on Ⓔ two nodes with minimum weight so select one according to your choice.

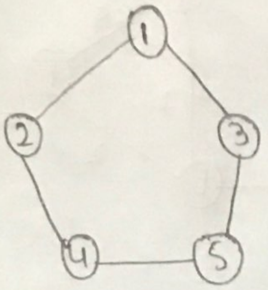


Minimum Spanning Tree:-

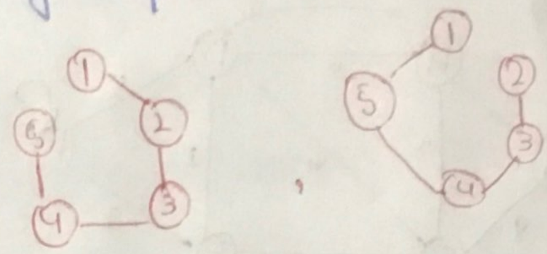
→ First what is spanning Trees.

- The graph represent as $G(V, E)$
- In spanning Graph represent as $G'(V', E')$, where

  $V' = V$ and $E' \subseteq E$. or $E' = |V| - 1$
- Spanning trees are without any loop.

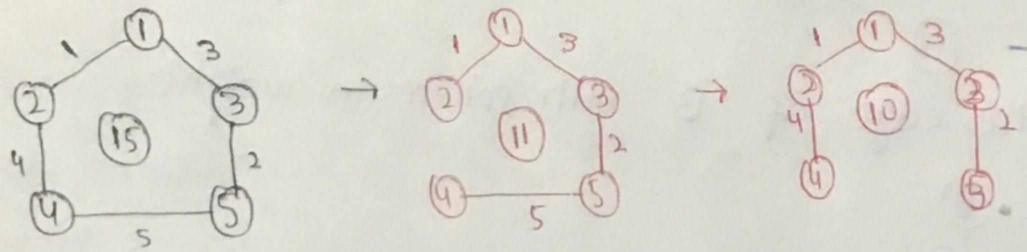


→ Many spanning trees are :-



→ Minimum spanning trees are trees having weight on edges.

→ The minimum spanning tree are the tree with minimum no. of cost.

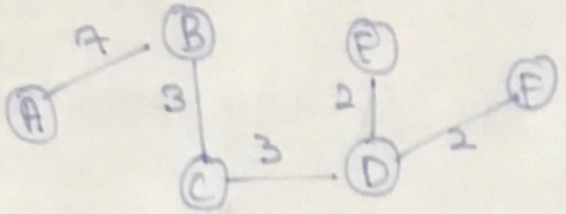


→ out of these this is the minimum spanning tree.

→ Construct Minimum Spanning Tree start with the minimum cost edge and move to the higher cost edge.

→ Two conditions for spanning tree one is with no cycle and the second is with no disconnected node.

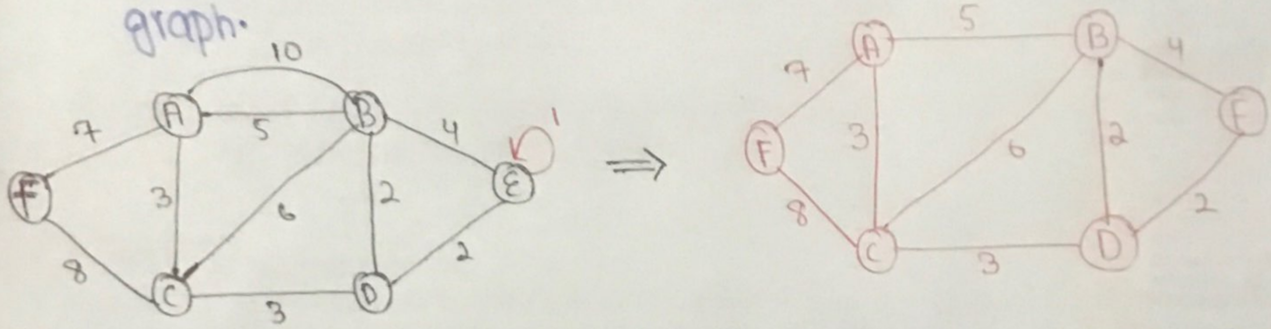
- Now check on F.
- But Before we compare all the previous edges and find the minimum unselected and them compare with F.



→ This is the Minimum spanning tree using Prim's Algorithm.

Kruskals Algorithm:-

- It is used to find the minimum spanning Tree.
- Step 1:-
  Remove all the loops and parallel edges from the graph.



- In kruskal's Algorithms arrange edges in increasing order
- We see the graph minimum weight is 2. (Note the edges).

  BD = 2
  DE = 2

- Now minimum weight is 3.

  AC = 3
  CD = 3

- For edge weight is 4.

  BE = 4

- 5 → AB = 5
- 6 → BC = 6
- 7 → AF = 7
- 8 → FC = 8

Step 3:-

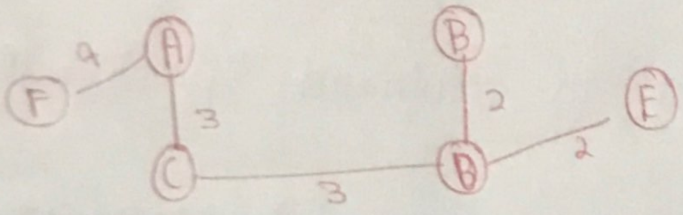
Choose minimum edge weight. and contain on graph.

Step 4:-

when you making graph make sure no loop/cycle make.

Step 5:-

Check and add one by one if edge contain cycle so simply leave the edge.



Note:-

→ This is minimum spanning tree of the given graph using Kruskal's Algo.