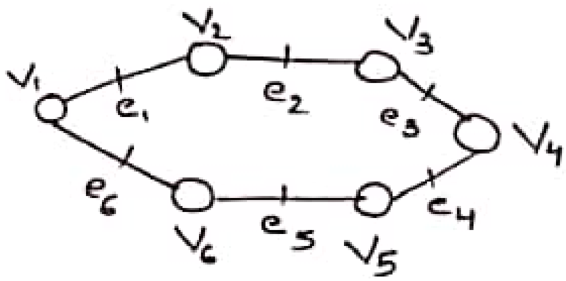
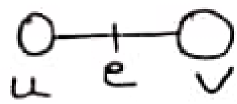
Unit-4

Graphs

A graph G consists of two things:-

1) A set V of elements called nodes (or points or vertices)
2) A set E of edges such that each edgee in E is identified with a unique (unordered) pair [u,v] of nodes in V, denoted by $\boxed{e=[u,v]}$





$e=[u,v]$

$$V=\{V_1, V_2, V_3 \ldots V_n\}$$
$$E=\{e_1, e_2, e_3 \ldots e_n\}$$

Suppose $e=[u,v]$, then nodes u,v are called end point of edge and u,v are called adjacent point.

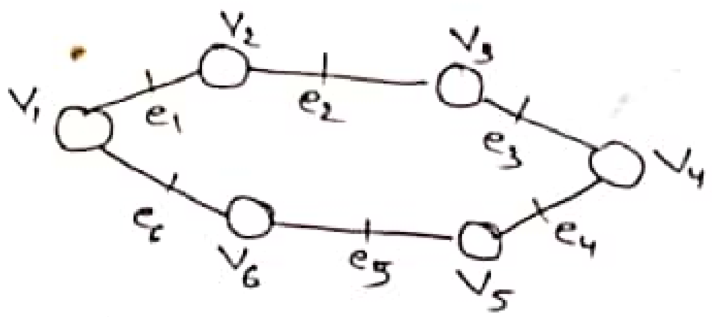
Application of Graph

1) Graphs are used to find shortest route.
2) Graphs are used to make an analysis of electrical circuit.

Types of Graph

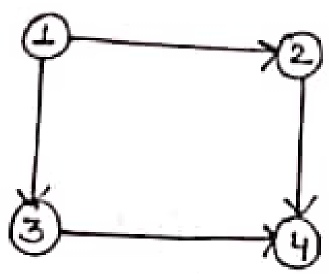
1) Undirected Graph
2) Directed Graph

i) Undirected Graph:- A graph which have unordered pair of vertices is called undirected graph.



[u,v] or [v,u]

ii) Directed Graph: It is a graph in which each edge is represented by an ordered pair of vertices or it is a graph in which each edge is assigned a direction.

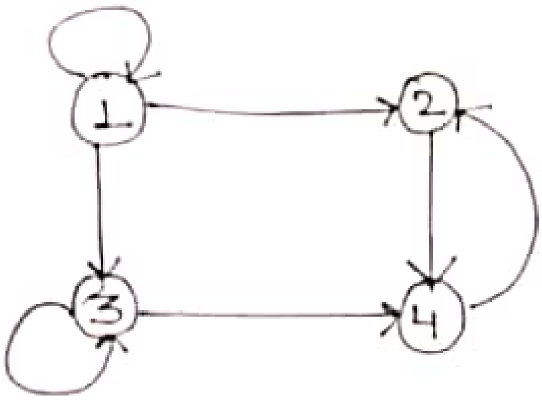


* In this case (1,2) & (2,1) different

(2,1) [No edge exist]

Multigraph :-

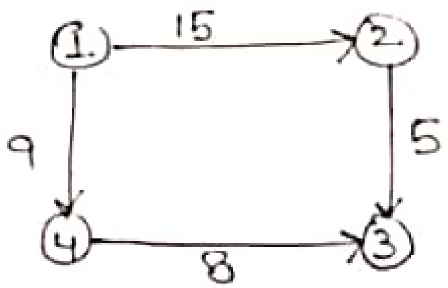
Multigraph contains multiple edges and loops.



b) Weighted Graph

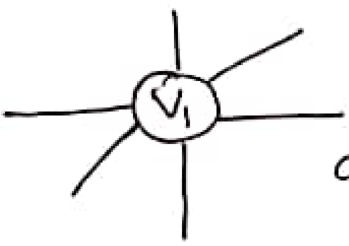
A graph is said to be weighted if every edge in the graph is assigned some non-negative numerical value as weight.

The weight may be the distance of the edges or the cost.



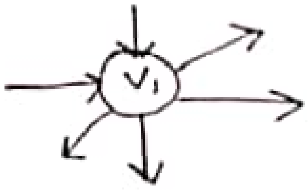
Properties of Graph

i) Degree of a node : Number of edges containing the node.



$degree(V_1) = 6$

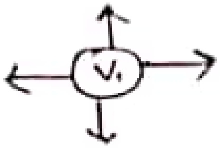
ii) Indegree and Outdegree :



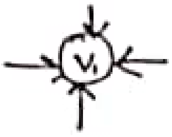
Indegree $(v_1) = 2$

Outdegree $(v_1) = 4$

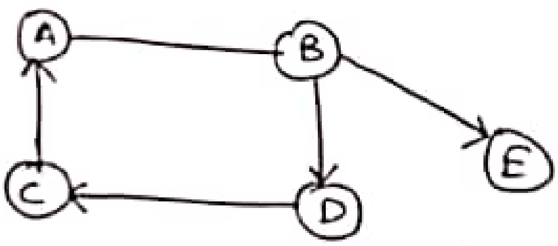
iii) Source vertex :- A vertex that has only source or that has indegree $= 0$



iv) Sink vertex :- A vertex that has outdegree $= 0$

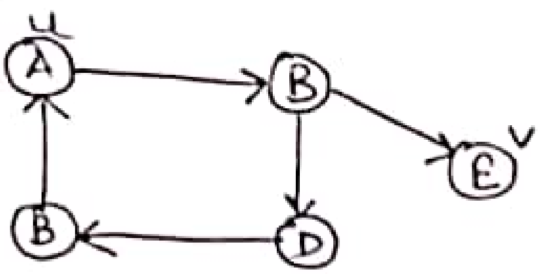


v) Pendant vertex :- It has its outdegree $= 0$ and indegree $= 1$.



$\Rightarrow$ E is pendant vertex.

(2) Path :- A path P of length n from node u to node v is defined as a sequence of $(n+1)$ nodes.



a) A path P is said to be closed if $V_0 = V_n$.

b) A path is said to be simple if all nodes are distinct.

Connected Graph

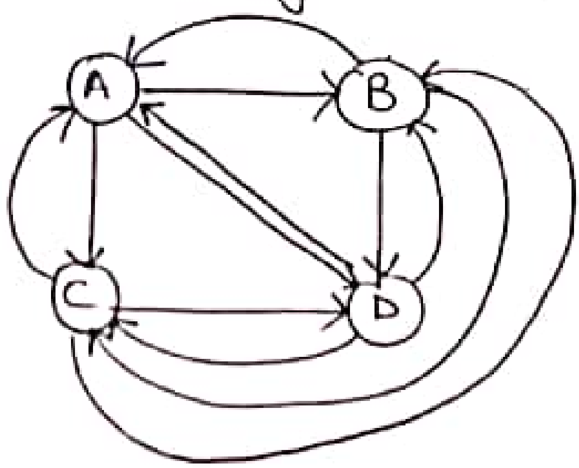
A graph is said to be connected graph iff there is a simple path b/w any two nodes in Graph 'G'. i.e., there is no isolated vertex.

Complete Graph

A graph G is said to be complete or fully connected if there is path from every vertex to every other vertex. A complete graph with n vertices will have $n(n-1)/2$ edges.

or

A graph is said to be complete if every node u in graph(G) is adjacent to every node v in graph (G).



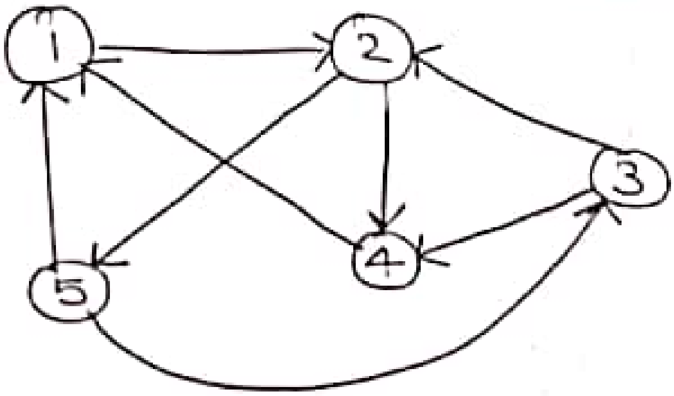
Memory Representation of Graph

i) **Sequential Representation**:- In Sequential Representation, we make use of 2-D array of order $n \times n$ where n is the total number of nodes in the graph.

Adjacency Matrix:-

Suppose G is simple directed graph with m nodes, and suppose the nodes of G have been ordered and are called $v_1, v_2, \ldots v_m$. Then, the adjacency matrix $A = (a_{ij})$ of Graph G is defined as:

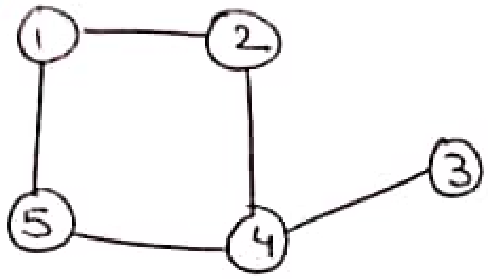
$$a_{ij} = \begin{cases} 1, & \text{if there is an edge from node } i \text{ to } j. \\ 0, & \text{otherwise.} \end{cases}$$



adjacency matrix

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

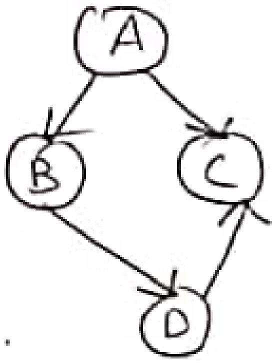
$m \times m$

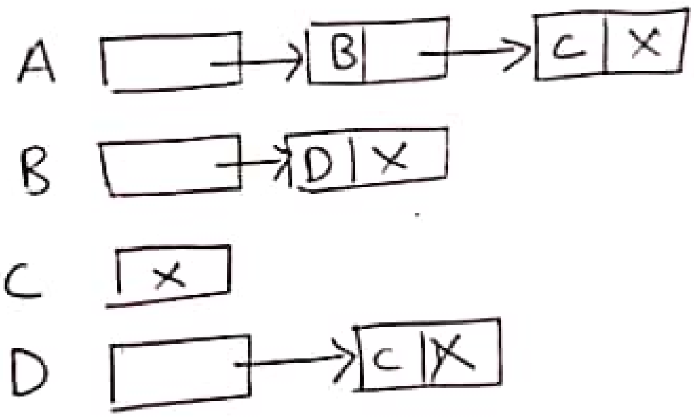


$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Linked representation :- It is by means of linked lists of neighbours.

Adjacency list :- In this representation of graphs, the n rows of adjacency matrix are represented as n linked lists.



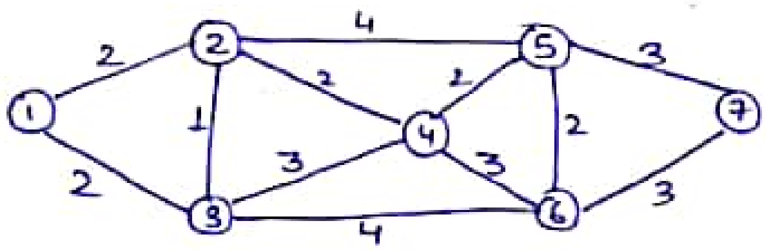


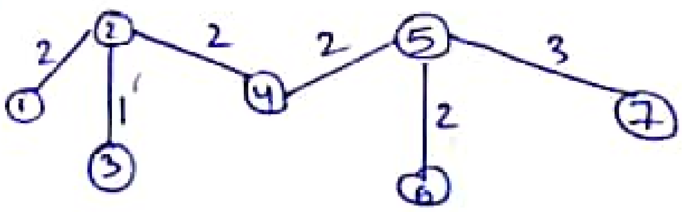
Kruskal's Algorithm

It finds the minimum cost spanning tree of graph G.

Steps :- /Algo.

1) Arrange all the edges in increasing order of their weight.
2) Add to MST the edge if it does not form a circuit.
3) Continue till all the edges are visited and an MST is formed.
4) Add the cost of all edges in MST to get the minimum cost of Spanning tree.





edge	Wght.	
(2,3)	1	✓
(1,2)	2	✓
(1,3)	2	✗
~~(4,5)~~		
~~(5,6)~~		
(2,4)	2	✓
(4,5)	2	✓
(5,6)	2	✓
(3,4)	3	✗
(4,6)	3	✗
(5,7)	3	✓
(6,7)	3	✗
(3,5)	4	✗
(3,6)	4	✗

=>

Prim's Algorithm

It also works like Krushkal algorithm except it applies nearest neighbour method to select new edge.

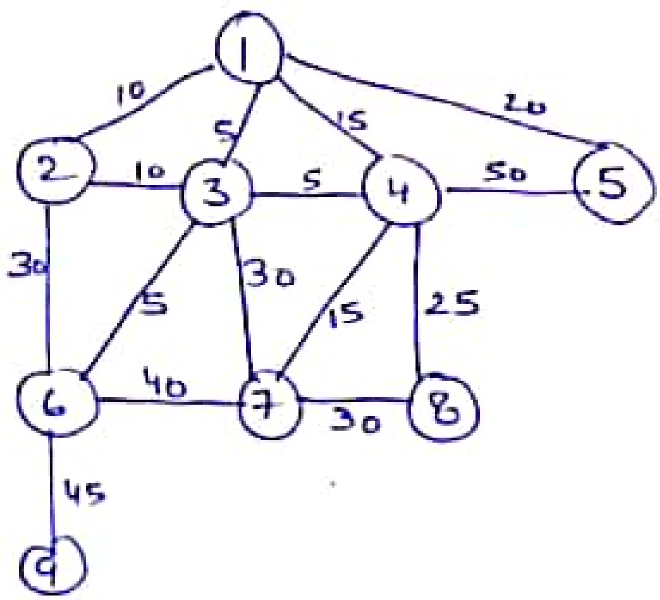
Steps:-

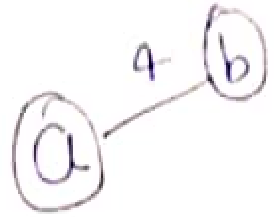
1.) Start with an vertex, say u.

2) Select another vertex, v, s.t. edge is formed from u and v and is of minimum wt., Connect uv and add it to set of for MST, edges A.

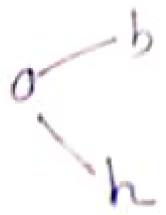
3) Now among the set of all vertices find other vertex V_i that is not included in A st. $(V_i\ V_j)$ is minimum labeled and is nearest neighbour of all vertices in set A and it does not form circuit add it to A.

4) Continue this process till you get a MST, MST obtained is of minimum cost.

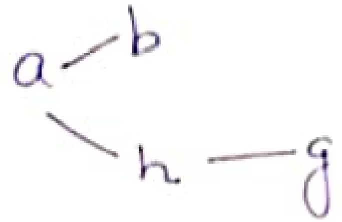


1) 

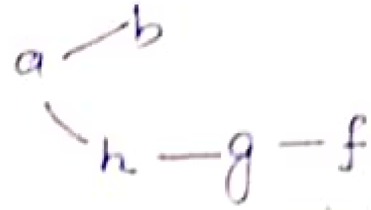
(ah, bh, bc) = ah
8 11 8

2) 

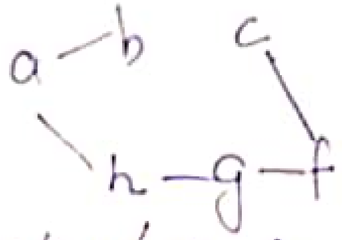
(bh, bc, hg, hi) - hg
11 8 1 7

3) 

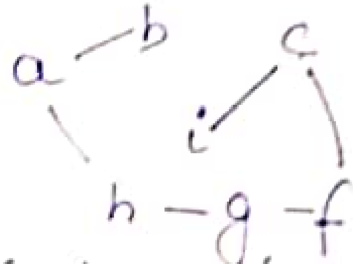
(bh, bc, hi, gi, gf.)
11, 8, 7, 6, 2

4) 

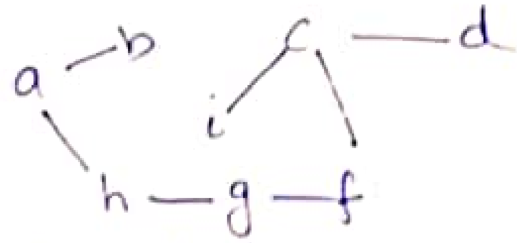
(bh, bc, hi, gi, fc, fd, fe)
8, 7 6 4 14 10

5) 

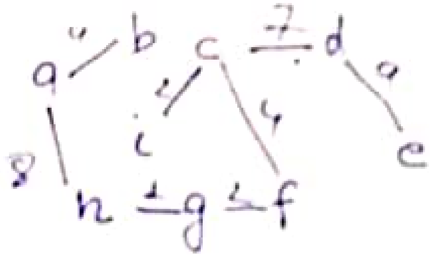
(hi, gi, fd, fe, cd, ci)
7 6 14 10 7 2

6) 

(fd, fe, cd,)
14, 10, 7

7) 

(fe, de)
10 9

8) 

= 37