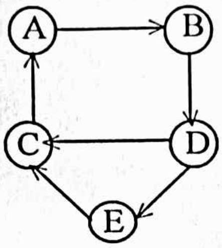
UNIT-IV
GRAPHS

Introduction:

Graph:- A graph is a collection of nodes and edges. (or) A graph is a pictorial representation of a set of objects. where pair of objects are connected by links. The objects are represented by vertices, and the links that connect the vertices are called edges.A graph can be represented by G ={V, E}.The represention of graph folows



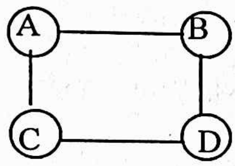
In the above graph A,B,C,D,E are called as vertices or nods.(A,B) (B,D) (D,C) (C,A) (D,E) (E,C) are called as edge.

G = (V,E)

V = set of nodes= {A,B,C,D,E}

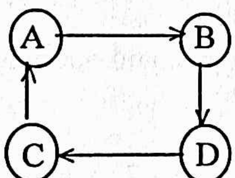
E = set of edge = {(A,B) (B,D) (C,A) (E,C) (D,E)(D,C)}

Undirected graph:- To draw a graph by using undirected edges is as undirected graph



Note:- In an undirected graph the edge (A,B) is same as (B,A)

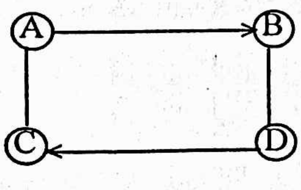
Directed graph: To draw a graph by using directed edge is called as directed graph.



Note- In directed graph the edge (A,B) is not same as (B,A)

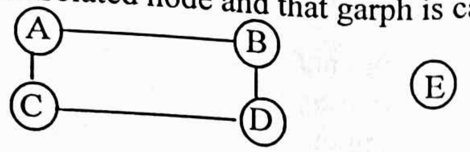
Mixed graph:- To draw a graph by using directed and undirected edges is called as mixed graph.

eg:



Isolated graphs:- In a graph, a node is not conected to any one of the node in a graph then that node is called as isolated node and that garph is called as isolated graph.

eg:



Null graph:-

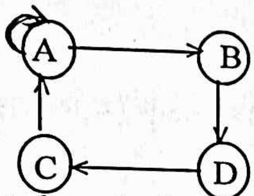
To draw a graph by using isolated nodes is called as null graph.

eg:

Loop:-

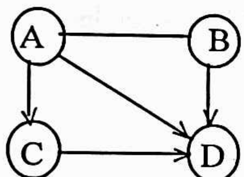
An edge in a graph starts and ends with the same node is called as loop.

eg:



Indegree:- In a directed graph, the indegree of a node is the number of edges that ends at the same node.

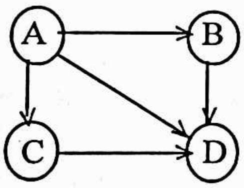
eg:



Ex:- The in degree of D is 3

Outdegree:- In a directed graph the out degree of a node is a number of edges start from that node.

eg:

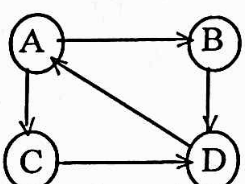


Ex:- The Out degree of A is 3

Total degree:-

In a directed graph, The sum of indegree and out degree of a node is called as total degree.

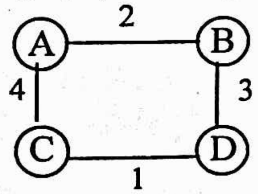
eg:



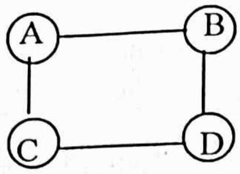
Ex:- The Total degree of A is 3

Weighted graph:- In a graph, each edge contain a number is called as weighted graph or labled graph.

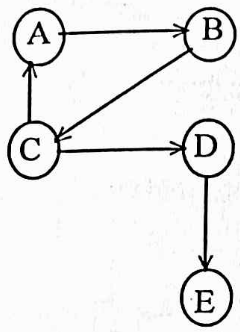
eg:



Connected Graph:-A connected graph is nothing but there is at least one edge from one node to another node in that graph



Path:- A path is a way to connect the nodes in sequntial order.



In the above graphs a path from Ato E is (A,B) (B,C) (C,D) (D,E)

Graph represention:-

Graph can be represented in two ways. They are

* Adjacency matrix
* Adjacency list

Adjacency matrix:-

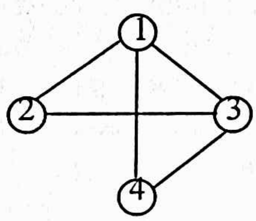
For a graph G={V,E} with n vertices, the adjacency matrix of G is the two dimentional nxn array

*Aij=1 if there is an edge from Ai toAj

*Aij=0 if there is no edge

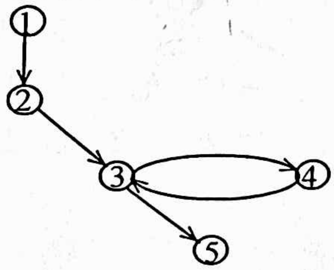
Undirected graph:

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



Directed graph:-

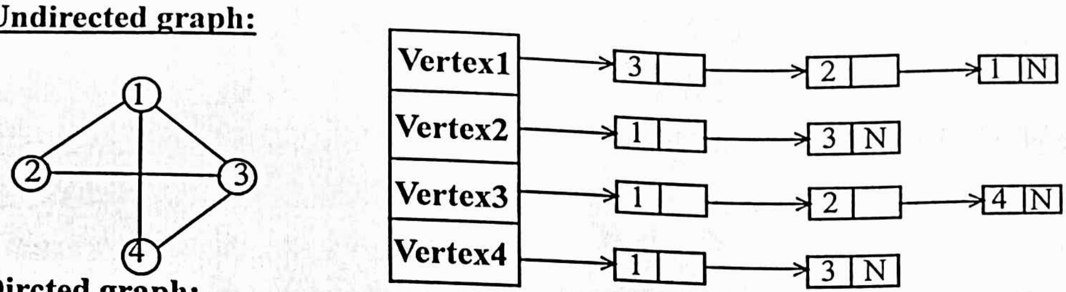
	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	0	0	1	0	1
5	0	0	0	0	0



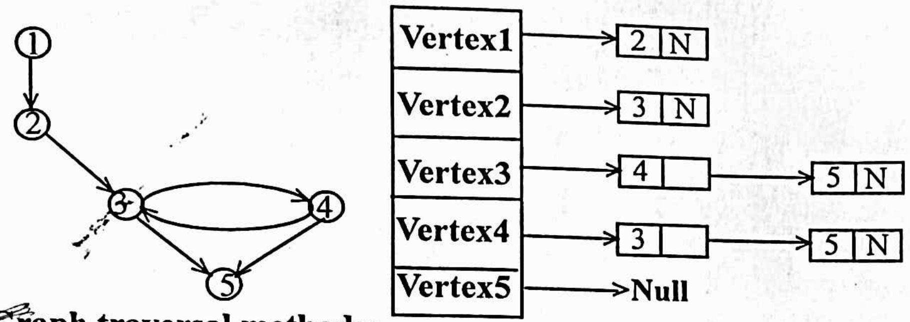
Adjacency list:-

In this represention the n rows of the adjacency matrix of G are represented as n linked list. there is one list for each vertex in the graph G. The nodes in the graph represents vertices that are adjacent from vertex. Each node has two fields.They are link and data fields.

Undirected graph:



Dircted graph:-



Graph traversal methods:-

Traversal means to visit all the nodes in the graph. Graphs supports two types of traversal methods. They are

* D.F.S (depth first serach)
* B.F.S (breath first seach)

D.F.S (depth first serach):-

D.F.S follows stack operation.it means last inserted element is first deleted . Most recently entered element treated as topmost element and all nodes arrranged in depth wise manner.

Algorithm D F S :

STEP 1: start

STEP 2: begin with top most node and push into the stack

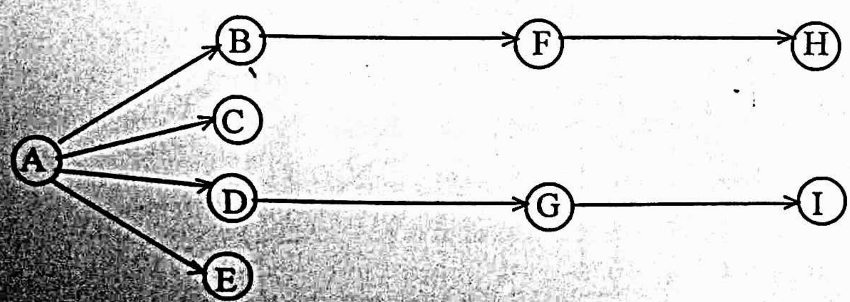
STEP 3 : repeat the steps 4 and 5 untiill the stack is empty.

STEP 4 ; POP in the top most node from the stack and print it.

STEP 5 : Push the adjacent nodes of the popped element into the stack

STEP 6 : Stop

Eg:



STEP 1 : Push A into the stack

D F S ; Empty

A

STEP 2 : pop A and print A, ans also push the adjacent nodes of A into the stack

D F S : A

E
D
C
B

STEP 3 : pop E and print E and also push the adjacent nodes of E into the stack

D F S : A,E

D
C
B

STEP 4 : pop D and print D and also push the adjacent nodes of D into the stack.

D F S : A,E,D

G
C
B

STEP 5 : pop G and print G and also push the adjacent nodes of G into the stack

D F S : A ,E, D, G

I
C
B

STEP 6 : pop I and print I and also push the adjacent nodes of I into the stack

D F S : A E D G I

C
B

STEP 7 : pop C and print C and also push adjacent nodes of C into the stack.

D F S : A E D G I C

B

STEP8 : pop B and print B and also push the adjacent nodes to 'B' to the stack

D F S : A E D G I C B

F

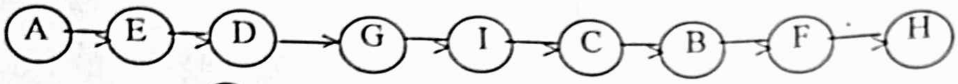
STEP 9 : pop F and print F and also push the adjacent nodes to 'F' to the stack

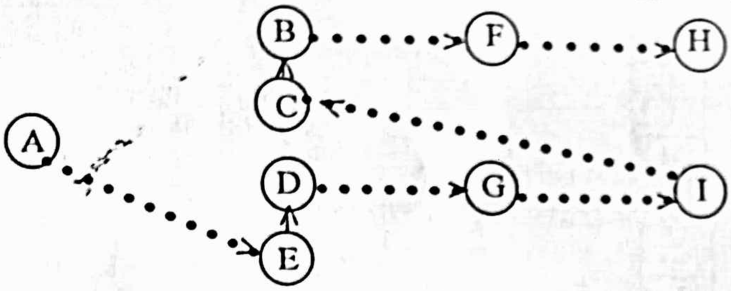
DFS: AEDGICBF

STEP 10 : pop H and print H and also push the adjacent nodes to 'H' to the stack.

DFS:AEDGICBFH

Now Stack is empty.The result of D F S traversing is





Applications of DFS:-

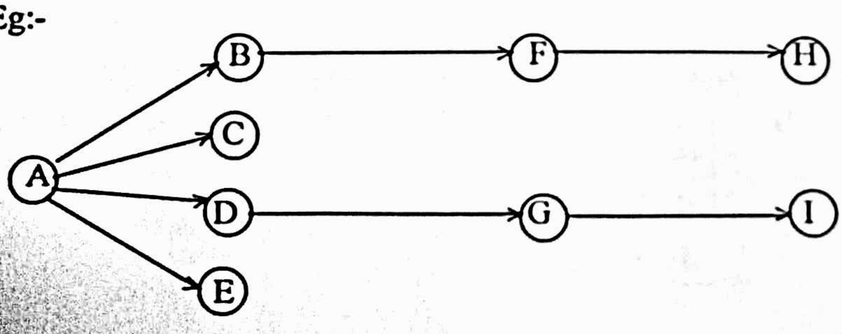
1.Finding a path between two specified nodes , u and v of an unweighted graph
2.Finding a path between two specified nodes , u and v of an weighted graph.
3.Finding wheather a graph is connected or not.
4.Computing the spanning tree of a connected graph.

<u>B F S (Bredth first search):</u> B F S traversal can follow queue operation it means first in first out (FIFO). In this metod elements are inserted at one end and delete the elements at another end. In this traversal elements are traversed in breath wise manner.

<u>Algorithm</u>:

STEP 1 : Start
STEP 2 : Begain at first node and insert that node into the queue
STEP 3 : repete the steps 4 and 5 untill the queue is empty
STEP 4 : delete first node and print it
STEP 5 : Insert adjacent nodes of a delete node into the queue

Eg:-



STEP 1 : Insert A into the queue
B F S: empty

Delete ←	A				← Insert

STEP 2 : Delete node A and print it and also insert all the adjacent nodes of A
B F S : A

Delete ←	B	C	D	E	← Insert

STEP 3 : Delete node B and print it and also insert all the adjacent nodes of the B.
B F S : A,B

Delete ←	C	D	E	F	← Insert

STEP 4 ; Delete node C and print it and also insert all the adjacent nodes of C
B F S : A,B,C

Delete ←		D	E	F	← Insert

STEP 5 : Delete D and print it also insert all the adjacent nodes of D
B F S : A, B, C,D

Delete ←	E	F	G		← Insert

STEP 6 : Delete E and print it also insert all the adjacent nodes of E
B F S A, B,C, D,E

Delete ←	F	G			← Insert

STEP 7 : Delete F and print it also insert all the adjacent nodes of F
B F S : A, B,C,D,E,F

Delete ←	G	H			← Insert

STEP 8 : Delete G and print it also insert all the adjacent nodes of G
B F S : A,B,C,D,E,F,G

Delete ←	H	I			← Insert

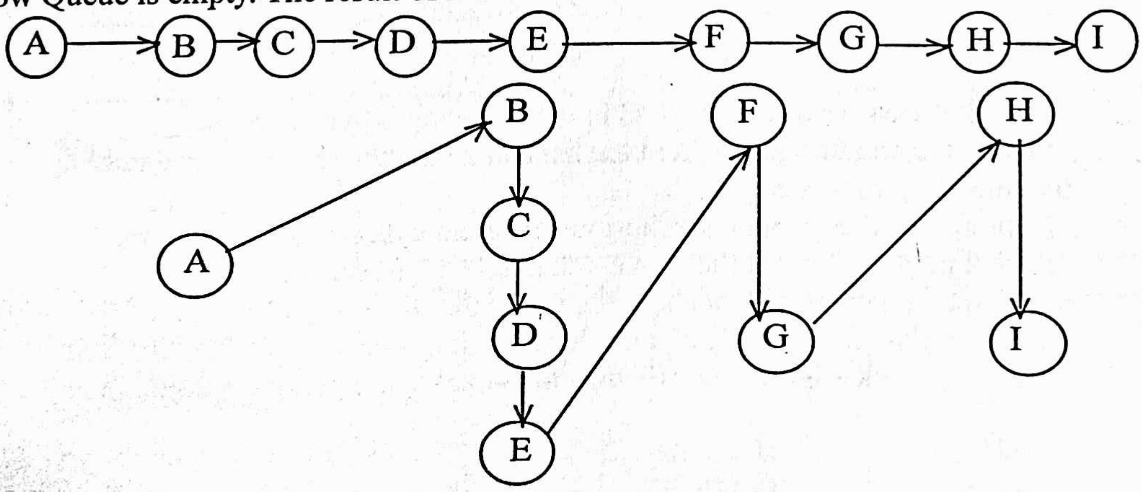
STEP 9 : Delete H and print it also insert all the adjacent nodes of H
B F S : A,B,C, D,E,F,G,H

Delete ←	I				← Insert

STEP 10 : Delete I and print it also insert all the adjacent node of I
B F S : A,B,C,D,E,F,G,H,I

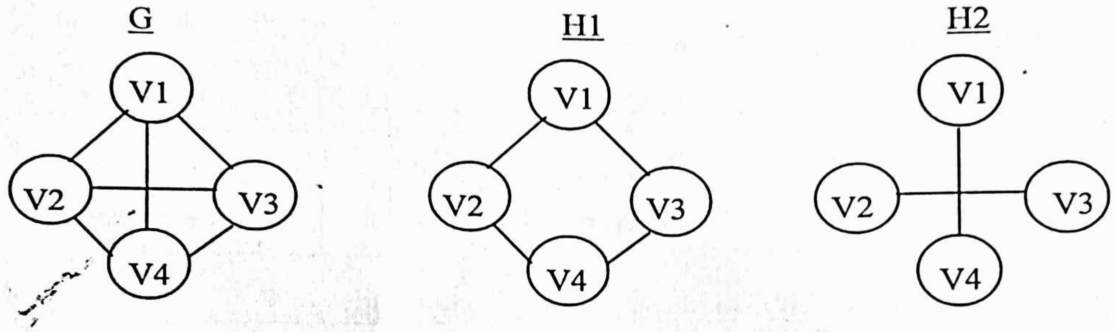
Delete ←					← Insert

Now Queue is empty. The result of B F S is



Applications of BFS:-
BFS can be used to solve many problems such as
1. Finding all connected components in a graph G
2. Finding all nodes with in an individual connected components.
3. Finding the shortest path between two nodes u and v , of an unweighted graph.
4. Finding the shortest path between two nodes u and v , of an weighted graph.

Spanning trees:- "A spanning tree H is a subset of graph G, which has all the vertices covered with minimum possible number of edges" (or) "it is a connected graph using all vertices in which there is no cycle"



These two H1 and H2 are the spanning trees of graph G.

Minimum spanning tree:- A minimum spanning tree is a spanning tree that has the minimum weight than all other spanning trees of the graph" (or)

Minimum sapnning tree of an undirected graph G is a tree formed from graph edges that connects all the vertices of graph G at lowest total cost.

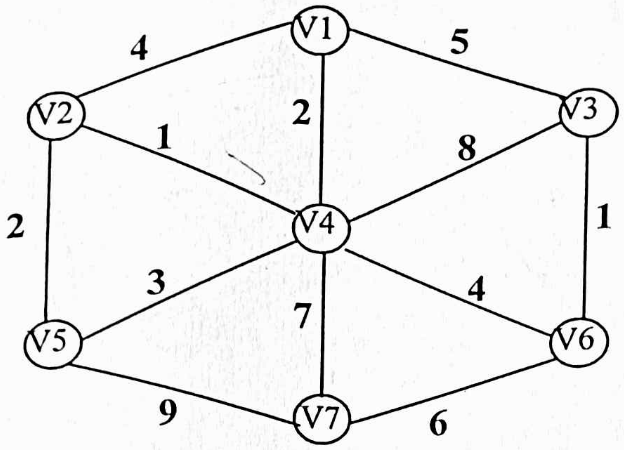
There are several methods available for finding minimum spanning trees of a graph. Out of them two methods are known to be very effected.They are

* Kruskal's algorithm and
* Prims algorithm

Kruskal's algorithm: Let as assume an undirected weighted graph with 'n' vertices where initially the spanning tree is empty

Algorithm:

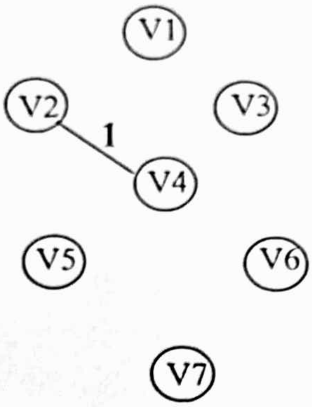
STEP 1 : List all the edges of the graph G in the incresing order of weights
STEP 2: Slecte the smallest edge from the list and it into the spanning tree initially it is empty if the including of this edge.
STEP 3 : If the selected edge with smallest weight from cyle remove it from the list.
STEP 4 : Repete step 2 - 3 untill the area cantians n-1 edge (or) list is empty.
STEP 5: If the tree cantains use then n-1 edges and the list is empry no spanning tree os possible for the graphs else return the minimum spanning tree
The following diagram shows the entire information about the kurskal's algorithem



EDGE	WEIGHT	SELECTON
V2 ⟶ V4	1	✓
V3 ⟶ V6	1	✓
V2 ⟶ V5	2	✓
V1 ⟶ V4	2	✓
V4 ⟶ V5	3	X
V1 ⟶ V2	4	X
V4 ⟶ V6	4	✓
V1 ⟶ V3	5	X
V6 ⟶ V7	6	✓
V4 ⟶ V7	7	X
V4 ⟶ V3	8	X
V5 ⟶ V7	9	X

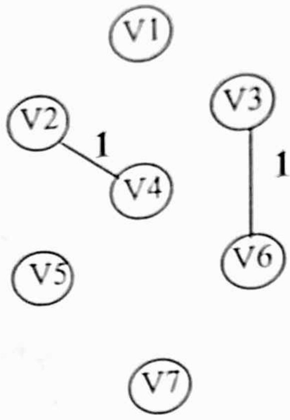
Let us consider an example for the instruction of the above algorithm for a graph G . A list of edges and there weights are maintain. Form this list we have select (✓) OR reject (X) depending on either from a cycle Or The length of a spannning tree as option can be caluculate as .

STEP 1:



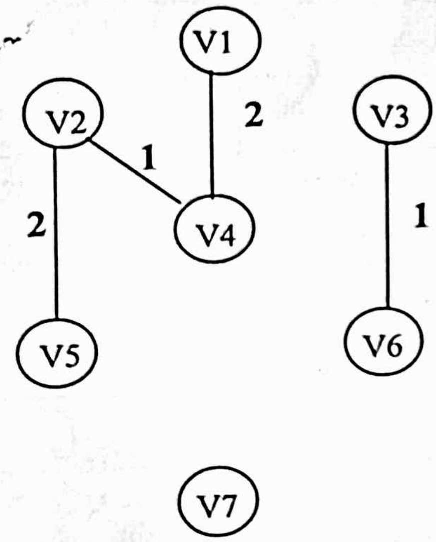
No cycle is form so consider the the edge v2 ans v4

STEP 2 :



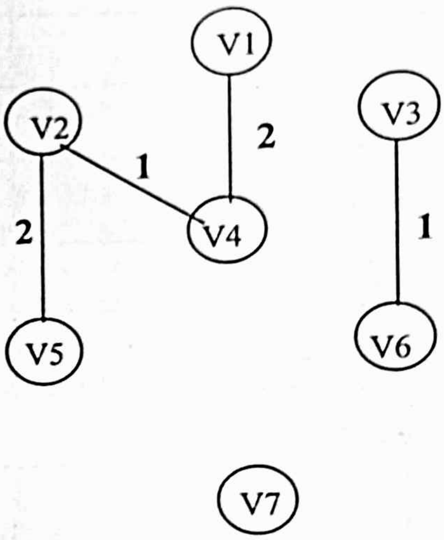
No cycle is form so consider the edge v3 and v6

STEP 3 :



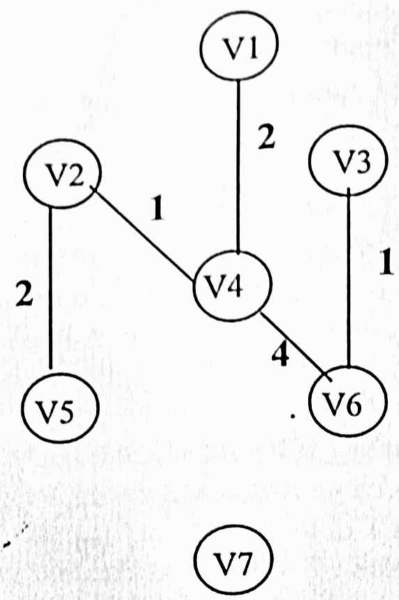
no cycle is form so consider the edges (V2 , V5) , (V1 , V4)

STEP 4 :



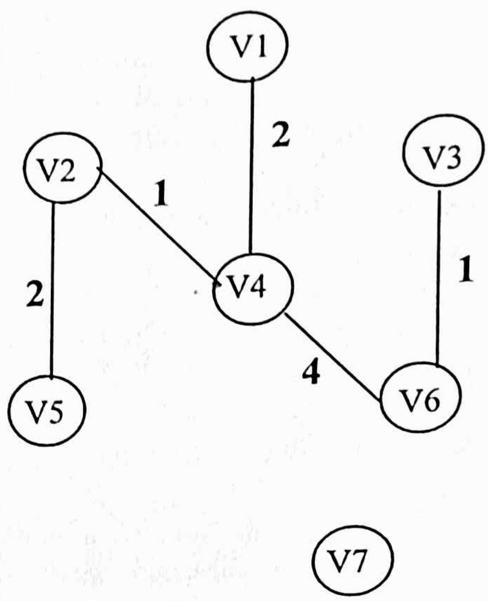
The cycle is form with the edges (V4 , V5) and (V1 , V2) so do not consider the edges

STEP 5 :



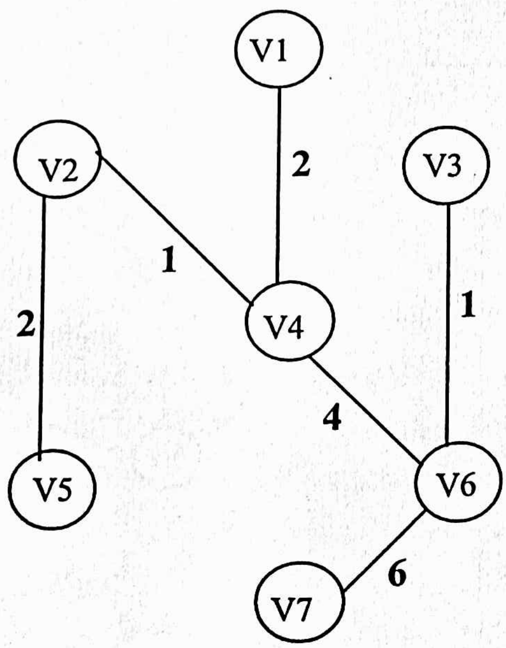
No cycle form with the edge (V4,V6) is considerd

STEP 6 ;



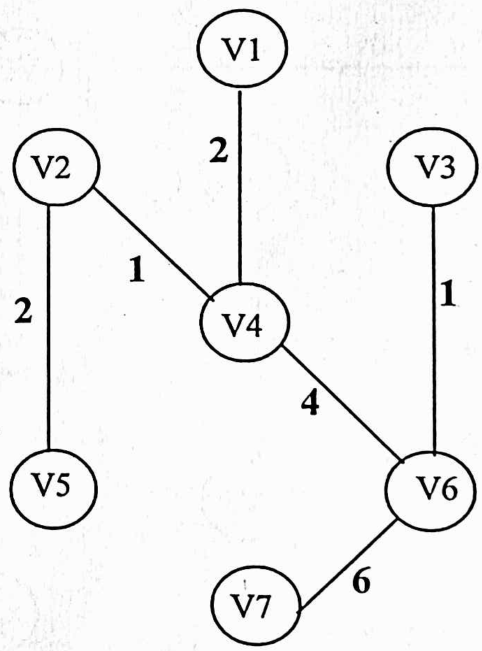
The cycle is form with the edge (V1 , V3) so we need not consider the edge

STEP 7 :



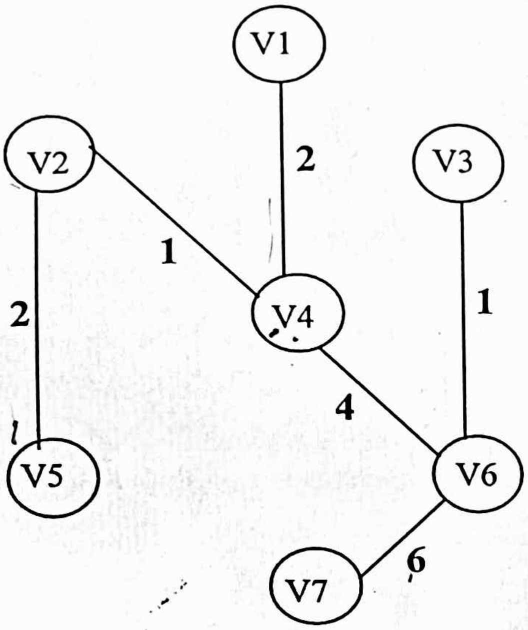
No cycle is form with the edge (V6 , V7) so consider that the edge

STEP 8 :



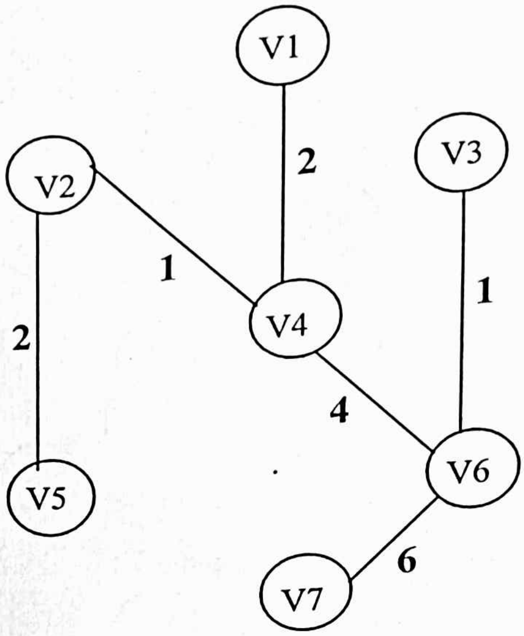
Cycle is form with the edge (V4 , V7) so we need not consider the edge .

STEP 9 :



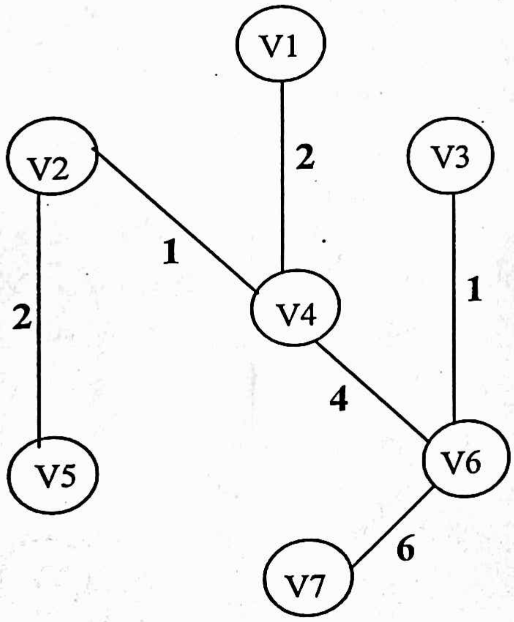
cycle form with the edge (V4 , V3)
so not consider edge

STEP 10 :



The cycle is formed with the edge (V5 ,V7)
so we need not consider the edge

Result: (V1 V4) + (V2 V4) + (V2 V5) + (V4 V6) + (V7 V6) + (V3 V6)
Result : 2+1+2+4+6+1=16



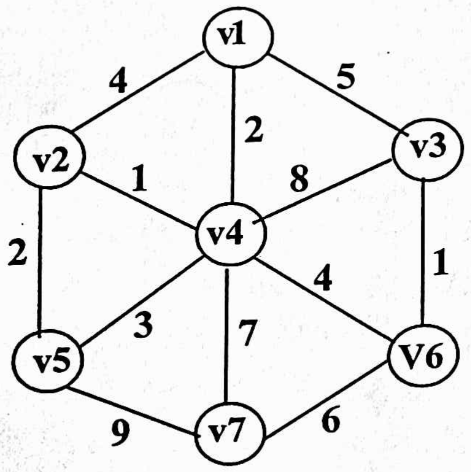
Prims algorithm:

According to the prims alogrithm, minimum spanning tree grows in successive stages at any stage in the algorithm.

The prims algorithm can easly be implemented using the adjacency matrix represention of a graph. Suppose there is an n x n adjancey matrix for a given undirected weighted graph and vertices are labled as V1,V2,V3 ....Vn. Start from vertex V1 say get it's nearest neibhour that is the vertex which has the smallest entry in the row of vertex V1.

Let it be Vi .Now consider V1 & Vi as one sub graph .Next obtain the closest neighbour of this sub graph that is vertex other than V1 & Vi that has the smallest entry among all enries in the rows of V1 &Vi. Let this new vertex be Vj therefore the tree now grows with vertex V1 & Vi & Vj as one sub graph.Continous this process untill the all vertices have been conneted with n-1 edges. The following is the algorithm

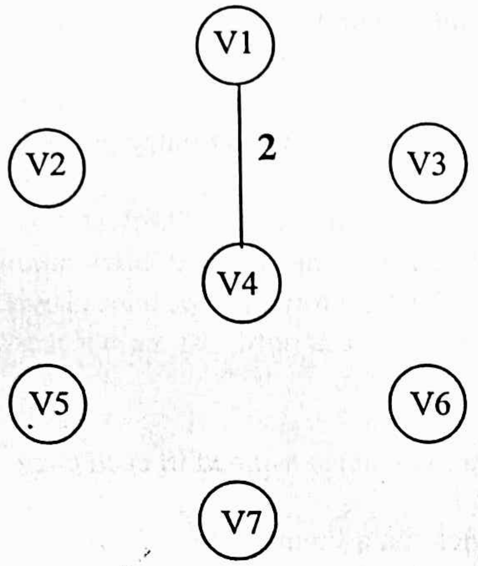
STEP 1 : Let G be a connected graph with non negative values asigned to each edge.
STEP 2 : Let T be the tree consiting of any vertex V1
STEP 3 : Among all the edges not in T that are incident on a vertex T and do not from a circuit and is added to T . Select one of the minimum cost and adding to T
STEP 4 : This process terminates that is step 3 repeted untill n-1 edges are selected. Where ' n ' is no. of vertices in G



Adjencey Matrix

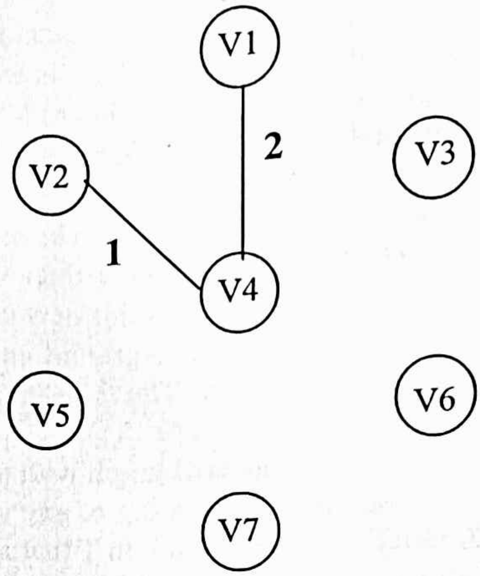
	v1	v2	v3	v4	v5	v6	v7
v1	-	4	5	2	-	-	-
v2	4	-	-	1	2	-	-
v3	5	-	-	8	-	1	-
v4	2	1	8	-	3	4	7
v5	-	2	-	3	-	-	9
v6	-	-	1	4	-	-	6
v7	-	-	-	7	9	6	-

STEP 1 :



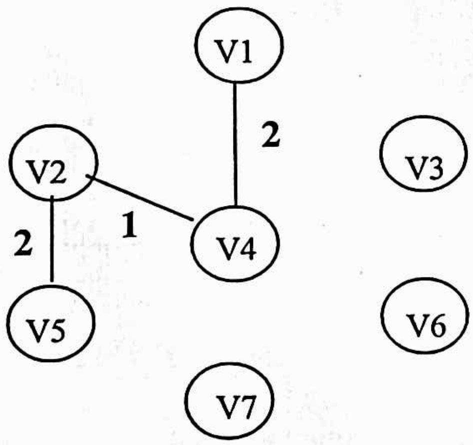
Set (V1,V4)

STEP 2 :



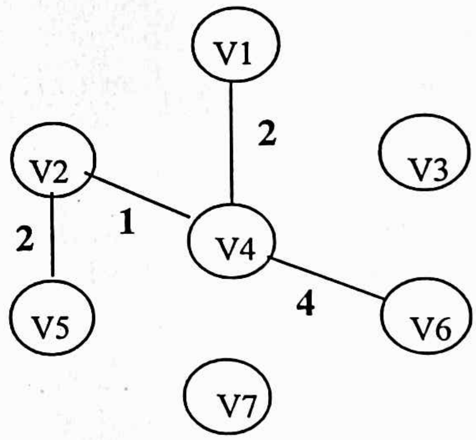
Set (V1,V4,V2)

STEP 3 :



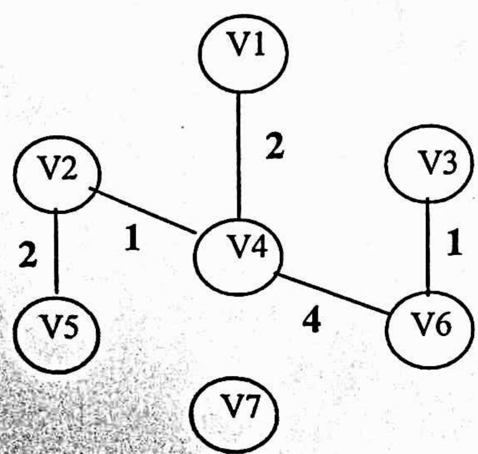
Set (V1,V4,V2,V5)

STEP 4 :



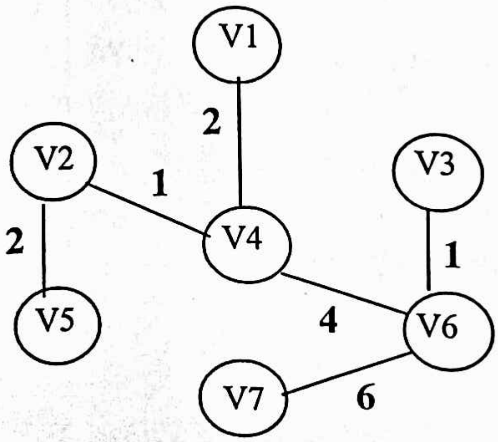
Set (V1,V4, V2, V5,V6)

STEP 5 :



Set (V1,V4,V2,V5,V6,V3)

STEP 6:



Set (V1,V4,V2,V5,V6,V3,V7)

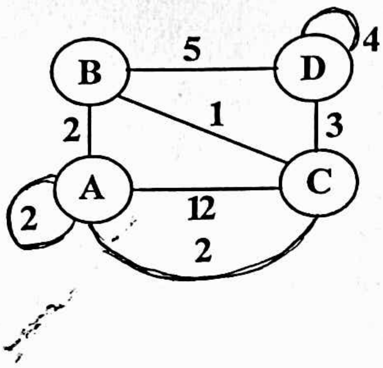
The resultent span tree are prims algorithenm is

(V2—> V5) + (V2—> V4) + (V1—> V4) + (V4—> V6) + (V3—> V6) + (V6—> V7)

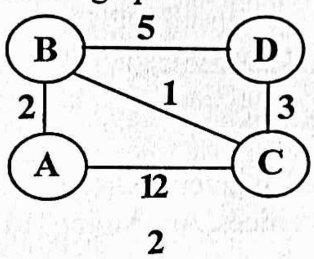
2+1+2+4+1+6=16

We can not consider the edges (V4—>V7), (V4—> V3),(V5—>V7),(V1—>V2) because the circute is form so we can not consider the edges

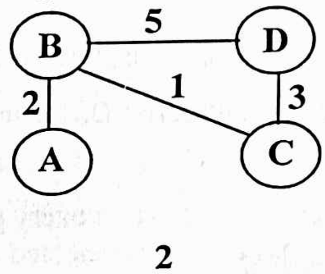
* *To find the shortest path for the given graph using krushkal algorithm.*



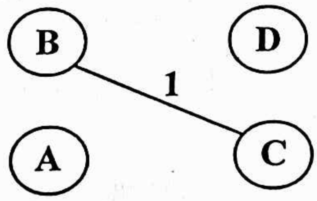
Step 1:- Remove all loops in the graph



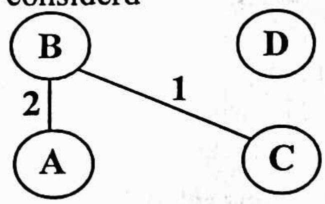
Step 2:- Remove parllel Edge of Max weight



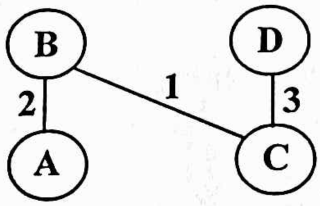
Step 3:- No cycle form with the edge (B,C) is considerd



Step 4:- No cycle form with the edge (B,A) is considerd



Step 5:- No cycle form with the edge (C,D) is considerd



Result : 2+1+3=6

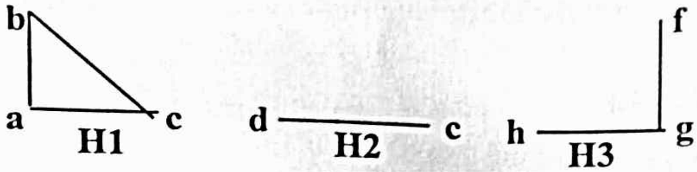
<u>Appication of minimum spannning tree</u> :

- Network desiging
- Telephone communication, electrical, t.v cable, computer etc......

Connected components :-

The connected components graph 'G' is a connected sub-graph of 'G' i.e.., not a proper sub-graph of another connected graph of 'G'. A graph 'G' that is not connected has two (or) more connected components that are disjoint and have 'G' as their unions.

E.g.:- The graph 'H' is the union of three disjoint sub-graph H_1,H_2,H_3 none of these which are proper sub-graphs of a larger connected sub-graph of 'G'. These 3 sub-graphs are the connected components of 'H'.



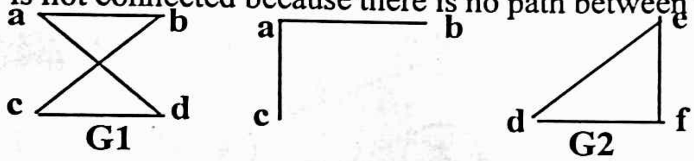
Different Types of Connected componenets-

- **Connected Undirected Graph:-**

An Undirected Graph is called connected with there is a path between every pair of vertices. An Undirected Graph that is not connected is called disconnected.

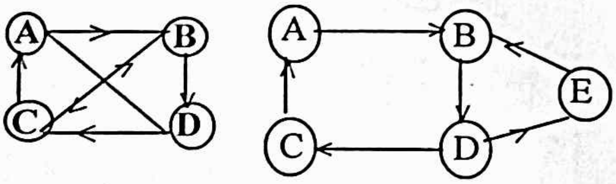
E.g.:- 'G_1' is connected because there is a path between any pair of it's vertices.

However 'G_2' is not connected because there is no path between vertices 'a' and 'f'

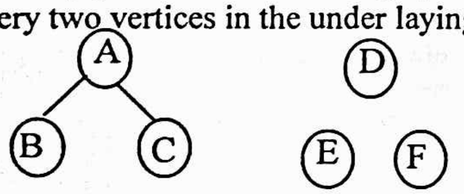


Connected Direct Graph:-There are two types

1.Strongly connected component:-A directed grapg is strongly connected if there is a directed path from a vertex to every other vertex.A directed graph can be borken down into strongly connected components.



2.Weakly connected components:- A directed graph is weakly connected if there is a path between every two vertices in the under laying undirected graph.



The above graph is not directly connected since there is no directed path