Chapter-2

Elementary Graph operation

graph traversal methods:-

traversal means to visit all the nodes in the graph graph supports two types of traversal methods they are

* DFS (Depth first search)
* BFS (Breath first search)

DFS (depth first search):- (LIFO)

DFS follows stack operation it means last inserted element first deleted i.e; most recently entered element treated as top most element and all nodes arranged in depth wise manner

Algorithm DFS (v)

Step1 : Start

Step2 : begin with topmost node and push into the stack.

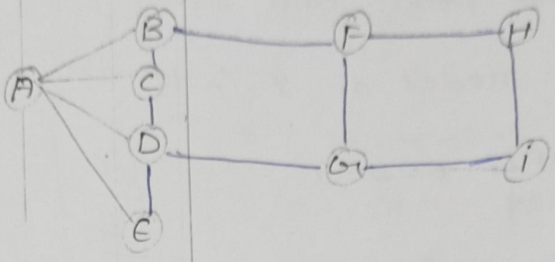
Step3 : repeat the steps 4 and 5 until the stack is empty.

Step4 : pop the top most node after stack and print it

Step5: push the adjacent nodes of the popped element into the stack.

Step6: stop()

Eg:-

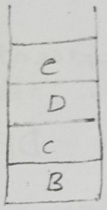


Step1: push A into the stack

DFS: empty

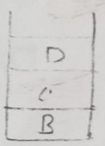
Step2: pop A and print 'A', and also push the adjacent nodes of 'A' into the stack

DFS: A



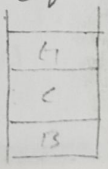
Step3: pop 'e' and print e and also push the adjacent nodes of 'e' into the stack.

DFS: AE



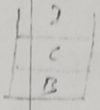
Step-4: pop D and print and also push the adjacent nodes of 'D' into the stacket

DFS: AED



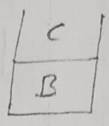
Step 5: pop G and print and also push the adjacent nodes of 'G' into the stack

DFS: A C D G



Step 6: pop I and print and also push the adjacent nodes of I in to the stack.

DFS: A C D G I



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

DFS: A C D G I C

Step 8: pop B and print and also push the adjacent node to 'C' to the stack.

DFS: A C d g I C B

Step 9: pop F and print and also push the adjacent node to 'F' to the stack

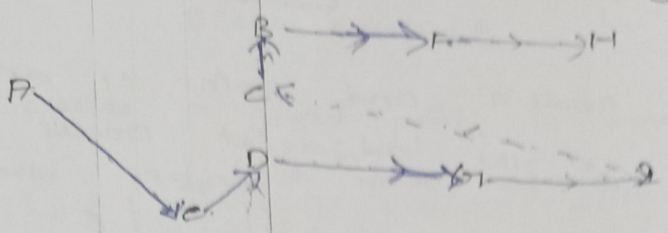
DFS: A e d g I C B F

Step 10: pop H and print and also push the adjacent node to H to the stack

DFS: A C d g I C B F H

The result of DFS traversing is

A -> E -> d -> g -> 2 -> c -> B -> F -> H



B.F.S (Breath first search)

BFS traverse can follow queue operation it means first in first out (FIFO) in this method. elements are inserted at one end and deleted another end. In this traversal elements are traversed in Breath wise manner.

Algorithem:-

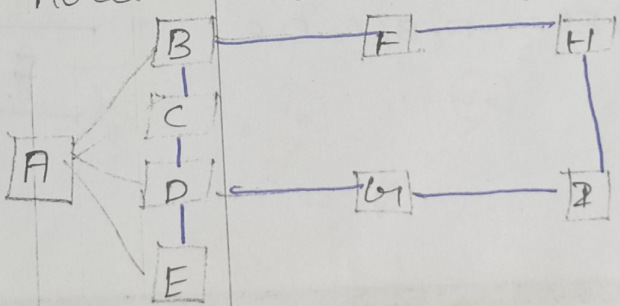
Step1: Start

Step2: begin at first node and insert that node into the queue

Step3: repeat the steps ④ and ⑤ until the queue is empty.

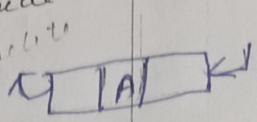
Step4: delete first node and print it

Step4: Insert adjacent nodes of a deleted node into the queue.



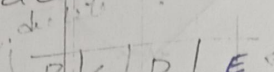
Step1: Insert 'A' into the queue

BFS: empty



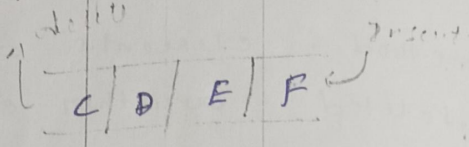
Step2: Delete node 'A' and print it and also Insert all the adjacent nodes of 'A'

BFS: A



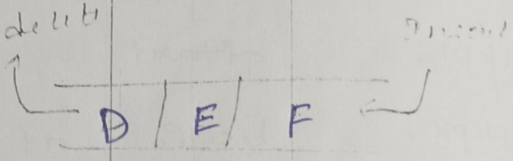
Step3: Delete node 'B' and print it and also Insert all the adjacent nodes of B.

BFS: A, B



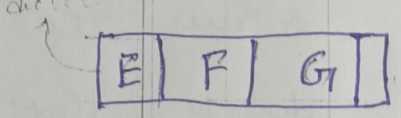
Step4: Delete node 'C' and print it and also Insert all the adjacent nodes of 'C'

BFS: A, B, C



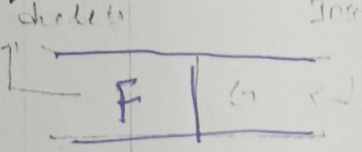
Step5: Delete 'D' and print it and also Insert all the adjacent nodes of 'C'

BFS: A, B, C, D



Step6: Delete 'E' and print it and also Insert all the adjacent nodes of E

BFS = A, B, C, D, E



Step7: Delete 'F' and print it and also insert all the adjacent nodes of F

BFS = A B C D E F

delete → G | H ← Insert

Step8: Delete 'G' and print it and also insert all the adjacent nodes of 'G'

BFS = A B C D E F G

delete → H | I ← Insert

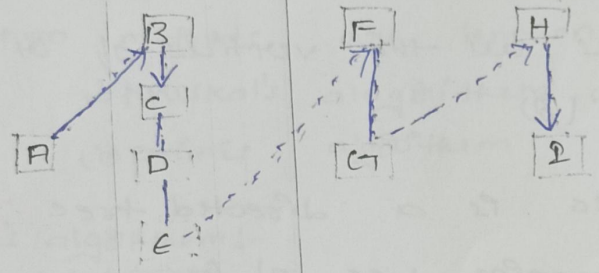
Step9: delete H and print it and also insert all the adjacent nodes of H

BFS = A B C D E F G H

I

Step10: delete I and print it and also insert all the adjacent nodes of I.

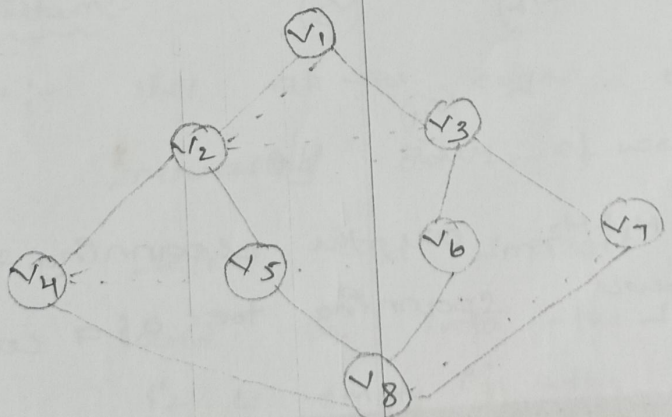
BFS : A B C D E F G H I



The result of BFS is

A -> B -> C -> D -> E -> F -> G -> H -> I

Eg:-



BFS = empty

BFS = v_1

BFS v_1 v_2

BFS : v_1 v_2 v_3

BFS v_1 v_2 v_3 v_4

BFS v_1 v_2 v_3 v_4 v_5

BFS v_1 v_2 v_3 v_4 v_5 v_6

BFS v_1 v_2 v_3 v_4 v_5 v_6 v_7

BFS v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8

Spanning tree:-

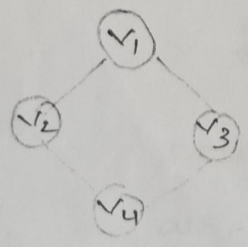
The sub graph H of a graph G is called a spanning tree H if

a) H is a tree and

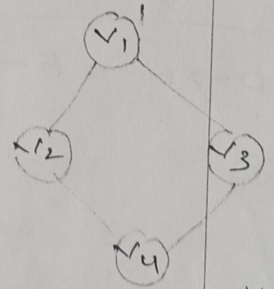
b) H contains all the vertices of G

(8)

A spanning tree is a directed tree is called a directed spanning tree of G.



H1



H2

These two H_1 & H_2 are the spanning trees of group H to produce spanning tree of a connected

graph we use methods called spanning of graph.

(81)

The spanning tree of a graph G can be defined as a tree which includes all the vertices of 'G' in during the discussion

Minimum spanning tree:-

minimum spanning tree of an undirected graph 'G' is a tree formed from graph edges that connects all the vertices of graph at lowest total cost. A minimum spanning tree exist if and only if G is connected.

There are several methods available for finding minimum spanning tree of a graph out of them two methods are known to be very efficient

(i) Kruskal's alogorithem and
(ii) prim's algorithem

Kruskal's algorithem:-

let us assume an undirected weighted graph with 'n' vertice where initially the spanning tree is empty

Algorithem:-

Step-1:- list all the edges of the graph G in the increasing order of weight

Step-2:- select the smallest edge from the list and it into the spanning tree initially it is empty if the increasing of this edge

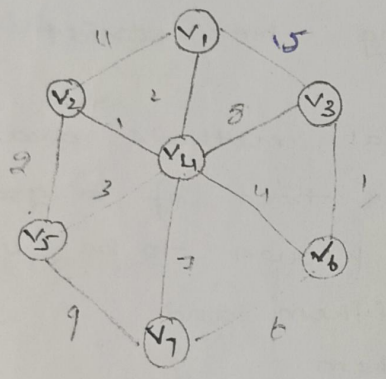
does not make a cycle

Step3:- If the selected edge with smallest weight from cycle remove it from the list.

Step4:- Repeat step ③-⑤ until the tree contains n-1 edges (or) list is empty.

Step5:- If the tree contains less than n-1 edges & the list is empty no spanning tree is possible for the graph else return the minimum spanning tree.

The following diagram shows the entire information about the Kruskal's algorithm.



Edge	weight	selection
$V_2 \rightarrow V_4$	1	✓
$V_3 \rightarrow V_6$	1	✓
$V_2 \rightarrow V_5$	2	✓
$V_1 \rightarrow V_4$	2	✓
$V_4 \rightarrow V_5$	3	×
$V_1 \rightarrow V_2$	4	×
$V_4 \rightarrow V_6$	4	✓
$V_1 \rightarrow V_3$	5	×
$V_6 \rightarrow V_7$	6	✓
$V_4 \rightarrow V_7$	7	×

$v_4 \longrightarrow v_3$

$v_5 \longrightarrow v_7$

8

9

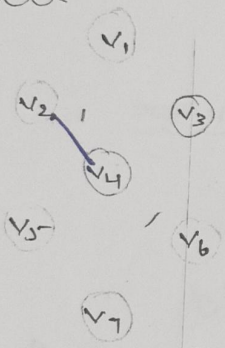
x

x

Introduction

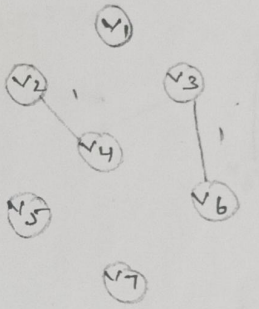
Let us consider an example for the working of the above algorithem for a graph from this list we have select (✓) (or) reject (x) depending on either a from a cycle (or) the weight of a spanning tree has option can be calculated has 16.

Step 1:-



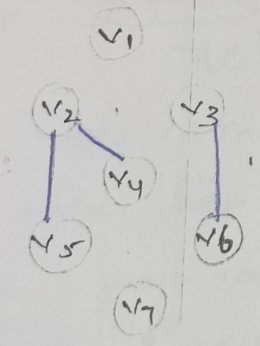
no cycle is from so consider the edge v_1 and v_4

Step 2:-



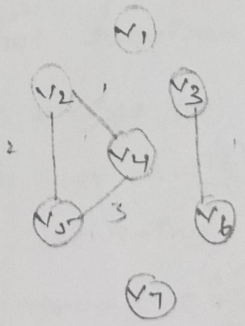
no cycle is from so consider the edge v_3 and v_6

Step: 3



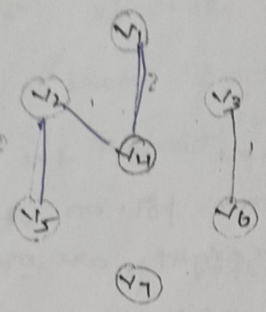
NO cycle is form so consider edge v_2 & v_5

Step: 4



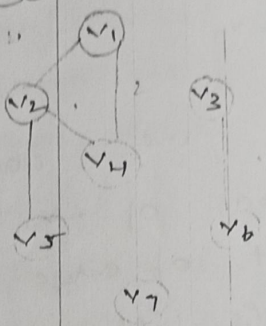
This cycle is from with edges v_4 & v_5 and v_2 & v_5 as v_2, v_4 so do not consider the edge.

Step 5:



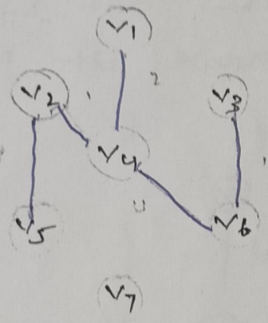
No cycle is form so the edge v_4, v_1 is consider.

Step 6:



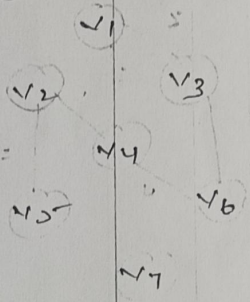
The cycle is form with the edge v_1 & v_2 so we need not consider the edge

Step 7:



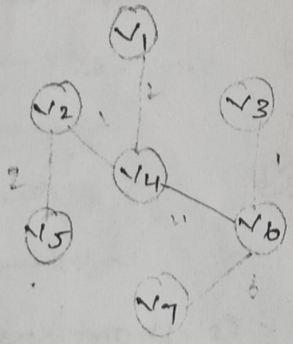
No cycle is form with the edge v_2 & v_6 so consider that the edge

Step 8:



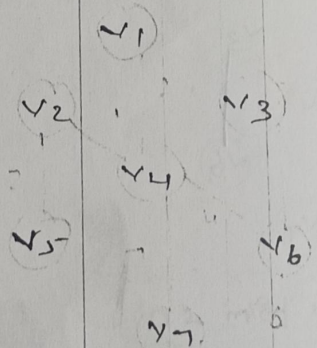
cycle is from with the edge v_1 & v_3 so we need not consider the edge $v_1 v_3$

Step 9:-



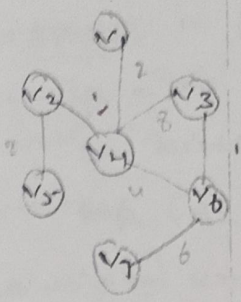
No cycle is form with the edge v_6 & v_7 so consider that edge

Step 10:-



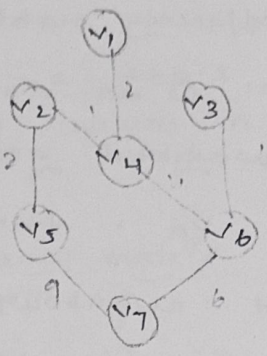
The cycle is form with the edge v_4 & v_7 so we need consider the edge v_4 & v_7

step 11:-



The cycle is form with the edge V_3 & V_4 so we need not consider the edge

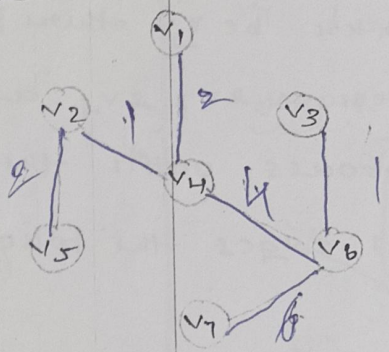
step 12:-



cycle is form with the edge V_5 & V_7 so we need not consider the edge

Result:- $(V_1 \rightarrow V_4) + (V_2 \rightarrow V_4) + (V_2 \rightarrow V_5) + (V_4 \rightarrow V_6) + (V_7 \rightarrow V_6) + (V_3 \rightarrow V_6)$

result:- $2+1+2+4+6+1 = 16$



Prims algorithem:-

This approach does not required listing of all edges in increasing order of weight (or) checking of each step that is whether a newly selected edge from a circuit (or) not according to the prims algorithem. minimum spanning tree grows in successive stages at any stage in the algorithem. we can see that we have a set of vertices that have already been included in the tree the rest of the vertices have not.

The prims algorithm can easily be implemented using the adjacency matrix representation of a graph suppose there is an nxn adjacency matrix for a given undirected weighted graph & vertices are labbled as $v_1, v_2, v_3 - - - v_n$. Start from vertex 'vi' say get its nearest neighbour that is the vertex nearest to the smallest entry in row of vertex v_i (if the minimum one smallest entry are there arbitrarily select any one) let it be v_j now consider v_i & v_j as one sub graph next obtain the closest neighbour of this sub graph i.e; vertex other than v_i & v_j that has the smallest entry among all entries in the rows of v_i & v_j let this new vertex be v_k there for the three now grows with vertex v_i & v_j & v_k as one sub graph continous this process until the all vertices have been connected n-1 edges the following are the algorithm.

Step:1:-

let G be a connected graph with non-negative value assigned to each edge.

Step2:-

let 'T' be the tree consisting of any vertex v_i (G)

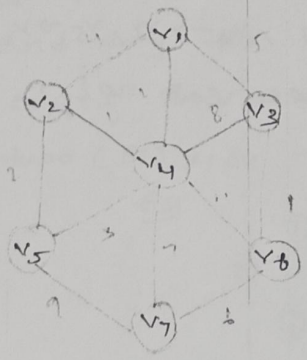
Step3:-

Among all the edges not in T' that are incident on a vertex in 'T' and do not form a circuit and added to 'T' select one of minimum cost and add it

TO 'T'

Step 4:-

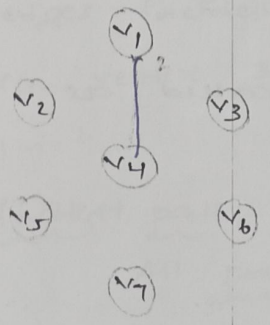
This process terminates i.e step-3 repeated until 'n-1' edges are selected. where n' is no. of vertices in "G"



adjacency matrix

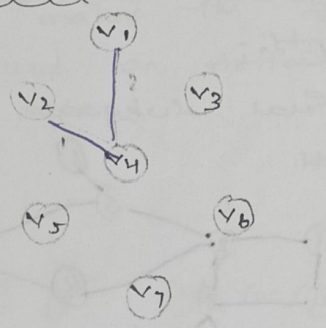
	V_1	V_2	V_3	V_4	V_5	V_6	V_7
V_1	–	4	5	2	–	–	–
V_2	4	–	–	1	2	–	–
V_3	5	–	–	8	–	1	–
V_4	2	1	8	–	3	4	7
V_5	–	2	–	3	–	–	9
V_6	–	–	1	4	–	–	6
V_7	–	–	–	7	9	6	–

Step 1:-



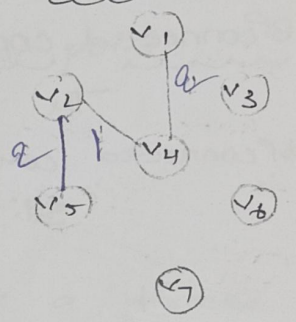
Set (V_1, V_4)

Step 2:



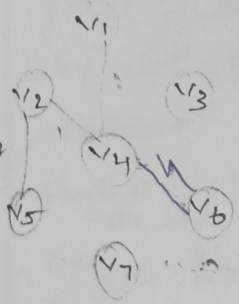
Set (V_1, V_4, V_2)

Step 3:-



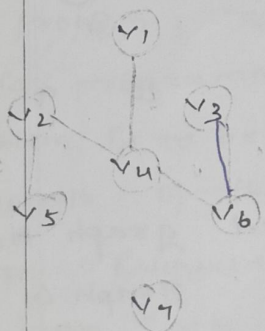
Set (V_1, V_4, V_2, V_5)

Step 4:-



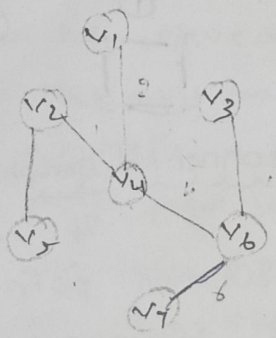
Set $(V_1, V_4, V_2, V_5, V_6)$

Step 5:-



Set $(V_1, V_4, V_2, V_5, V_6, V_3)$

Step 6:-



Set $(V_1, V_4, V_5, V_6, V_2, V_7)$

Answer:

The resultant spann tree are prims algorithm is

$(v_2 \to v_5) + (v_2 \to v_4) + (v_1 \to v_4) + (v_4 \to v_6) + (v_3 \to v_6) + (v_6 \to v_7)$

$= 2 + 1 + 2 + 4 + 1 + 6 = 16$ ✓

We can not consider the edges $v_4 \to v_7$, $v_4 \to v_3$, $v_5 \to v_7$, $v_1 \to v_2$ because the circuit is form so we can not consider the edges.

Applications of Minimum spanning tree:-

Network designing

telephone communication, electrical, TV cable, computer etc.