1. Solve - Adjancency list convert (int [ ] [ ] a) Time Depo

```
l = a[0] length                                  1
adglist = new Arraylist (l)                      1
for i = 0 to i < l do                            n
    adj list . add (new Arraylist ()) n-1
for i = 0 to a[0] . length                       n
    for j = 0 to a . length do                   n^2
        if a [i] [i] = = 1                       n^2 - 1
        adjlist . got [i] . add [j]              n^2 - 1
return adjlist                                   1
```

The time complexity of the algorithm is

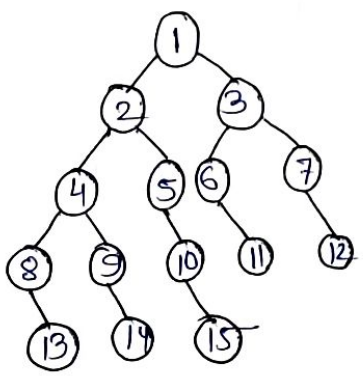
$O(N^2)$

2. Solve - Ady Matrix convert (adj [ ] , V) Time Req.

```
matrix [ ] [ ] = new [V] [V]                     1
for i = 1 to V                                   n
    for j = 1 to adj [i] da                      m
        matrix [:] [j] = 1                       n m - 1
    return matrix
```

The time complexity is $O(N^2M)$

(3)



(i) BFS Traversal

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 13 → 14 → 15

DFS Traversal

1 → 2 → 4 → 8 → 13 → 8 → 4 → 9 → 14 → 9 → 4 → 2 → 5 → 10 → 15 → 10 → 5 → 2 → 1 → 3 → 6 → 11 → 6 → 3 → 7 → 12

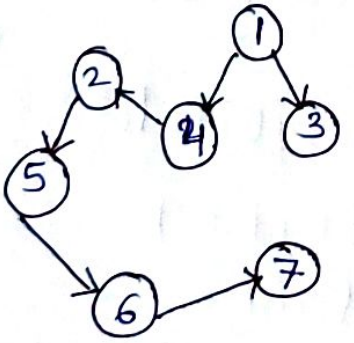
(ii) Maximum size of Queue in BFS is 5
Maximum size of stak in DFS is 4

(iii) for searching node 6, we preffect BFS as it will take less time to reath node 6 then DFS

(iv) for searching node 14, we preffer DFS on is topes less time to reach node 14 than BFS

4.

(i)



(ii) The Edges: 1-3, 1-4, 4-2, 2-5, 5-6, 6-7
Back. edge: 2-1
forword Edge: 4-6
cross Edge - 7-3, 7-4

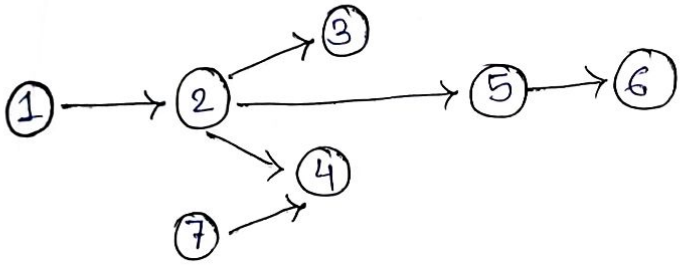
(iii)

(iv) Topologically sorted ordered of nodes of graph is
1 → 4 → 2 → 5 → 6 → 7 → 3

(5)

Let G be a graph with V vertices E edges.
Let T be a MST such that E_1 is not include in T (Let E_1 be the least weight w edge)
if we add E_1 ain T, it will form 1 cycle.
we have to remove on edge from T to get back. MST. So we chose to remove the most weight vertex (which is not E_1)
∴. E_1 must be included in T for it to be a minimum ~~sporing t~~ Spaning tree.

(6)

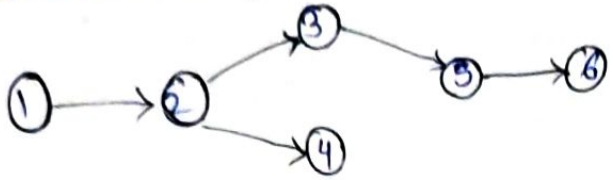


in graph G
Let we reverse the formed Edge 2 → 5 to back edge 2 ← 5 and we DFS traversal we traverse 1 → 2 → 3 → 5 in 5 we have an option to thourse back to 2. home, there exist a cycle in G.
So, G is not DAG
now, if we the given 2 → 5 edge is G and traverse.
after reaching 5, there wont be any cyclees to back to a visited vertex.
Hence G is Ocylic
∴ G is a DAG when there is no backedge

is of traversal of G_1

hence proved

DFS traversal for tree in G_1

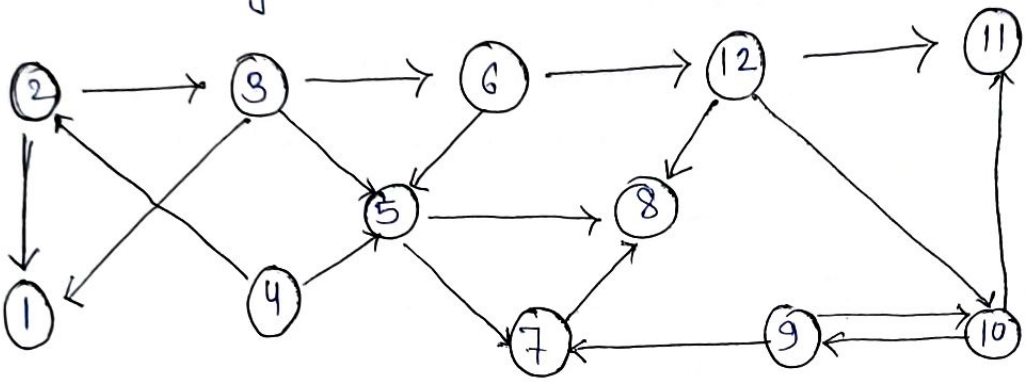


Tree Edge : 1-2, 2-3, 3-5, 5-6, 2-4

forward edge : 2-5

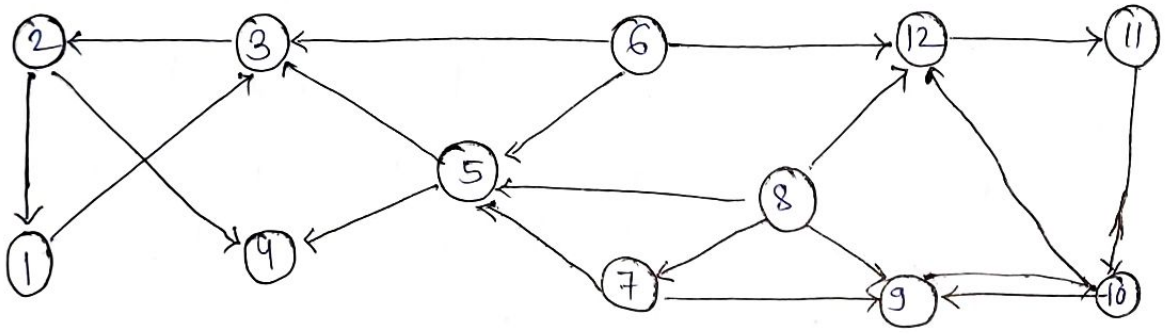
cross edge : 7-4, 4-5

(7)



G_1

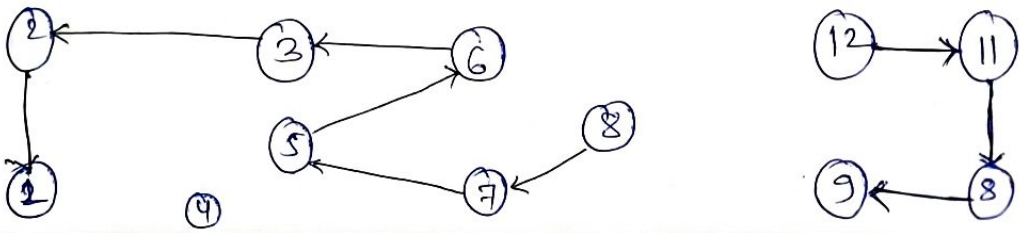
Transpose of G_1 is



Topologically sorted of nody in G_1

$1 \to 2 \to 3 \to 6 \to 5 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12 \to 4$

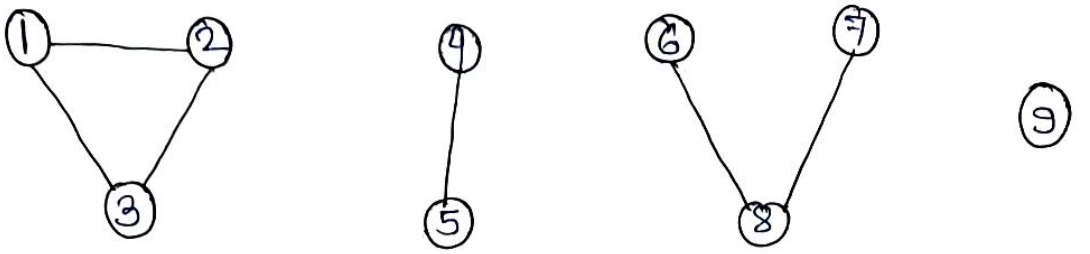
Let this be the order of element in A



Let that be T_1, T_2 and T_3 as the there tree here T_1, T_2, and T_3 represents the strongly Connected components of G

Heuee proved

(8) From the given adjency matrix graph G is



There are 4 connected components

edges = [[1,2],[2,3],[1,3][4,5][6,8][7,8]]

array = [1, 2, 3, 4, 5, 6, 7, 8, 9]

edge 1-2
array = [2, 2, 3, 4, 5, 6, 7, 8, 9]

edge 2-3
array = [2, 3, 3, 4, 5, 6, 7, 8, 9]

edge 1-3
array = [2, 3, 3, 4, 5, 6, 7, 8, 9]

edge 4-5 array [2, 3, 3, 5, 6, 7, 8, 9]

edge 6-8 array [2, 3, 3, 5, 5, 8, 8, 9]

unique parants = [3, 5, 8, 9]

∴ our graph G has 4 components.

(9) i) Let Adj [1...|V|] be a new adjacency list of the transposed G^T
for each vertex v ∈ Adj [u]

Insert [adj[v], u]
Time complexity : $O(|E|+|V|)$

(2)

(10) $BB^T(i,j) = \sum_{e \in E} b_{ie} b^T_{aj}$

$= \sum_{e \in E} b_{ie} b_{je}$

if $i=j$

then $b_{ie} b_{je} = 1$ whenever i enter or leaves vertex i, and 0 otherwise.

if $i=j$

then $b_{ie} b_{je} = -1$ when $e=(i,j)$ as $e=(j,i)$ or 0 otherwise

Then

$BB^T(i,j) = \begin{cases} \text{indegree}[i] + \text{outdegree}(i) & \text{if } i=j \\ -(\text{nb of edge connr tnyi and j}) & \text{if } i \neq j \end{cases}$

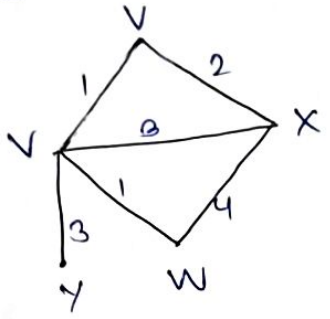
(ii) Although the edge weight of every edge in increases by R, The shortest path spaning tree T, remains the some.

∵ R is increasses on edge weight of every edge is G_1

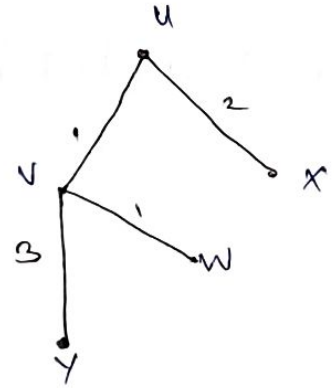
∴ The less weight edge of G_1 still remains the less weight edges.

∴ There is no change in T

example

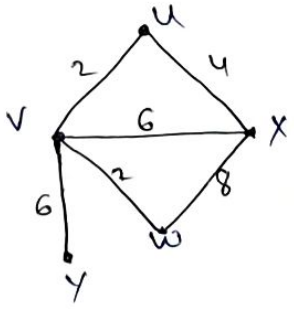


Shortest path spanning tree T from G_1 in

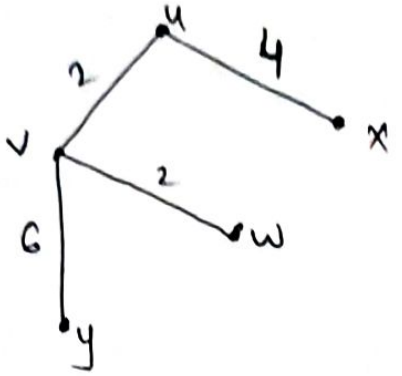


W be increment on edge weight of every edges of E

Let R = 2



Now the shortest path spaning-tree is



(12) Let G be an undirected connected graph but not a tree.

Then G must contain a cycle C.

Suppose C consists of R node or

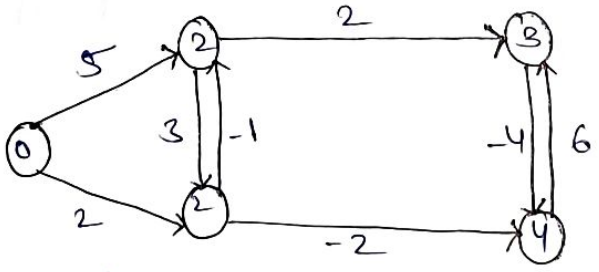
$$V_1 \to V_2 \to V_R \to V_1$$

now in DFS tree, node V_1 $V_2 \to V_R$ will all be on the some path from root to a leaf.

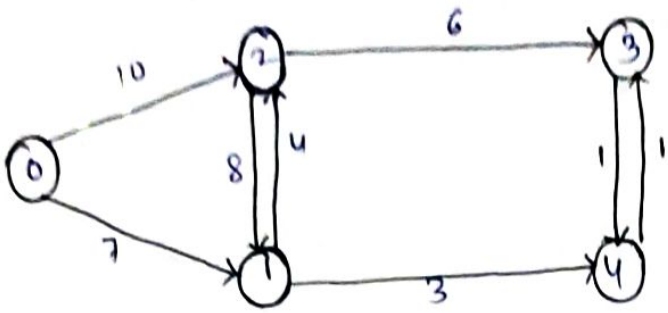
But, in BFS tree, nodes V_1 V_2 — V_R will form atleast two ~~fore~~ branches, branching from the node first visited.

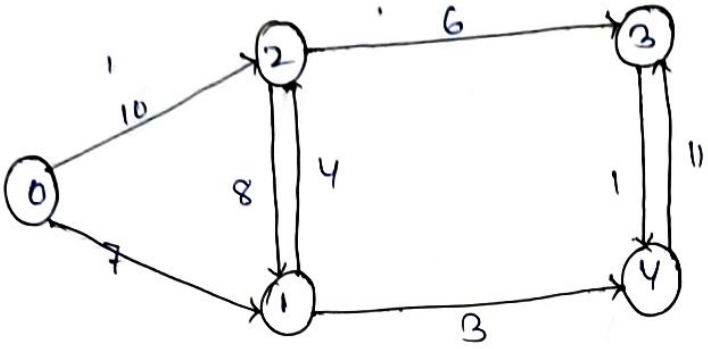
∴ BFS and DFS will product the same tree T if and only if G = = T.

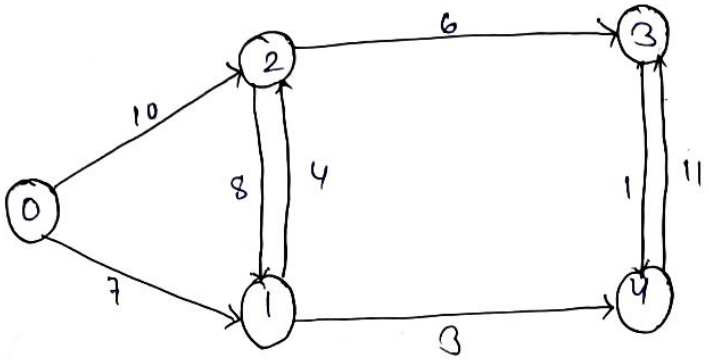
(13) Let G be a directed graph with negative edge cost for some edge.



The next minimum cost in -4 so lets add 5 to every nodes.



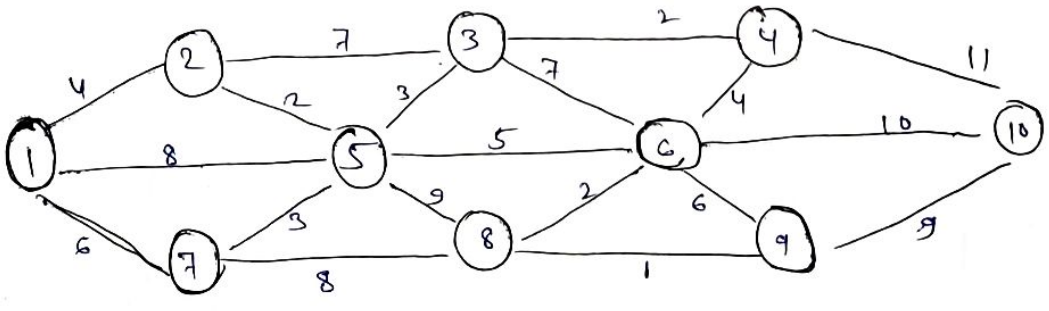




Distance Algorithm fails on a graph with negative weighted edges.

Since, Distance Algorithm followers grady approach. It doesnot recomder a vertex node even if shorter pat exists.

(14)

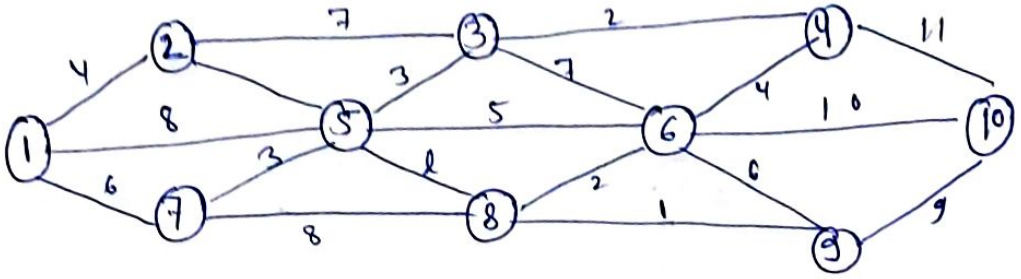


Let R be the shortest path from source to the farthest node.

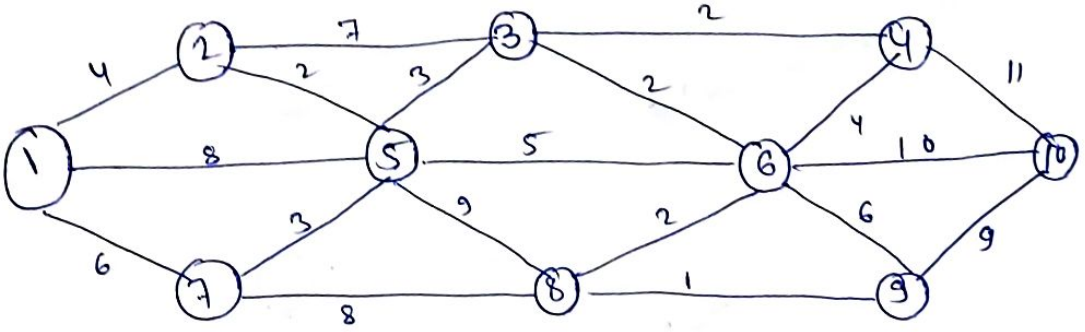
$\therefore R \leq |E|$

Applying Diypstras Algorithm

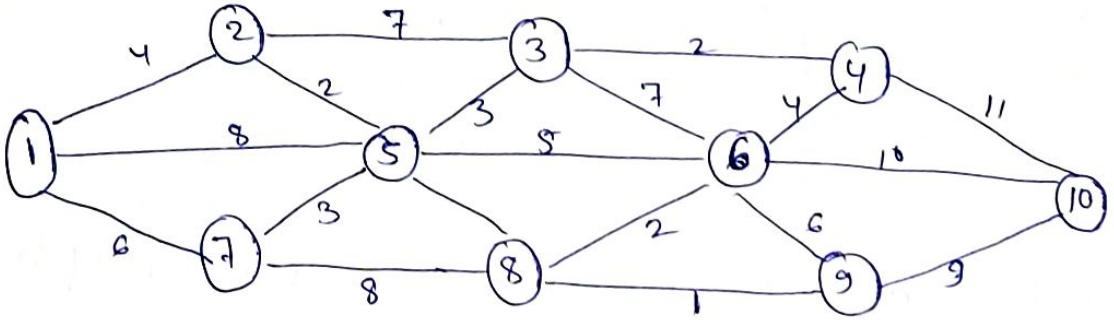
I)



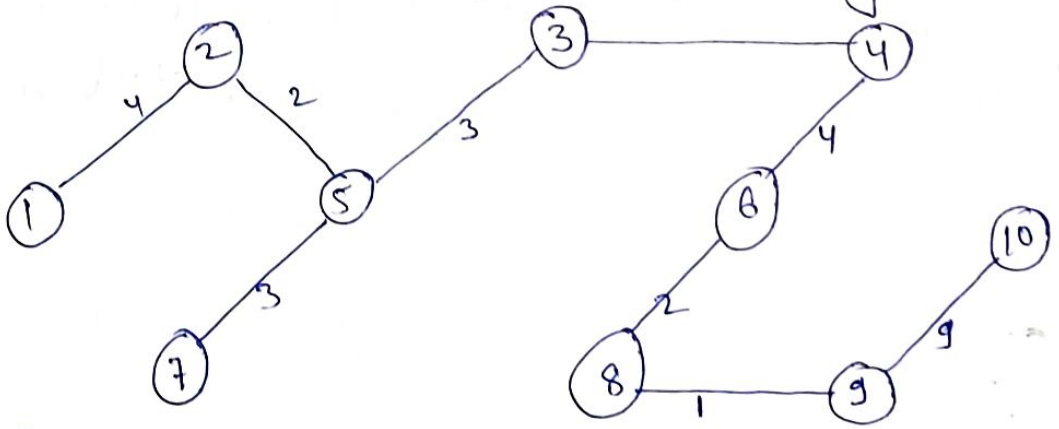
II



III



$\therefore$ our shortest path spanning tree is



$\therefore R = 6$ (∵ our root is 5 & furthest node is 10 our tree has 9 edges.

(5) Kruspole Algorithm

1. $A \leftarrow \phi$
2. for each kertage $V \in V[G]$
3. do Create-Set (V)
4. Sort each edge of E by non dureaing weight W
5. for each edge $(u,v) \in E$ inorderd by nondecrea sing weight.
6. if find. set(u) $\neq$ Find. set (v)
7. $A \leftarrow A \cup \{(u,v)\}$
8. union (u, v)
9. return A,

Correctnuse of krusksalis Algorithum

Loop inuoriont: At the start of each iter ation of for loop at line 2, the ~~A~~ Array A[0: vcos] does not ~~dear~~ contain vertix v.

Initialization: before the first enteration the array A[0: V(r)] is enpty.

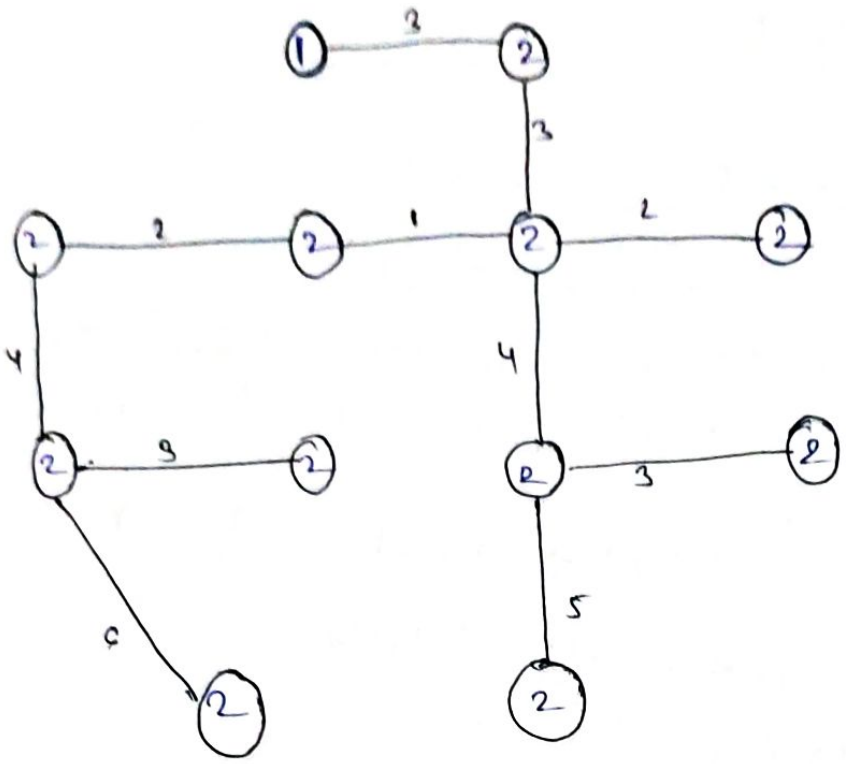
so, the sub array does not contain vertex v

Montainomce: At line 3, the loop creater a particular set for every verten in V(0).

∵ $V \in V(G)$, There exists a ~~position~~ particular set for verten v.

Terminotion: The loop termenoter when all the verten of graph G pass been assign a ~~post~~ particular set.

The algorithum selects the edge with least weight and convert these two verten. by the end, we got a Minimum spearing tree.



∴ This is even minimum spanning tree.

Steps :-

① Connect all verten with edge w = 1
② Connect all verten with edge w = 2
③ Connect all verten with edge w = 3
④ Connect all those verten with edge ~~w~~ widgh
⑤ Connect all those verten with weight 5 and 6 & not formity any cycle.
⑥ Stop when all the the edge are connected, we get our tree.