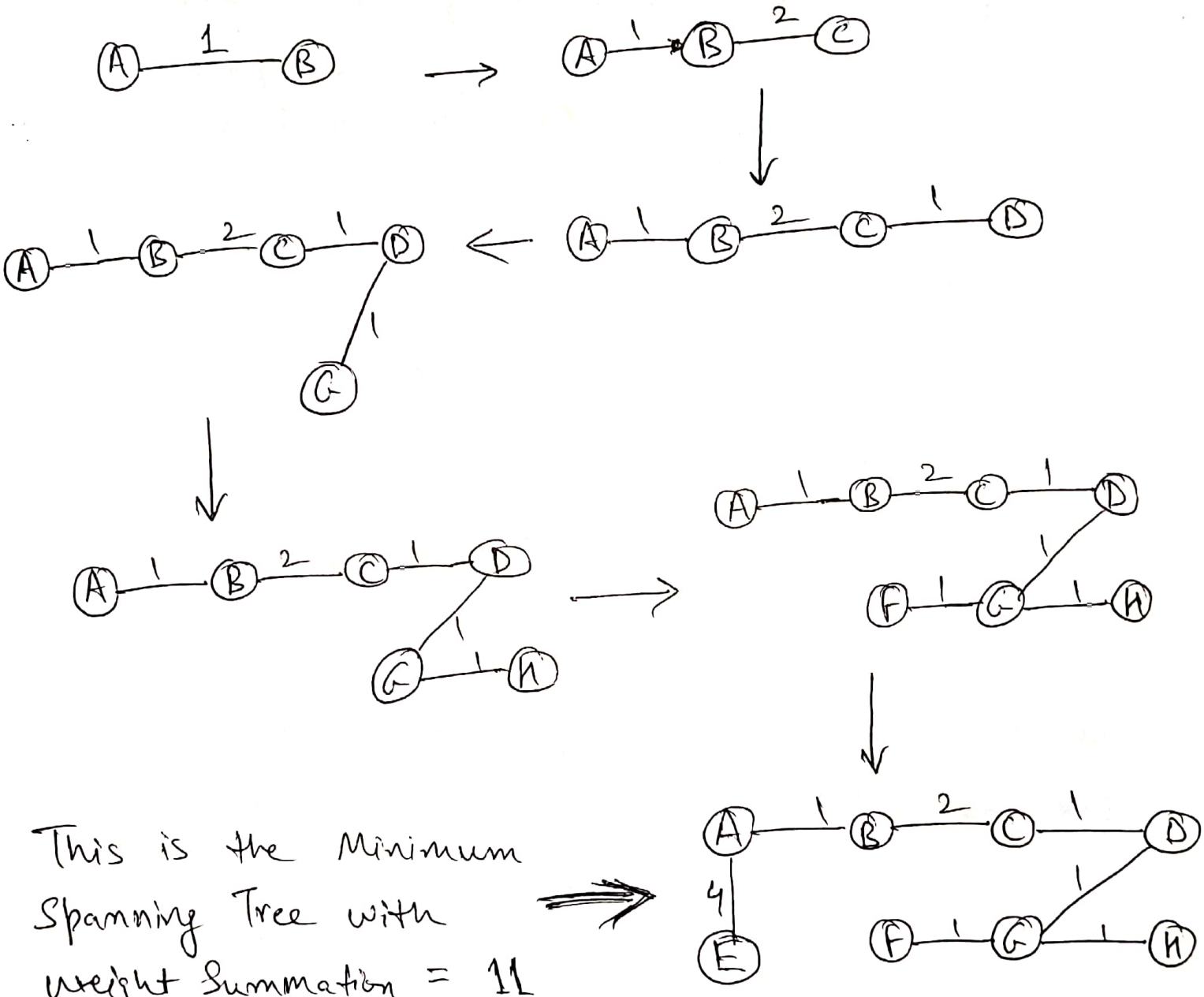
DAA Assignment -5

Submitted by –
BHUVAN CHADHA

1. Minimum Spanning Tree using Prim's Algorithm –
The starting node is A.



This is the Minimum Spanning Tree with weight Summation = 11

2. Greedy Algorithm to determine the number of conference rooms required —

```
Rooms (Sort)
{
    Sort according to the finishing duration/time.
    temp = 0
    for i in range(2n):
        if (order[i][1] = 0):
            temp = temp + 1
            rooms = max (rooms, temp)
        else
            if (temp):
                temp = temp - 1

    return rooms
}
```

This algorithm can always generate optimal solution. If k rooms are needed, the time complexity will be

$$T(n) = O(n \log n).$$

3. An edge of minimum weight in a graph is trivially the minimum weight edge of some cut. Also, if that edge is unique, then it is the minimum-weight edge that separates its endpoints.

So, if there is one edge of a weight smaller than all the other edges, that edge will be chosen first every time a minimum spanning tree is made. Hence, it will be part of every minimum spanning tree.

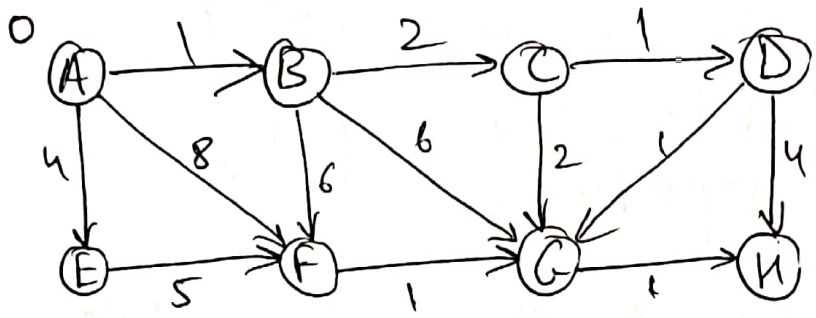
4. In both, Breadth-first Search and Prim's Algorithm, we explore the graph starting from a single vertex and choose the next vertex to explore according to some priority function. In Prim's Algorithm, the shortest edge is chosen from the starting node, always. ~~whereas~~ In Breadth-first Search, the priority is the best distance from the origin node whereas in Prim's Algorithm, the priority is the distance from already explored nodes no matter which connecting ~~node~~ edge is chosen. In both the algorithms, a smaller value is better.

Hence, it is <u>sometimes</u> true that the order in which the nodes are added to the MST by Prim's Algorithm is the same as the order in which they are encountered in BFS.

5. We are given a group of cities modled by a strongly connected undirected graph $G=(V, E)$.
A Capital city $a \in V$.
An algorithm can be made using Dijkstra's Method of finding shortest path between all pairs of cities.
Here, all the paths pass through the Capital city 'a'.

```
CitiesPath ( )
{
    for all u∈V      // u is an individual node.
      {
        dist [u] = ∞
        prev [u] = nuld
        dist [s] = 0 }
    H = makequeue (V)      // Making Heap.
    while (H is not empty)
    {
        u = deletemin(H)
     }
     for all edges (u,v) ∈ E   // (u,v) is an edge
       {                       // between u and v.
           if dist [v] > dist [u] + L [u,v]
              dist [v] = dist [u] + L [u,v]
              prev [v] = u }
     decreasekey (H, v)
}
```

6. (a) Applying Dijkstra's Algorithm on the given graph starting from node A.



Initializing node A = 0 and all other nodes to ∞.

Iteration-1

	A	B	C	D	E	F	G	H
0	(0)	∞	∞	∞	∞	∞	∞	∞

Iteration-2

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	(1)	∞	∞	4	8	∞	∞

Node B = 1 is the minimum cost value.

Iteration-3

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	4	8	∞	∞
2	0	1	(3)	∞	∞	7	7	∞

Node C = 3 is the minimum cost value.

Iteration - 4

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	4	8	∞	∞
2	0	1	3	∞	4	7	7	∞
3	0	1	3	④	4	7	5	∞

Node D=4 is the minimum cost value.

Iteration - 5

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	4	8	∞	∞
2	0	1	3	∞	4	7	7	∞
3	0	1	3	4	4	7	5	∞
4	0	1	3	4	④	7	5	8

Node E=4 is the minimum cost value.

Iteration - 6

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	4	8	∞	∞
2	0	1	3	∞	4	7	7	∞
3	0	1	3	4	4	7	5	∞
4	0	1	3	4	4	7	5	8
5	0	1	3	4	4	7	⑤	8

Node G=5 is the minimum cost value

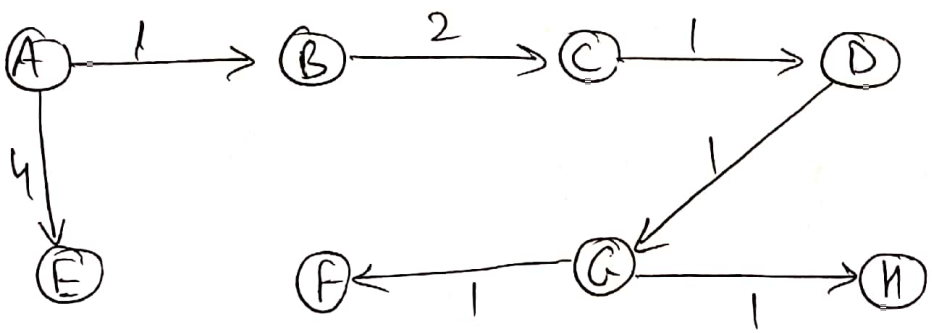
Iteration - 7

	A	B	C	D	E	F	G	H
0	0	∞	∞	∞	∞	∞	∞	∞
1	0	1	∞	∞	4	8	∞	∞
2	0	1	3	∞	4	7	7	∞
3	0	1	3	4	4	7	5	∞
4	0	1	3	4	4	7	5	8
5	0	1	3	4	4	7	5	8
6	0	1	3	4	4	(6)	5	(6)

Nodes F=6 and H=6 are minimum cost values.

Hence, after visiting all the nodes, the above table represents the Shortest distance cost of each node.

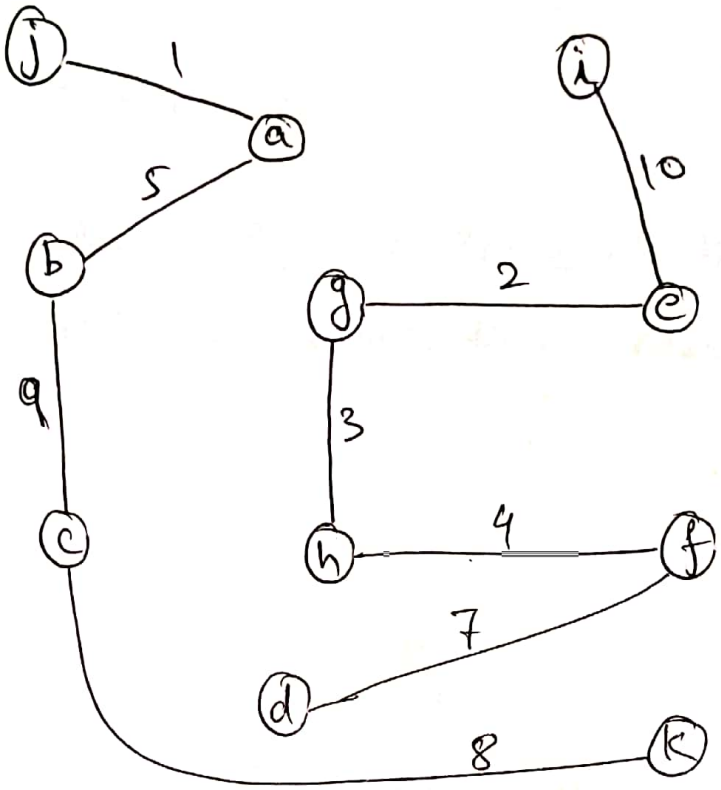
(b) Final Shortest path tree is -



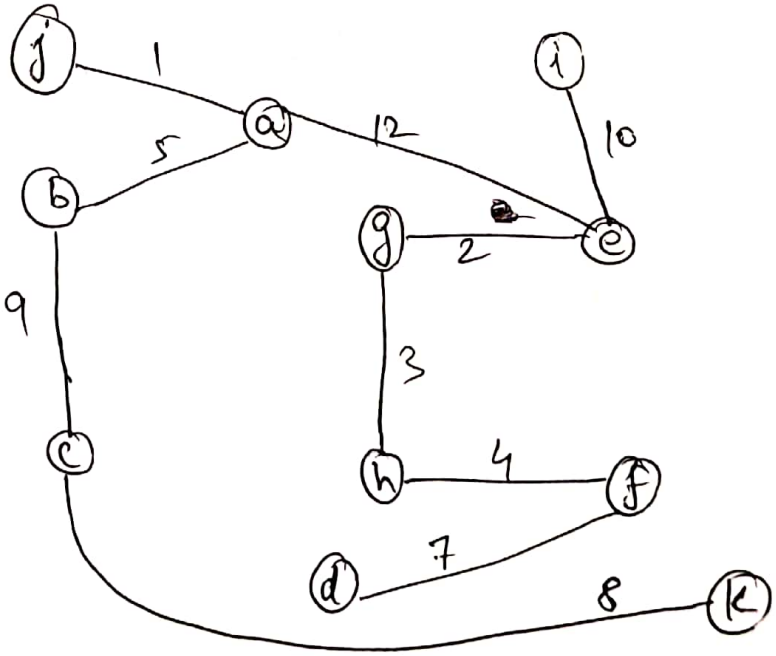
The cost of this Minimum Spanning Tree = 11

7. Using Kruskal's Algorithm, to find the minimum Cost of the given road network,
Starting with the minimum-weight edge (j, a) = 1.

Now, intermediate step when (e, i) edge is selected -



Now, final result after joining all the edges -



Minimum Cost = 61

8. We are given -
No. of files = n
Size of each CD = B bytes
Size of each file, i = S_i bytes ($S_i < B$)
The files need to be stored in minimum number of CDs, which can be done using greedy algorithm.

Algorithm DiskStorage
{
Take Array, A[] to store all the files,
Sort A[] in descending order
A[] = {1, 2, 3, ... n files}

CD1 ⟵ A[1]

Check for(A[2] to A[n]) // for loop.
{If Any file can be made available with
size equal to the left size in CD1,
Store A[] in CD1

Else
create CD2
A[2] ⟵ CD2.
}

Return no. of CDs used, CDn.
}.

9. The Prim's Algorithm is designed in such a way that it is able to check the graph's connectivity by itself. If the algorithm reaches the entire graph's vertices then the graph is connected, Otherwise the graph is disconnected, because there will be no path from one sub-graph to reach to another sub-graph. Hence, there will be some nodes left to be traversed.

10. Maximum Spanning Tree is the tree with the largest possible edge weight of a weighted connected graph. The following algorithm finds the maximum spanning tree of a graph, G.

```
MaxSpanningTree (G, w, r)
{
    for each u ∈ G.V        // u is an individual node.
    {                       // V is the set of vertices.
      u.key = -∞
      u.π = π
    }
    r.key = 0
    Q = G.V
    while (Q is not empty)
    {
        u = extract maximum (Q)
        for (each v ∈ G.Adj[u])
            if (v ∈ Q & w(u,v) > v.key
                v.π = u ; v.key = w(u,v)
    }
}
```