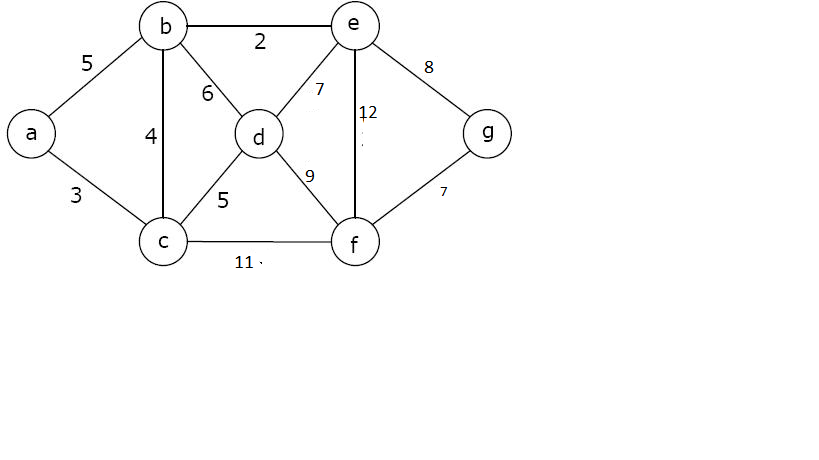
**Vipul Krishna**

**Assignment 10**

**INFO 6205 – Program Structures and Algorithms**

Question 1:

Given Graph:



1. Kruskal’s Algorithm:

Kruskal’s algorithm of Minimum Spanning Tree is a greedy algorithm. The steps for this algorithm is:

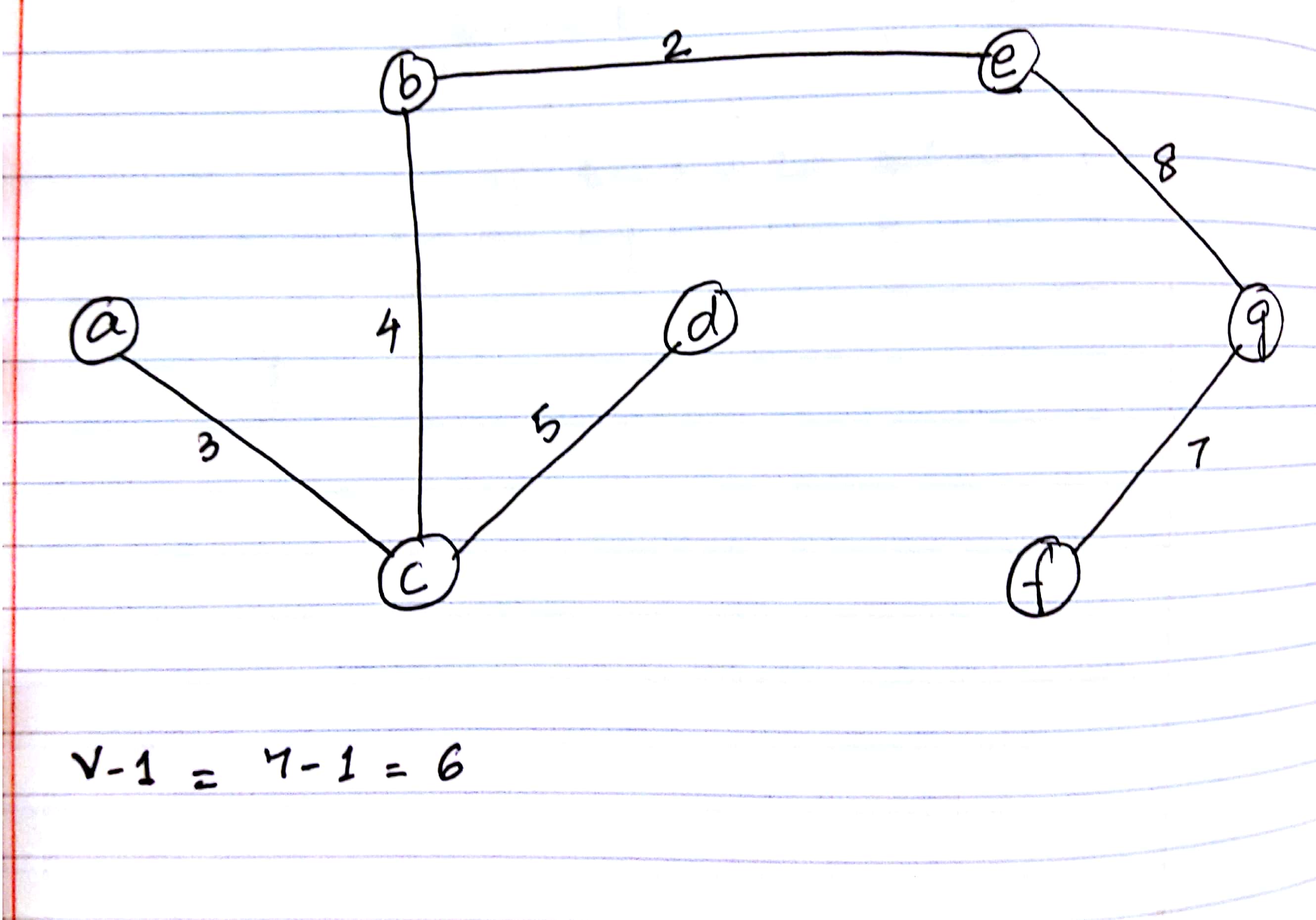
1. Arrange all vertices in non-descending order of their weight.

|  |  |
| --- | --- |
| b-e | 2 |
| a-c | 3 |
| b-c | 4 |
| a-b | 5 |
| c-d | 5 |
| b-d | 6 |
| d-e | 7 |
| f-g | 7 |
| e-g | 8 |
| d-f | 9 |
| c-f | 11 |
| e-f | 12 |

1. Pick the first edge from the table. Add it to the spanning tree only if it does not form a cycle. Keep picking the edges till the number of edge in MST = V - 1 => 7 – 1 = 6

**Working:**

1. Pick up b-e. It does not form a cycle. Add it to the MST. Pick a-c. It does not form a cycle. Add it to the MST. Pick b-c. It does not form a cycle. Add it to the MST. Pick a-b. It forms a cycle. Skip it. Pick c-d. It does not form a cycle. Add it to the MST. Pick b-d. it forms a cycle, so skip it. Pick d-e. It forms a cycle, so skip it. Pick f-g. It does not form a cycle. Add it to the MST. Pick e-g. It does not form a cycle. Add it to the MST.
2. Now that we have V-1 edges in the MST, the process stops and the resultant minimum spanning tree is:



1. Prim’s Algorithm:

This is also a greedy algorithm. The steps for this algorithm are as follows:

1. Create an mstSet to keep a track of all vertices that have been visited.
2. Assign s key value to all vertices in the given graph. Source is initialized to 0 and all other vertices is initialized to ∞.
3. Now pick the vertex with the minimum key value which is not present in mstSet, include it in mstSet.
4. Update the key value to all adjacent vertex (which is not there in mstSet) to the edge weight only of the edge weight < current key value of that vertex.
5. Repeat step iii and iv till all vertices are included in mstSet.

**Working:**

MstSet = []

|  |  |
| --- | --- |
| Vertices | Key Value Updates |
| a | 0 |
| b | ∞ 🡪 5 🡪 4 |
| c | ∞ 🡪 3 |
| d | ∞ 🡪 5 |
| e | ∞ 🡪2 |
| f | ∞ 🡪 11 🡪 9 🡪 7 |
| g | ∞ 🡪 8 |

Visit vertex a. Add to mstSet. MstSet = [a, ]

Update Values of adjacent vertices b and c to 5 and 3 respectively since they are less than ∞.

Now, c has minimum key value. Add it to mstSet. MstSet = [a, c, ]. Update values of b, d and f since the edge weights are less than their current index values.

Now, b has minimum key value. Add it to mstSet. MstSet = [a, c, b]. Update key value of e to 2 and skip d as d’s key value is already lesser than the edge weight which is 6.

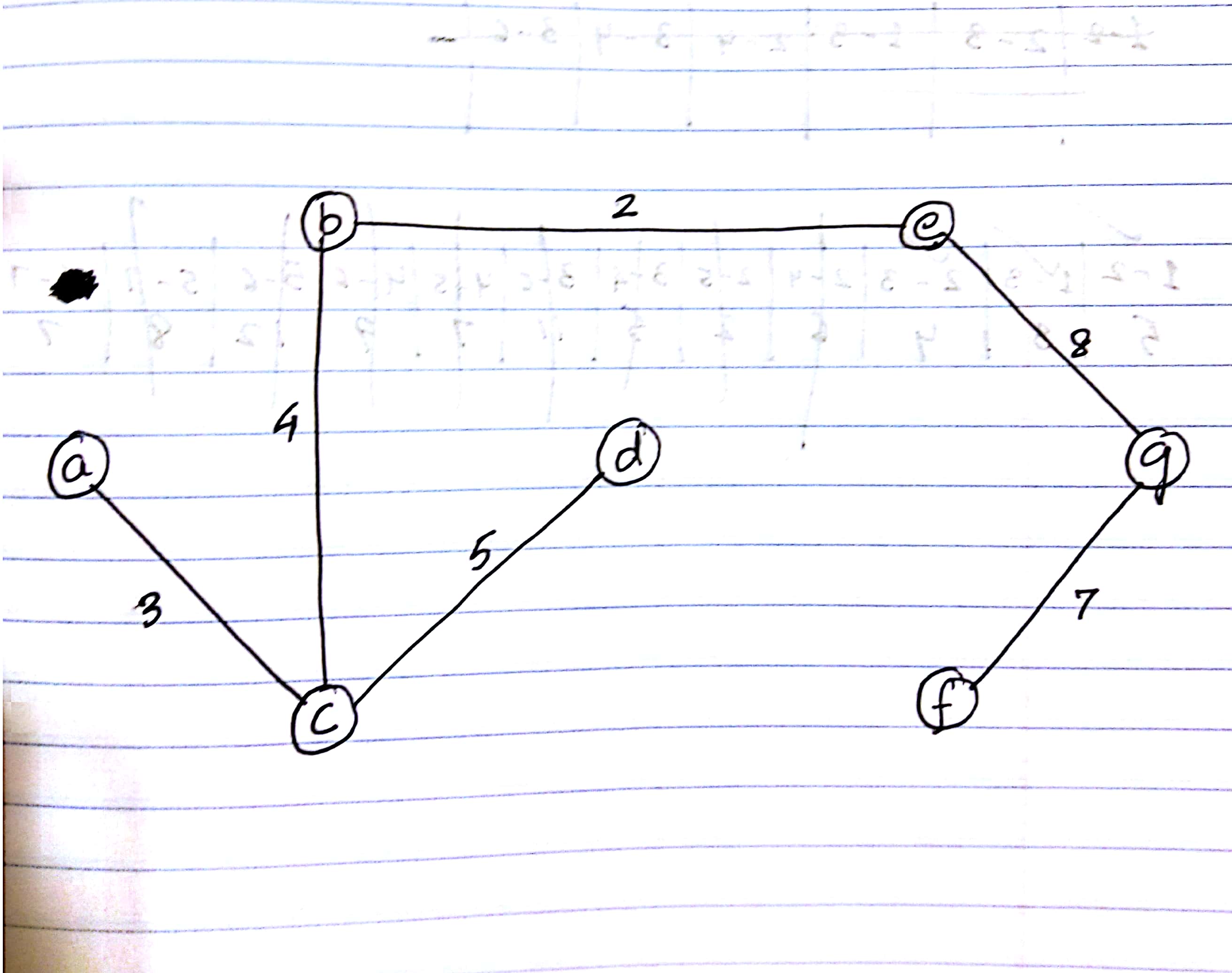
Now, e has minimum key value. Add it to mstSet. MstSet = [a, c, b, e]. Skip d and f and update g.

Now, d has minimum key value. Add it to mstSet. MstSet = [a, c, b, e, d]. Update f to 9.

Now, g has minimum key value. Add it to mstSet. MstSet = [a, c, b, e, d, g]. Update f to 7.

Only vertex left now if f and all its adjacent vertex is in mstSet. So, include f to mstSet.

MstSet = [a, c, b, e, d, g].



1. Java Code Attached.
2. Space and Time Complexity Comparison

Space Complexity:

Both Kruskal’s and Prim’s algorithm have same space complexity which is = O (E + V).

E is the number of edges and V is the number of vertices given input graph

Time Complexity:

Time Complexity of Kruskal’s Algorithm = O(ELogE) or O(ELogV).

Time complexity of Prim’s Algorithm = O(ELogV) with the use of binary heap and when input graph is represented using adjacency list.

Thus, we can say that both algorithms will have almost similar time complexity.