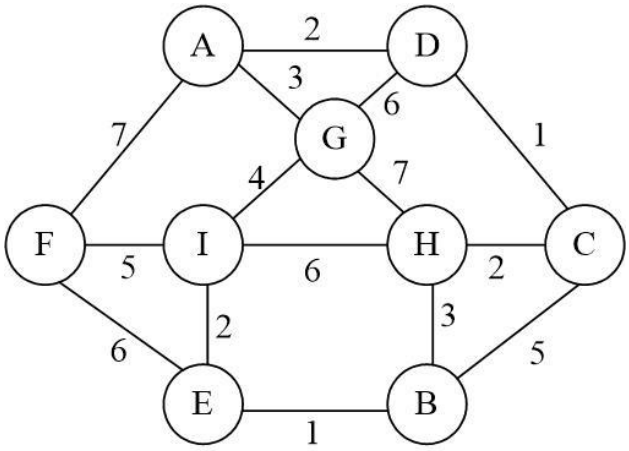
Question 1:

Execute Prim's minimum spanning tree algorithm by hand on the graph below showing how the data structures evolve specifically indicating when the distance from a fringe vertex to the tree is updated. Clearly indicate which edges become part of the minimum spanning tree and in which order. Start at vertex F.



Answer:

To execute Prim's algorithm, we start with a single vertex and gradually grow the minimum spanning tree by adding the nearest vertex not already in the tree. We keep track of the vertices that are not in the tree, but are adjacent to the tree (i.e., the fringe vertices). We also keep track of the distance of each fringe vertex to the tree.
start at vertex F, and initialize the minimum spanning tree (MST) with just F. The fringe vertices are the vertices adjacent to the MST.
A priority queue is maintained to keep track of the edge with shortest distance.

- In the strart, there is only F node in queue and all other nodes are initialized to infinity distance.

MST = {}

Q = {F}
{0}

Then, fringes or the adjacent nodes of F are added in the queue that are (A,7) , (I,5) and (E,6).
These nodes are added in the queue, and the visited node is added to MST.

MST = {F}

Q = {I,E,A}
{5,6,7}

Lightest node with weight 5 that is I is picked and its adjacent nodes are added to the queue and it is added to MST. As distance from I to E is less than the distance from F to E, we will update the distance from 6 to 2.

MST = {F,I}

Q = {E,G,H,A}
      {2,4,6,7}

Again, lightest node E is picked with fringe B as I is already visted. B has weight 1 so it will be placed at first in the priority queue.

MST = {F,I,E}

Q = {B,G,H,A}
      {1,4,6,7}

Moving forward from B, gives a new node C and updates the weight of fringe H to 3.

MST = {F,I,E,B}

Q = {H,G,C,A}
      {3,4,5,7}

Visiting node H with lightest weight, adds updates the weight of fringe C from 5 to 2., and is added to MST.

MST = {F,I,E,B,H}

Q = {C,G,A}
      {2,4,7}

Next node to be visited is C with weight 2. It gives add to a new fringe D with edge weight 1.

MST = {F,I,E,B,H,C}

Q = {D,G,A}
      {1,4,7}

When we visit the node D, it updates the distance of A from 7 to 2 and makes edge DA.

MST = {F,I,E,B,H,C,D}

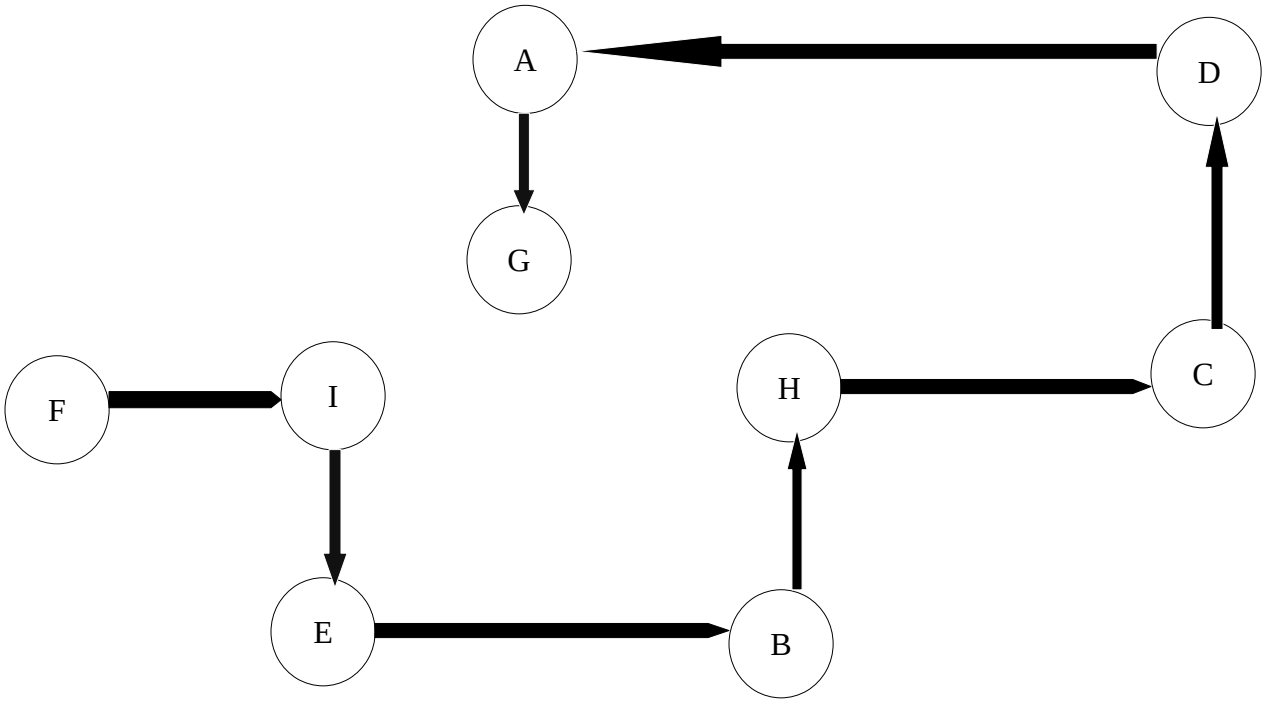
Q = {A,G}
      {2,4}

Second last node in the graph, updates the distance of last node G from 4 to 3.
MST = {F,I,E,B,H,C,D,A}

Q = {G}
{3}

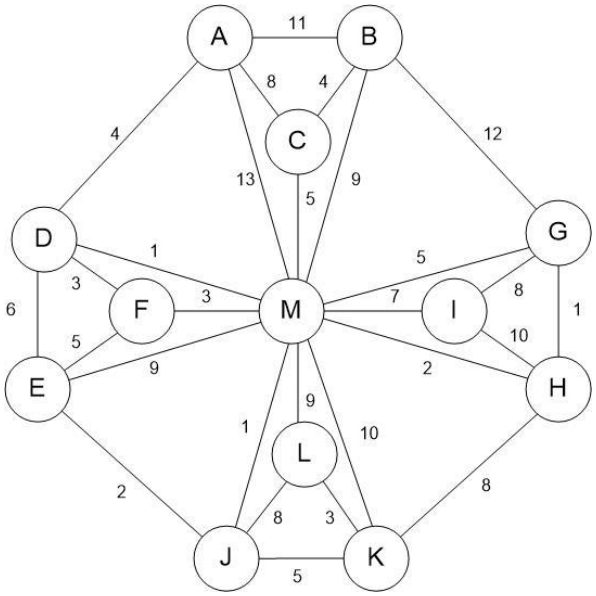
And in the end G is added to the MST being the last node from F to G.
MST = {F,I,E,B,H,C,D,A,G}

Finally, the edges being the part of MST and their order is as follows:



Question 2:

`Execute Kruskal's algorithm on the weighted tree shown below. Assume that edges of equal weight will be in the priority queue in alphabetical order and each edge name is ordered alphabetically. Clearly show what happens each time an edge is removed from the priority queue and how the dynamic equivalence relation changes on each step and show the final minimum spanning tree that is generated.



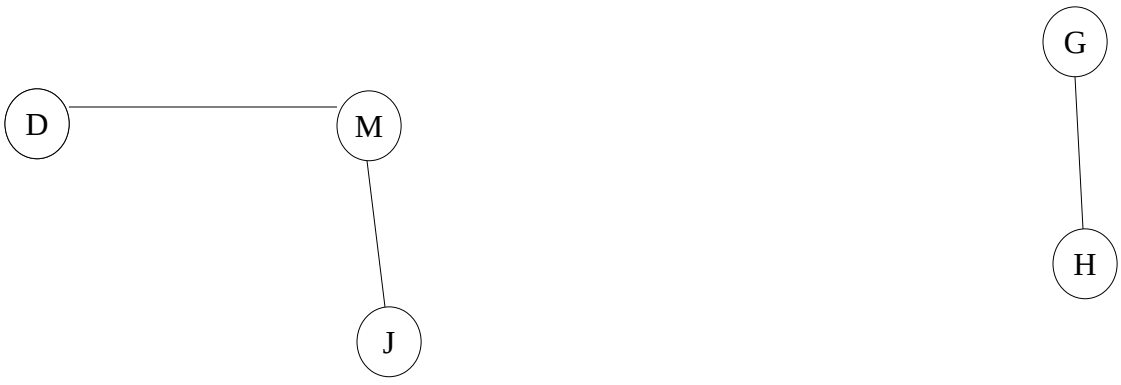
Answer:

A priority queue is maintained for the pairs of nodes with similar nodes in a fashion that each node will be in alphabetical order where pairs are also ordered alphabetically.

In kruskal's algorithm, we need to check a mandatory condition that no edge should make a cycle , else it will not be a minimum spanning tree.

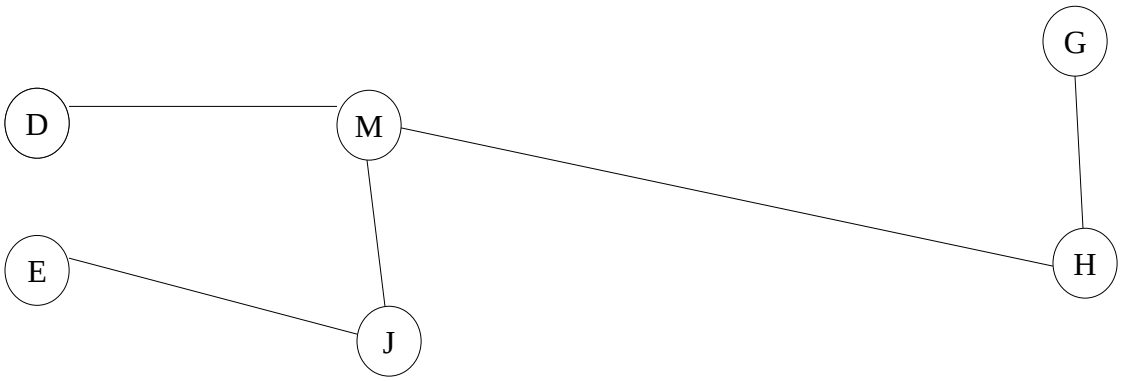
Starting with the pairs with least edge cost: 1.

Q = { (D,M) , (G,H) , (J,M) }



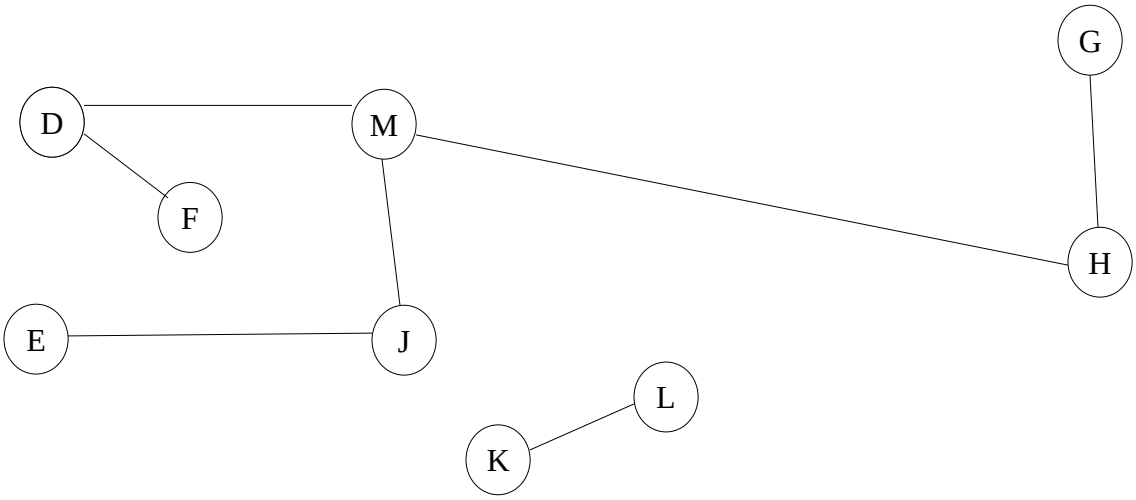
Next moving to Cost 2 , two pairs of nodes are added in priority queue.

Q = { (E,J), (H,M) }



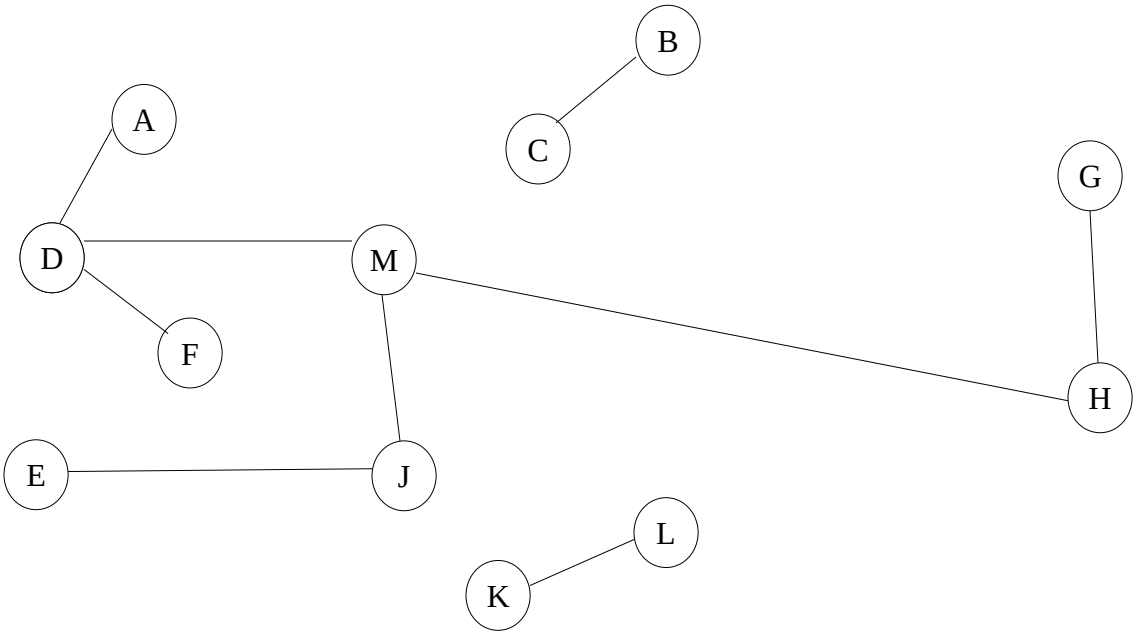
Moving ahead to cost 3, Three pairs of nodes are added in P. Queue. Where pair (F,M) will be discarded because it will make a cycle in tree.

Q = { (D,F) , (F,M) , (K,L)}



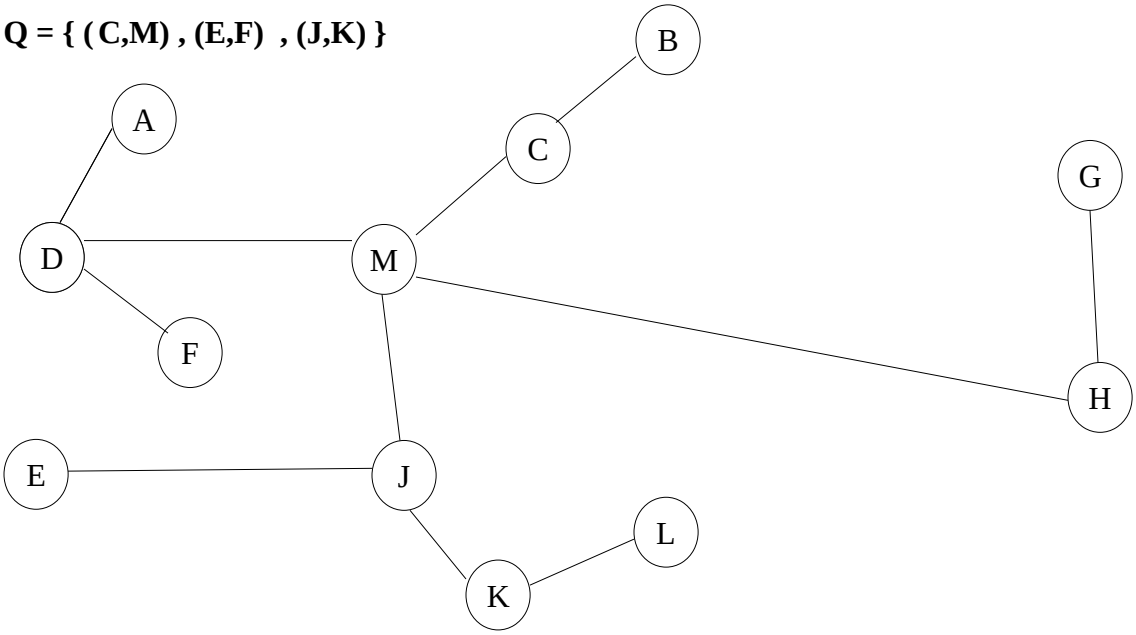
Next is cost 4 , where we get 2 pairs.

Q = { (A,D) , (B,C) }



Moving to cost 5 , three nodes pairs are added where one pair (E,F) will be discarded because it makes a cycle in MST.

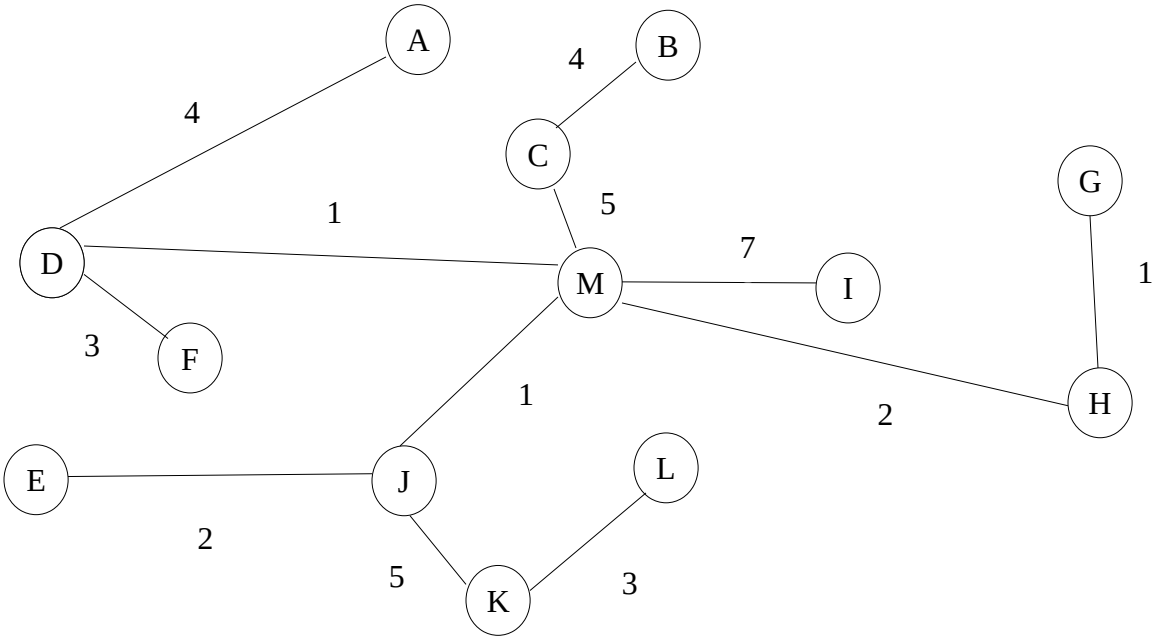
Q = { (C,M) , (E,F) , (J,K) }



Next, Cost 6 will give pair of (**D,E)** which is eliminated from queue because it makes a cycle in MST.

Pairing nodes of cost 7 gives a pair (I,M).

Q = { (I,M) }



At this stage, all nodes are added to the Tree, there is no need for more pairs with higher costs.

Thus, above tree is the **minimum spanning tree.**

Question 3:

Examine the minimum spanning trees generated in the previous two problems. In both cases, indicate whether the spanning tree is unique. If it is not unique provide all other minimum spanning trees. Explain how you made the determination whether the minimum spanning tree is unique.

Answer:

In general, the minimum spanning tree of a graph may not be unique. The uniqueness of the minimum spanning tree depends on the weights of the edges in the graph**.**

- In the case of Prim's algorithm, the minimum spanning tree is unique if the weights of the edges in the graph are distinct. This is because at each step of the algorithm, we choose the minimum weight edge that connects the tree to a vertex in the fringe. If the weights of the edges are distinct, then there can be only one edge with minimum weight at each step. Therefore, the minimum spanning tree produced by Prim's algorithm is unique.
- In the case of Kruskal's algorithm, the minimum spanning tree is also unique if the weights of the edges in the graph are distinct. However, if there are multiple edges with the same weight, then there may be multiple minimum spanning trees. In this case, Kruskal's algorithm can produce any one of the minimum spanning trees because the pairs with same weighs are arranged alphabetically.

To determine whether a minimum spanning tree is unique or not, we can examine the weights of the edges in the graph. If there are no ties in the edge weights, then the minimum spanning tree is unique. If there are ties, then we need to analyze the ties and the order in which the edges are added to the tree by the algorithm to determine if there are multiple minimum spanning trees.

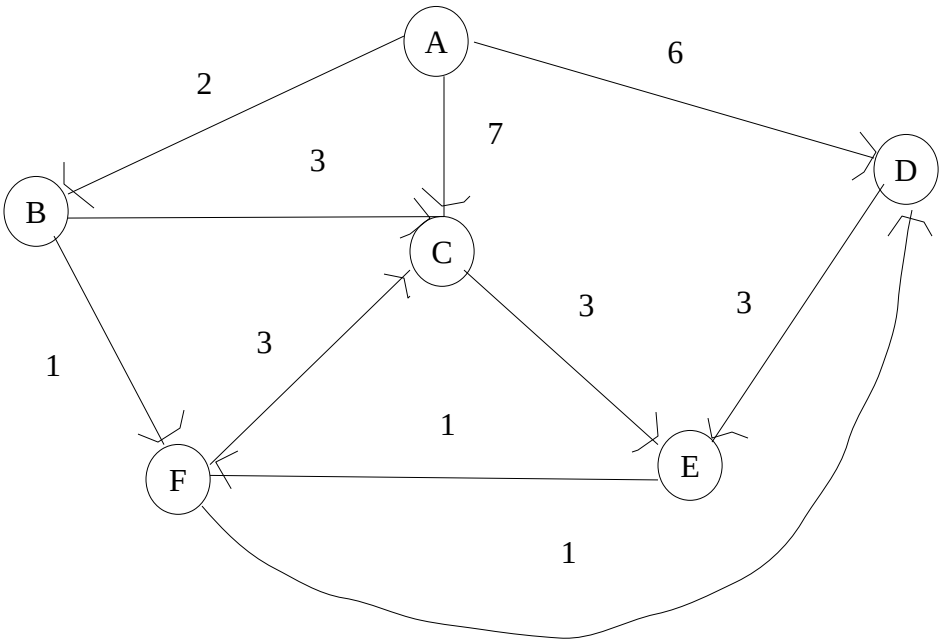
If there are ties, and we want to find all minimum spanning trees, we can modify the algorithm to handle ties in different ways. For example, we can sort the edges with the same weight in a specific order or use a tie-breaking rule to select the edges to be added to the tree.

Question 4:

Given the following adjacency lists (with edge weights in parentheses) for a directed graph:

A: B(2), C(7), D(6)
B: C(3), F(1)
C: E(3)
D: E(3)
E: F(1)
F: C(3), D(1)

Execute Dijkstra's shortest-path algorithm by hand on this graph, showing how the data structures evolve, with A as the starting vertex. Clearly indicate which edges become part of the shortest path tree and in which order.



Answer:

In Dijkstra's Algorithm, I will take care of three things, the unvisited nodes, the cost of visited nodes and the path through the visited nodes. For this, I will maintain a table where visited nodes will be shown with black color and unvisited with green color. Previous cost is added to next cost and minimum of all the costs is selected for next traversal.

We start with vertex A as the source vertex. The distance to A from itself is 0, and we update the distances to its neighbors (B, C, and D) as follows: B has a distance of 2, C has a distance of 7, and D has a distance of 6.

Vertex	Cost	Path
A	0	A
B	2	A
C	7	A
D	6	A
E		
F		

Next shortest of all these that is B is traversed to its adjacent nodes. Which updates the distance to C from 7 to 5.

Vertex	Cost	Path
A	0	A
B	2	A
C	7, 5	A, B
D	6	A
E		
F	3	B

We will repeat similar step for F now being the unvisited node with shortest cost.

Vertex	Cost	Path
A	0	A
B	2	A
C	7, 5	A, B
D	6, 4	A, F
E		

Vertex	Cost	Path
F	3	B

Repeating the iteration for D:

Vertex	Cost	Path
A	0	A
B	2	A
C	7, 5	A, B
D	6, 4	A, F
E	7	D
F	3	B

Now for C . As C has only 1 adjacent child node E, but has cost to reach there 8 from source, hence no update will occur in table and similarly for last node E , no update will happen for node F. Hence,

Vertex	Cost	Path
A	0	A
B	2	A
C	7, 5	A, B
D	6, 4	A, F
E	7	D
F	3	B

All the nodes are visited now.
The nodes and shortest-path to reach them are given:

Vertex
A → B
B → C , B → F
C
D → E
E
F → D

The order is :

A → B → (C , F → D → E)

So the shortest path TREE is :