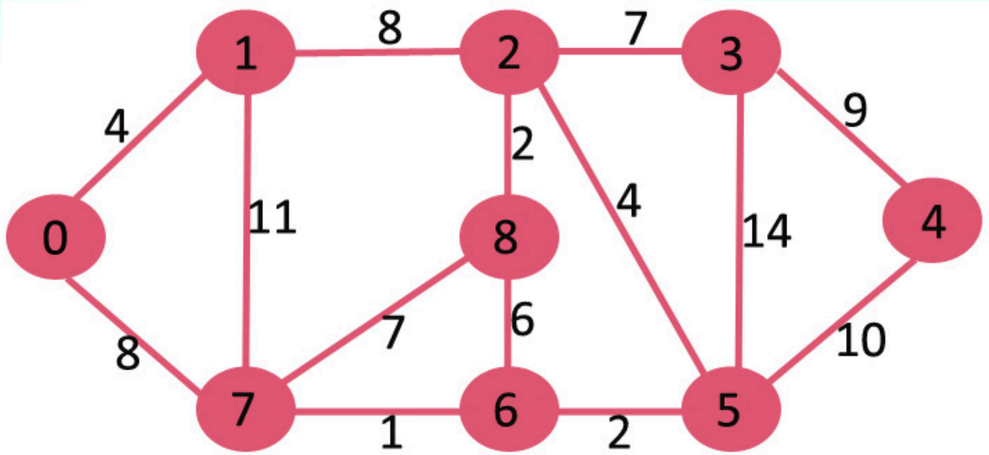
Dijkstra’s Algorithm

A weighted graph is a graph in which values area assigned to the edges and the length of a path in a weighted graph is the sum of the weight of the edges in the path.

The ALGORITHM

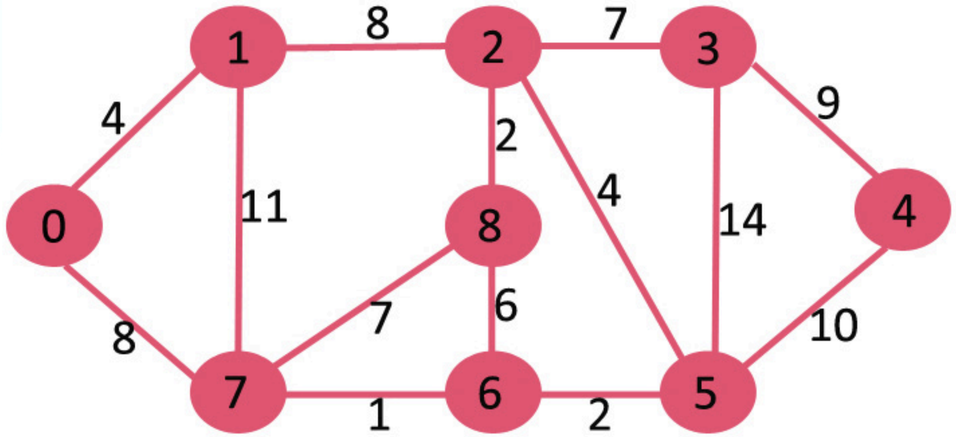
Consider the weighted graph G=(V,E) where each edge has a non-negative length. One of the vertex/node is the source vertex. Suppose, we are to determine a shortest path from a source vertex a to a destination vertex z. Let A and B be two sets of vertices denoting visited and unvisited vertices. Let A denote the set of visited vertices that have been already chosen and minimal distance from source vertex is already known for each vertex in A. The set B contains all other vertices whose minimal distance from the source is not yet known. Initially, a is the only vertex in A. At each step we add to A another vertex for which the shortest path from a has been determined.



Follow the steps below to solve the problem:

- Create a set **sptSet** (shortest path tree set) that keeps track of vertices included in the shortest-path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as **INFINITE**. Assign the distance value as 0 for the source vertex so that it is picked first.
- While **sptSet** doesn't include all vertices
  - Pick a vertex **u** which is not there in **sptSet** and has a minimum distance value.
  - Include u to **sptSet**.
  - Then update distance value of all adjacent vertices of u.
    - To update the distance values, iterate through all adjacent vertices.
    - For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

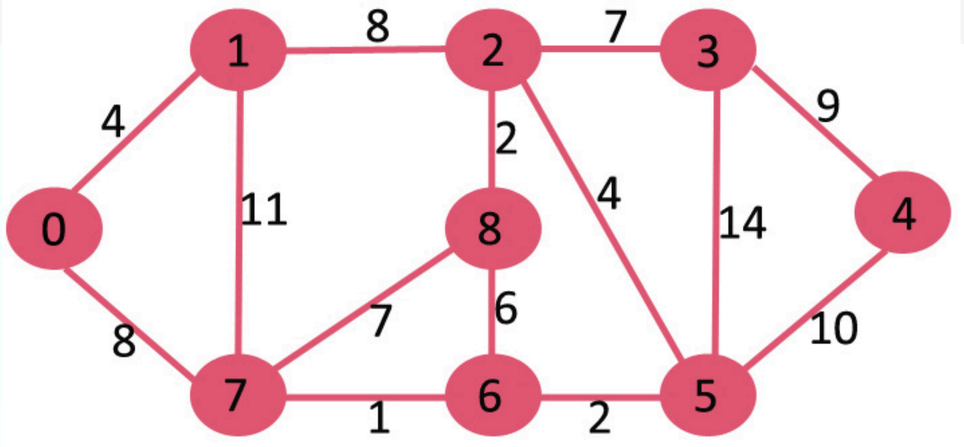
Find the shortest path from source to all nodes of the graph given in figure using Dijkstra's Algorithm.

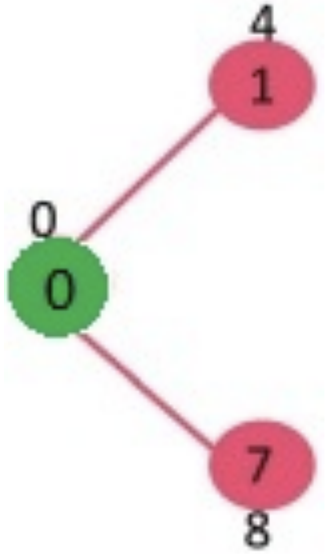


Step 1:

- *The set **sptSet** is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where **INF** indicates infinite.*
- *Now pick the vertex with a minimum distance value. The vertex 0 is picked, include it in sptSet. So sptSet becomes {0}. After including 0 to sptSet, update distance values of its adjacent vertices.*
- *Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8.*

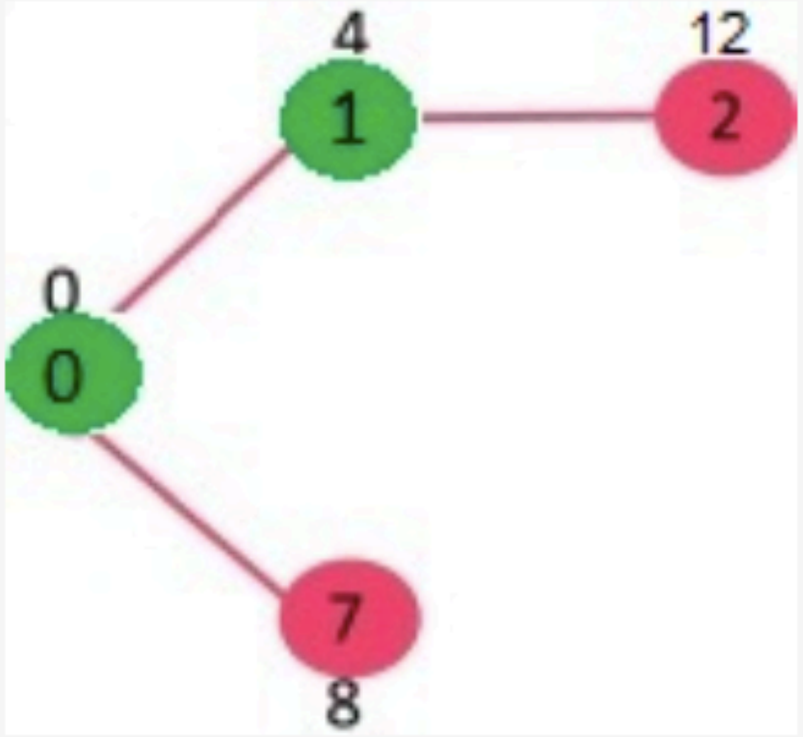
*The following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in **green** colour.*

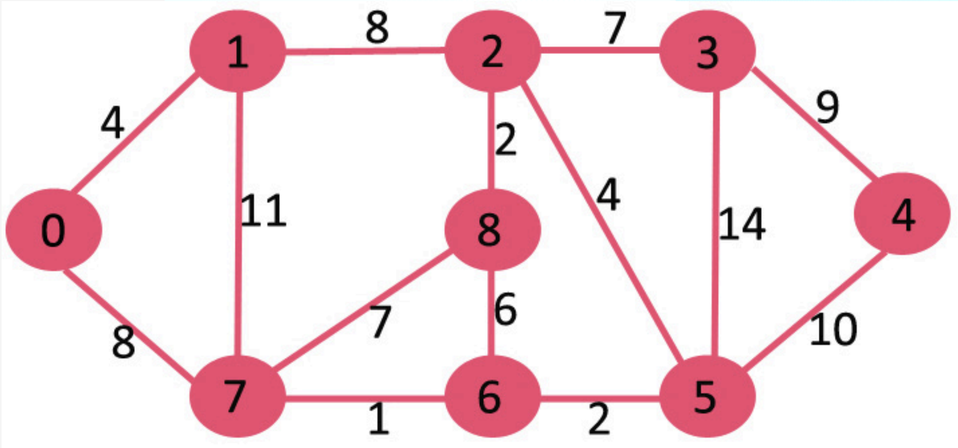




Step 2:

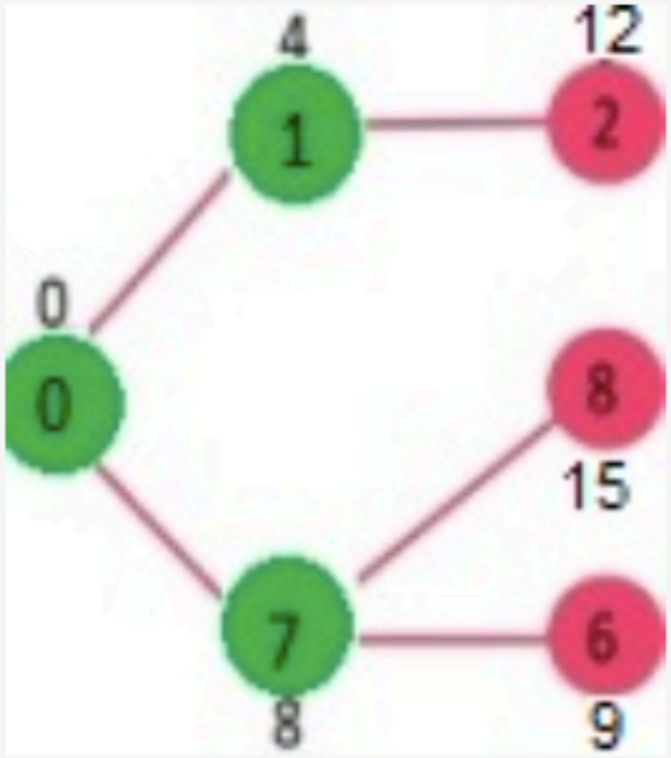
- *Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet.*
- *So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1.*
- *The distance value of vertex 2 becomes **12**.*

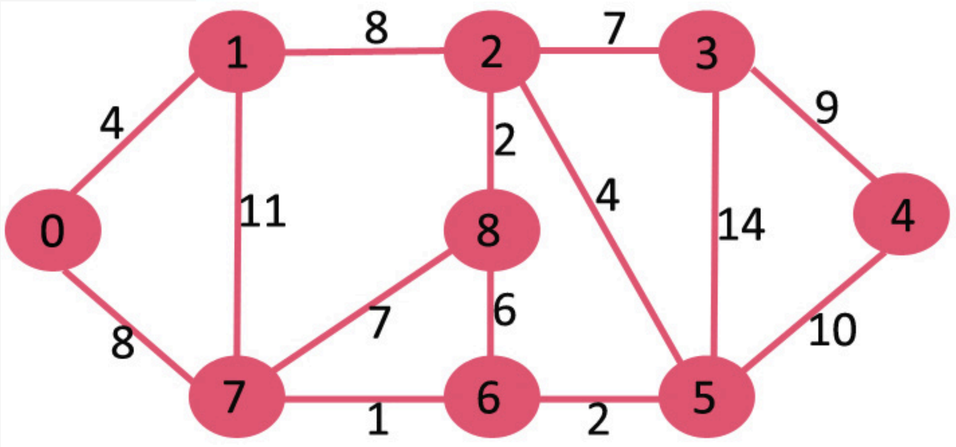




Step 3:

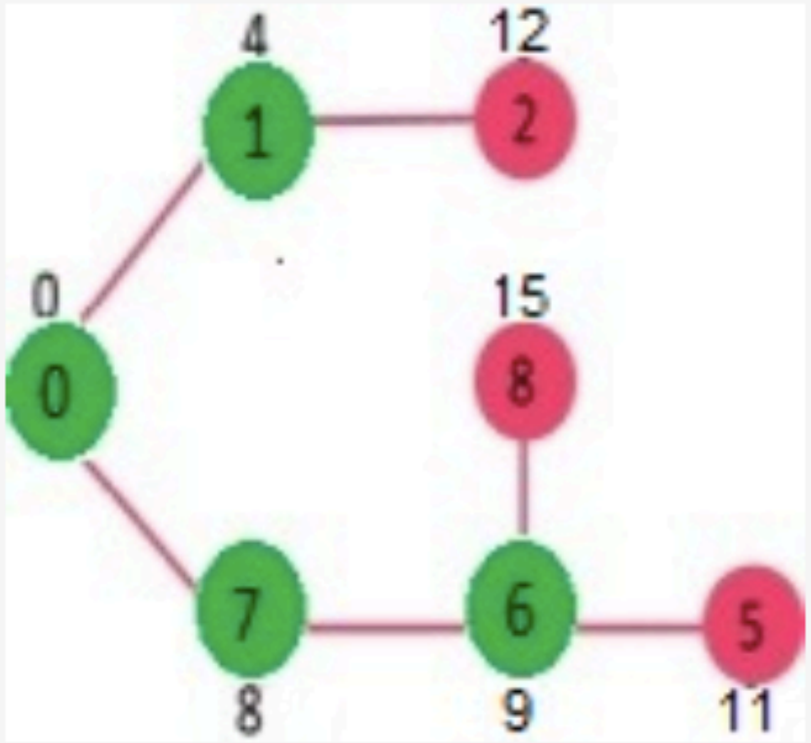
- *Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes* ***{0, 1, 7}.***
- *Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (****15 and 9*** *respectively).*

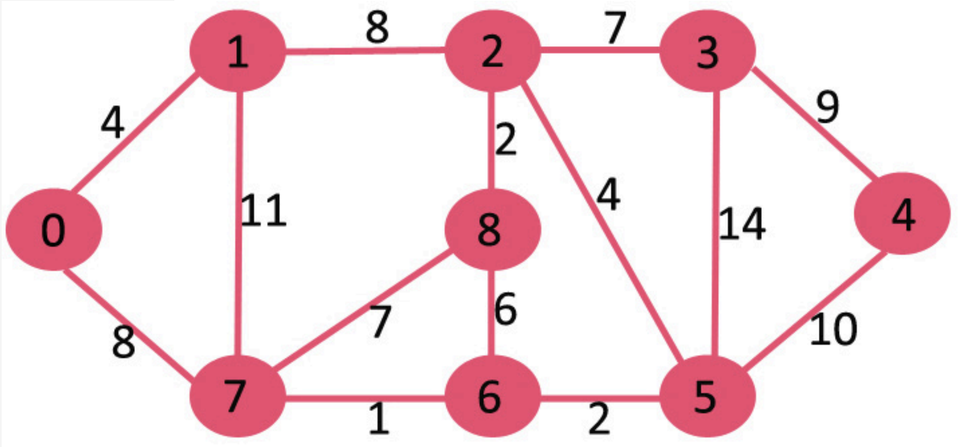




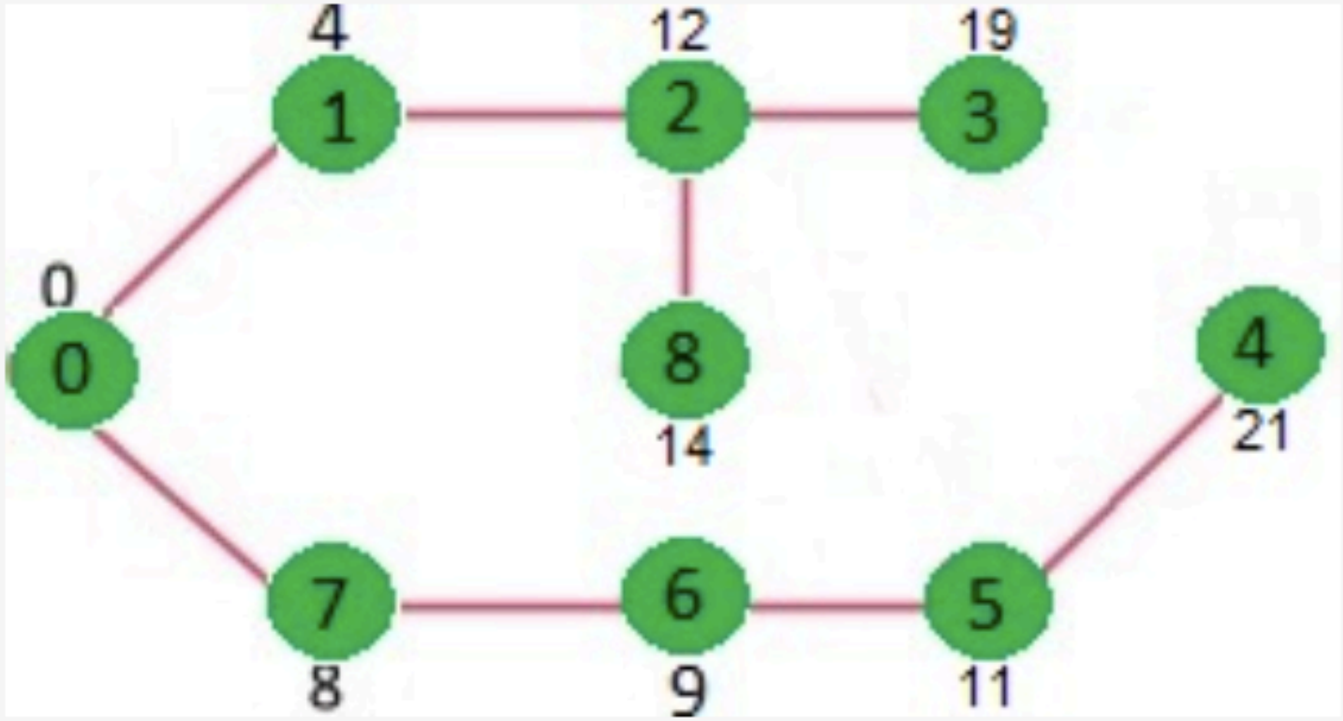
Step 4:

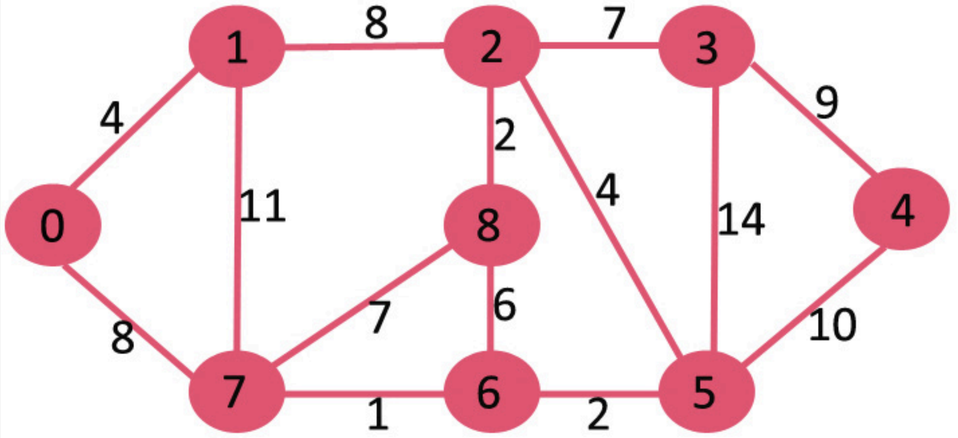
- *Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes* ***{0, 1, 7, 6}****.*
- *Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.*





*We repeat the above steps until sptSet **includes** all vertices of the given graph. Finally, we get the following Shortest Path Tree (SPT).*



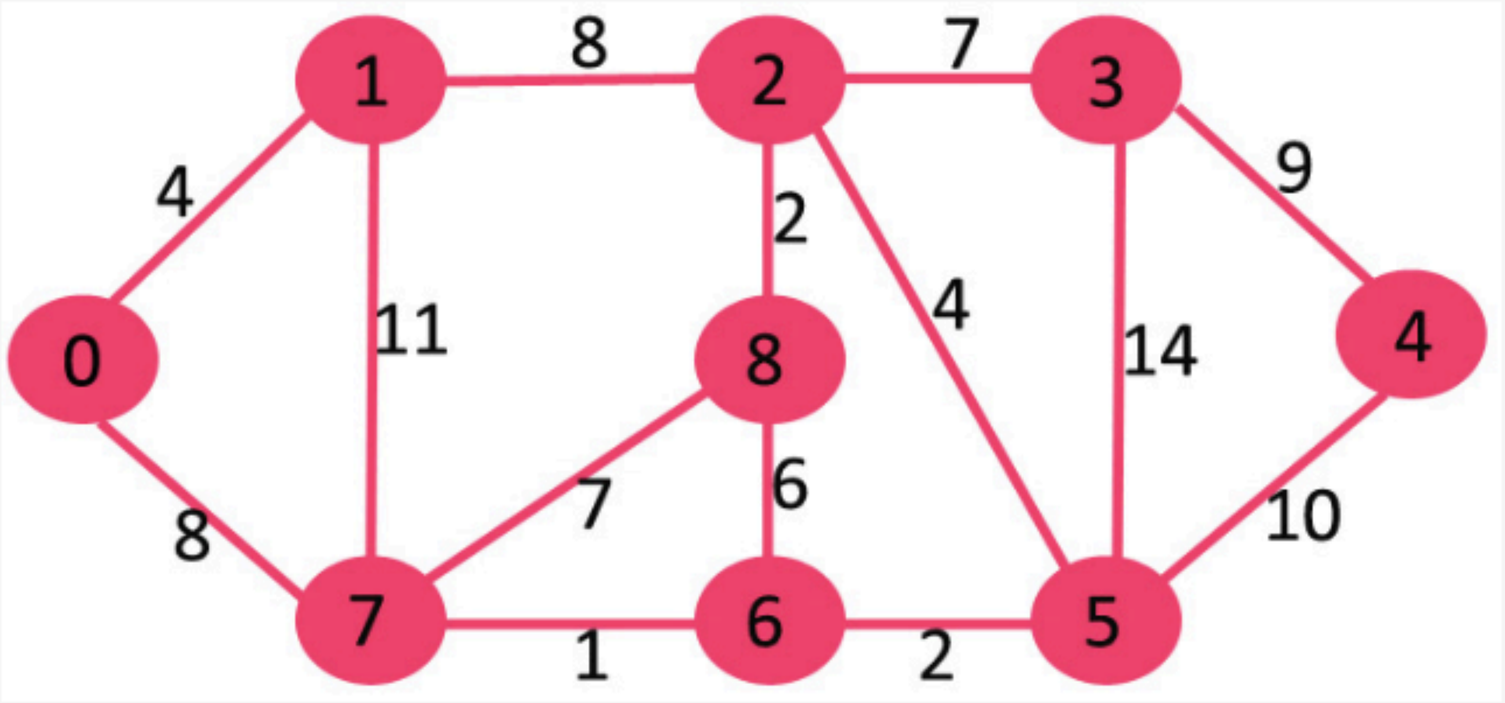


Follow the given steps to find MST using Prim's Algorithm:

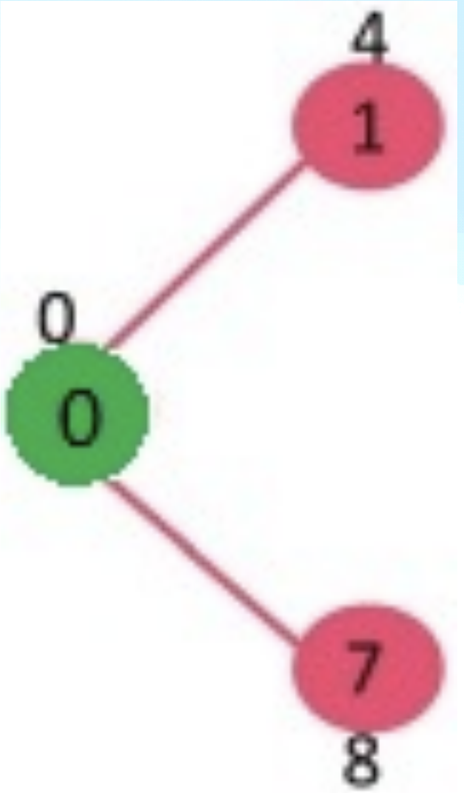
MST :=Minimum Spanning Tree

- Create a set *mstSet* that keeps track of vertices already included in MST.
- Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign the key value as 0 for the first vertex so that it is picked first.
- While mstSet doesn't include all vertices
  - Pick a vertex *u* which is not there in *mstSet* and has a minimum key value.
  - Include *u in the* mstSet.
  - Update the key value of all adjacent vertices of *u*. To update the key values, iterate through all adjacent vertices. For every adjacent vertex *v*, if the weight of edge *u*-*v* is less than the previous key value of *v*, update the key value as the weight of *u*-*v*

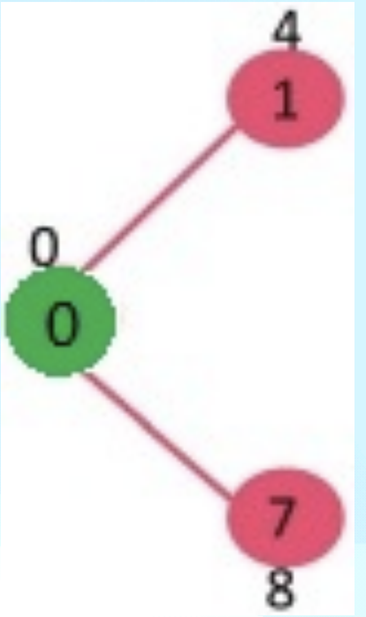
Input graph:

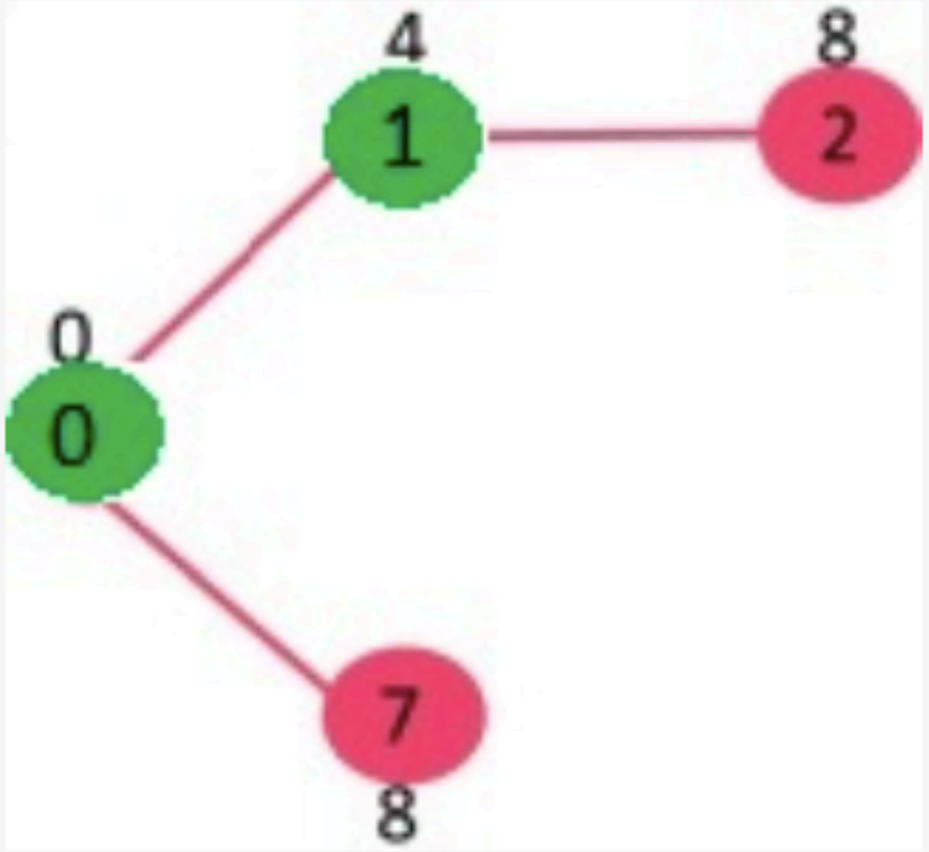


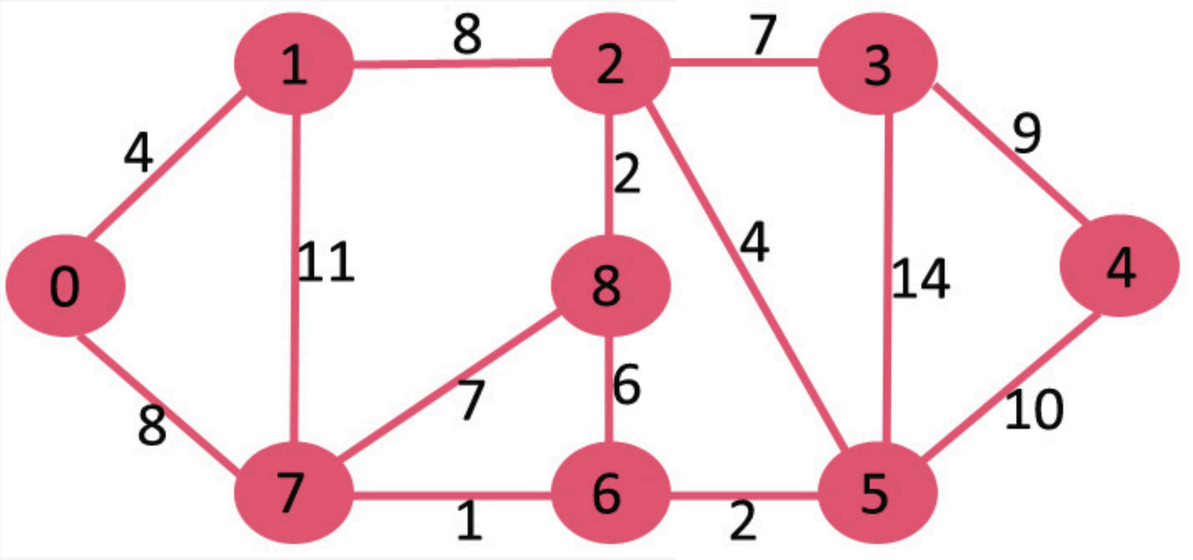
Step 1: *The set mstSet is initially empty and keys assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with the minimum key value. The vertex 0 is picked, include it in mstSet. So mstSet becomes {0}. After including it to mstSet, update key values of adjacent vertices. Adjacent vertices of 0 are 1 and 7. The key values of 1 and 7 are updated as 4 and 8. Following subgraph shows vertices and their key values, only the vertices with finite key values are shown. The vertices included in MST are shown in green color.*



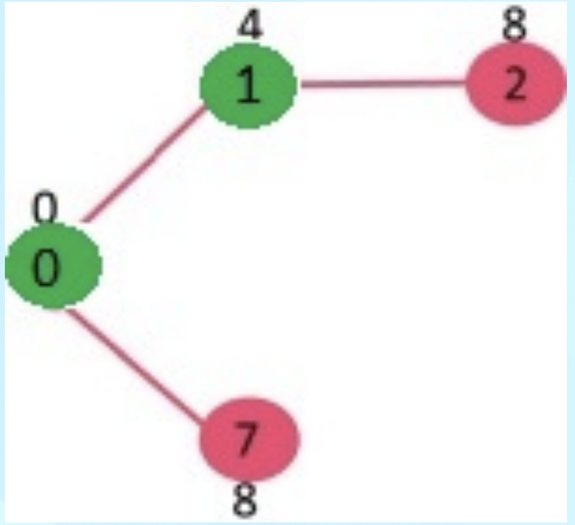
Step 2: *Pick the vertex with minimum key value and which is not already included in the MST (not in mstSET). The vertex 1 is picked and added to mstSet. So mstSet now becomes {0, 1}. Update the key values of adjacent vertices of 1. The key value of vertex 2 becomes 8.*

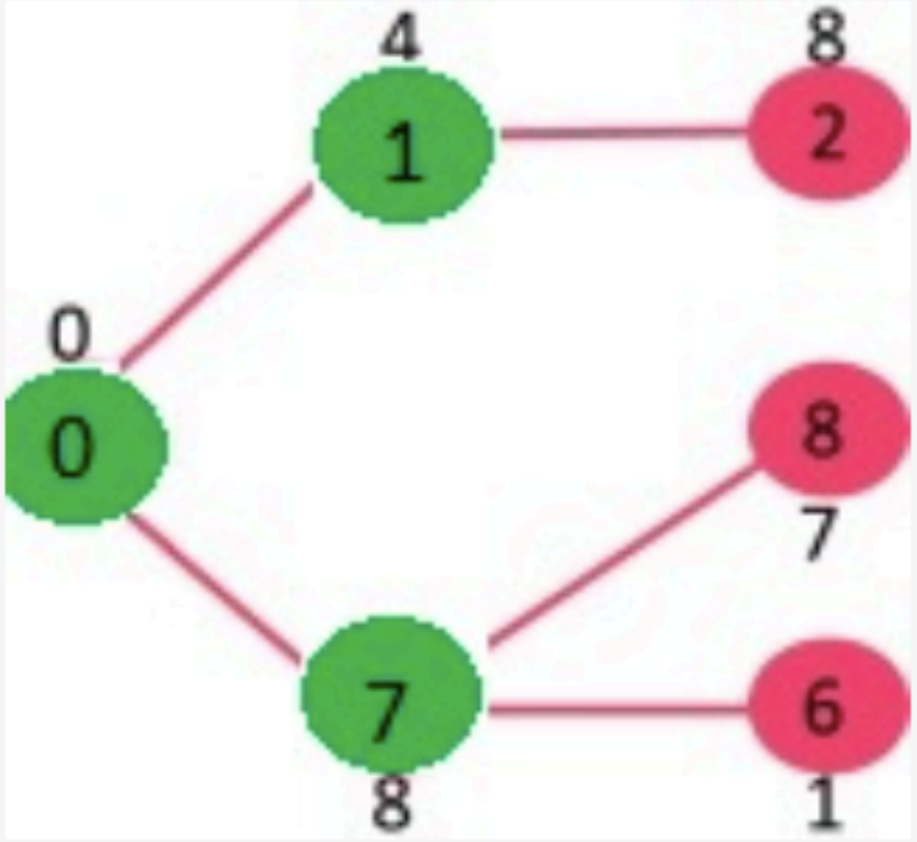


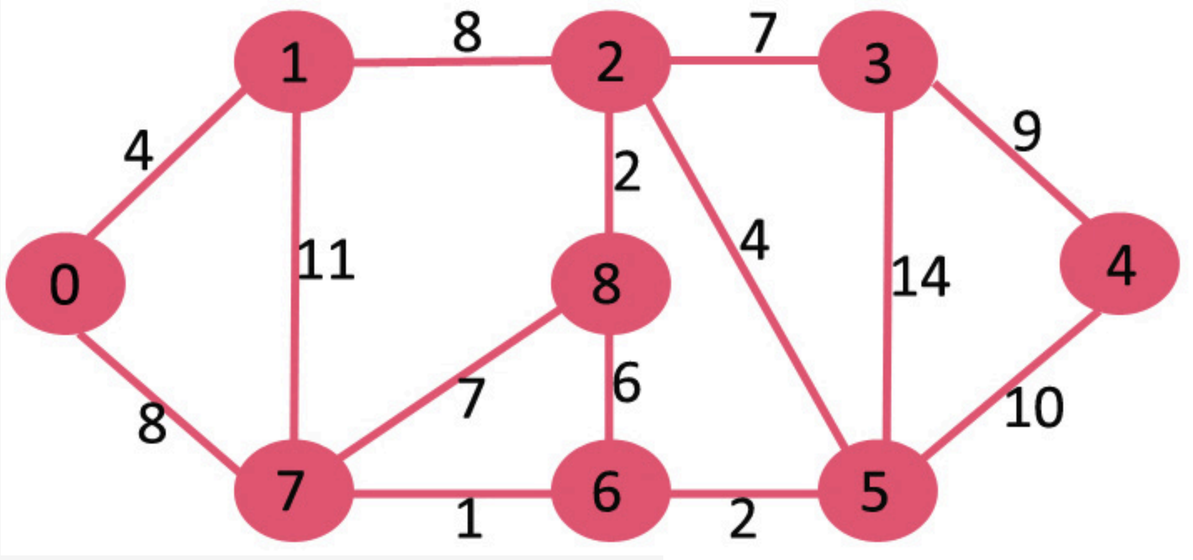




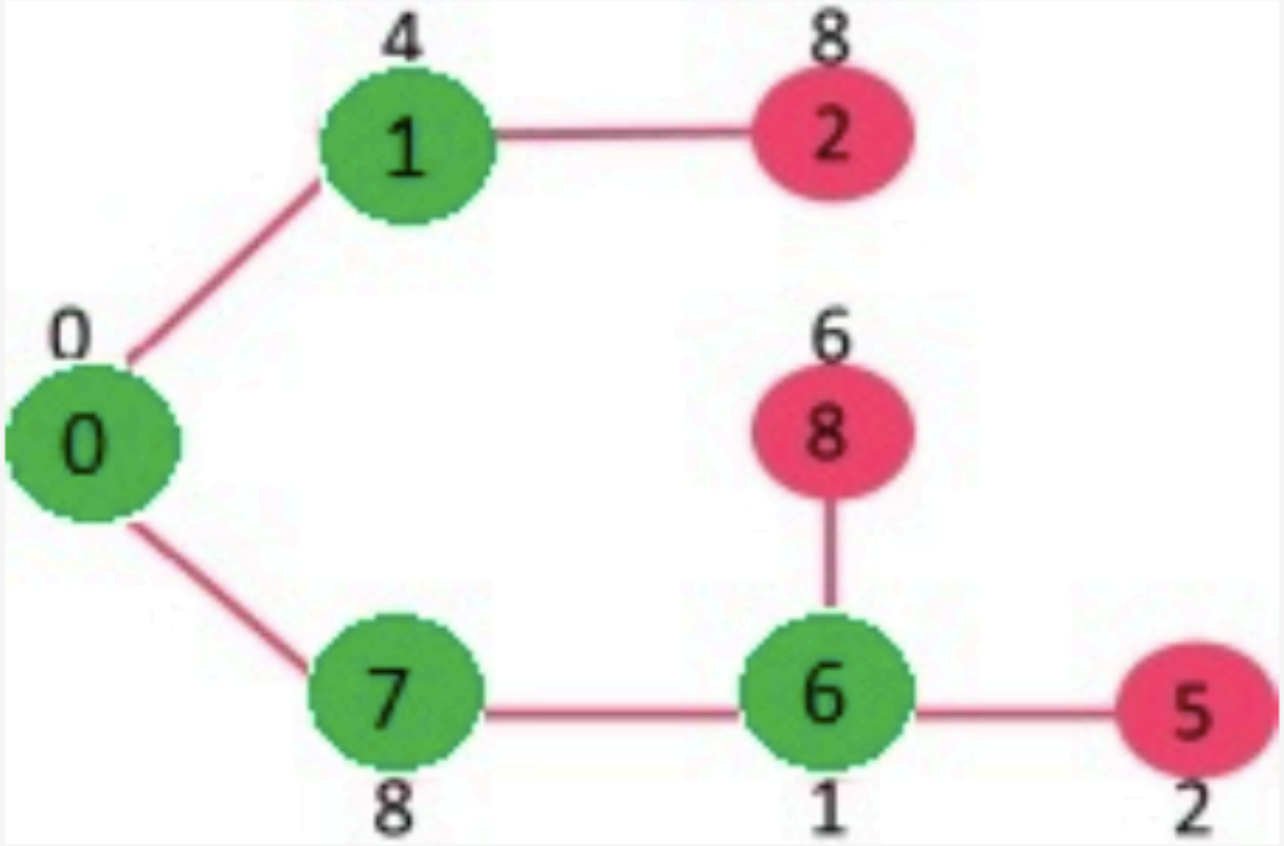
Step 3: *Pick the vertex with minimum key value and which is not already included in the MST (not in mstSET). We can either pick vertex 7 or vertex 2, let vertex 7 is picked. So mstSet now becomes {0, 1, 7}. Update the key values of adjacent vertices of 7. The key value of vertex 6 and 8 becomes finite (1 and 7 respectively).*

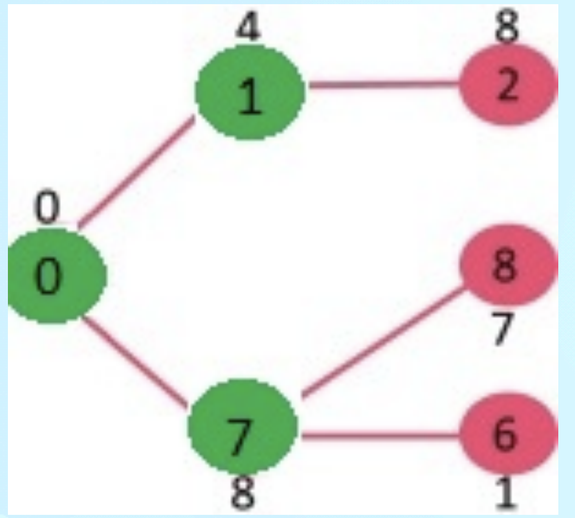


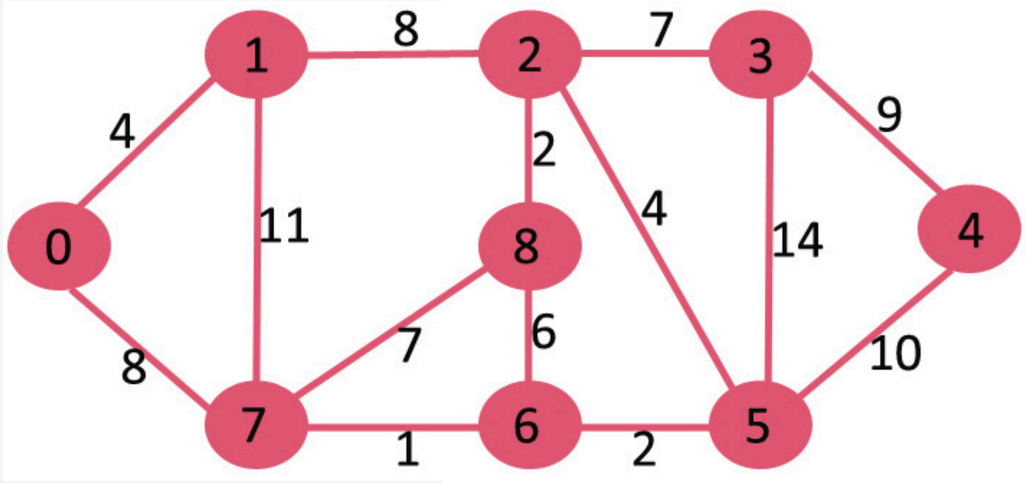




Step 4: *Pick the vertex with minimum key value and not already included in MST (not in mstSET). Vertex 6 is picked. So mstSet now becomes {0, 1, 7, 6}. Update the key values of adjacent vertices of 6. The key value of vertex 5 and 8 are updated.*

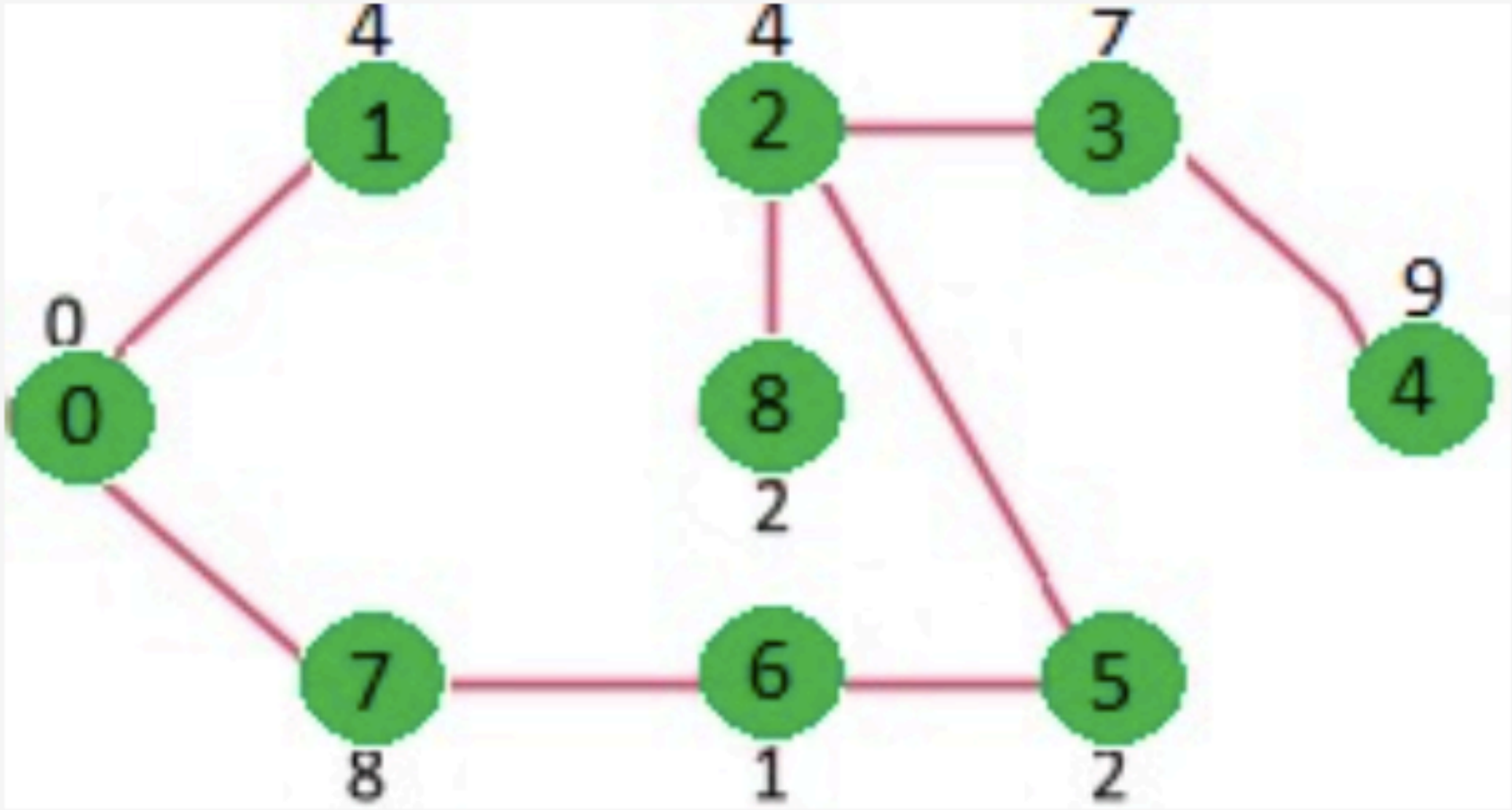






Step 5: *Repeat the above steps until mstSet includes all vertices of given graph.*

Finally, we get the following graph.



How to find MST using Kruskal's algorithm?

Below are the steps for finding MST using Kruskal's algorithm:

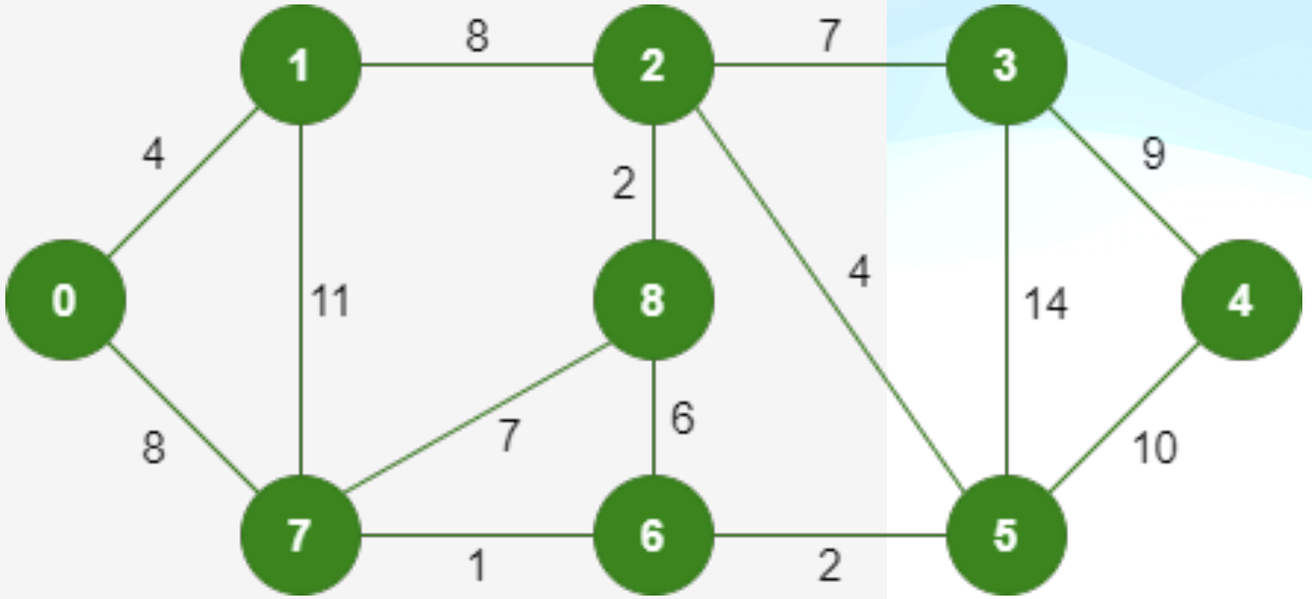
1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are (V-1) edges in the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example:

The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having (9 – 1) = 8 edges.

After sorting:

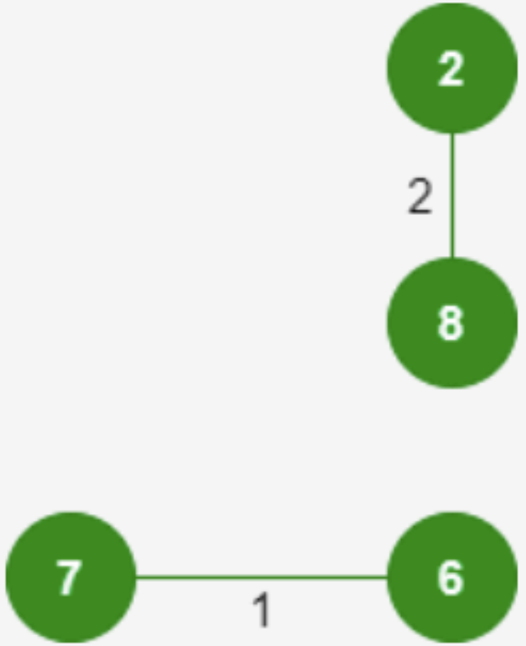
Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5



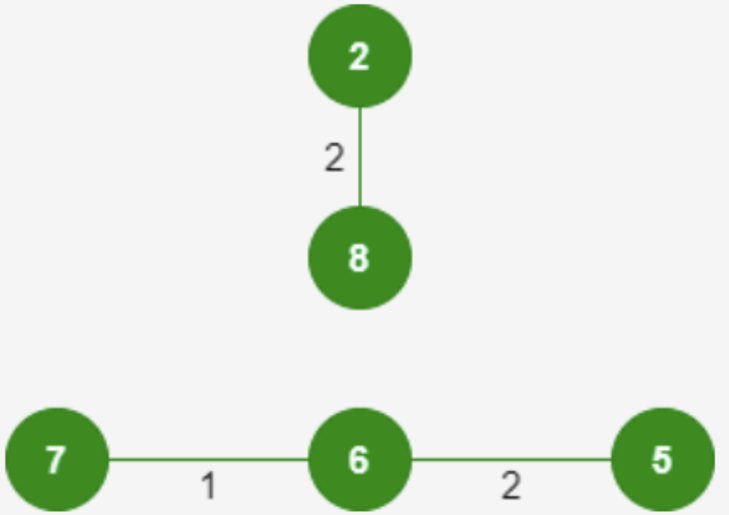
Now pick all edges one by one from the sorted list of edges

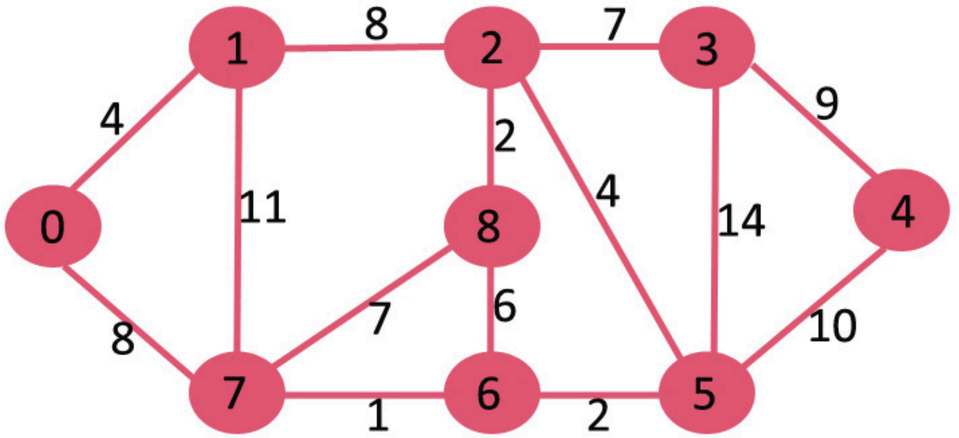
Step 1: *Pick edge 7-6: No cycle is formed, include it.*

Step 2: *Pick edge 8-2: No cycle is formed, include it.*

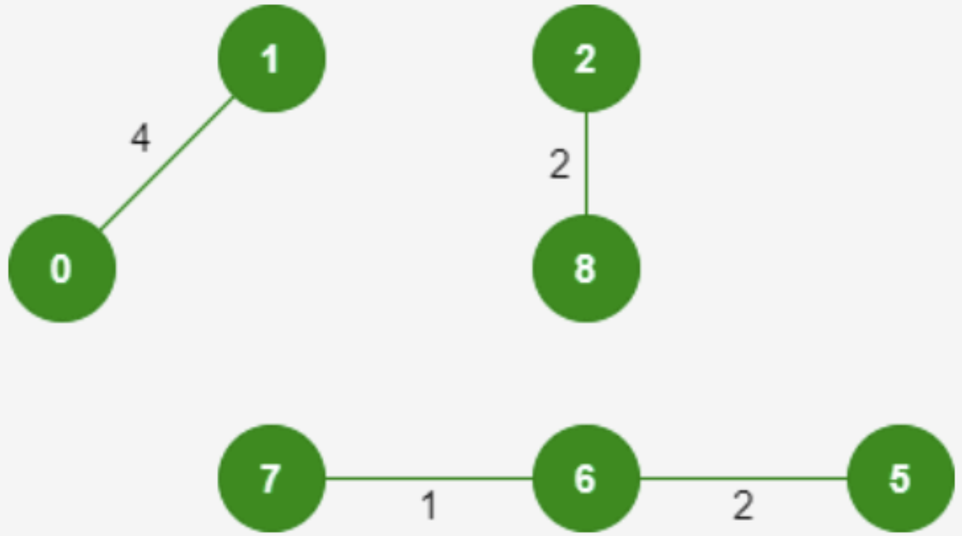


Step 3: *Pick edge 6-5: No cycle is formed, include it.*

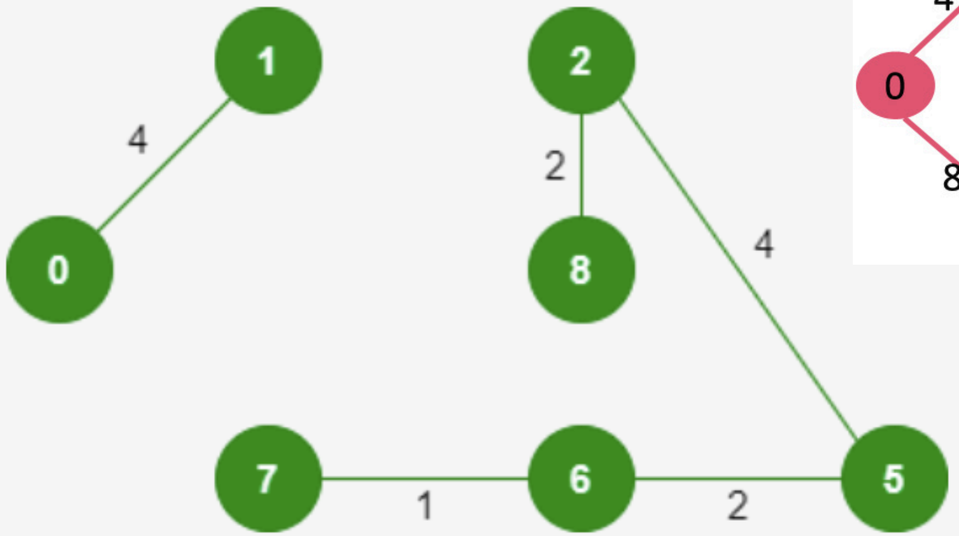


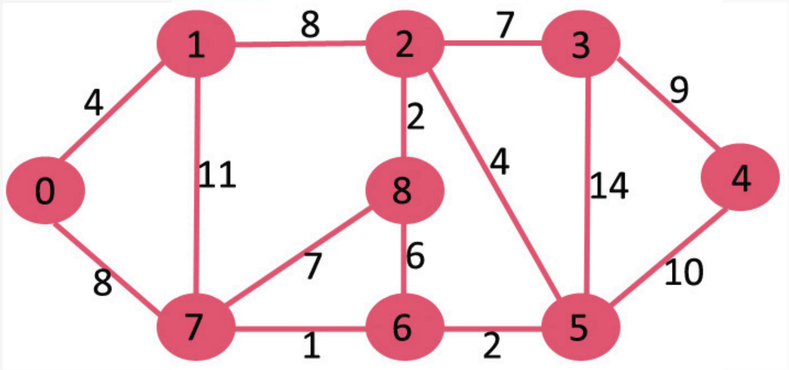


Step 4: *Pick edge 0-1: No cycle is formed, include it.*



Step 5: *Pick edge 2-5: No cycle is formed, include it.*



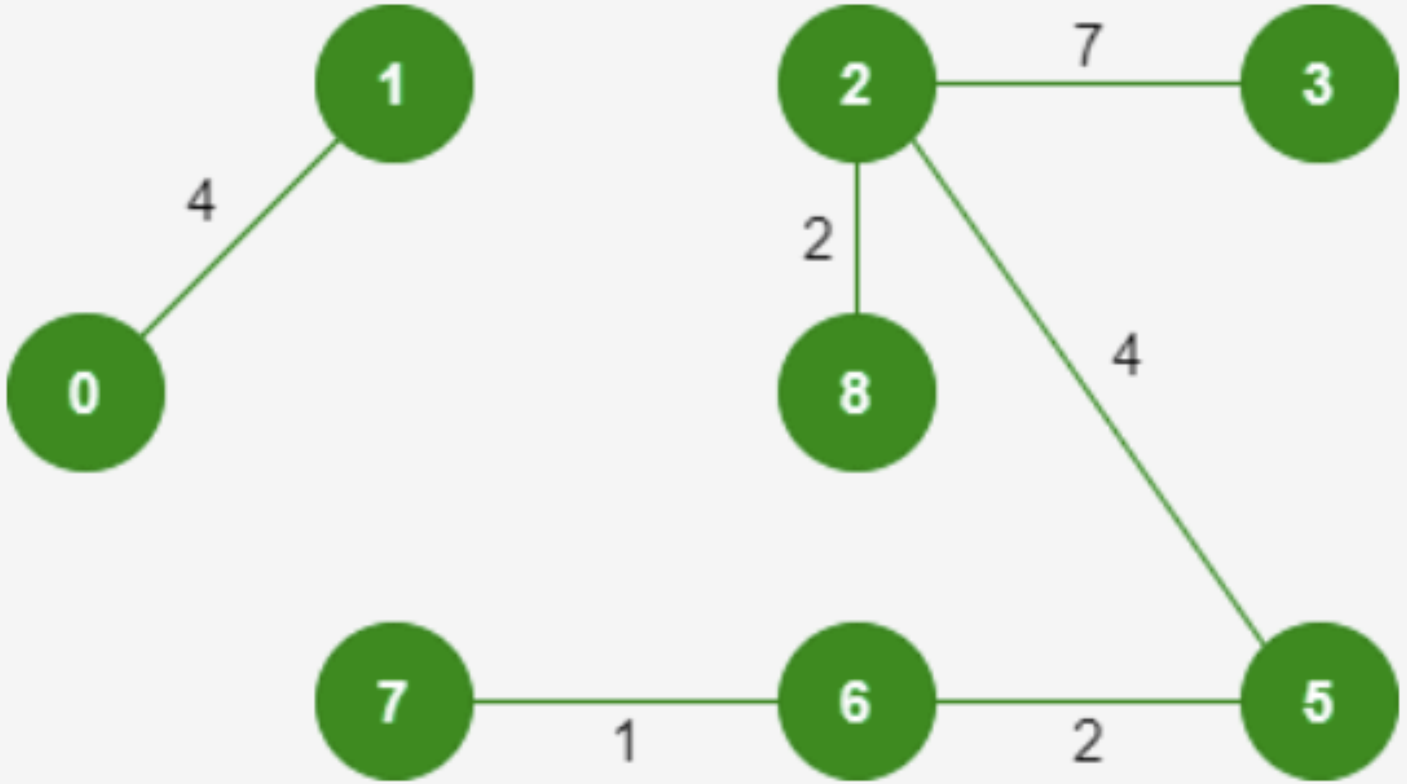


After sorting:

Weight	*Src*	*Dest*
1	*7*	*6*
2	*8*	*2*
2	*6*	*5*
4	*0*	*1*
4	*2*	*5*
6	*8*	*6*
7	*2*	*3*
7	*7*	*8*
8	*0*	*7*
8	*1*	*2*
9	*3*	*4*
10	*5*	*4*
11	*1*	*7*
14	*3*	*5*

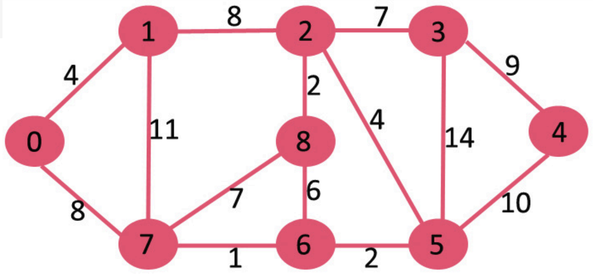
Step 6: *Pick edge 8-6: Since including this edge results in the cycle, discard it.*

Step 7: *Pick edge 2-3: No cycle is formed, include it.*



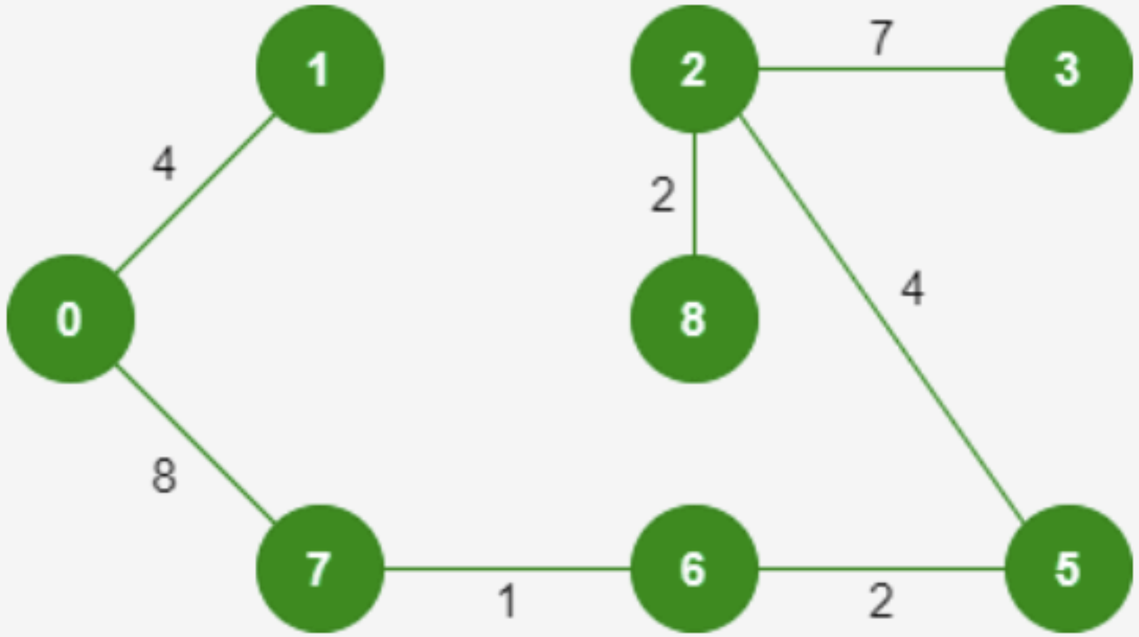
After sorting:

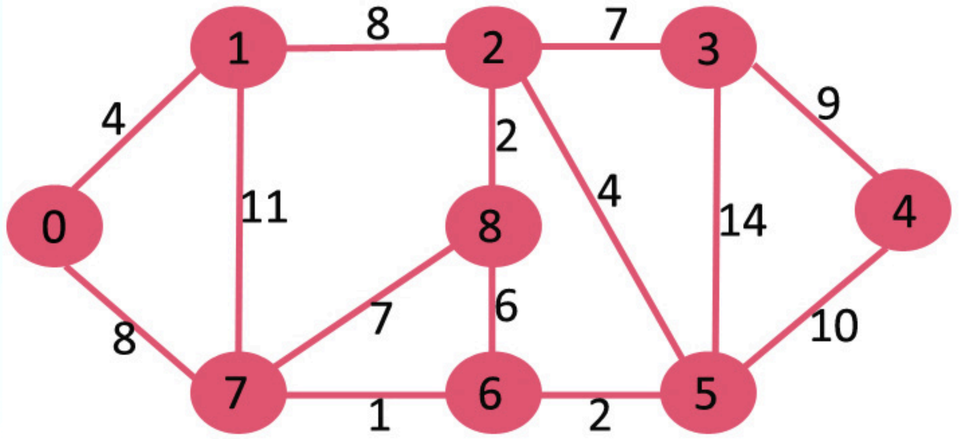
Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5



Step 8: *Pick edge 7-8: Since including this edge results in the cycle, discard it.*

Step 9: *Pick edge 0-7: No cycle is formed, include it.*



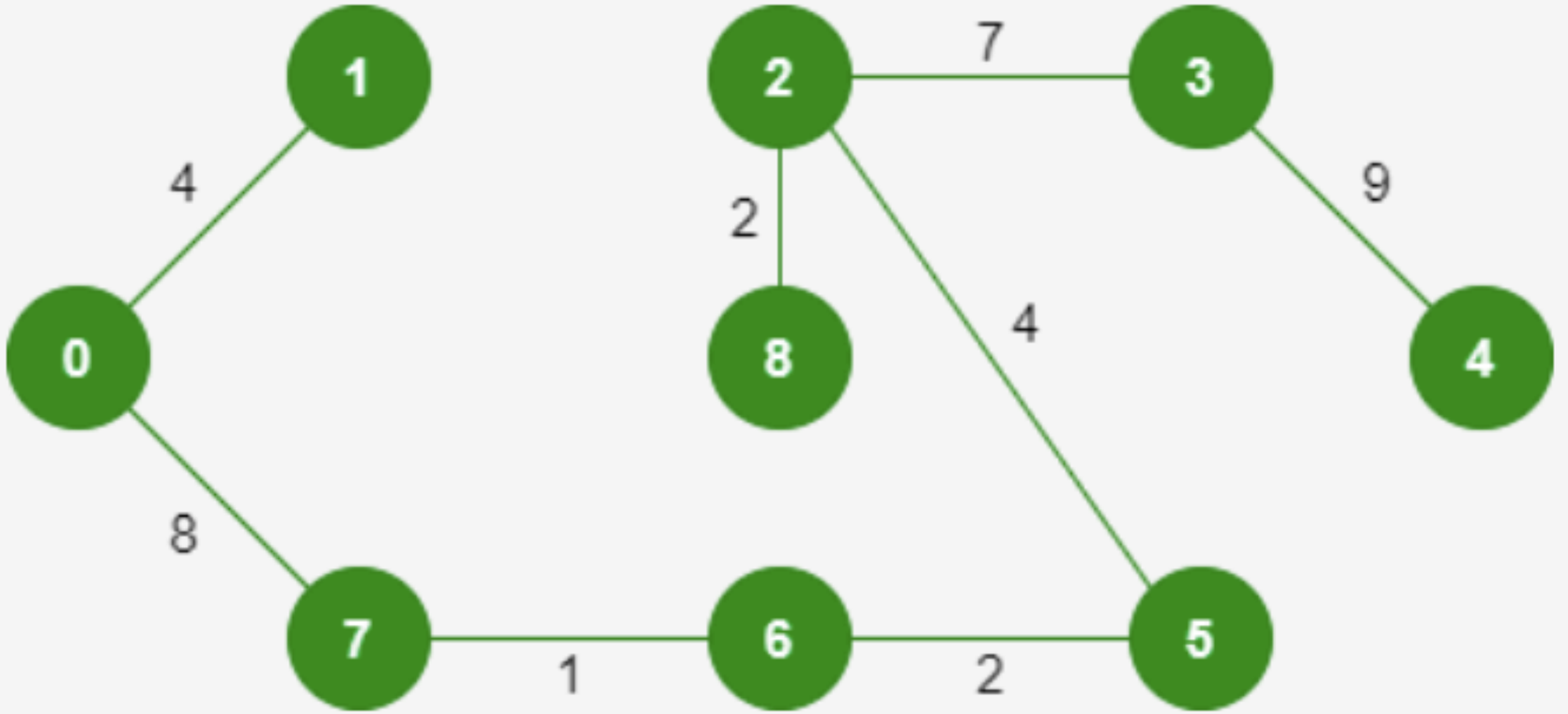


After sorting:

Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Step 10: *Pick edge 1-2: Since including this edge results in the cycle, discard it.*

Step 11: *Pick edge 3-4: No cycle is formed, include it.*



Note: *Since the number of edges included in the MST equals to (V - 1), so the algorithm stops here.*

After sorting:

Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5