Prim's Algorithm

Mashiwat Tabassum Waishy

Lecturer

Department *of*

CSE

STAMFORD UNIVERSITY BANGLADESH

Prim's Algorithm

- Prim's algorithm builds the MST by adding **leaves one at a time** to the current tree
- We start with a root vertex r: it can be any vertex
- At any time, the subset of edges A forms a **single tree**(in Kruskal it formed a forest)

Prim's Algorithm

- While making MST, our graph is divided in **S** and **V-S**
- The Big Question for this algorithm is: "How to decide a node from **V-S** to include in **S**"

Prim's Algorithm

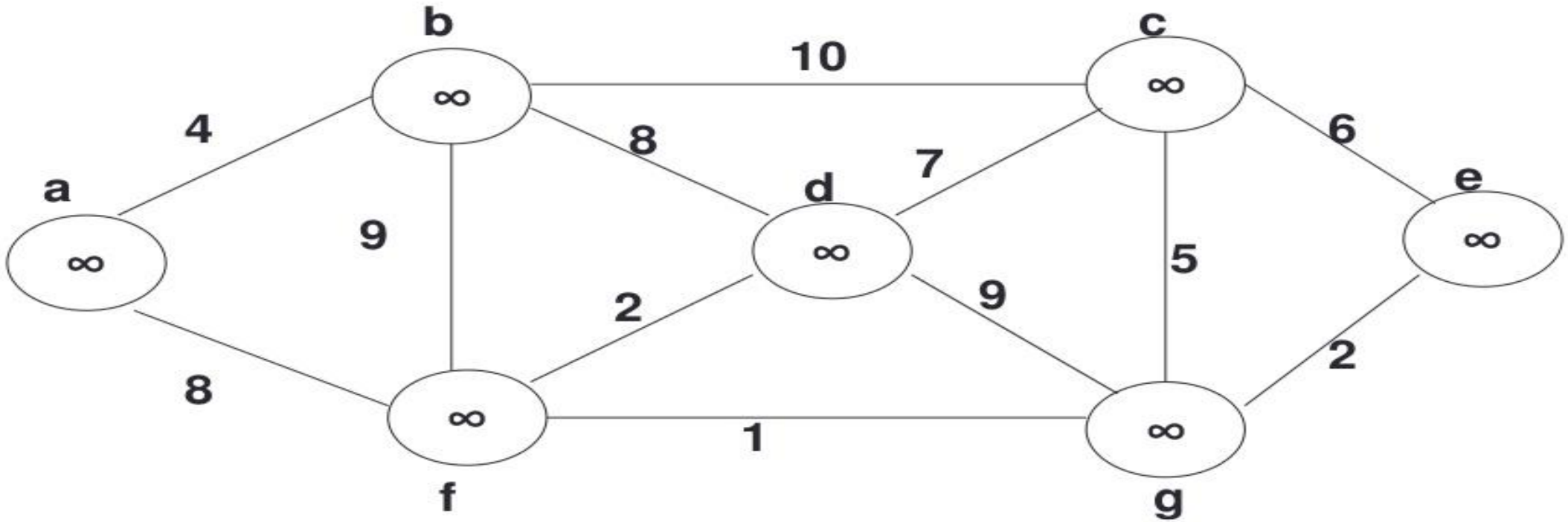
- We also need to know which vertices are in **S** and which are not
- To do this we will assign color to each vertex
- If the color of vertex is black then it is in **S** otherwise if the color of vertex is white it is in **V-S**

Prim's Algorithm

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

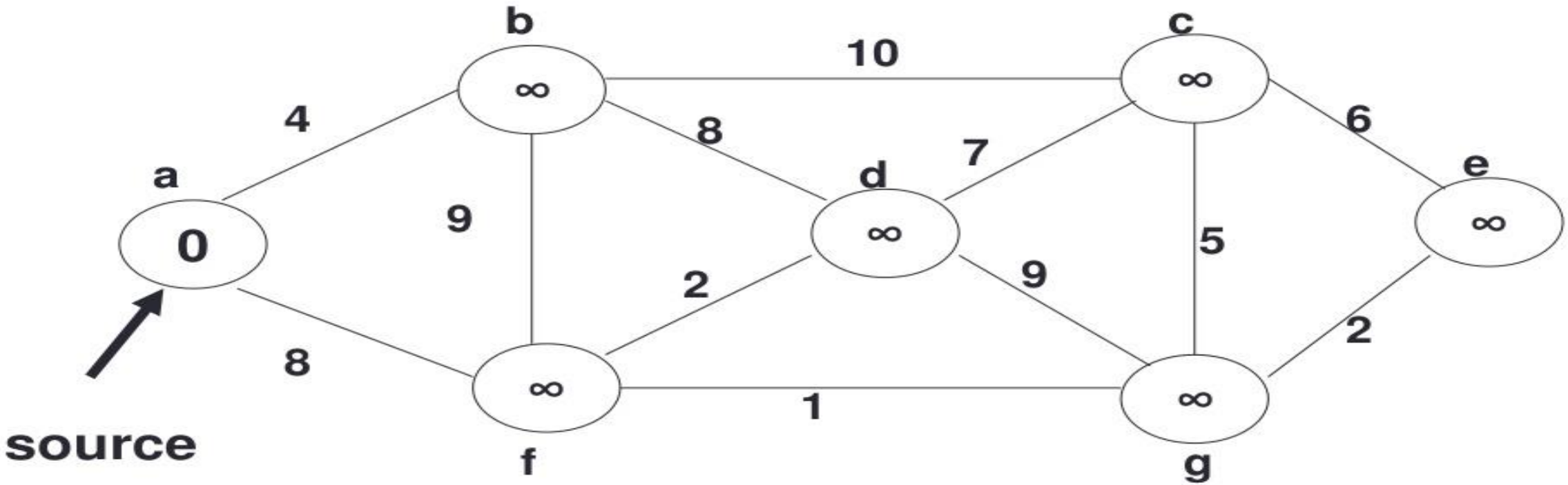
- Grow the minimum spanning tree from the root vertex r.
- Q is a priority queue, holding all vertices that are not in the tree now.
- key[v] is the minimum weight of any edge connecting v to a vertex in the tree.
- Π[v] names the parent of v in the tree.
- When the algorithm terminates, Q is empty; the minimum spanning tree A for G is thus $A=\{(v,parent[v]):v\in V-\{r\}\}$.
- Running time: $O(||E||\lg |V|)$.

Prim's Algorithm



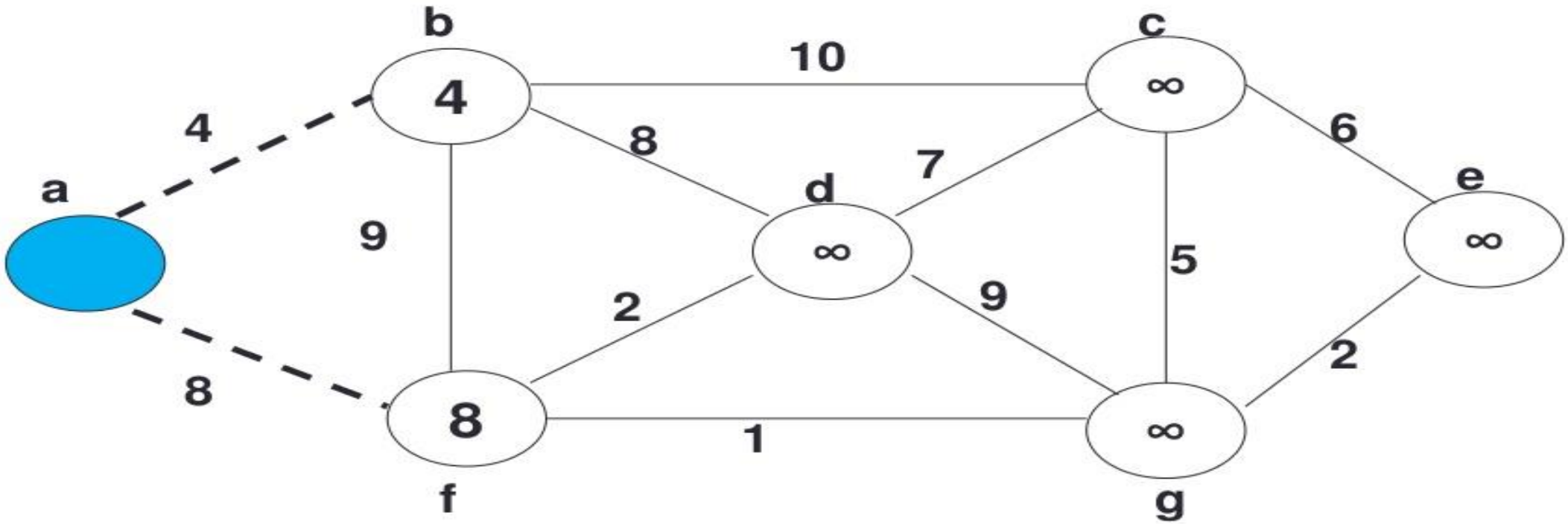
Prim's Algorithm

PQ: 0,∞,∞,∞,∞, ∞,



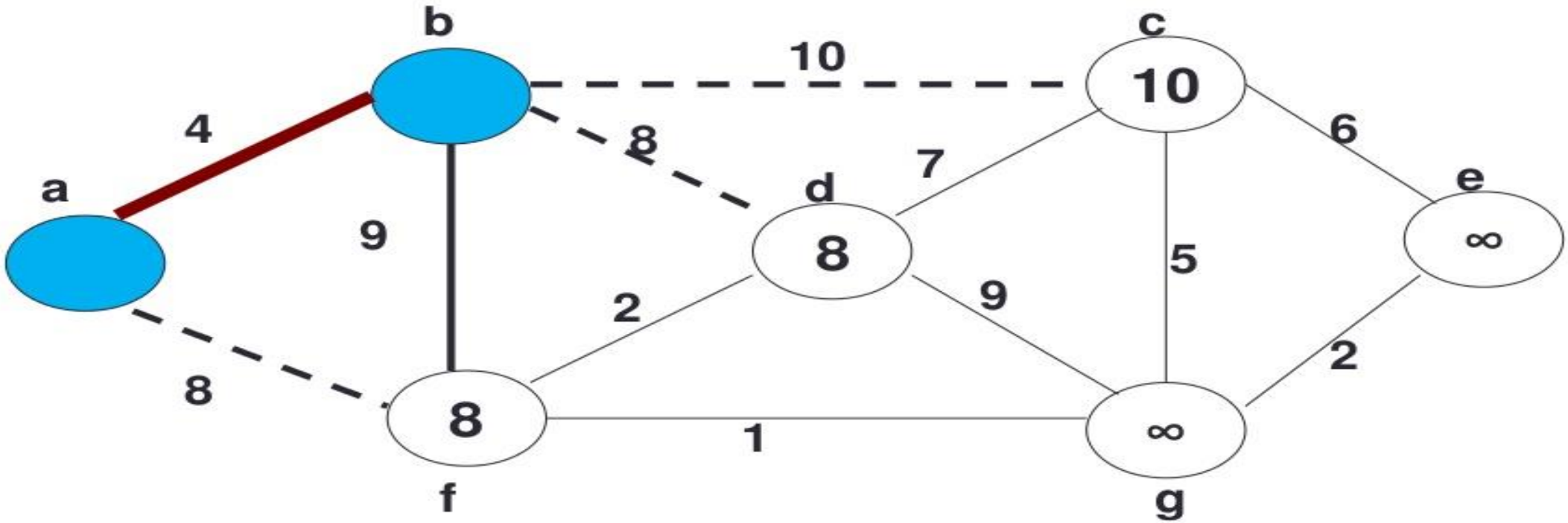
Prim's Algorithm

PQ: 4,8,∞,∞,∞,∞,



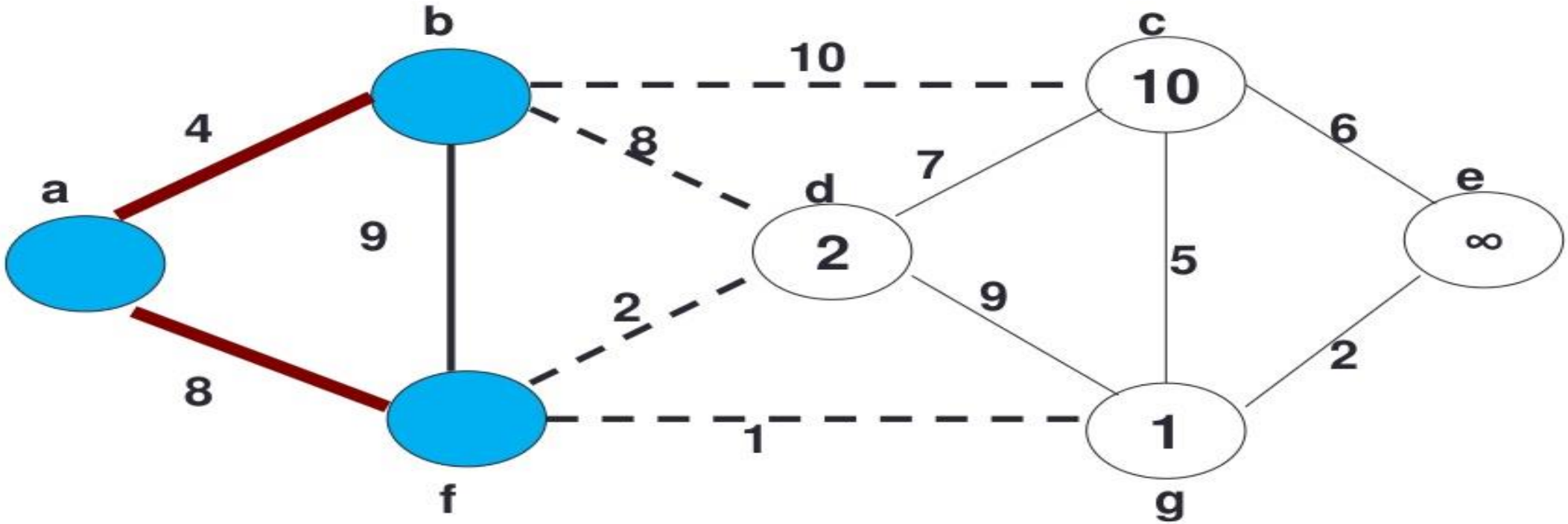
Prim's Algorithm

PQ: 8,8,10,∞,∞,



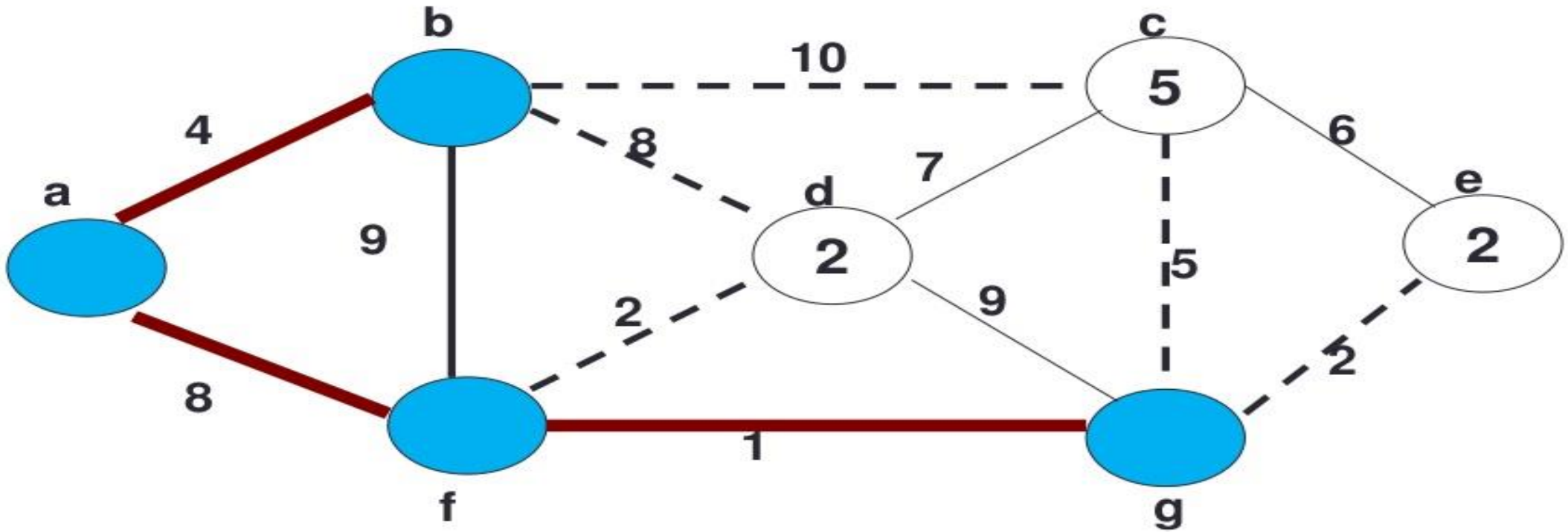
Prim's Algorithm

PQ: 1,2,10,∞



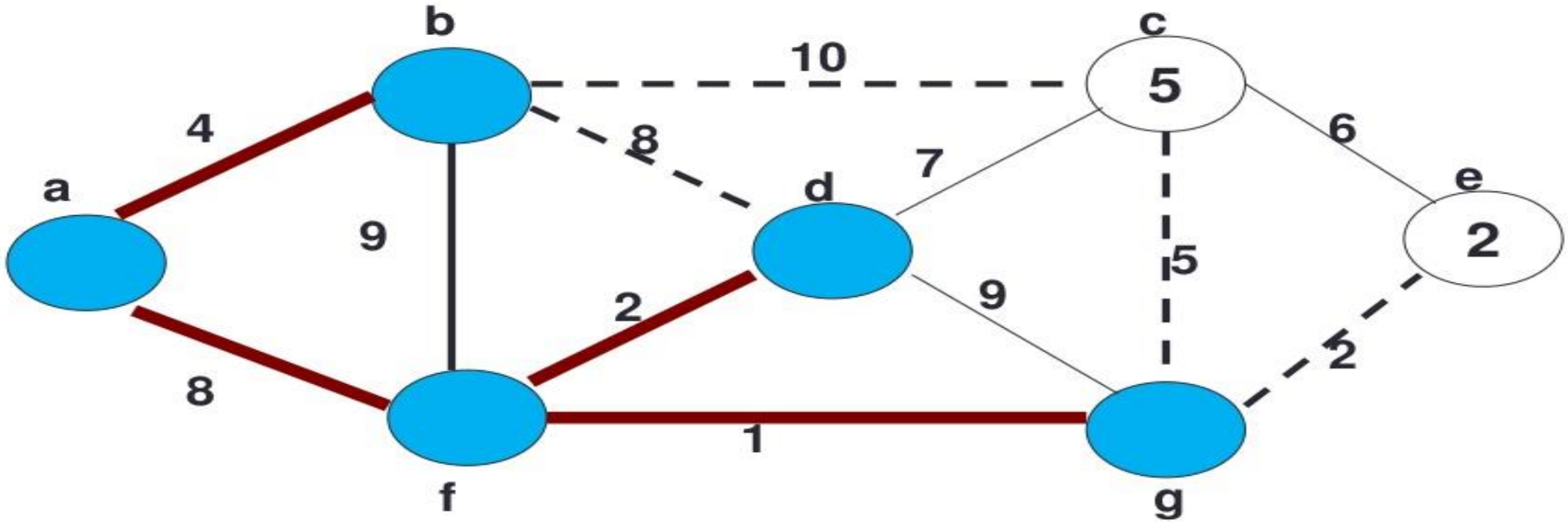
Prim's Algorithm

PQ: 2,2,5



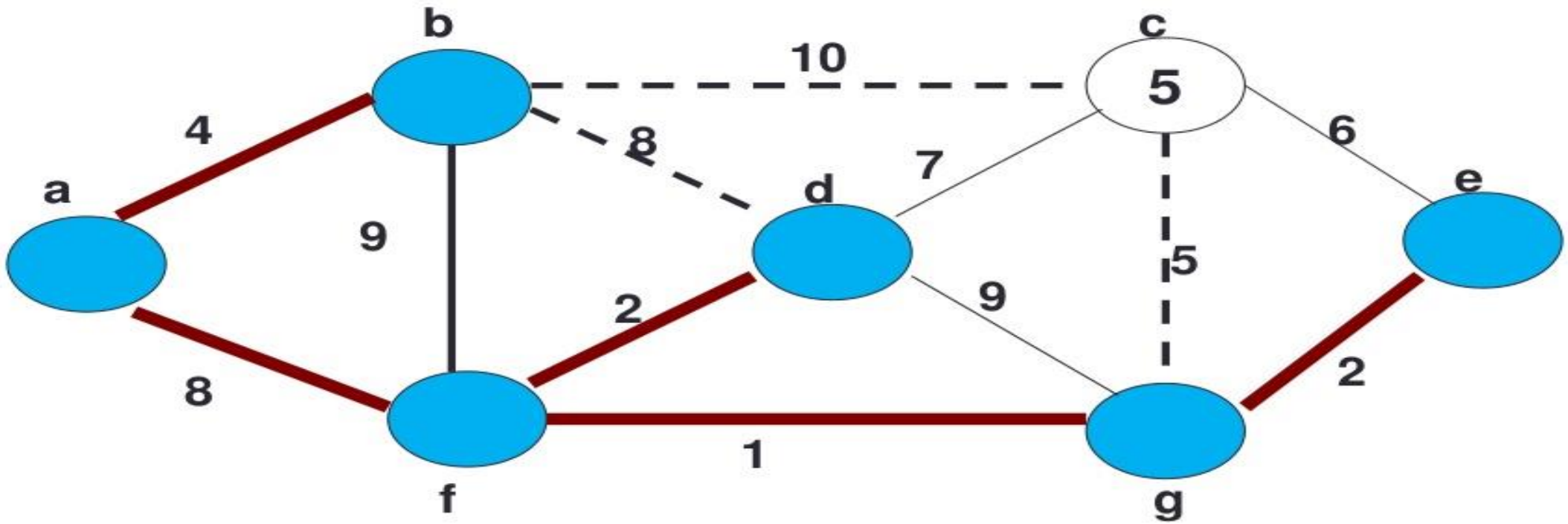
Prim's Algorithm

PQ: 2,5



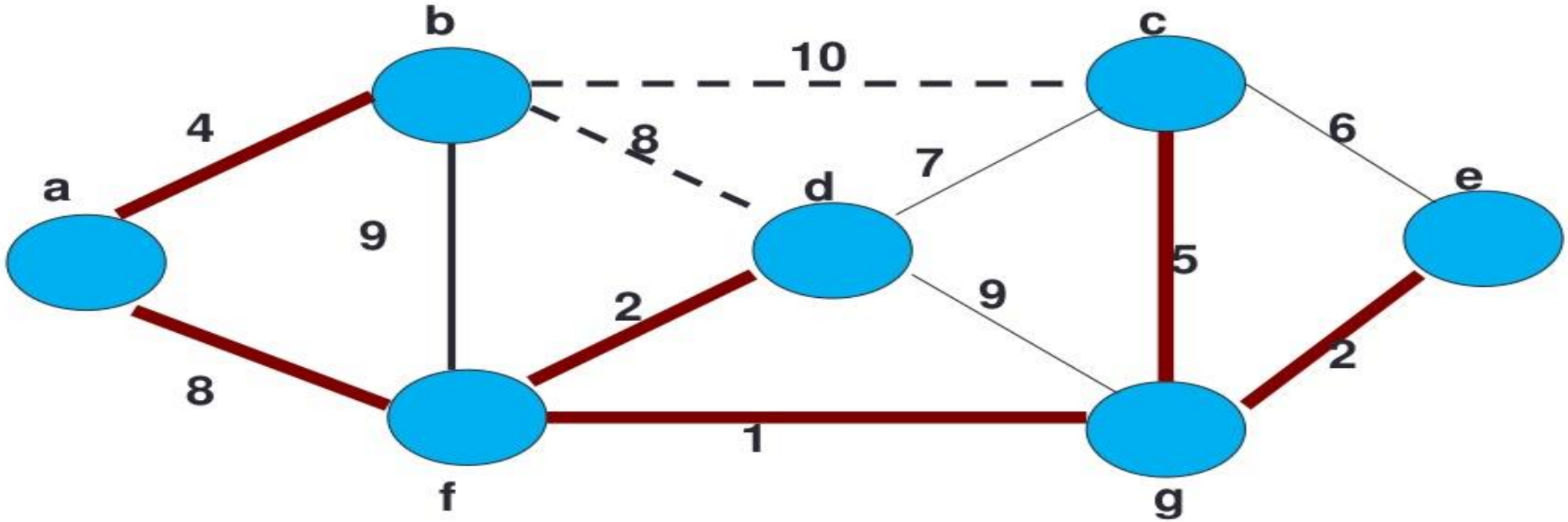
Prim's Algorithm

PQ: 5



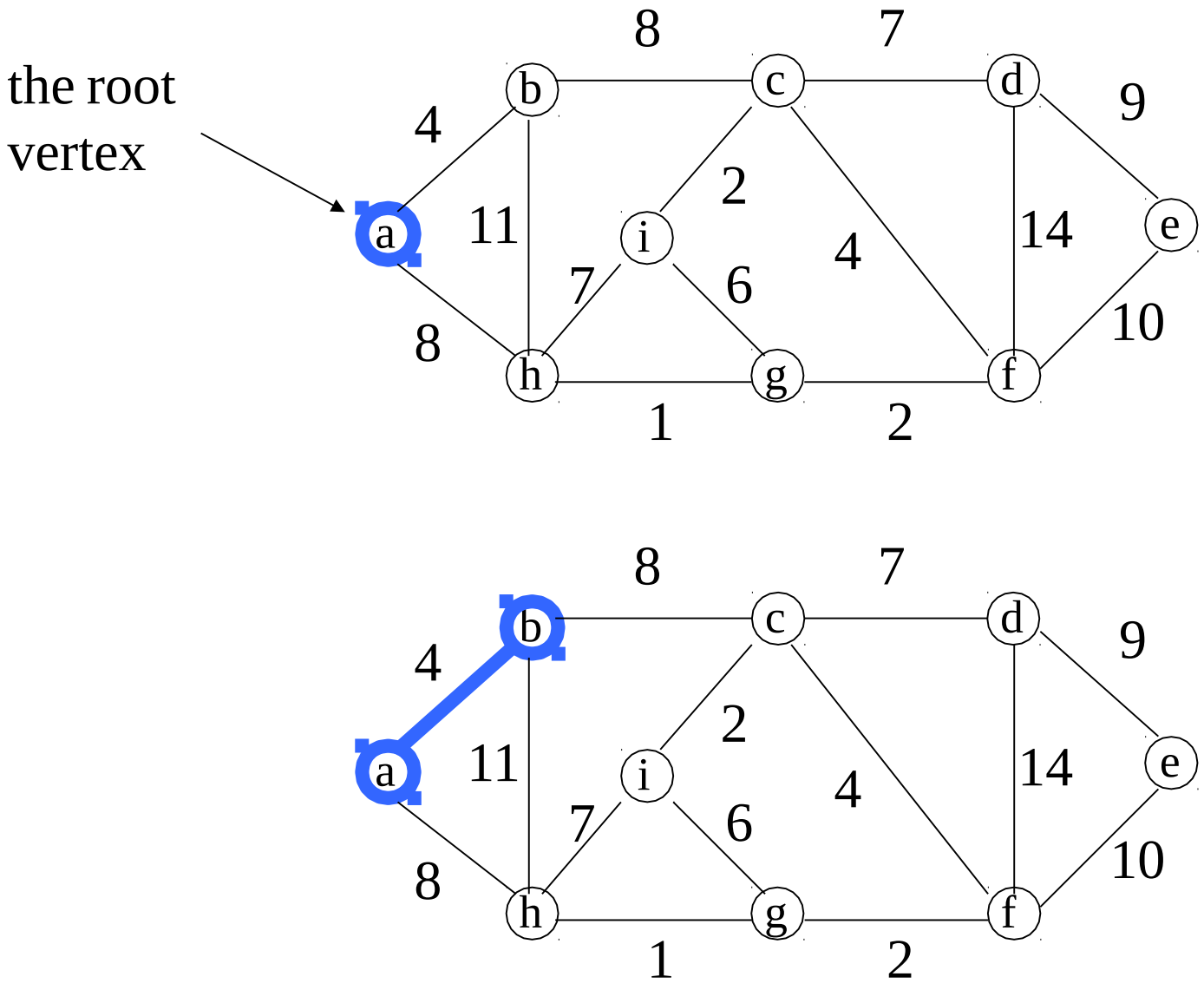
Prim's Algorithm

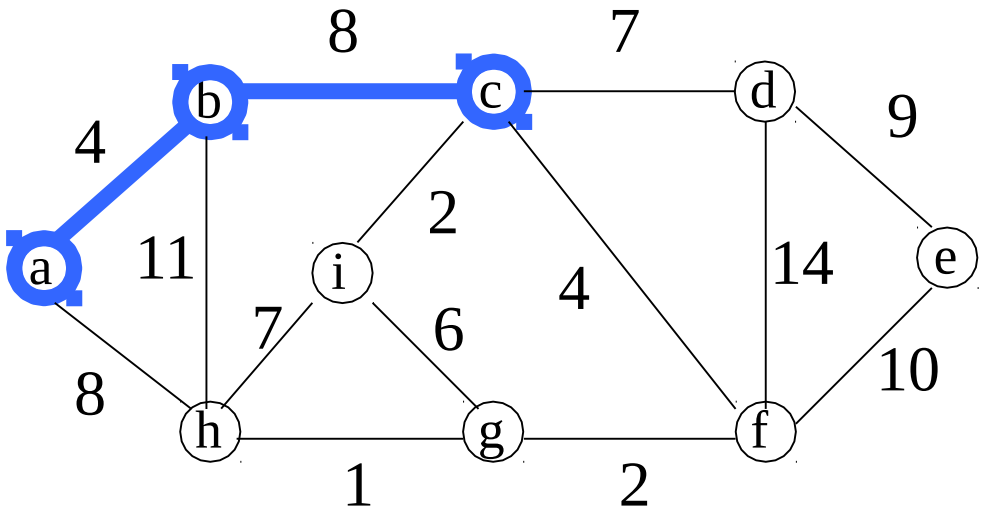
PQ: Φ

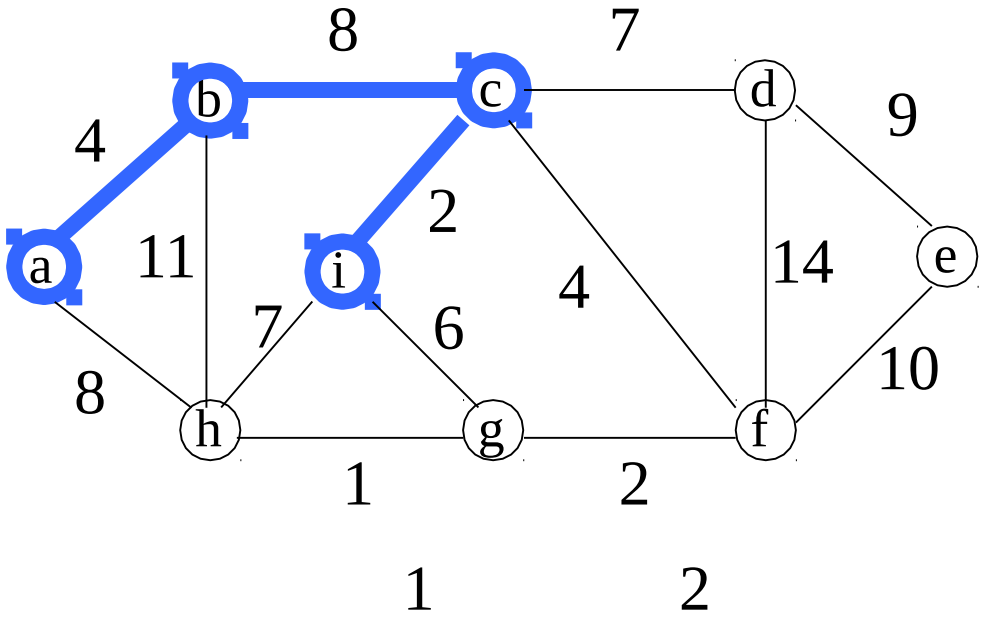


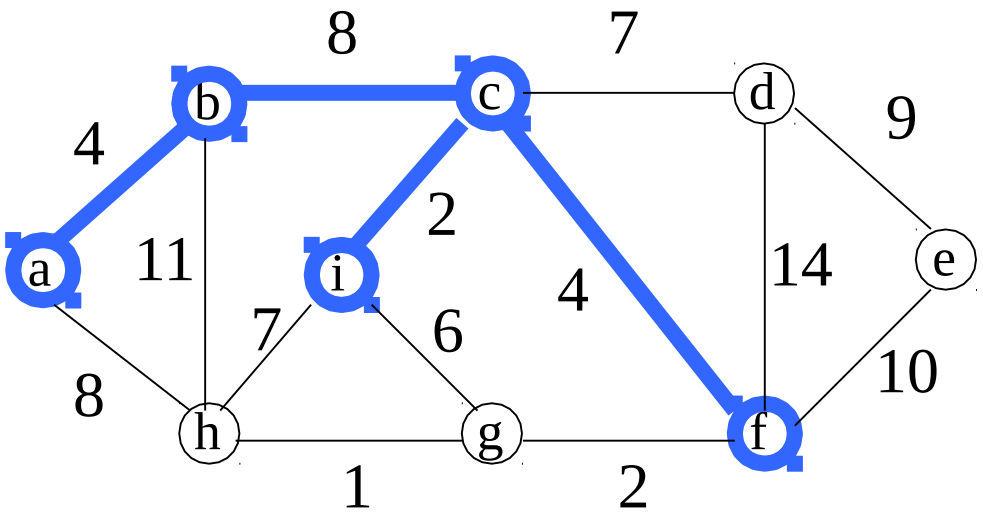
Minimum Cost=22

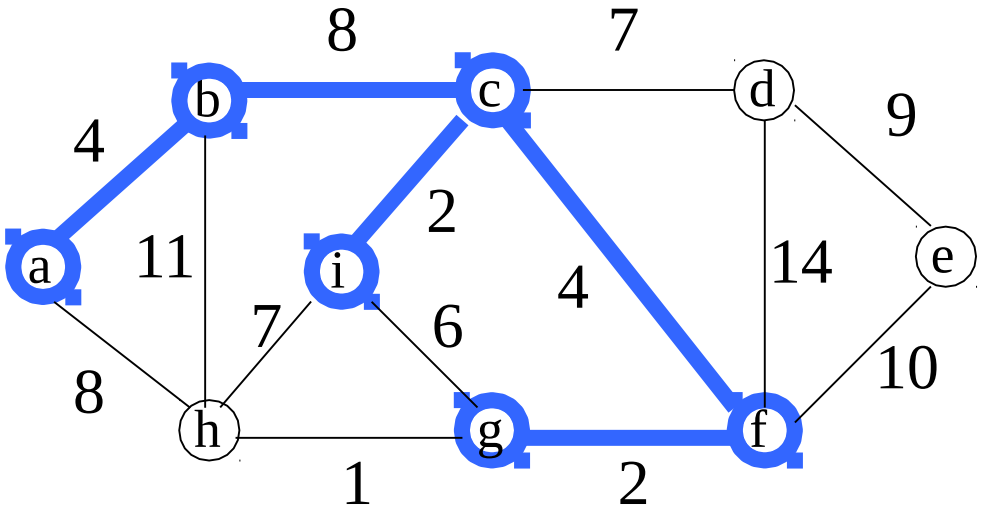
The execution of Prim's algorithm

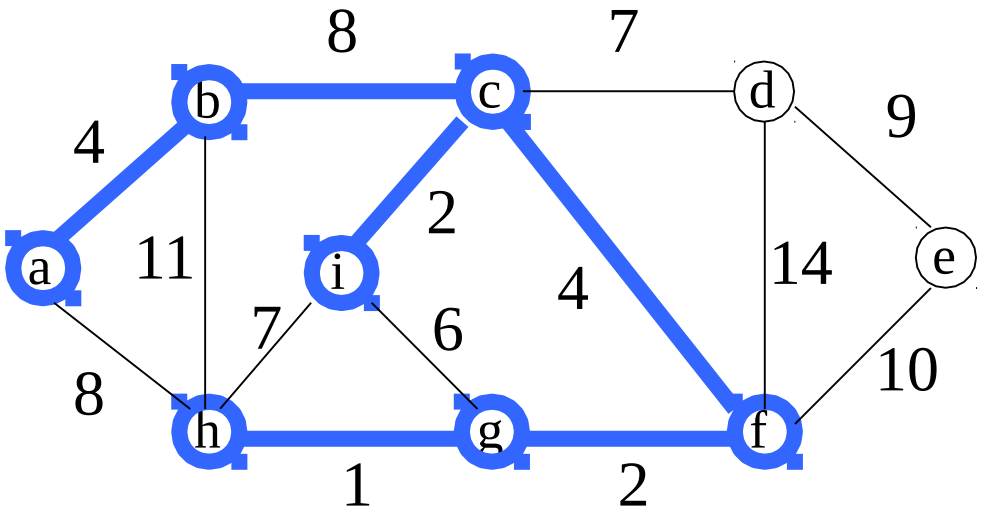


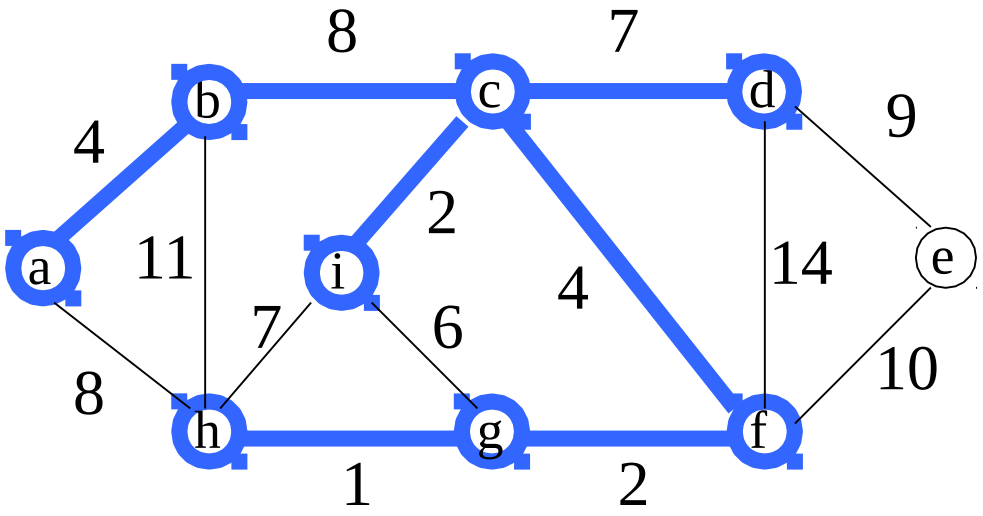

8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2

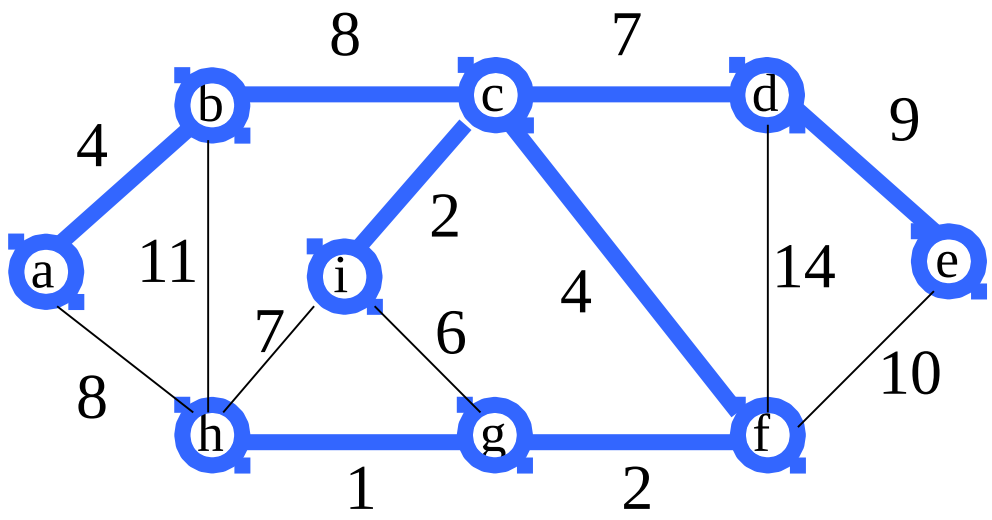

8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2
1
2


8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


8
7
b
c
d
4
9
2
a
11
i
4
14
e
7
6
8
10
h
g
f
1
2

Analysis

```
•     MST-PRIM(G,w,r)
1.    for each uεV[G]                                  |V|
2.        do key[u]←∞
3.           color[u]←white
4.           ∏[u]←Nil
5.    key[r]←0
6.    Q←V[G]
7.    while Q≠Φ                                        |V|
8.       do u← EXTRACT-MIN(Q)                          O(log V)
9.         for each v ε Adj[u]
10.        do if color[v]==white &w(u,v)<key[v]
11.             then ∏[v]← u
12.              key[v]← w(u,v);Decrease-key(v);
13.          color[u]=black
```

Prim's Algorithm

- Another way to MST using Prim's Algorithm.
- This algorithm starts with one node. It then, one by one, adds a node that is unconnected to the new graph to the new graph, each time selecting the node whose connecting edge has the smallest weight out of the available nodes' connecting edges.

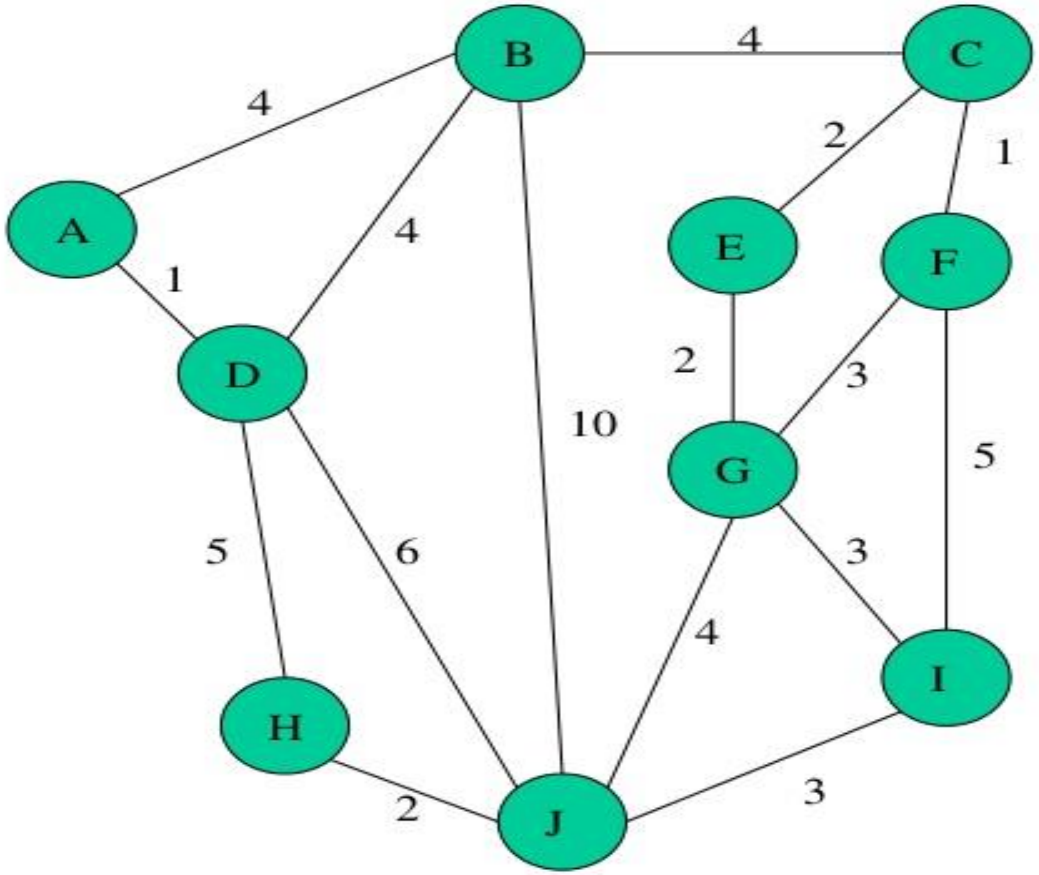
Prim's Algorithm

The steps are:

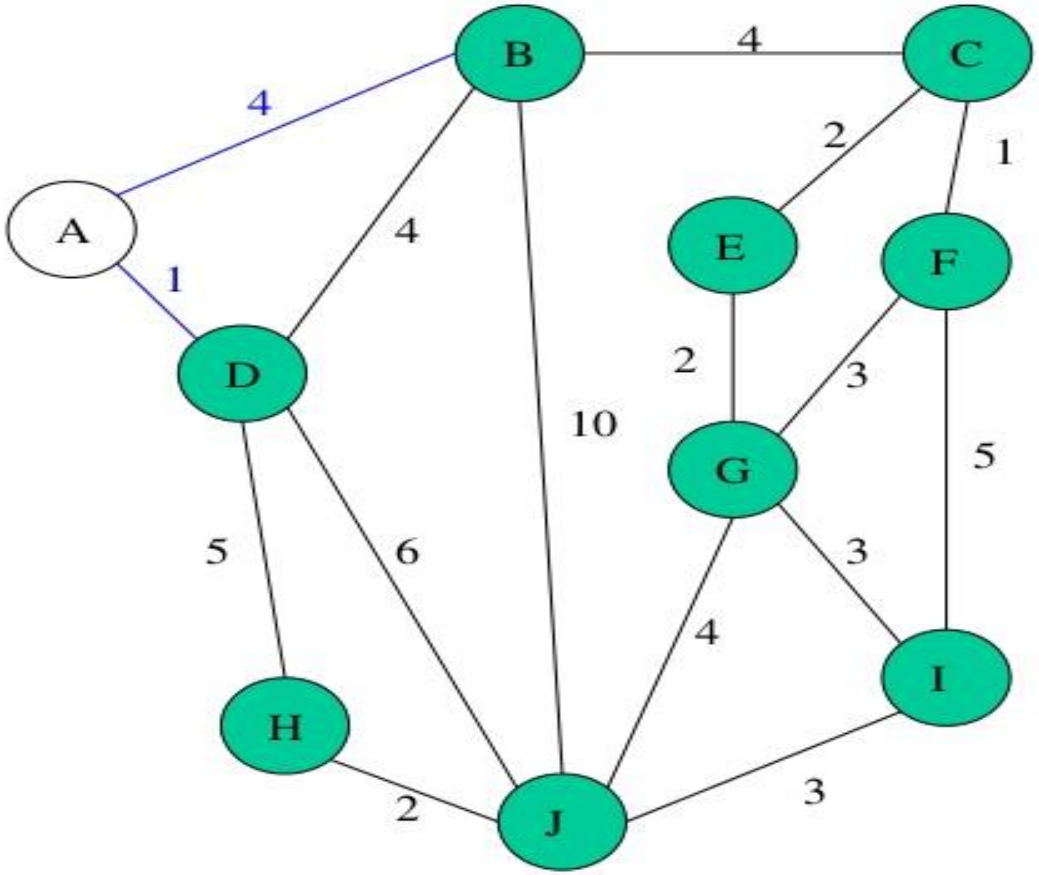
1. The new graph is constructed - with one node from the old graph.
2. While new graph has fewer than n nodes,
    1. Find the node from the old graph with the smallest connecting edge to the new graph,
    2. Add it to the new graph

Every step will have joined one node, so that at the end we will have one graph with all the nodes and it will be a minimum spanning tree of the original graph.

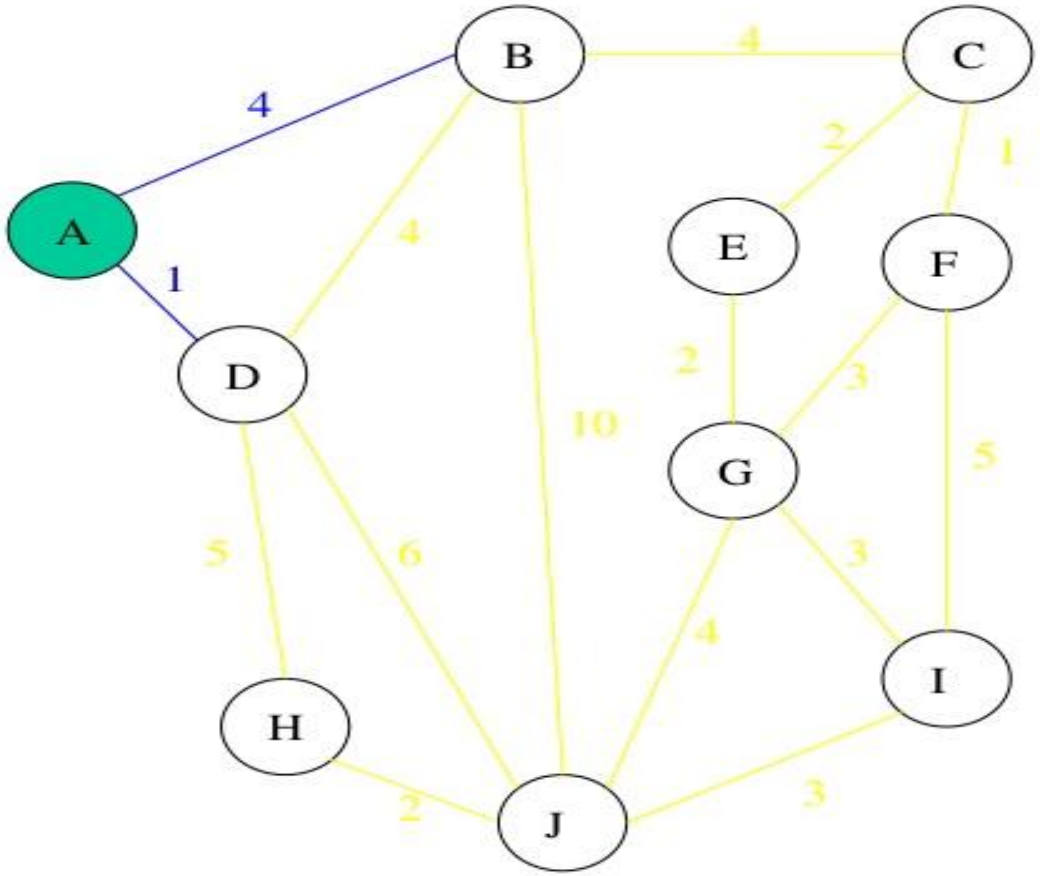
Complete Graph



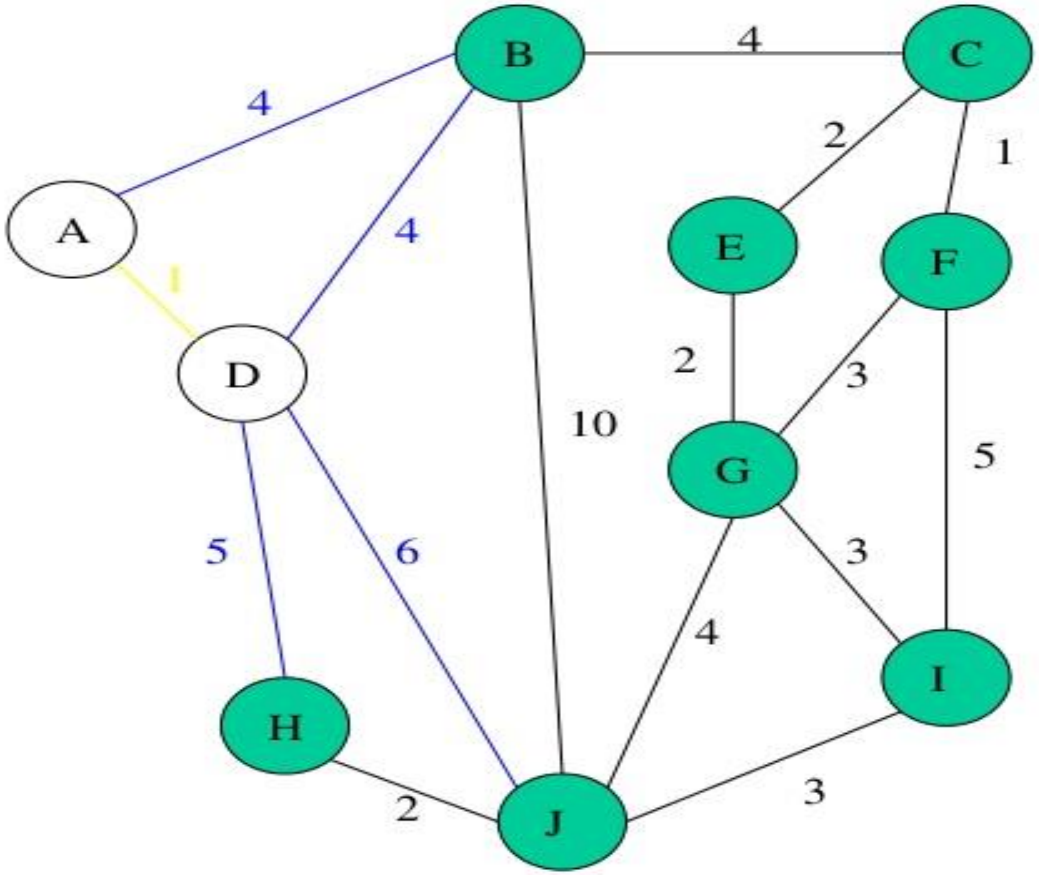
Old Graph



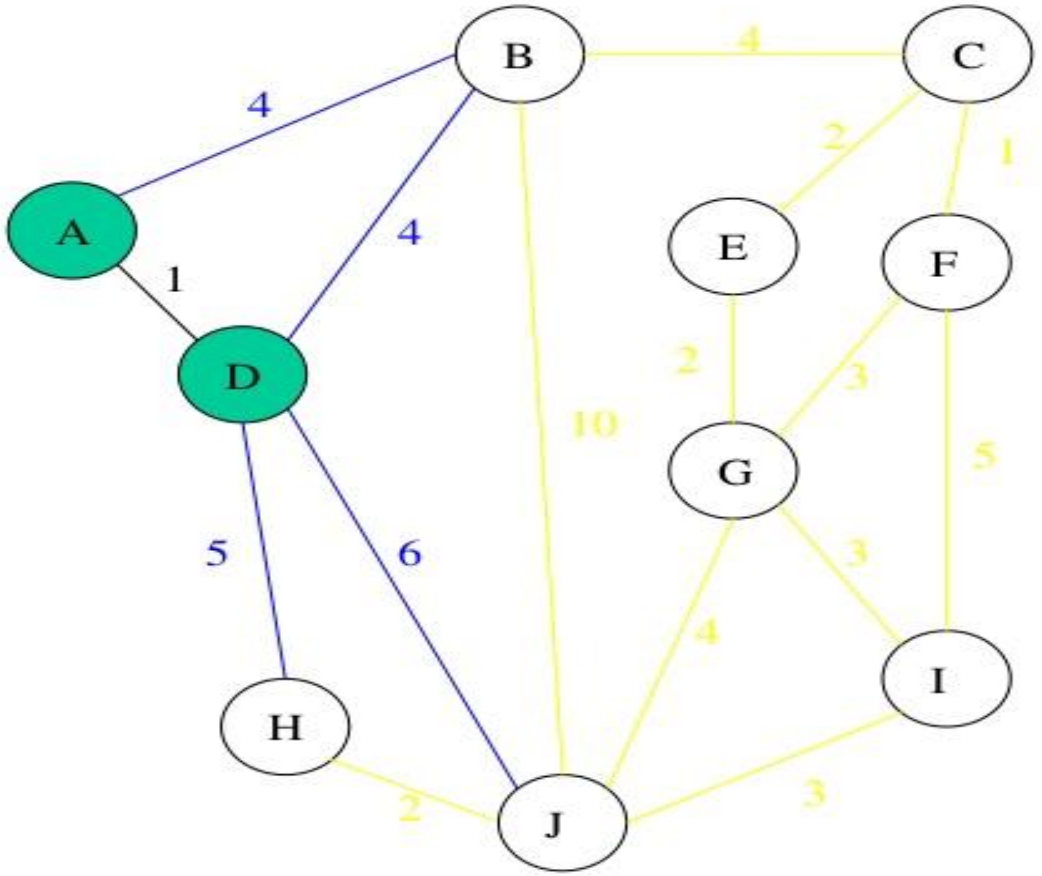
New Graph



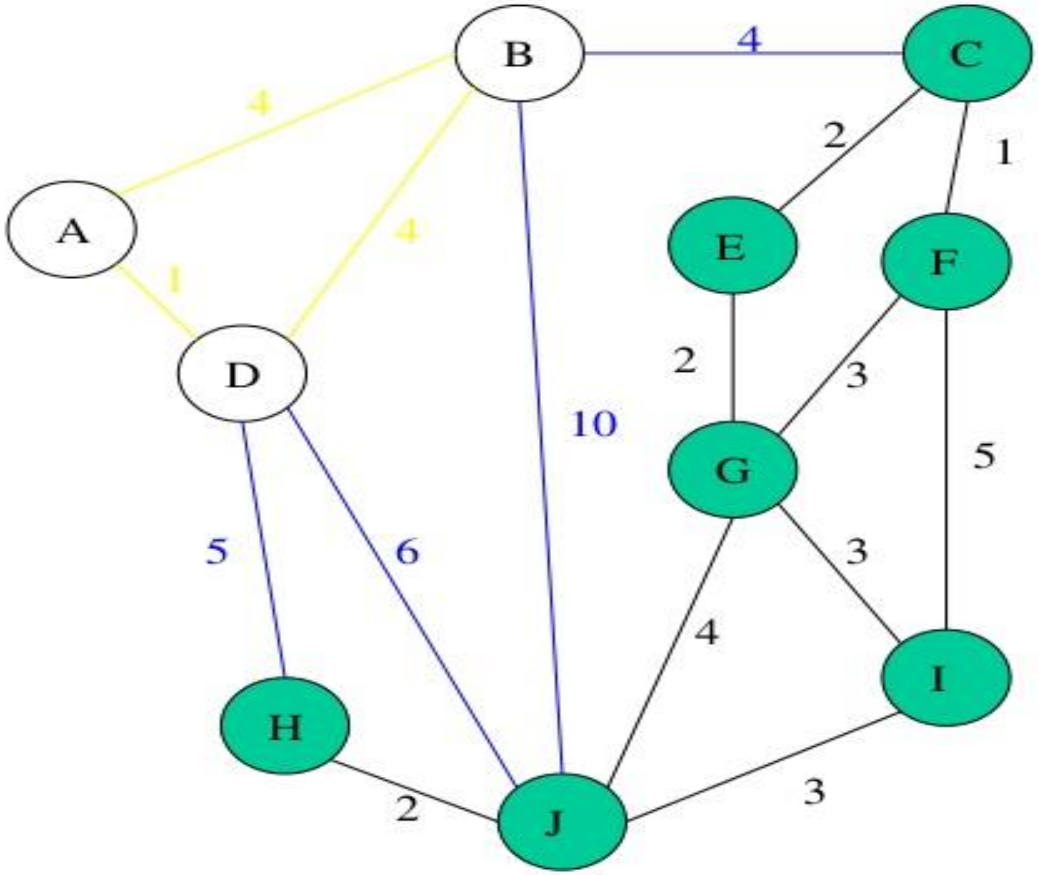
Old Graph



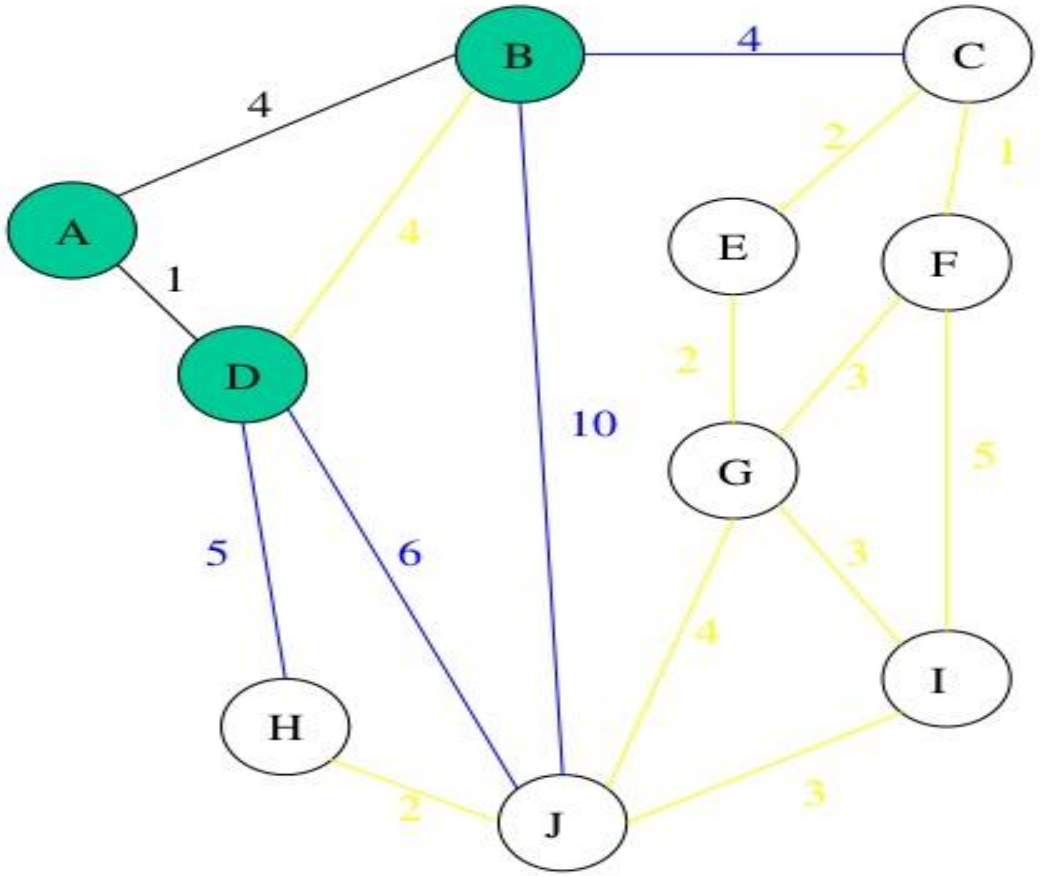
New Graph



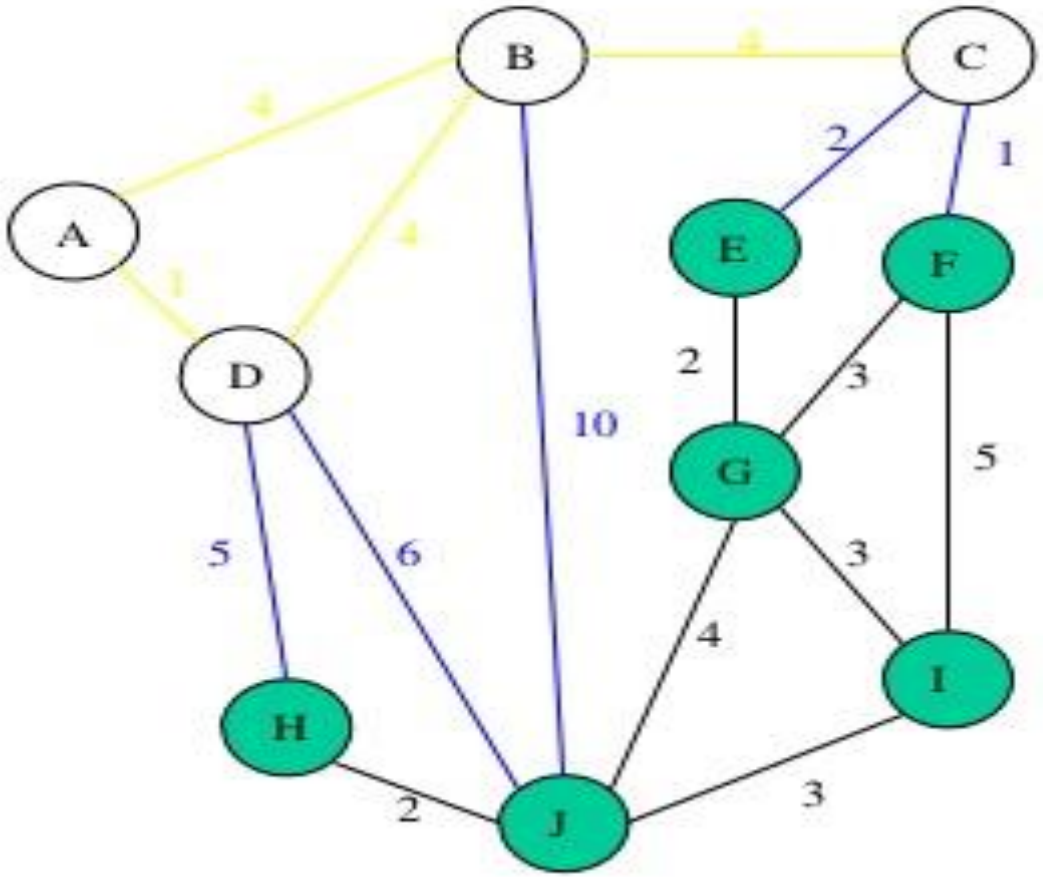
Old Graph



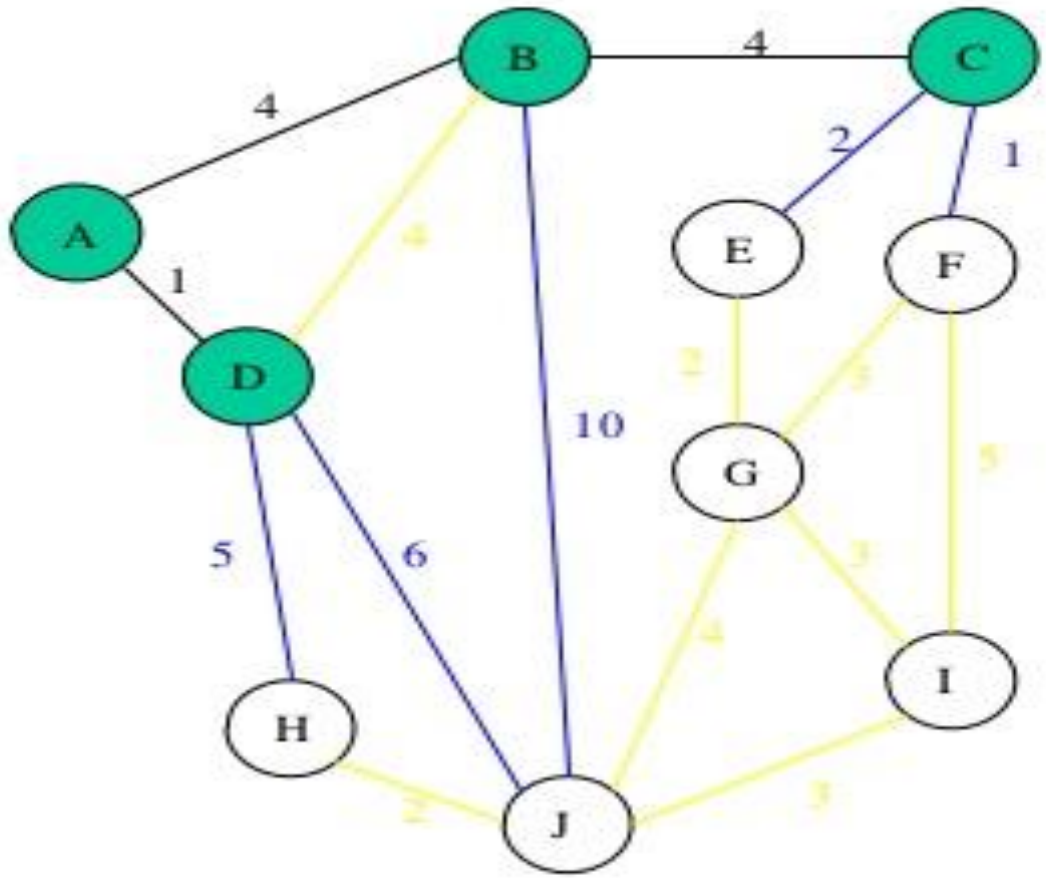
New Graph



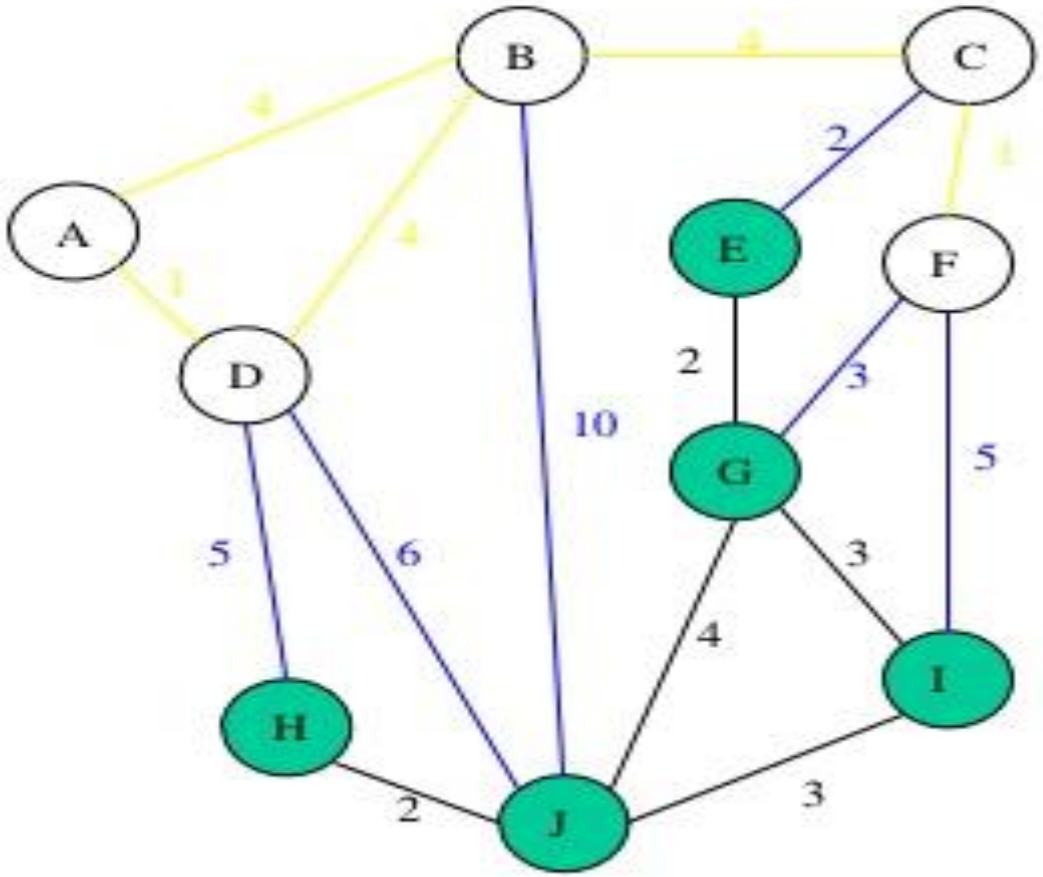
Old Graph



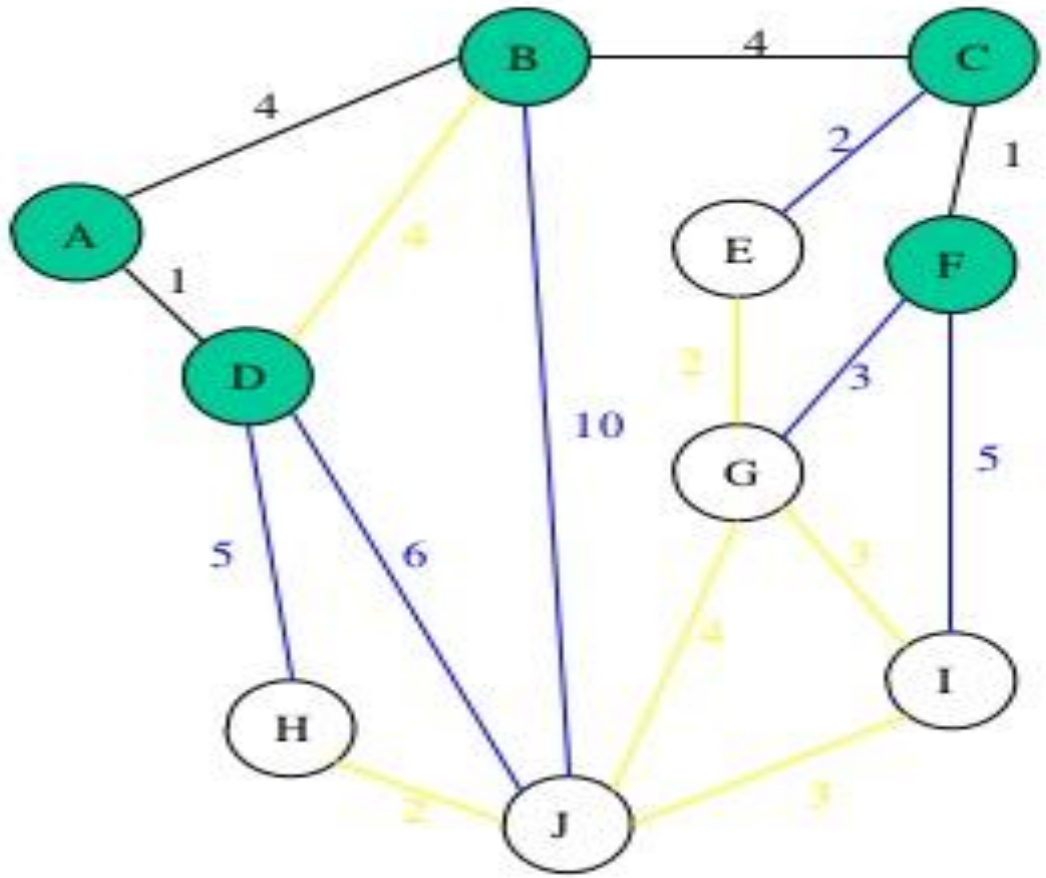
New Graph



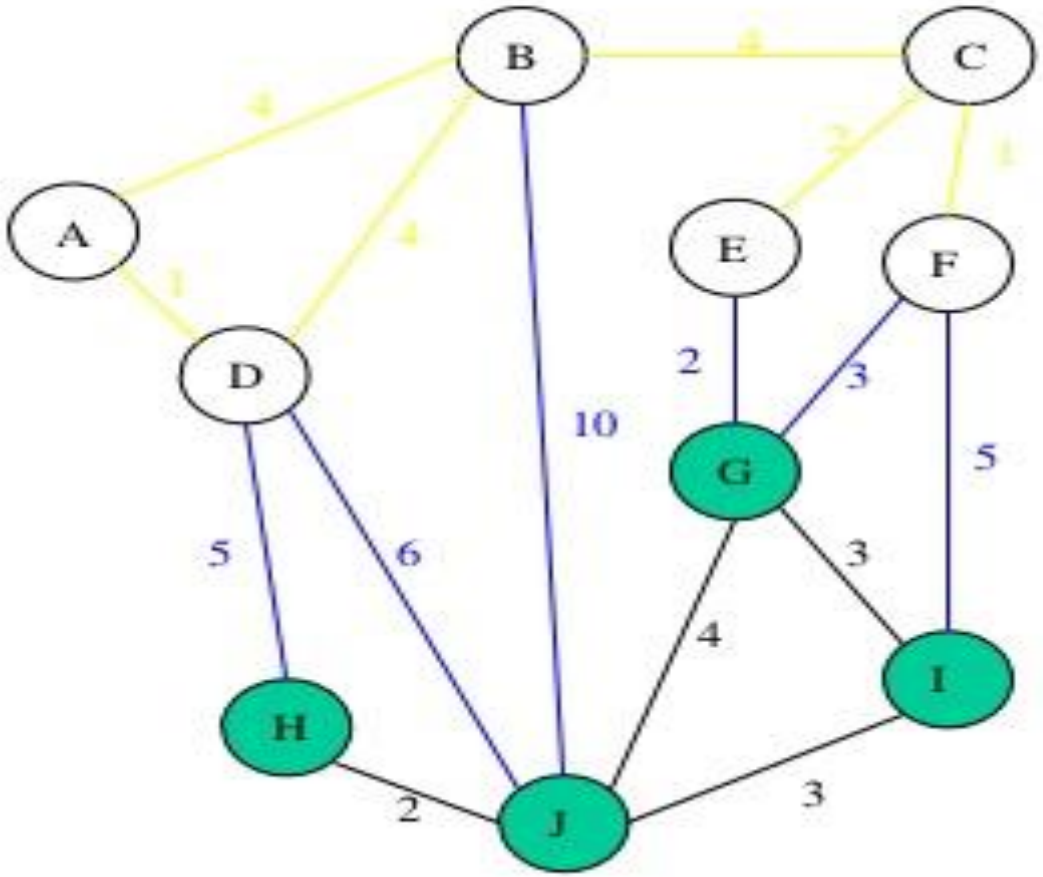
Old Graph



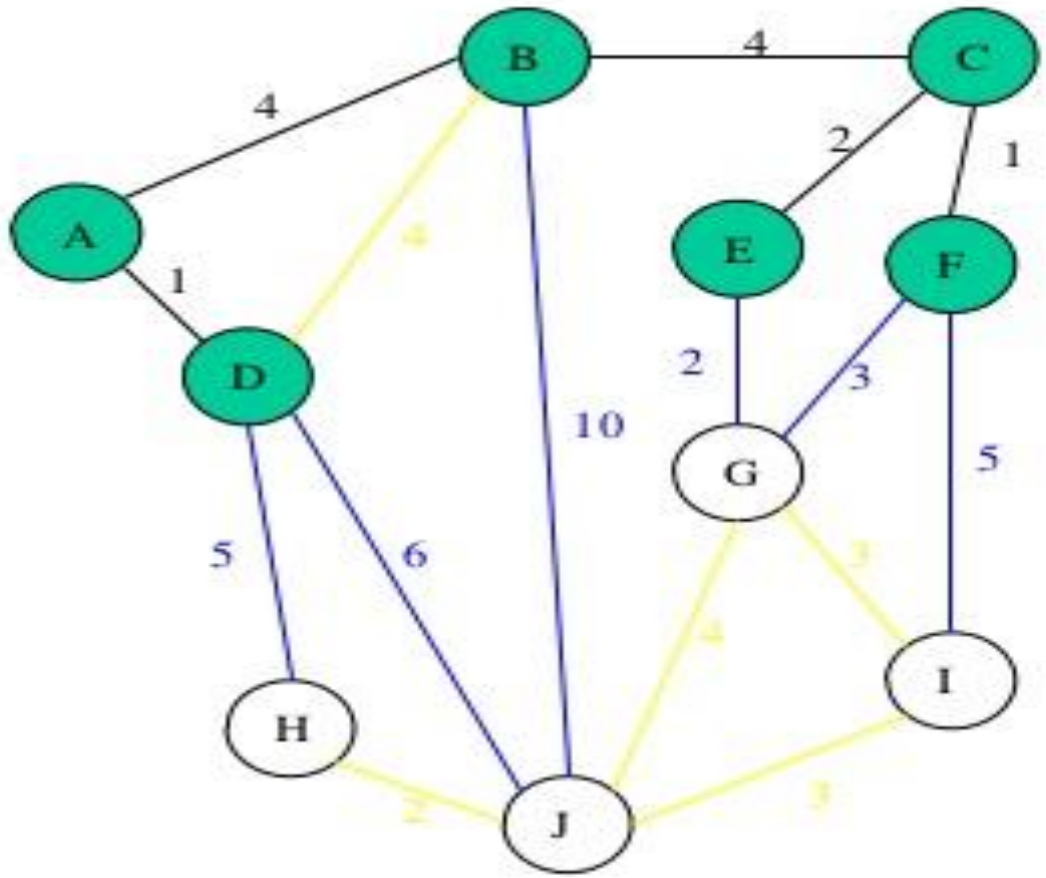
New Graph



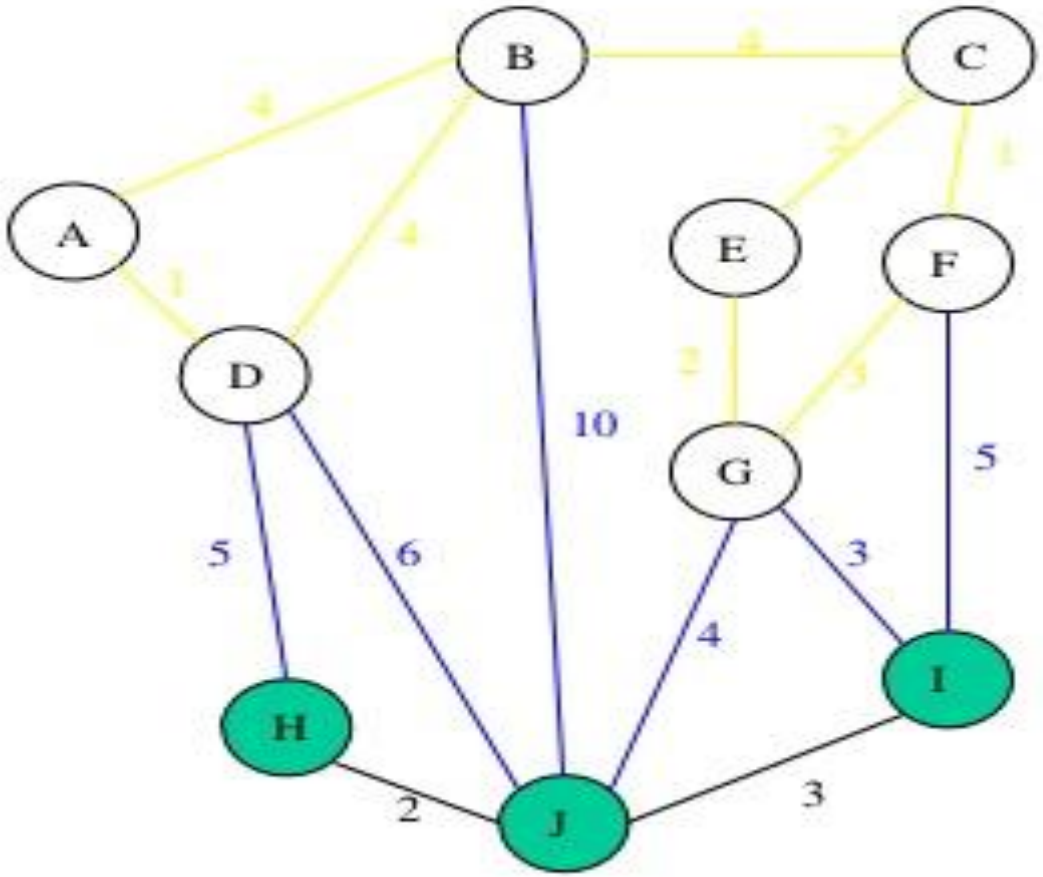
Old Graph



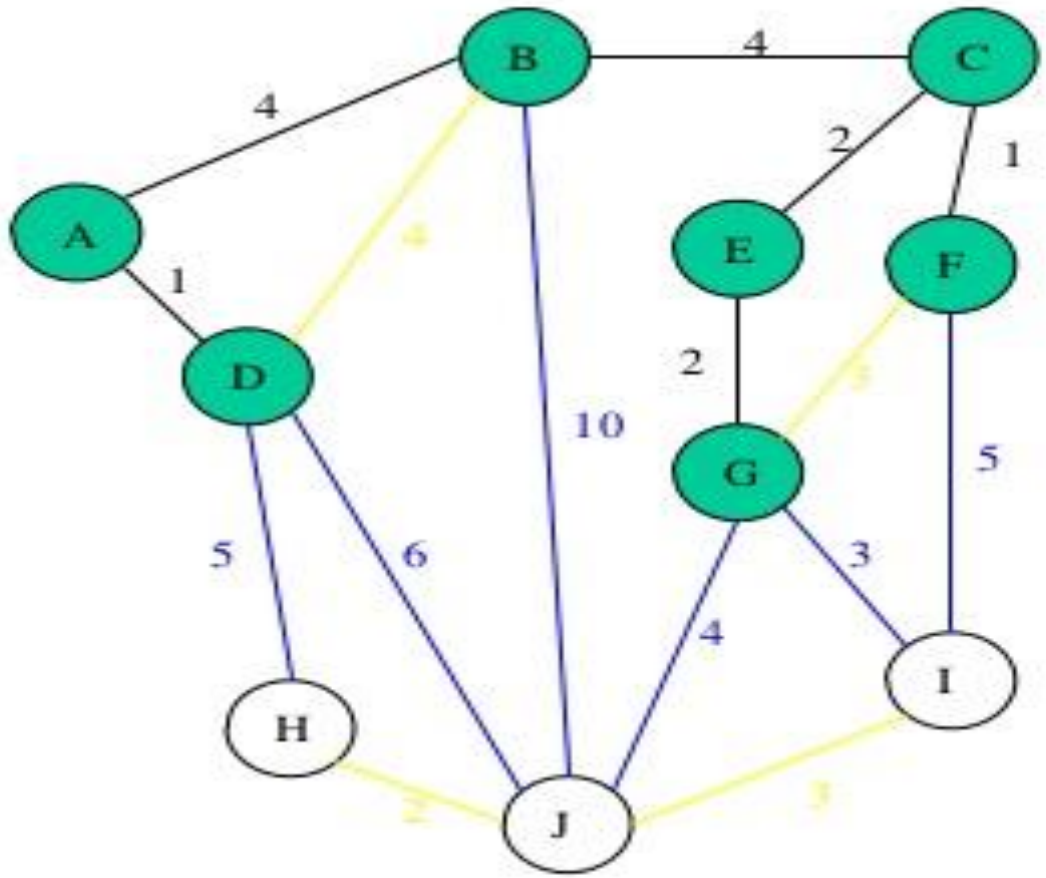
New Graph



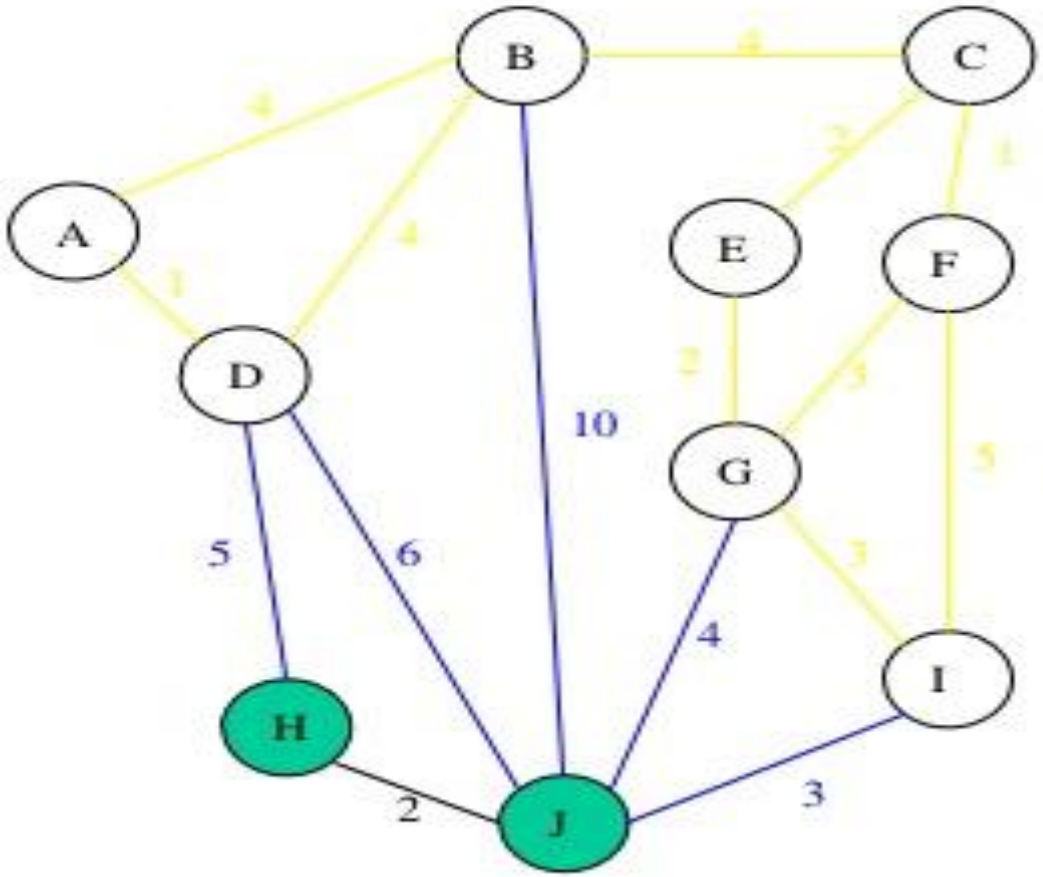
Old Graph



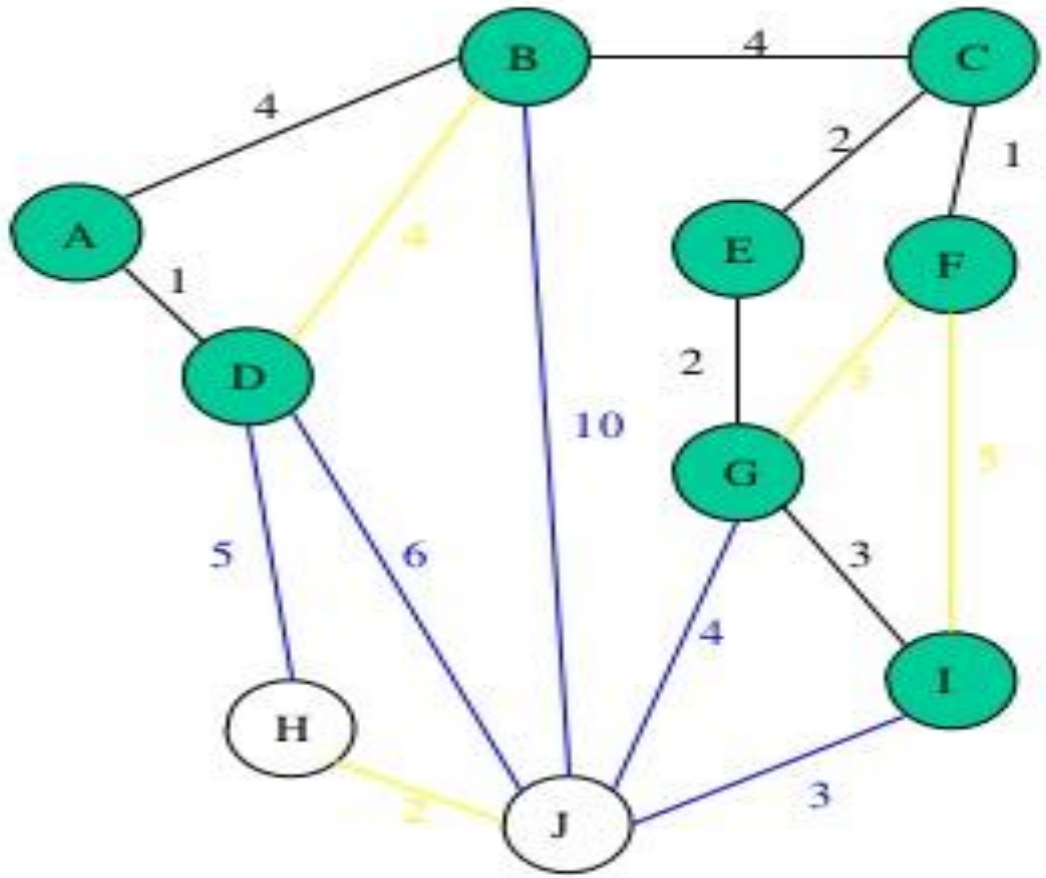
New Graph



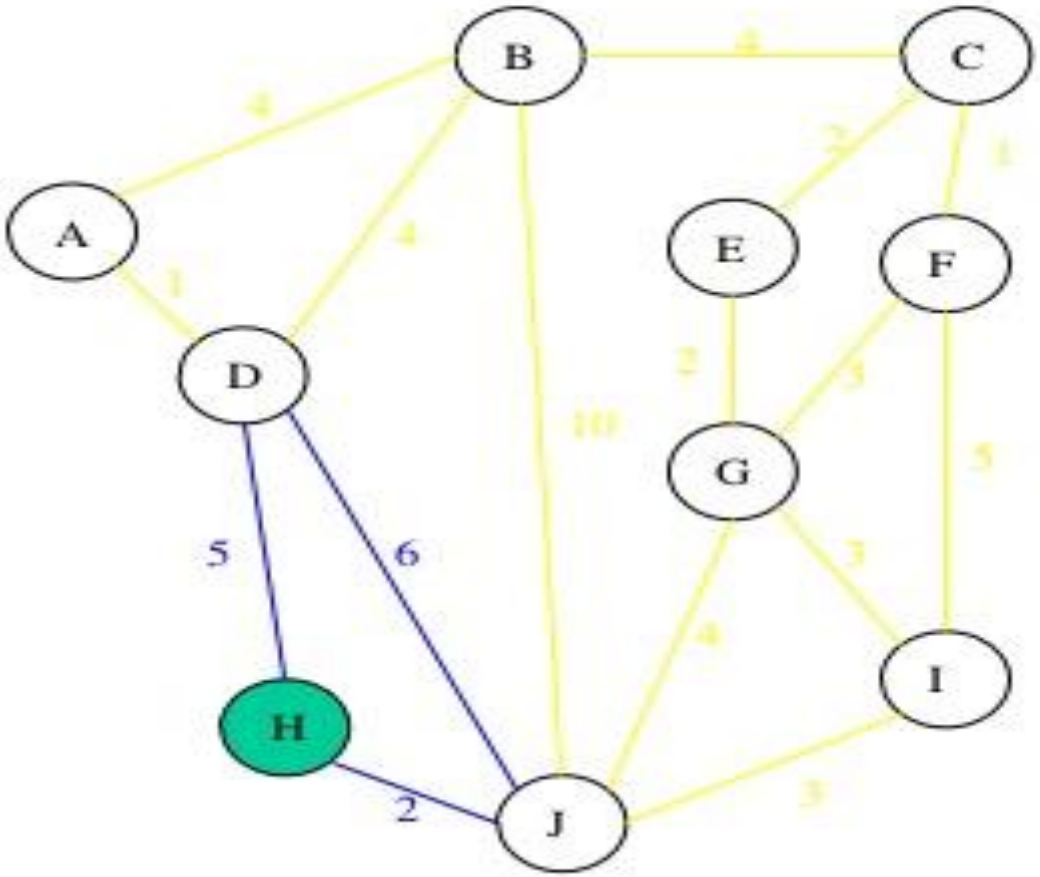
Old Graph



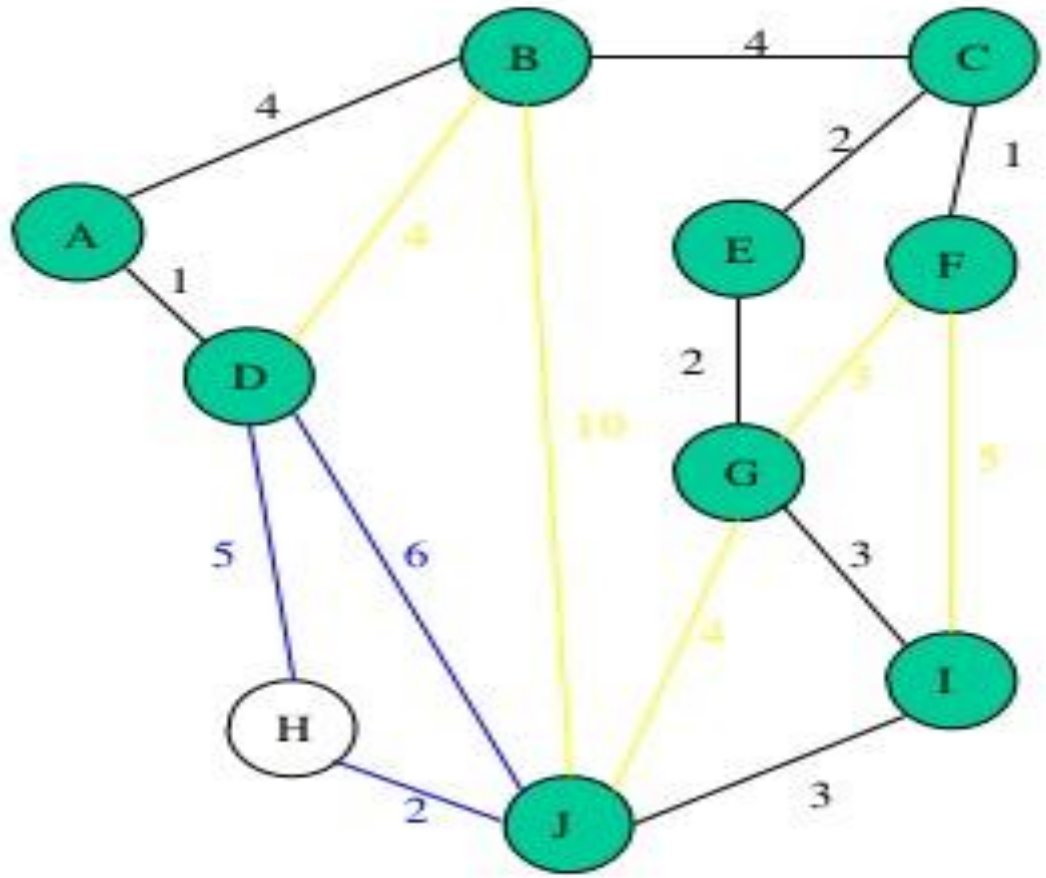
New Graph



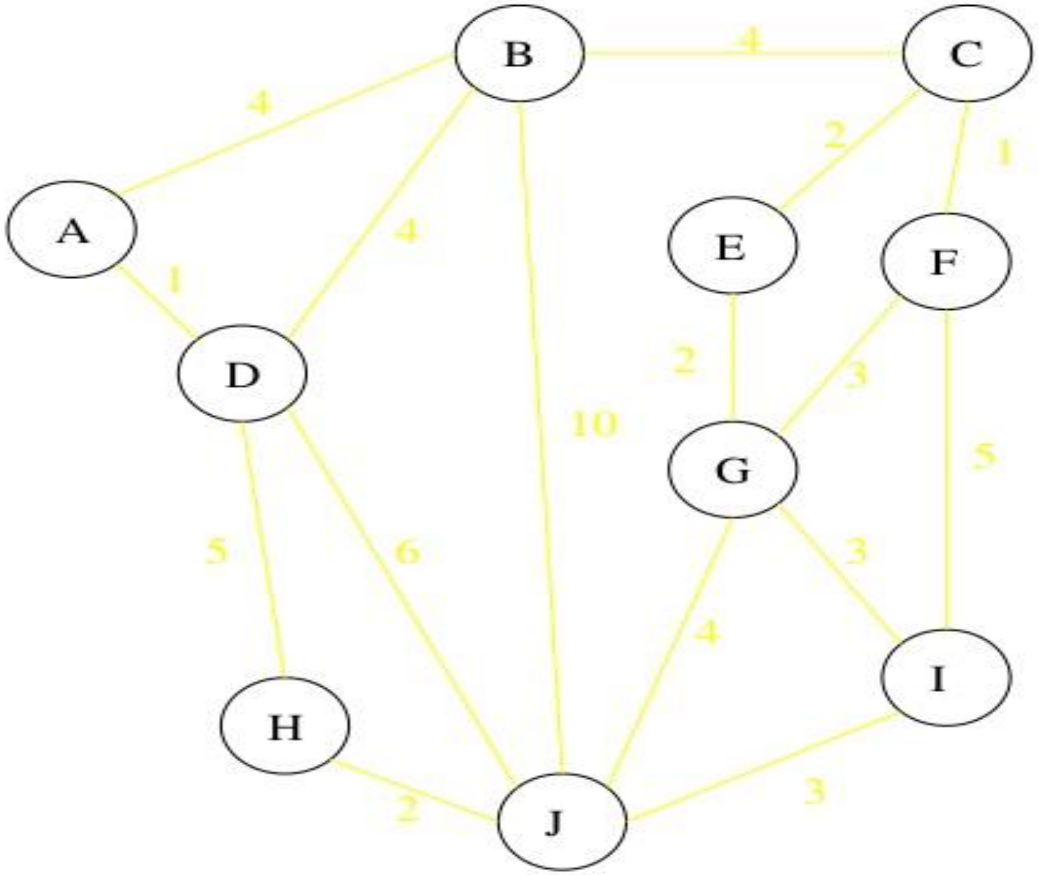
Old Graph



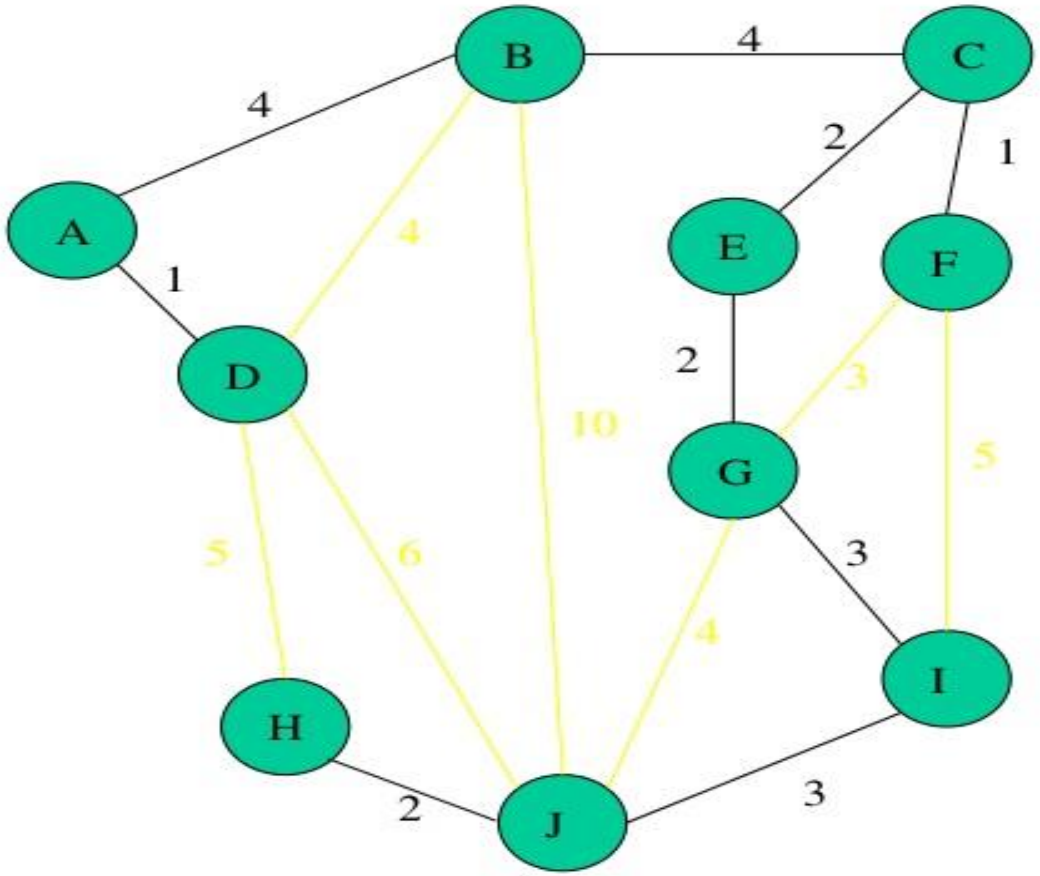
New Graph



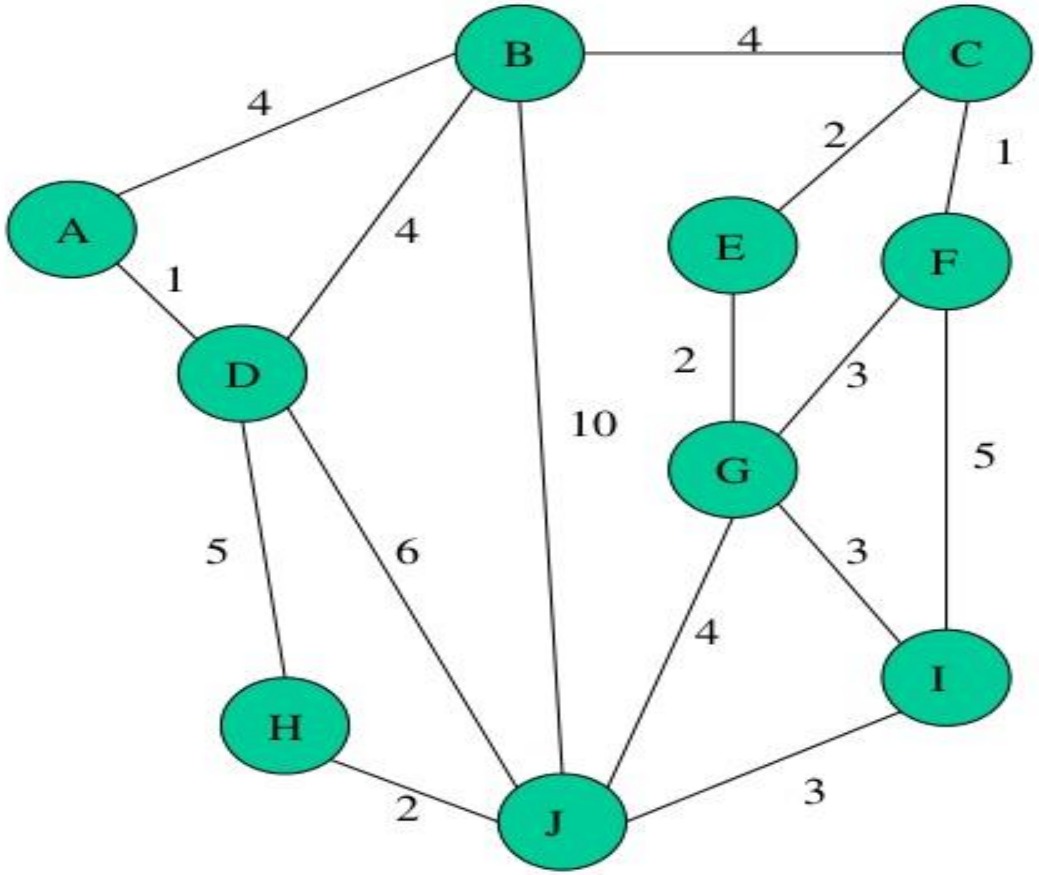
Old Graph



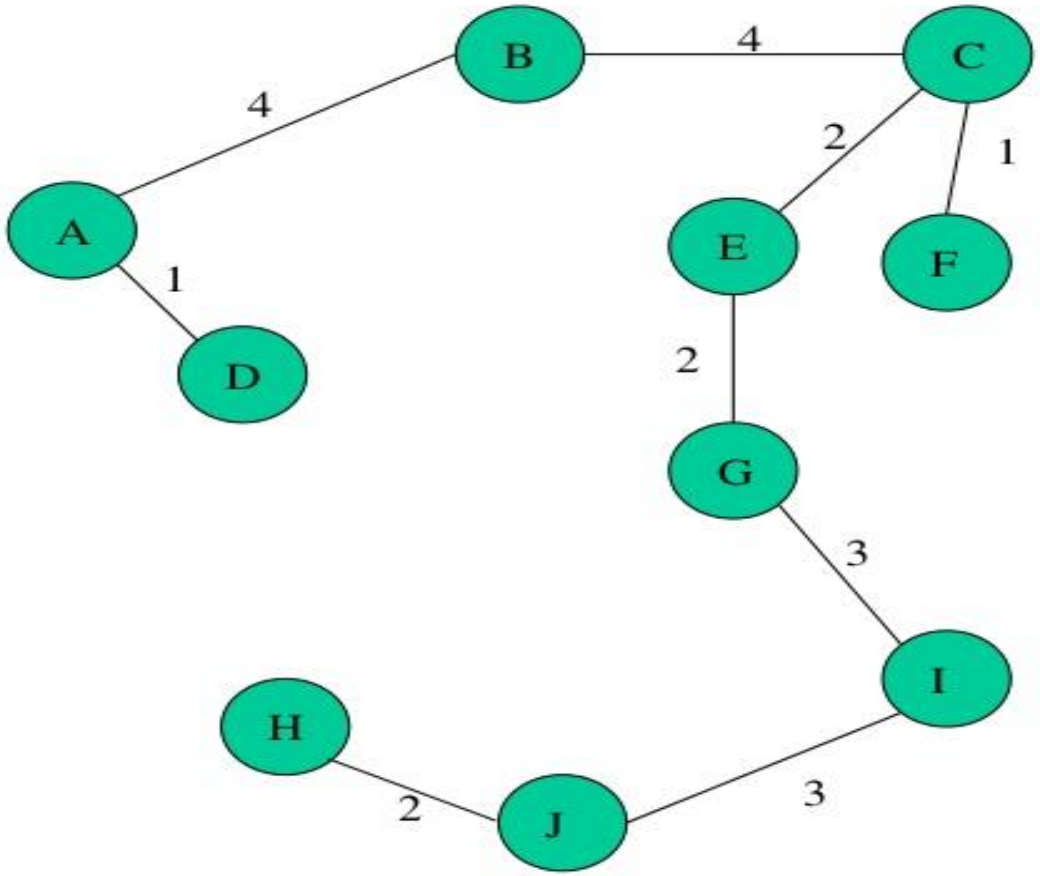
New Graph



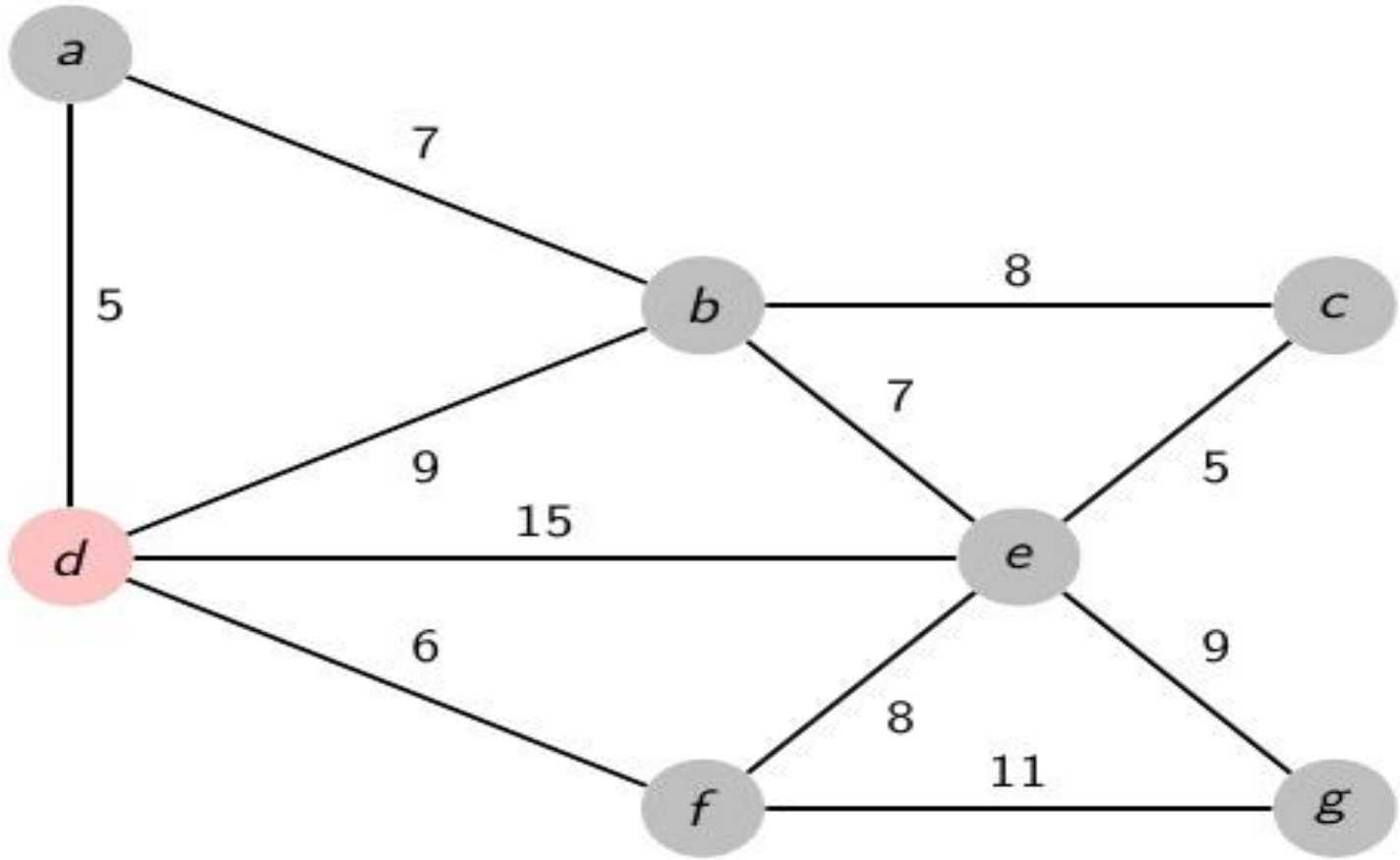
Complete Graph



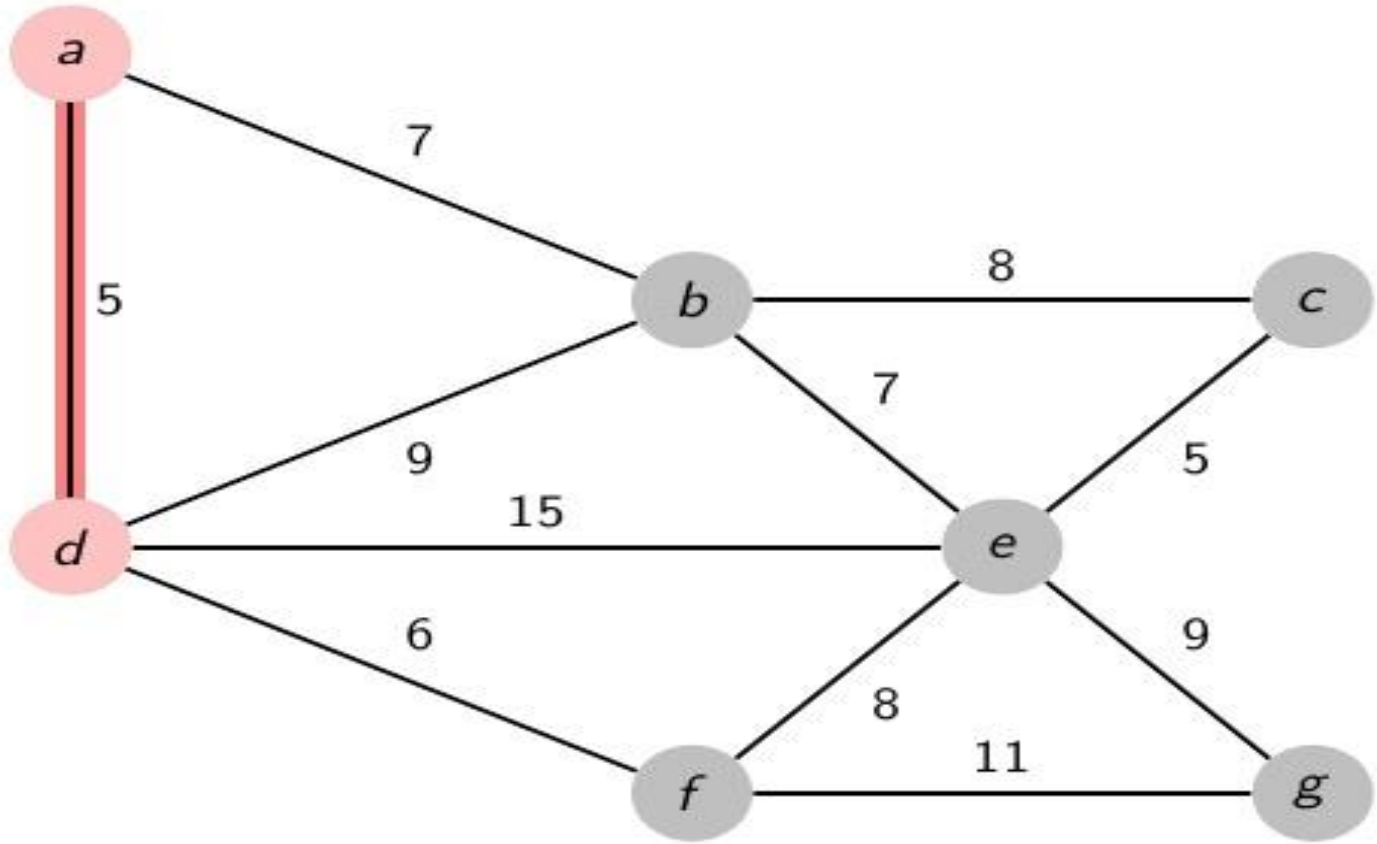
Minimum Spanning Tree



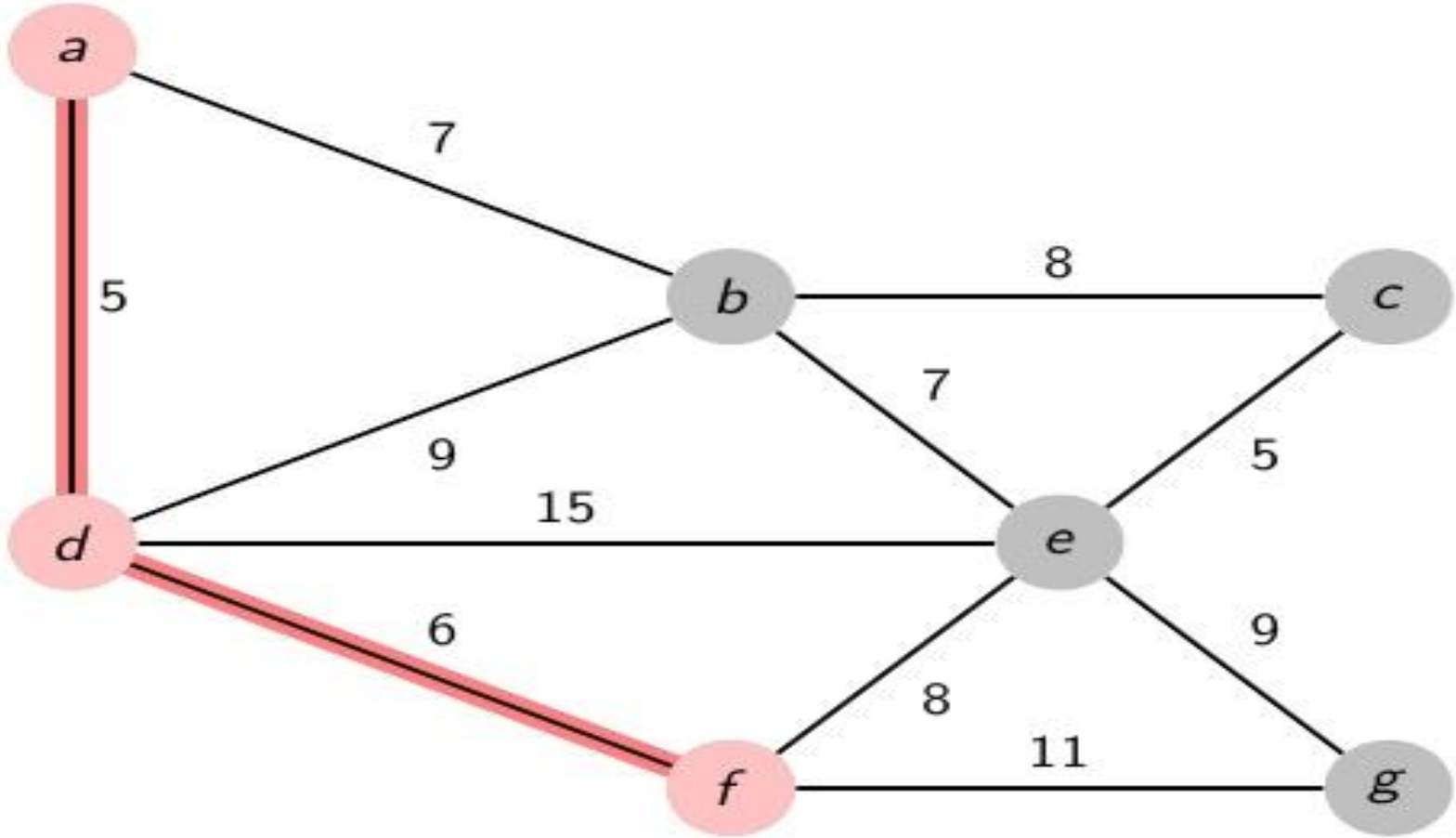
Prim's algorithm



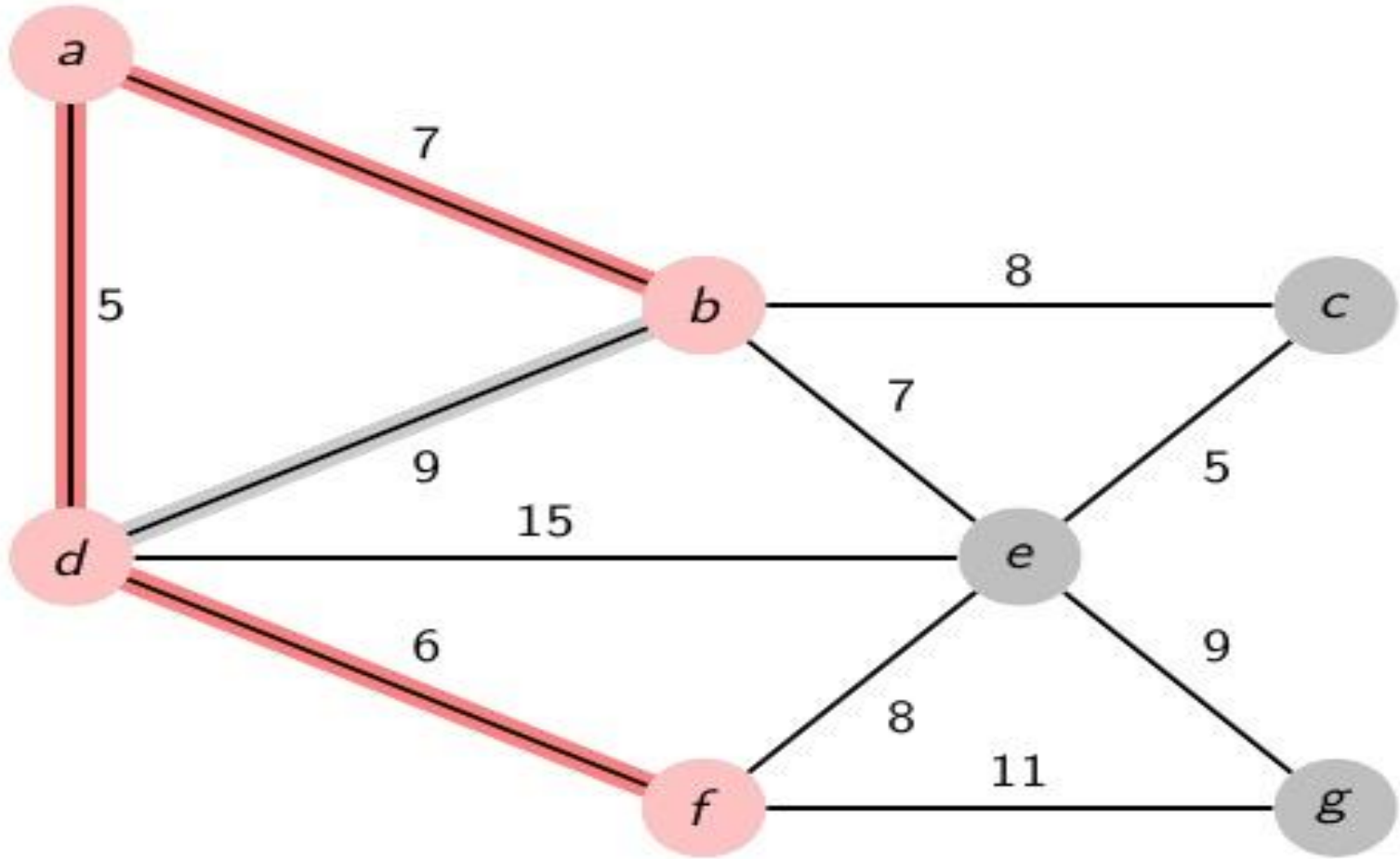
Prim's algorithm



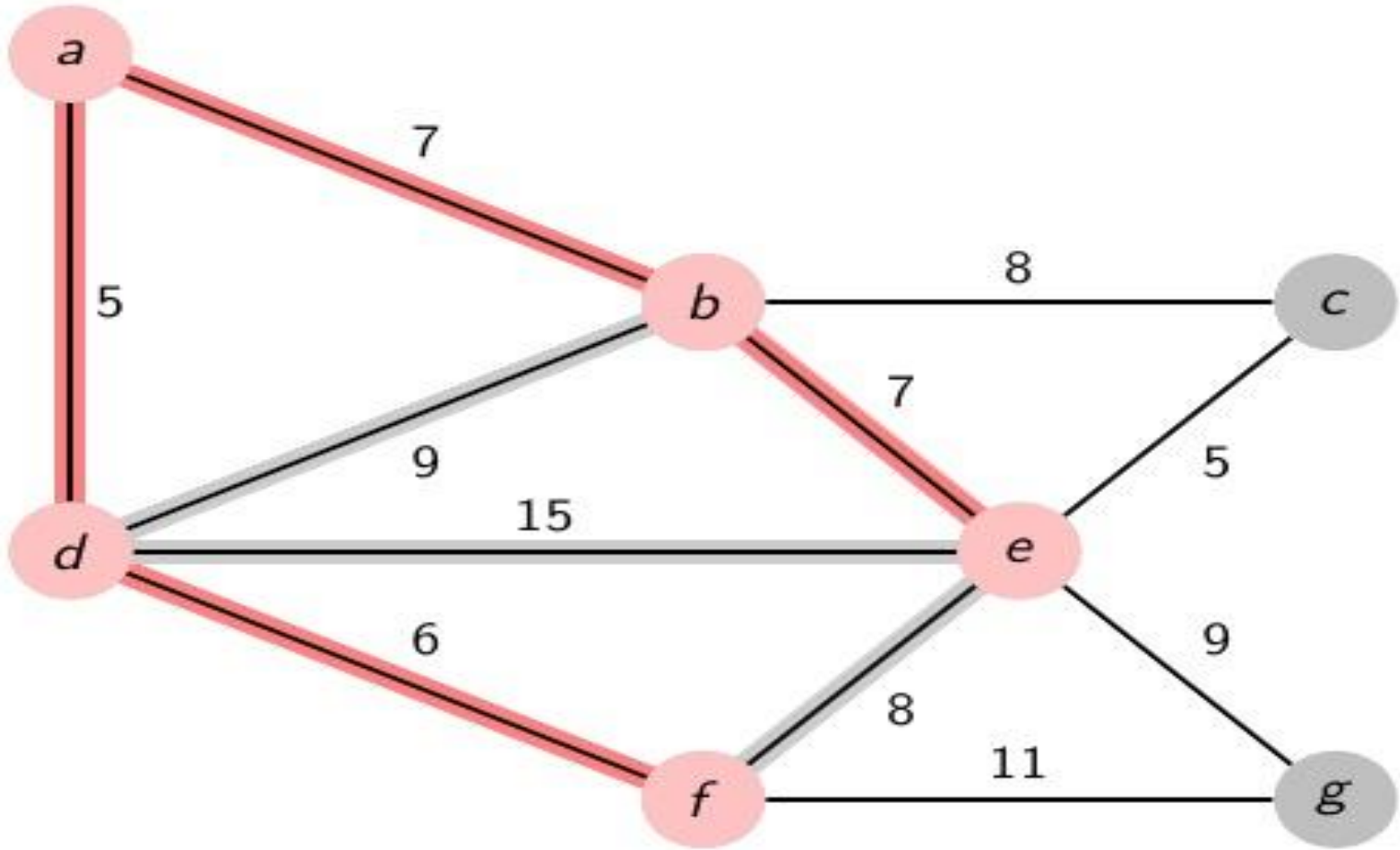
Prim's algorithm



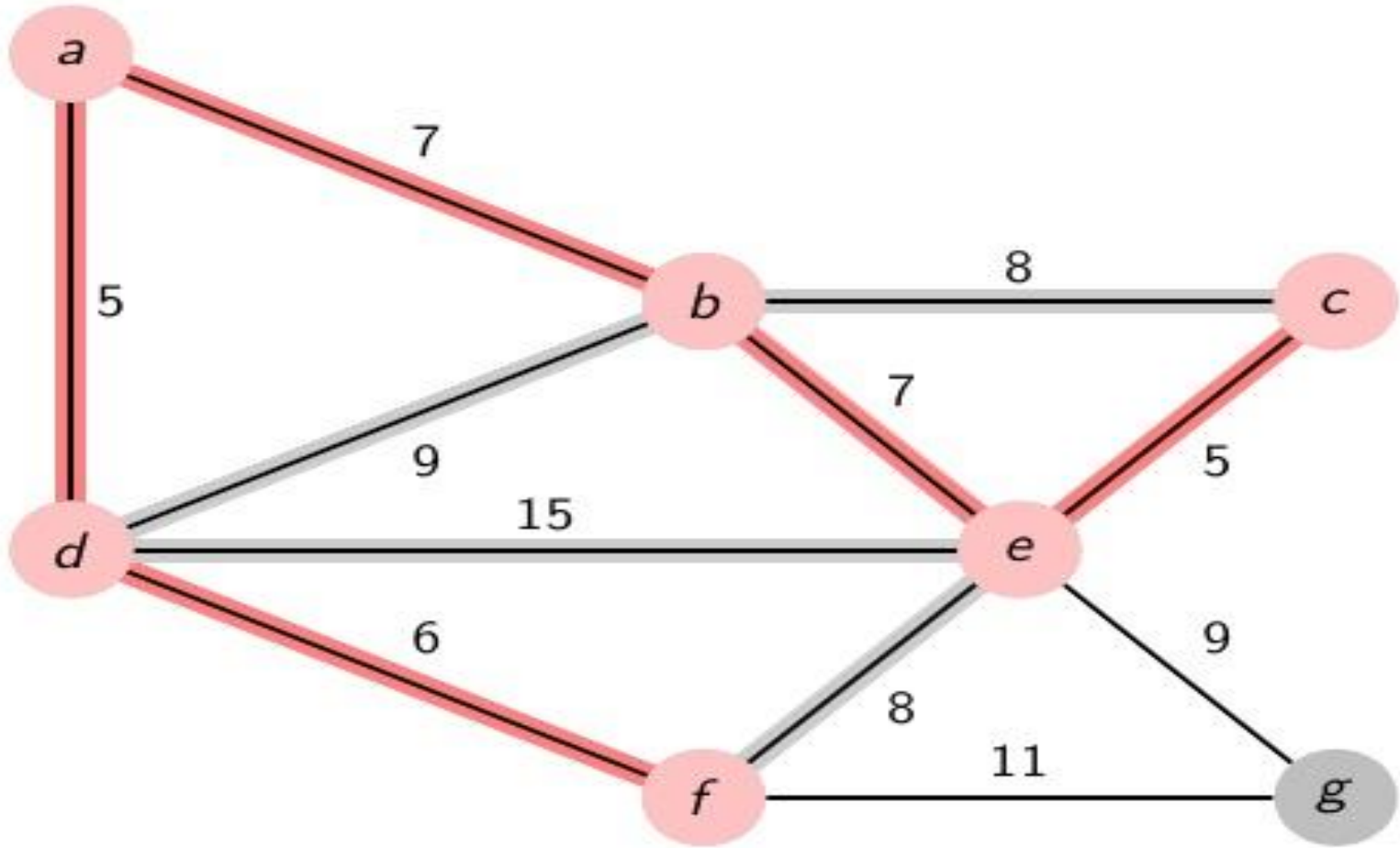
Prim's algorithm



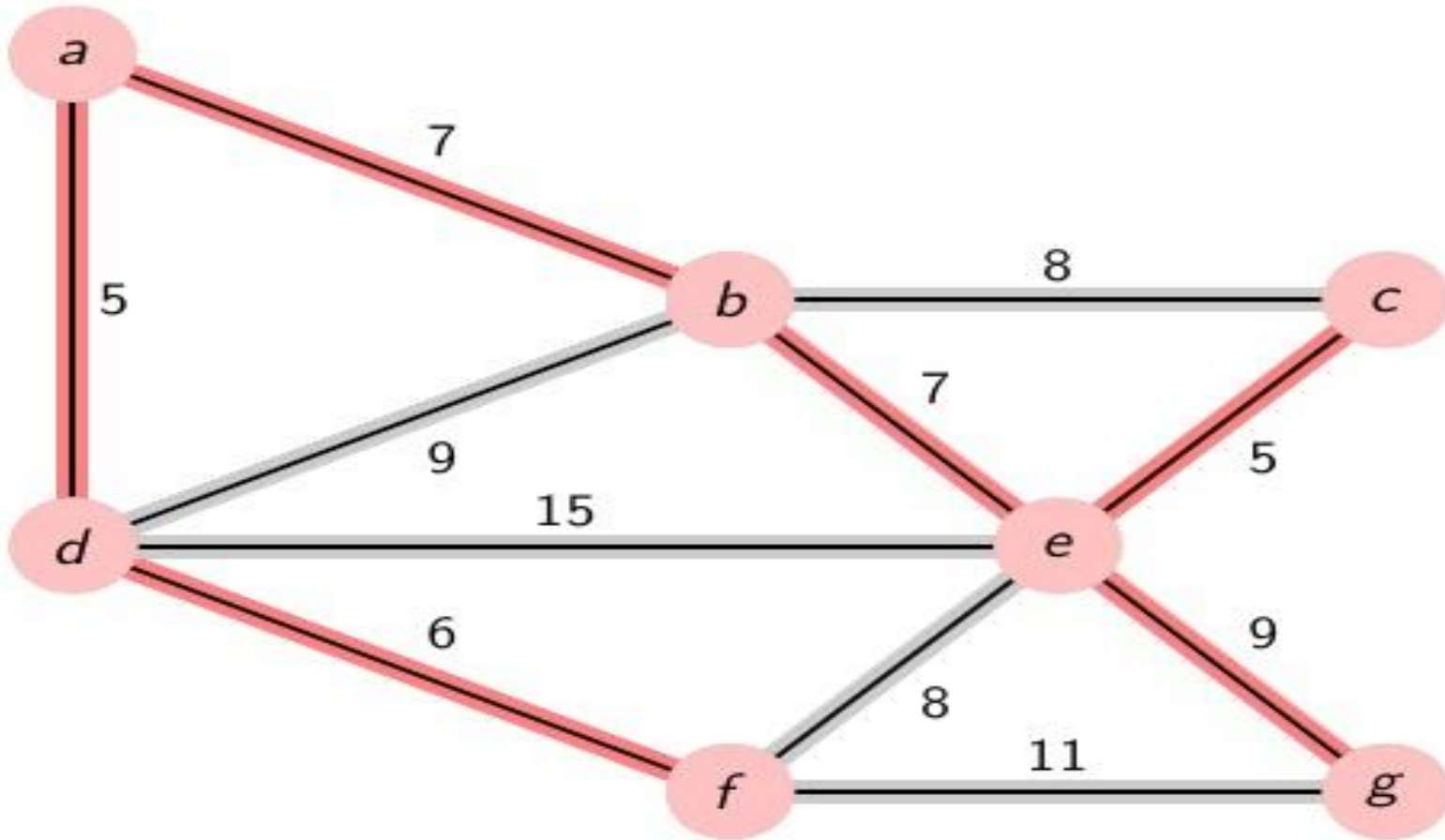
Prim's algorithm



Prim's algorithm



Prim's algorithm



Prim's Algorithm

- Similar to Dijkstra's Algorithm except that d_v records edge weights, not path lengths


Thank You!