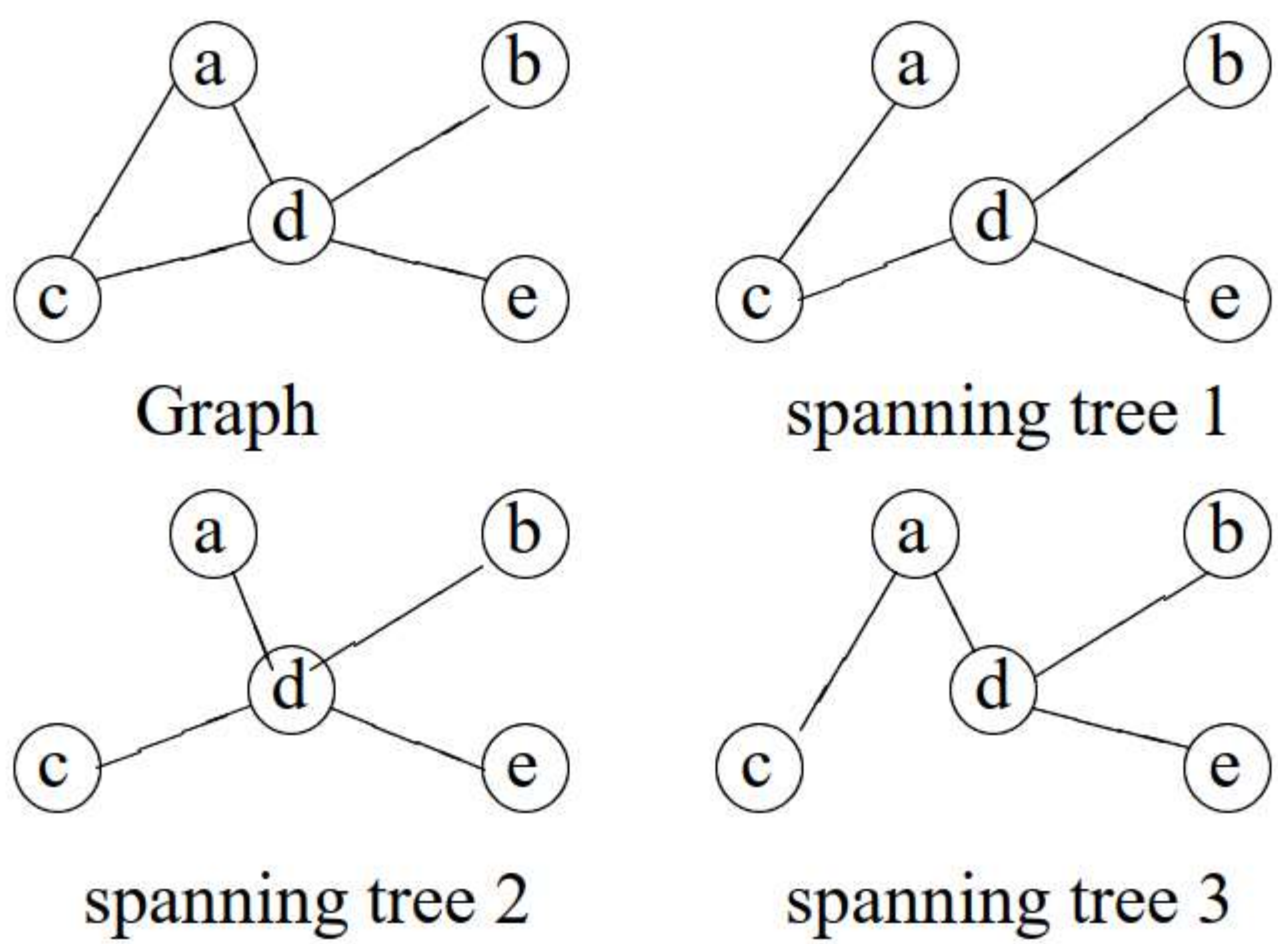
Minimum Spanning Trees
Prim’s Algorithm
Kruskal’s Algorithm

Spanning Trees

Spanning Trees: A subgraph **T** of a undirected graph **G= (V,E)** is a spanning tree of **G** if it is a tree and contains every vertex of **G**. If the number of Vertices is n then (n-1) edges will be required to find connected graph.

Example:



Graph

spanning tree 1

spanning tree 2

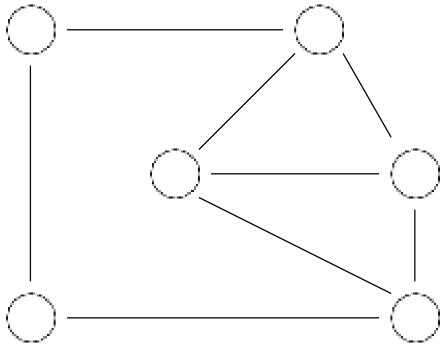
spanning tree 3

Spanning Trees

Theorem: Every connected graph has a spanning tree.

Question: Why is this true?

Question: Given a connected graph **G**, how can you find a spanning tree of **G**?

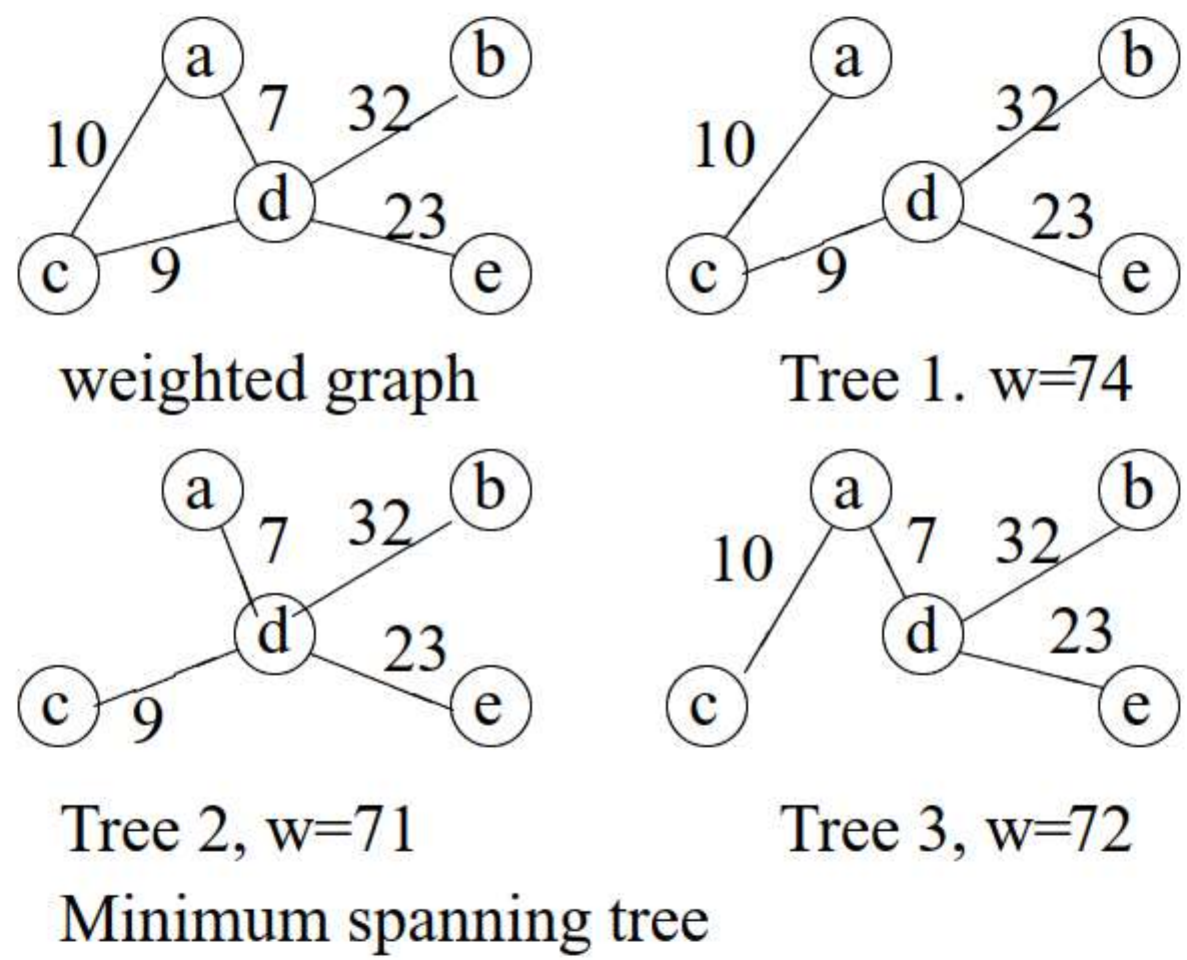


Weighted Graphs

Weighted Graphs: A weighted graph is a graph, in which each edge has a weight (some real number).

Weight of a Graph: The sum of the weights of all edges.

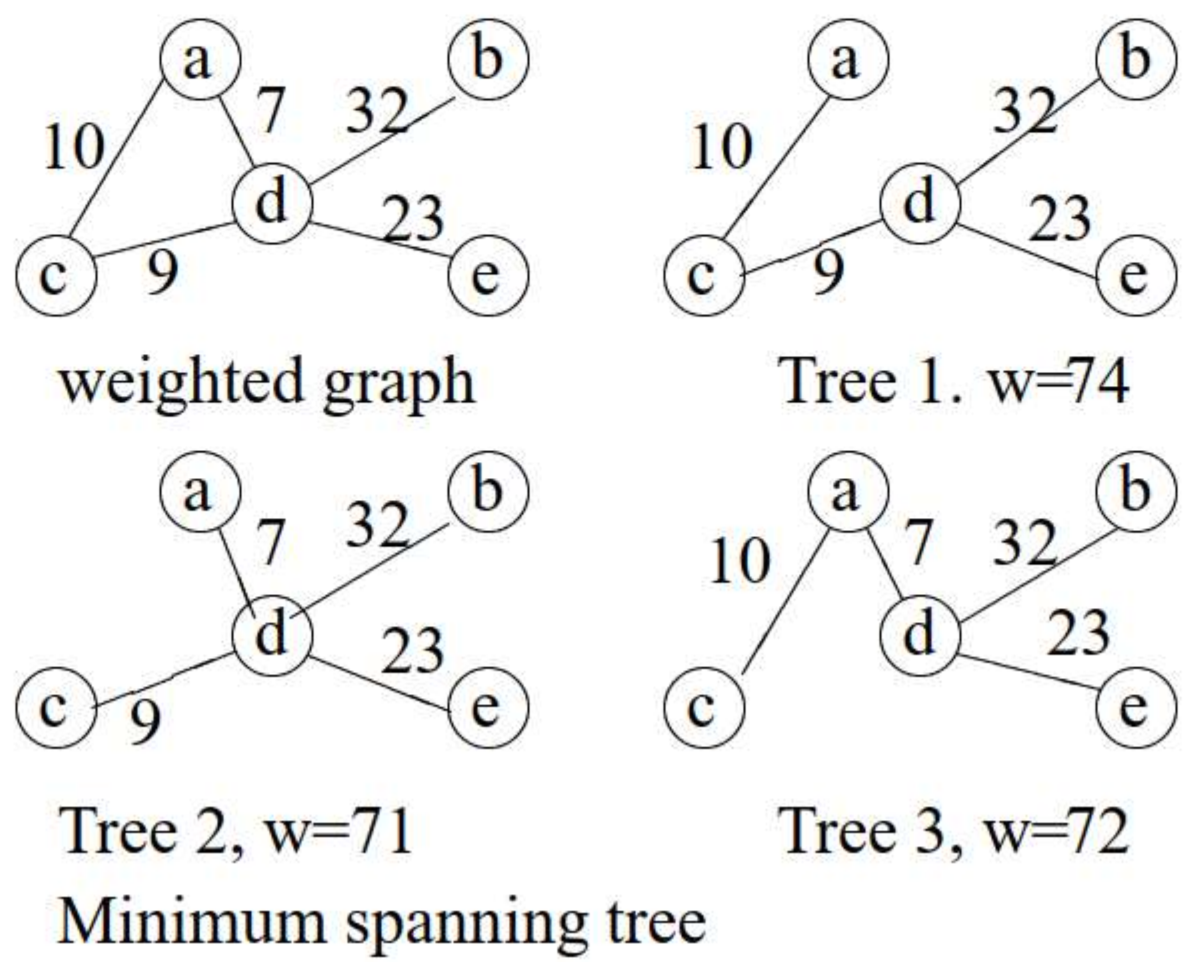
Example:



Minimum Spanning Trees

A **Minimum Spanning Tree** in an undirected connected weighted graph is a spanning tree of minimum weight (among all spanning trees).

Example:



weighted graph

Tree 1. w=74

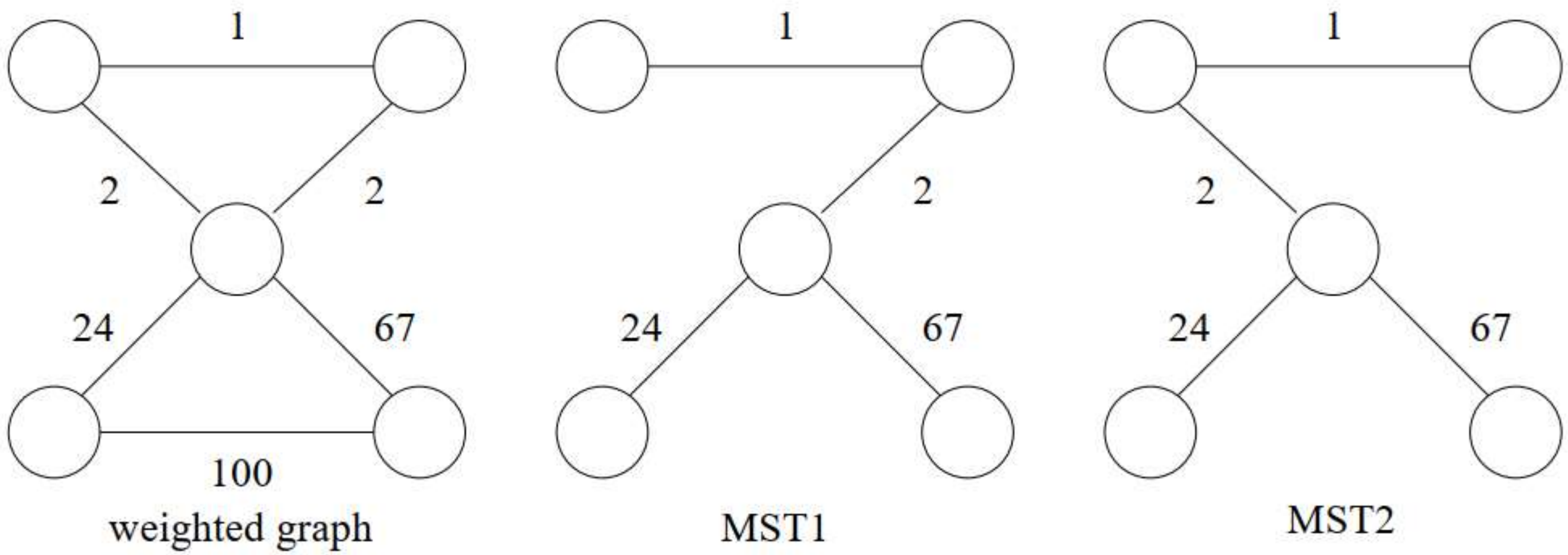
Tree 2, w=71
Minimum spanning tree

Tree 3, w=72

Minimum Spanning Trees

Remark: The minimum spanning tree may not be unique. However, if the weights of all the edges are pairwise distinct, it is indeed unique (we won't prove this now).

Example:



Minimum Spanning Tree Problem

MST Problem: Given a connected weighted undirected graph **G**, design an algorithm that outputs a minimum spanning tree (MST) of **G**.

Question: What is most intuitive way to solve?

Generic approach: A tree is an acyclic graph.
The idea is to start with an empty graph and try to add edges one at a time, always making sure that what is built remains acyclic. And if we are sure every time the resulting graph always is a subset of some minimum spanning tree, we are done.

Generic Algorithm for MST problem

Let **A** be a set of edges such that **A ⊆ T**, where **T** is a MST. An edge **(u,v)** is a *safe edge* for A, if **A ∪ {(u,v)}** is also a subset of some MST.

If at each step, we can find a safe edge **(u,v)**, we can 'grow' a MST. This leads to the following generic approach:

```
Generic-MST(G, w)
Let A=EMPTY;
 while A does not form a spanning tree
 find an edge (u, v) that is safe for A  add (u, v)
  to A
 return A
```

How can we find a safe edge?

Prim's Algorithm : How to grow a tree

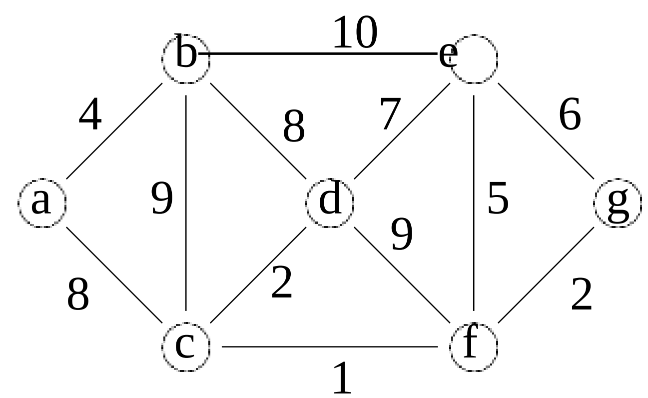
Grow a Tree

- Start by picking any vertex **R** to be the root of the tree.
- While the tree does not contain all vertices in the graph
  - find shortest edge leaving the tree and add it to the tree .

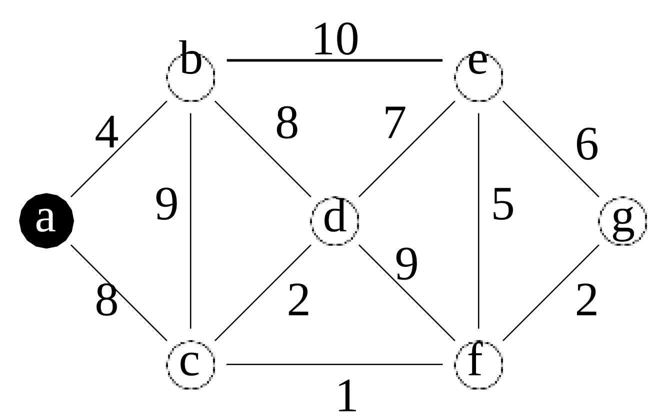
Running time is $O((|V| + |E|)\log |V|)$.

Prim’s Algorithm

Worked Example



Connected graph



Step 0

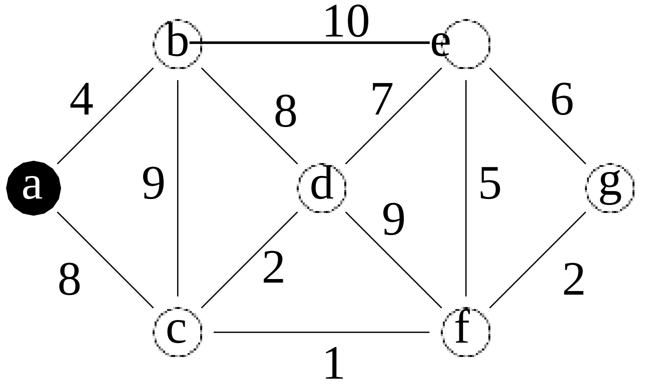
S={a}

V \ S = {b,c,d,e,f,g}

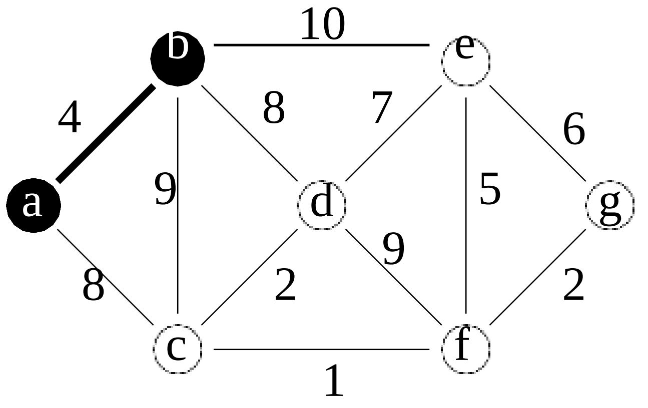
lightest edge = {a,b}

Prim’s Algorithm

Prim’s Example – Continued



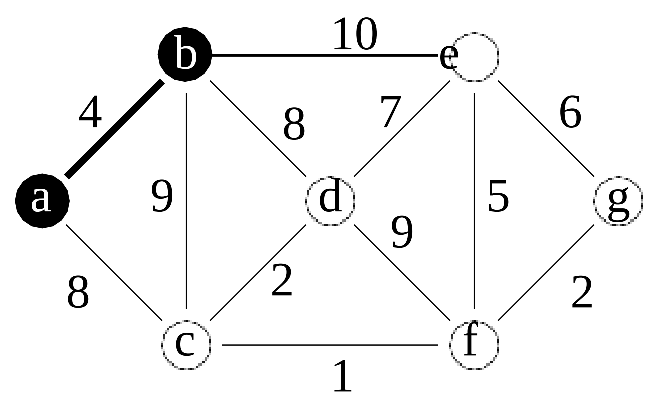
Step 1.1 before
S={a}
V \ S = {b,c,d,e,f,g}
A={}
lightest edge = {a,b}



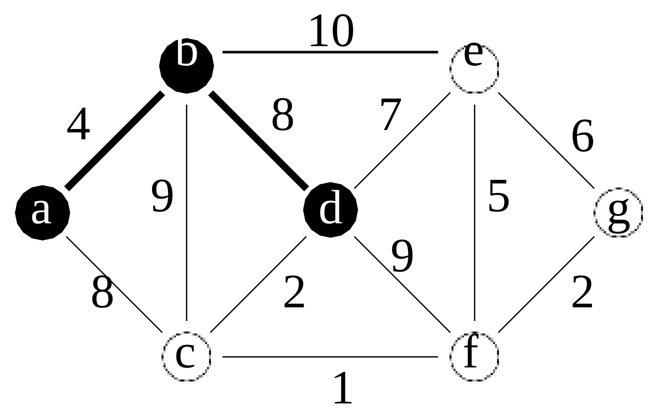
Step 1.1 after
S={a,b}
V \ S = {c,d,e,f,g}
A={{a,b}}
lightest edge = {b,d}, {a,c}

Prim’s Algorithm

Prim’s Example – Continued



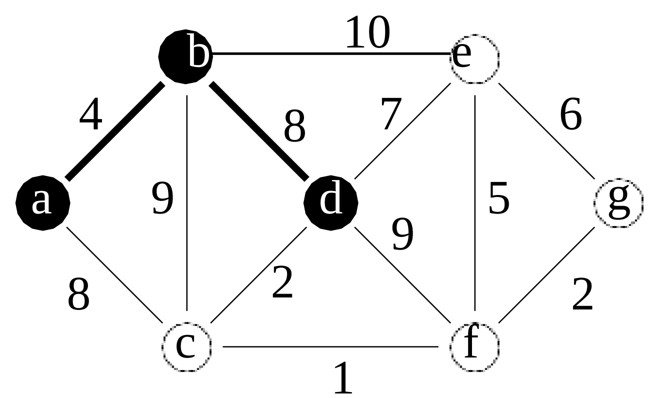
Step 1.2 before
S={a,b}
V \ S = {c,d,e,f,g}
A={{a,b}}
lightest edge = {b,d}, {a,c}



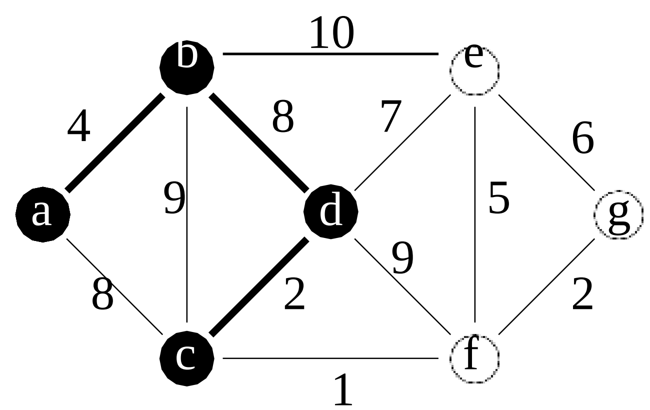
Step 1.2 after
S={a,b,d}
V \ S = {c,e,f,g}
A={{a,b},{b,d}}
lightest edge = {d,c}

Prim’s Algorithm

Prim’s Example – Continued



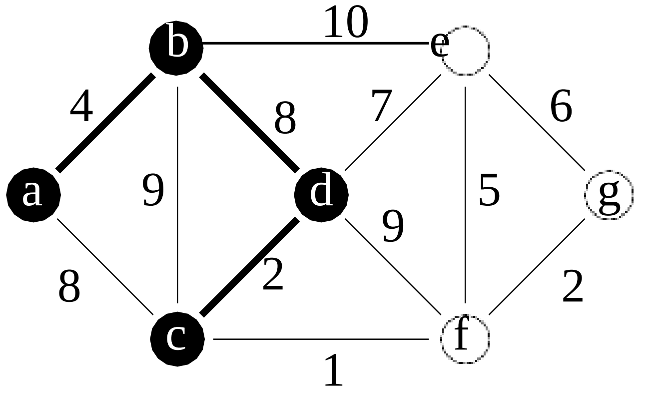
Step 1.3 before
S={a,b,d}
V \ S = {c,e,f,g}
A={{a,b},{b,d}}
lightest edge = {d,c}



Step 1.3 after
S={a,b,c,d}
V \ S = {e,f,g}
A={{a,b},{b,d},{c,d}}
lightest edge = {c,f}

Prim's Algorithm

Prim's Example – Continued



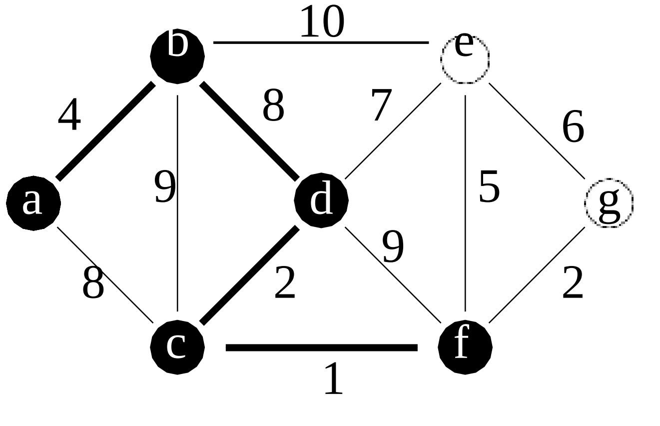
Step 1.4 before

S={a,b,c,d}

V \ S = {e,f,g}

A={{a,b},{b,d},{c,d}}

lightest edge = {c,f}



Step 1.4 after

S={a,b,c,d,f}

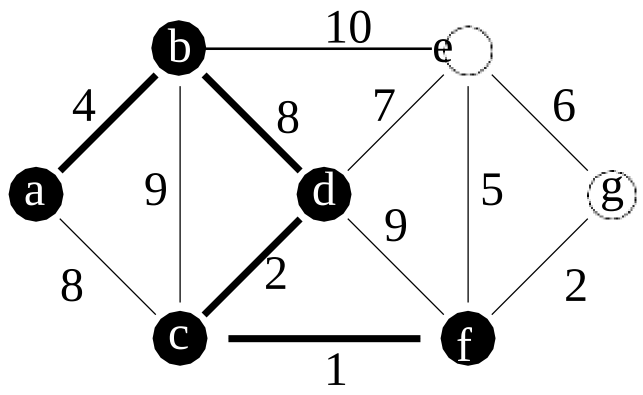
V \ S = {e,g}

A={{a,b},{b,d},{c,d},{c,f}}

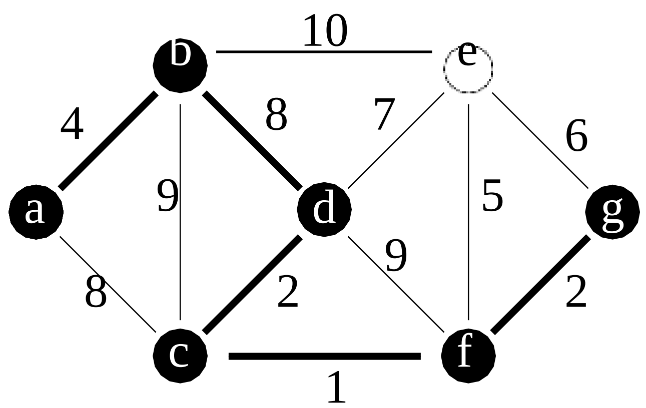
lightest edge = {f,g}

Prim’s Algorithm

Prim’s Example – Continued



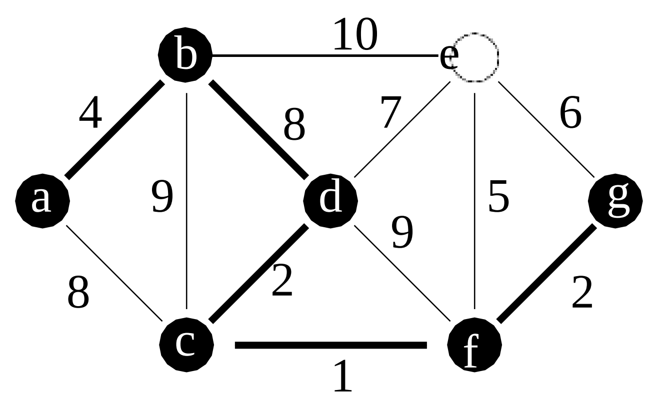
Step 1.5 before
S={a,b,c,d,f}
V \ S = {e,g}
A={{a,b},{b,d},{c,d},{c,f}}
lightest edge = {f,g}



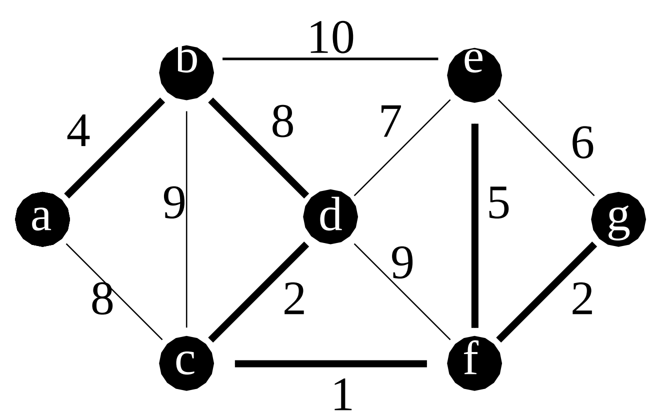
Step 1.5 after
S={a,b,c,d,f,g}
V \ S = {e}
A={{a,b},{b,d},{c,d},{c,f},
{f,g}}
lightest edge = {f,e}

Prim’s Algorithm

Prim’s Example – Continued



Step 1.6 before
S={a,b,c,d,f,g}
V \ S = {e}
A={{a,b},{b,d},{c,d},{c,f},
{f,g}}
lightest edge = {f,e}



Step 1.6 after
S={a,b,c,d,e,f,g}
V \ S = {}
A={{a,b},{b,d},{c,d},{c,f},
{f,g},{f,e}}

MST completed

Prim's Algorithm

Step 0: Choose any element **r** and set $S = \{r\}$ and $A = \emptyset$.
(Take **r** as the root of our spanning tree.)

Step 1: Find a lightest edge such that one endpoint is in **S** and the other is in $V \setminus S$. Add this edge to **A** and its (other) endpoint to **S**.

Step 2: If $V \setminus S = \emptyset$, then stop and output the minimum spanning tree **(S, A)**

Otherwise go to Step 1

Kruskal’s Algorithm

- Kruskal’s Algorithm is a famous greedy algorithm.
- It is used for finding the Minimum Spanning Tree (MST) of a given graph.
- To apply Kruskal’s algorithm, the given graph must be weighted, connected and undirected.

Kruskal’s Algorithm

Step-01:

- Sort all the edges from low weight to high weight.

Step-02:

- Take the edge with the lowest weight and use it to connect the vertices of graph.
- If adding an edge creates a cycle, then reject that edge and go for the next least weight edge.

Step-03:

- Keep adding edges until all the vertices are connected and a Minimum Spanning Tree (MST) is obtained.

Kruskal’s Algorithm

- Worst case time complexity of Kruskal’s Algorithm is O(ElogV) or O(ElogE)

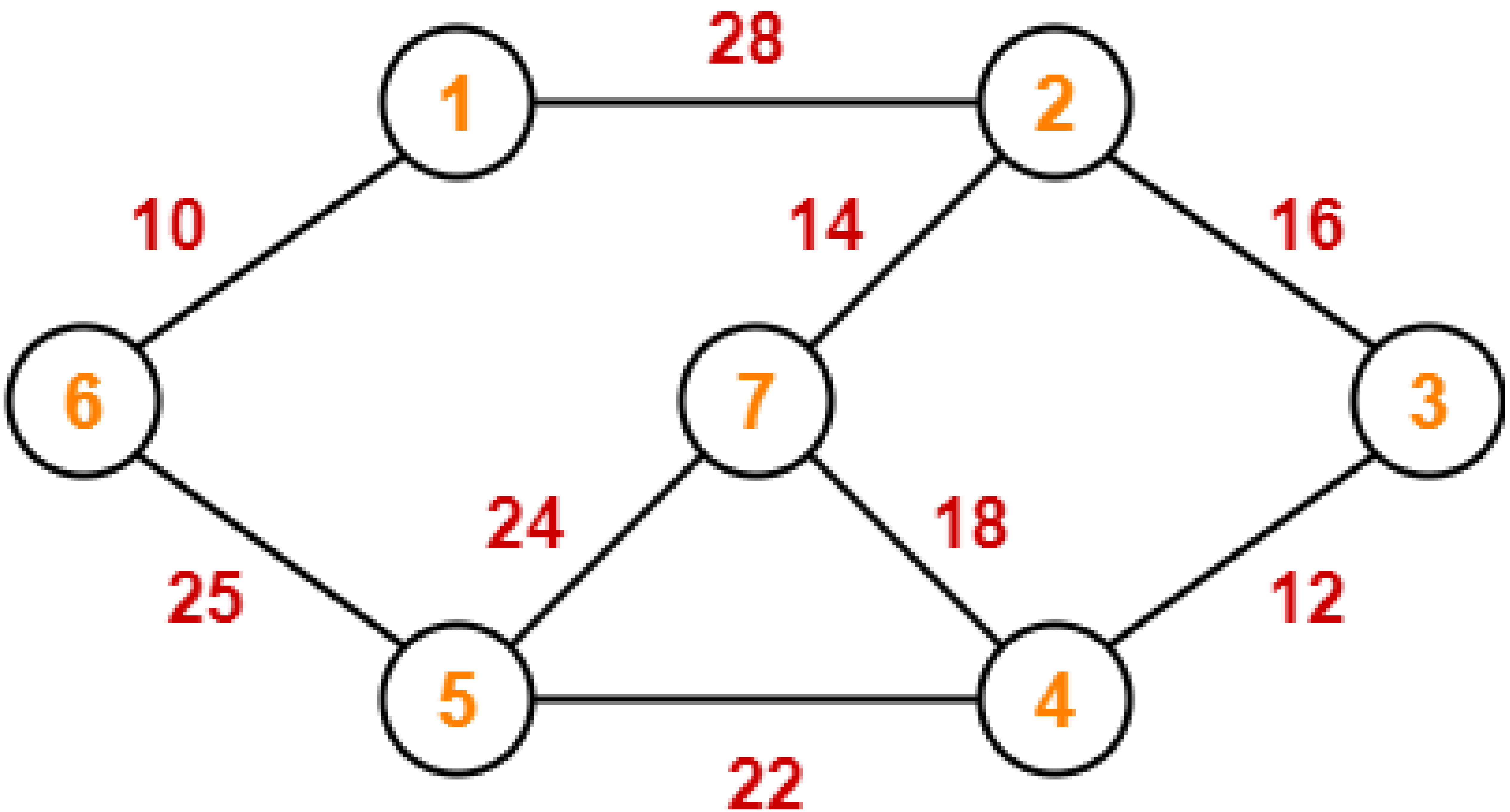
Solution :

- The edges can be maintained as min heap.
- The next edge can be obtained in O(logE) time if graph has E edges.
- Reconstruction of heap takes O(E) time. So, Kruskal’s Algorithm takes O(ElogE) time.
- The value of E can be at most $O(V^2)$. So, O(logV) and O(logE) are same.

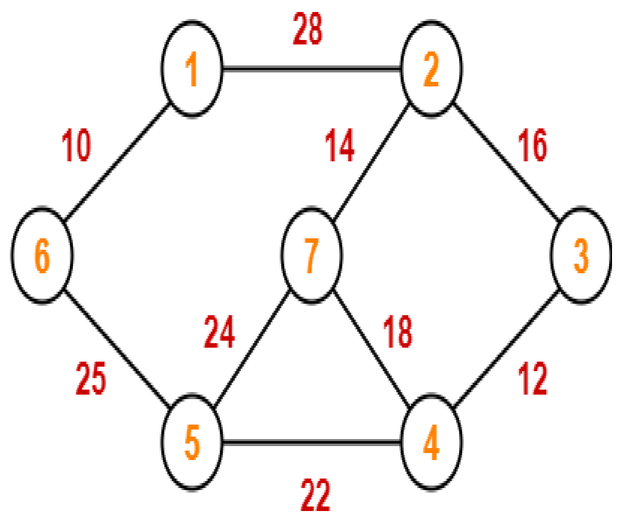
Alternate Solution:

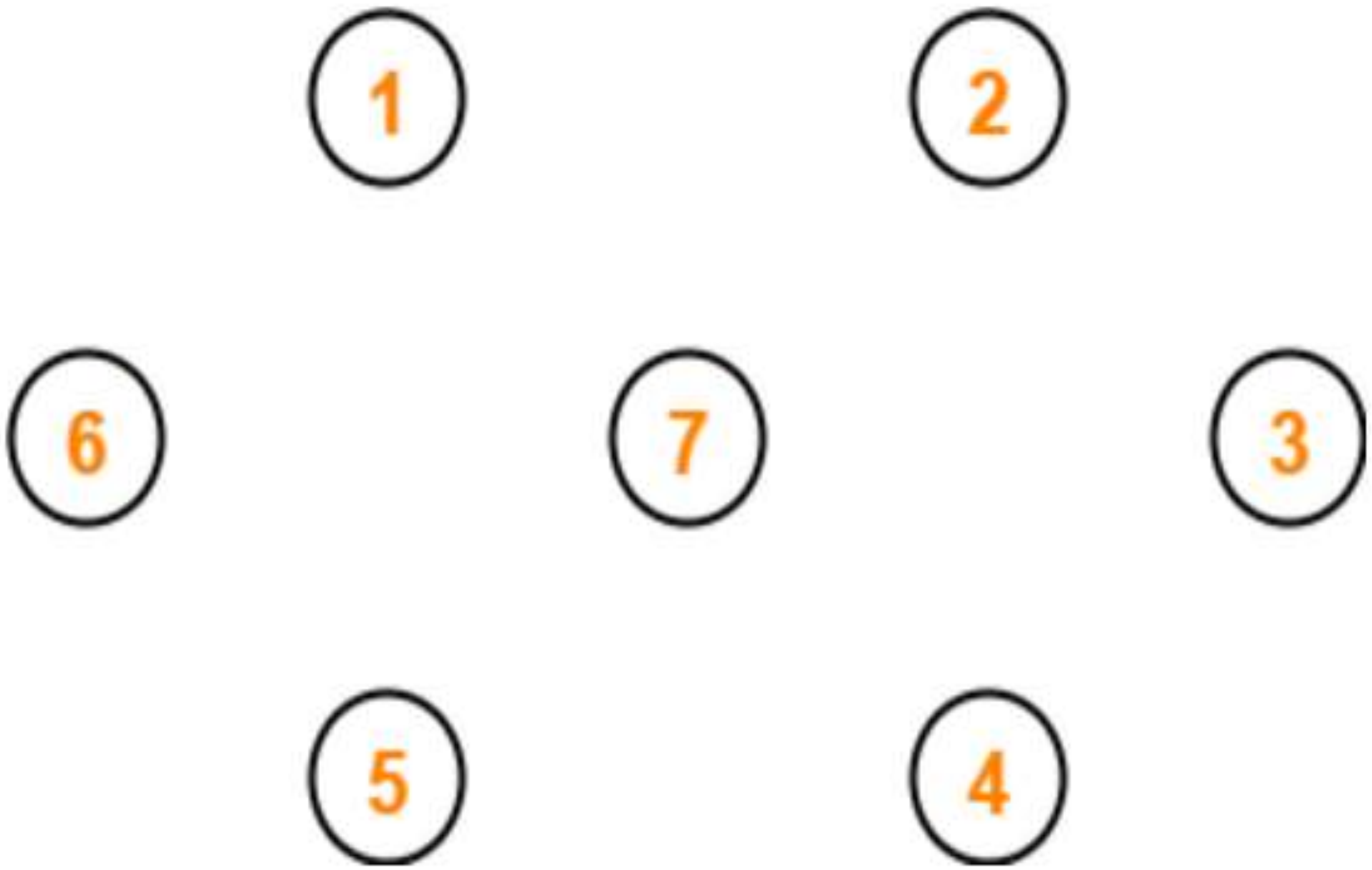
- If the edges are already sorted, then there is no need to construct min heap.
- So, deletion from min heap time is saved.
- In this case, time complexity of Kruskal’s Algorithm = O(E + V)

Example

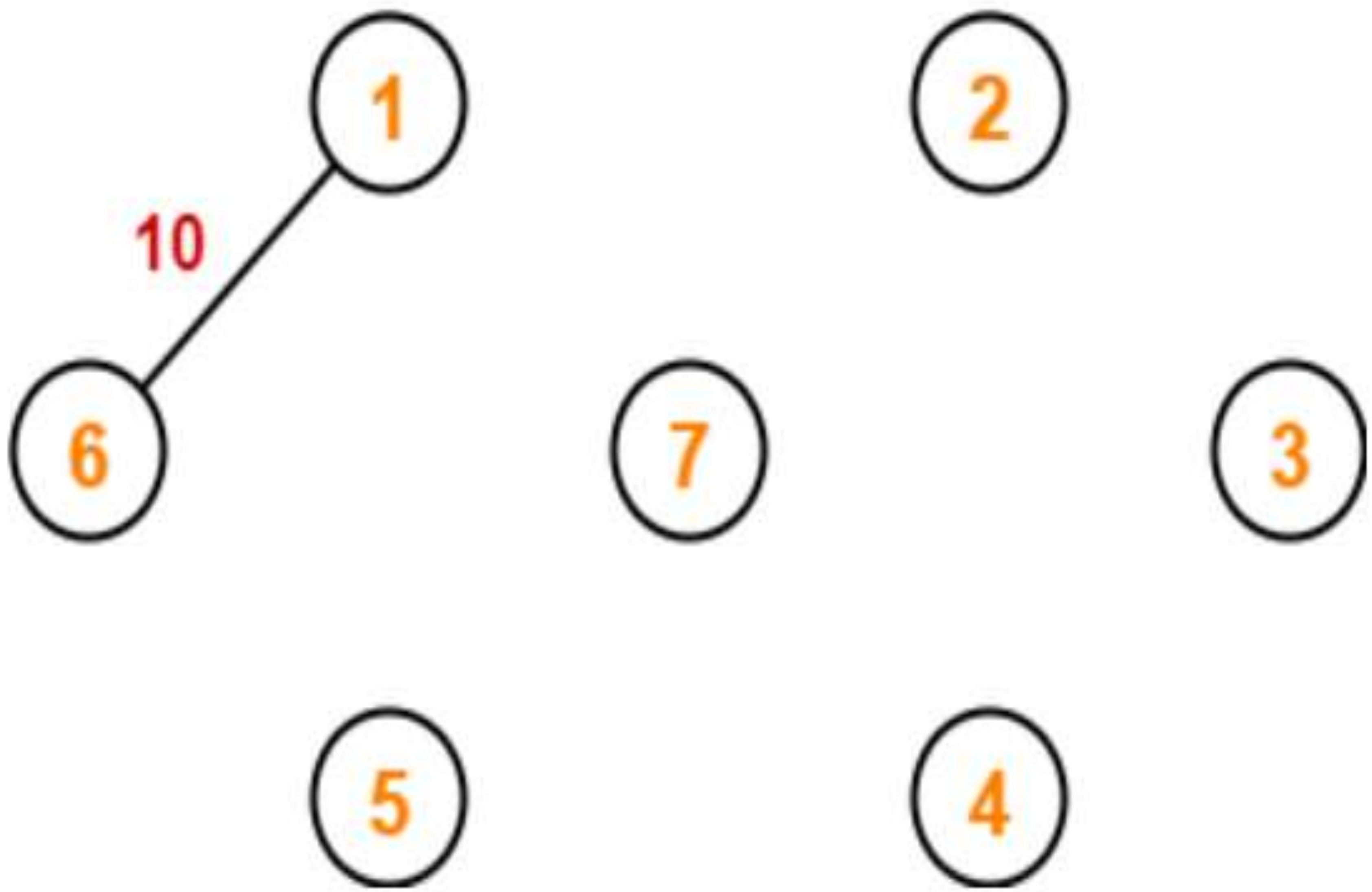


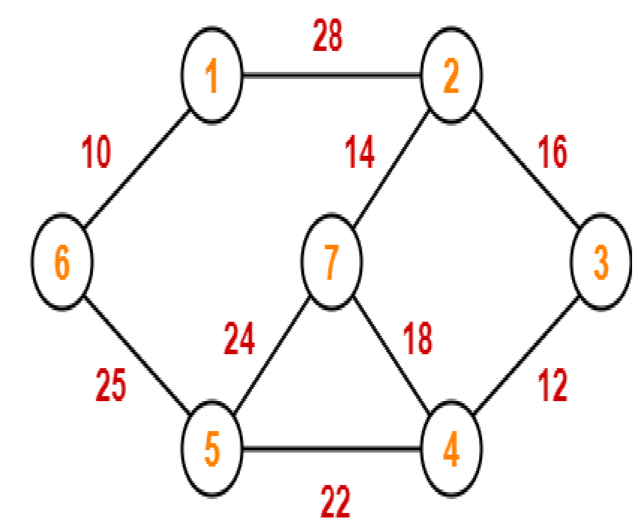
Example



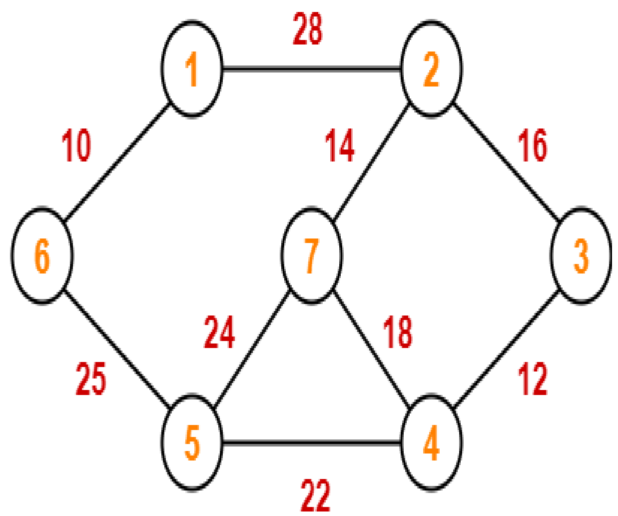


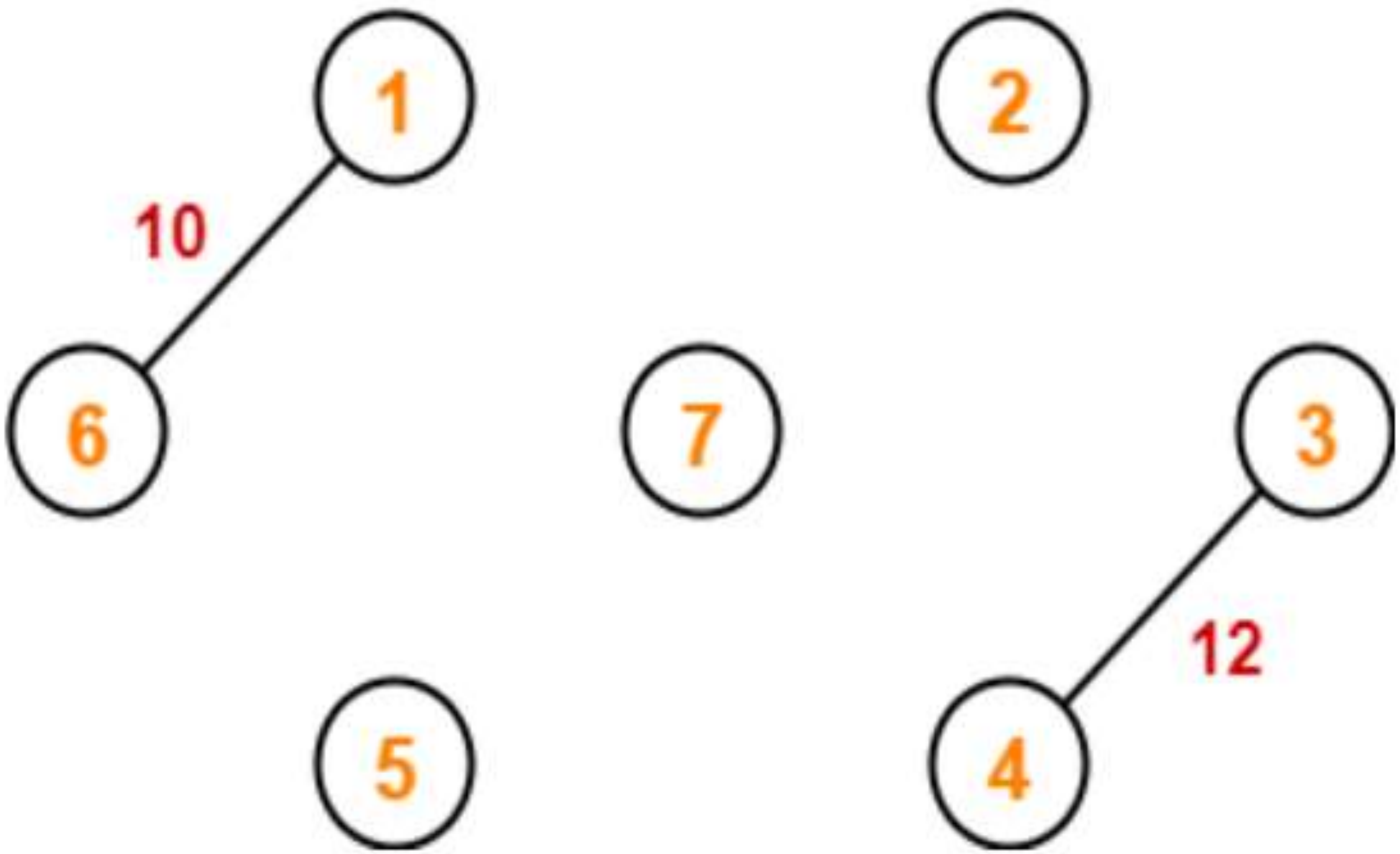
Example



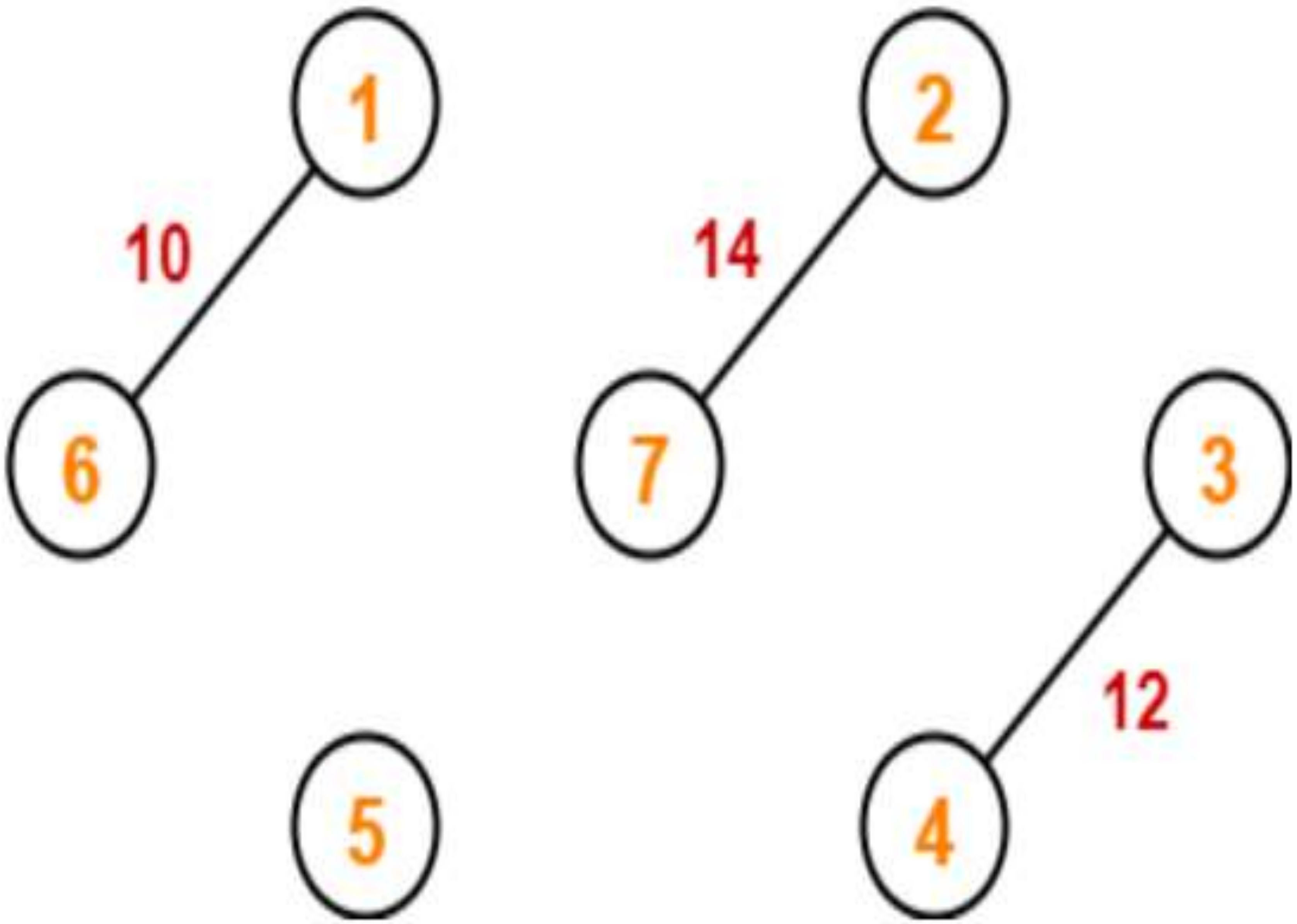


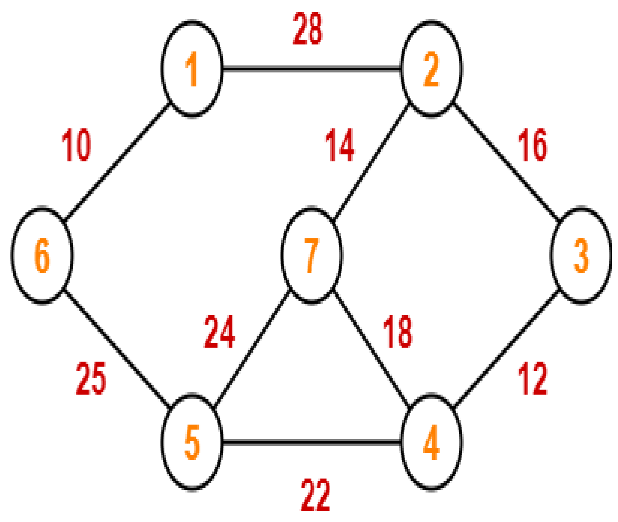
Example



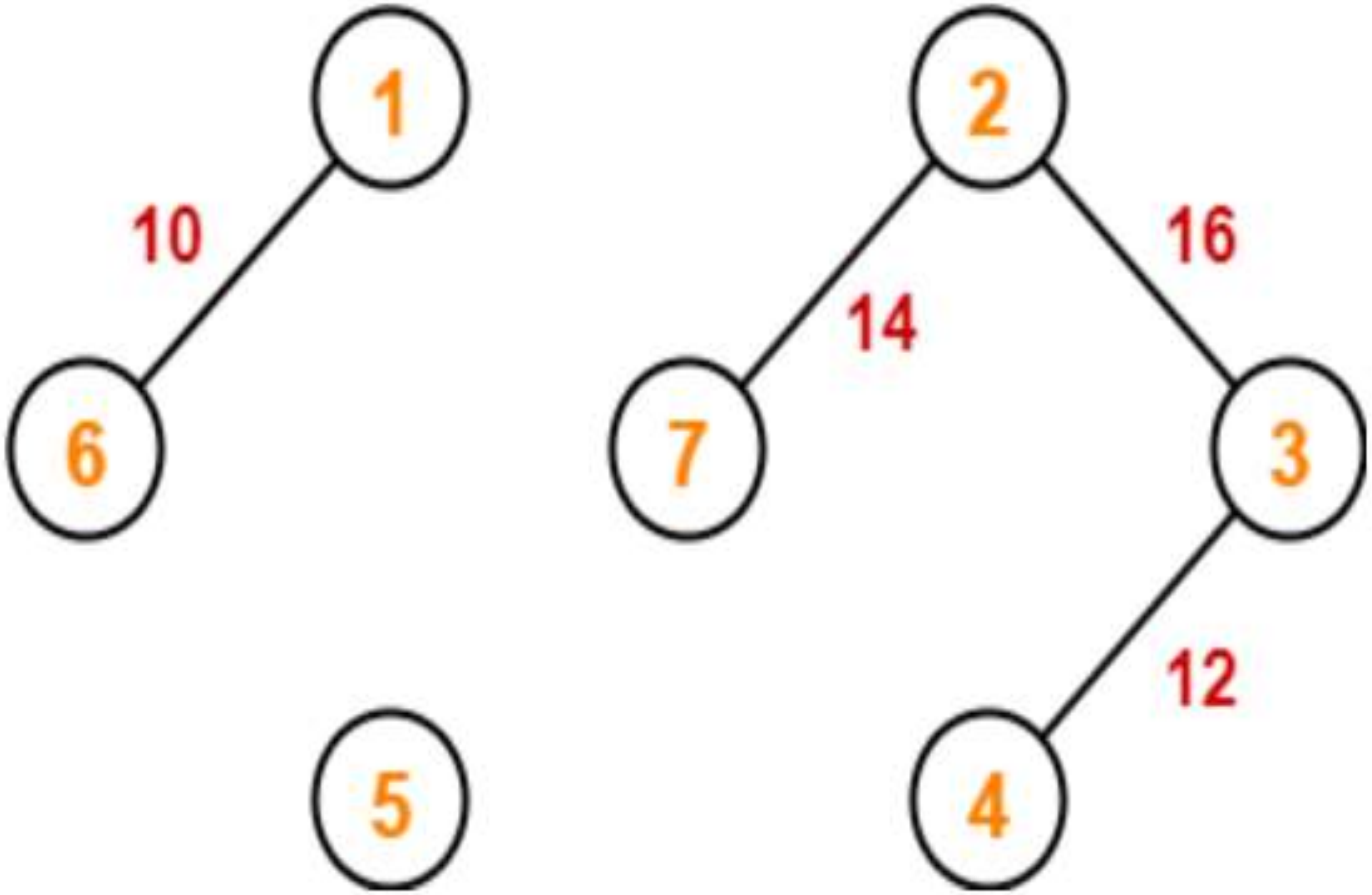


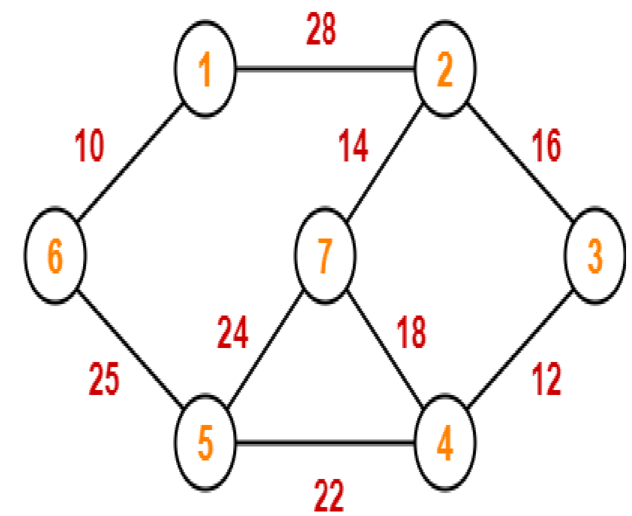
Example



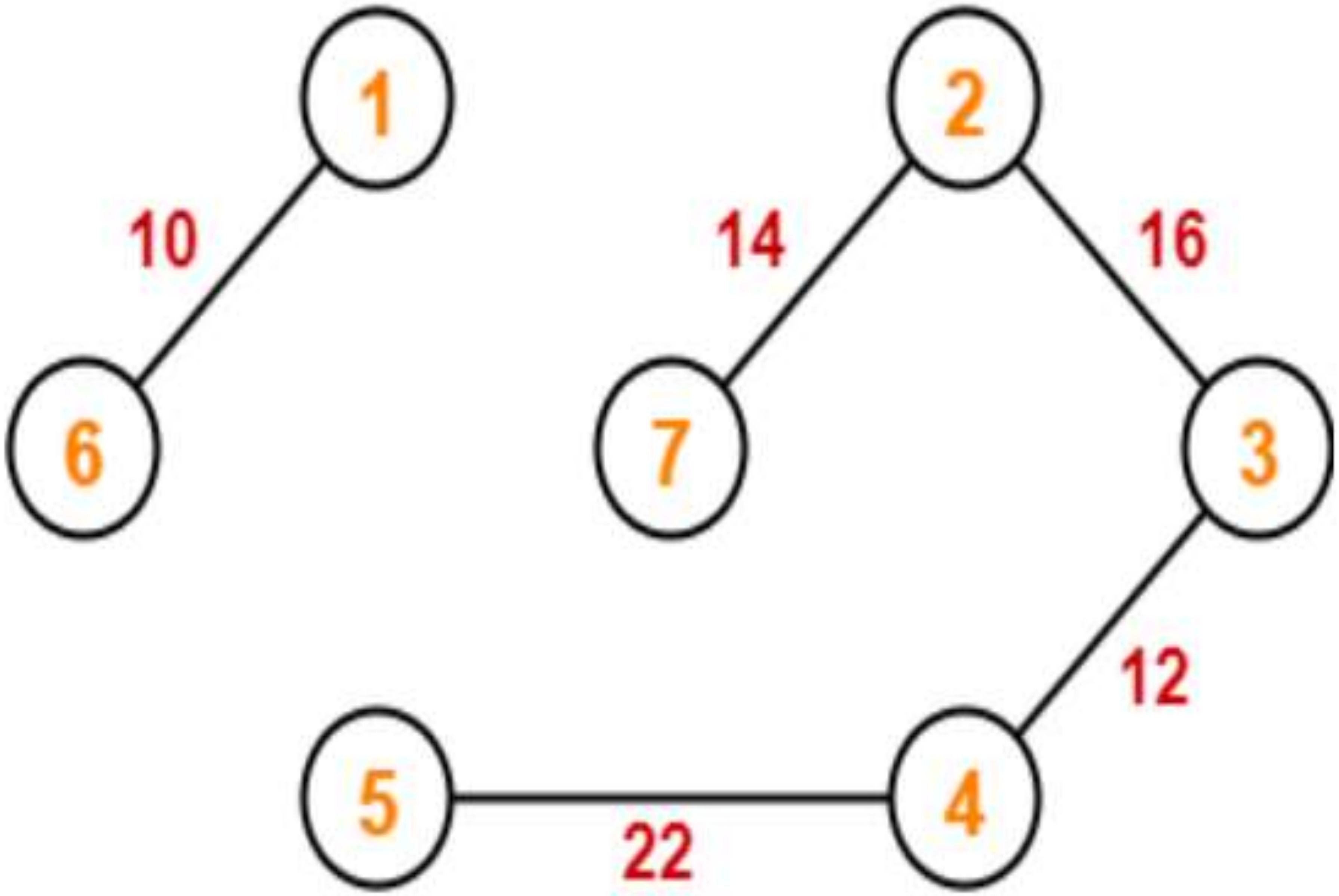


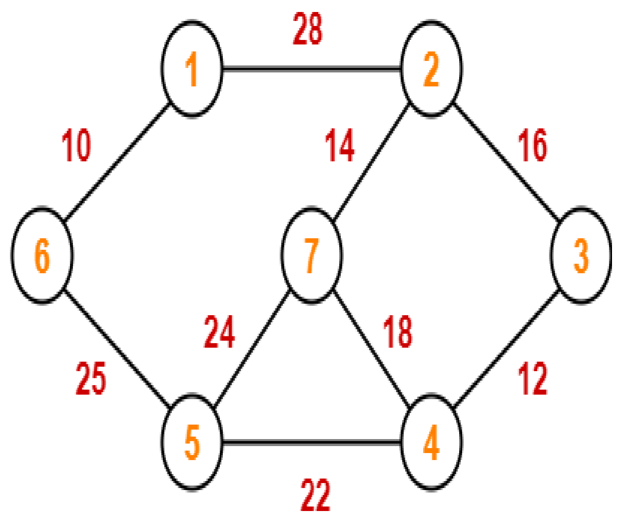
Example



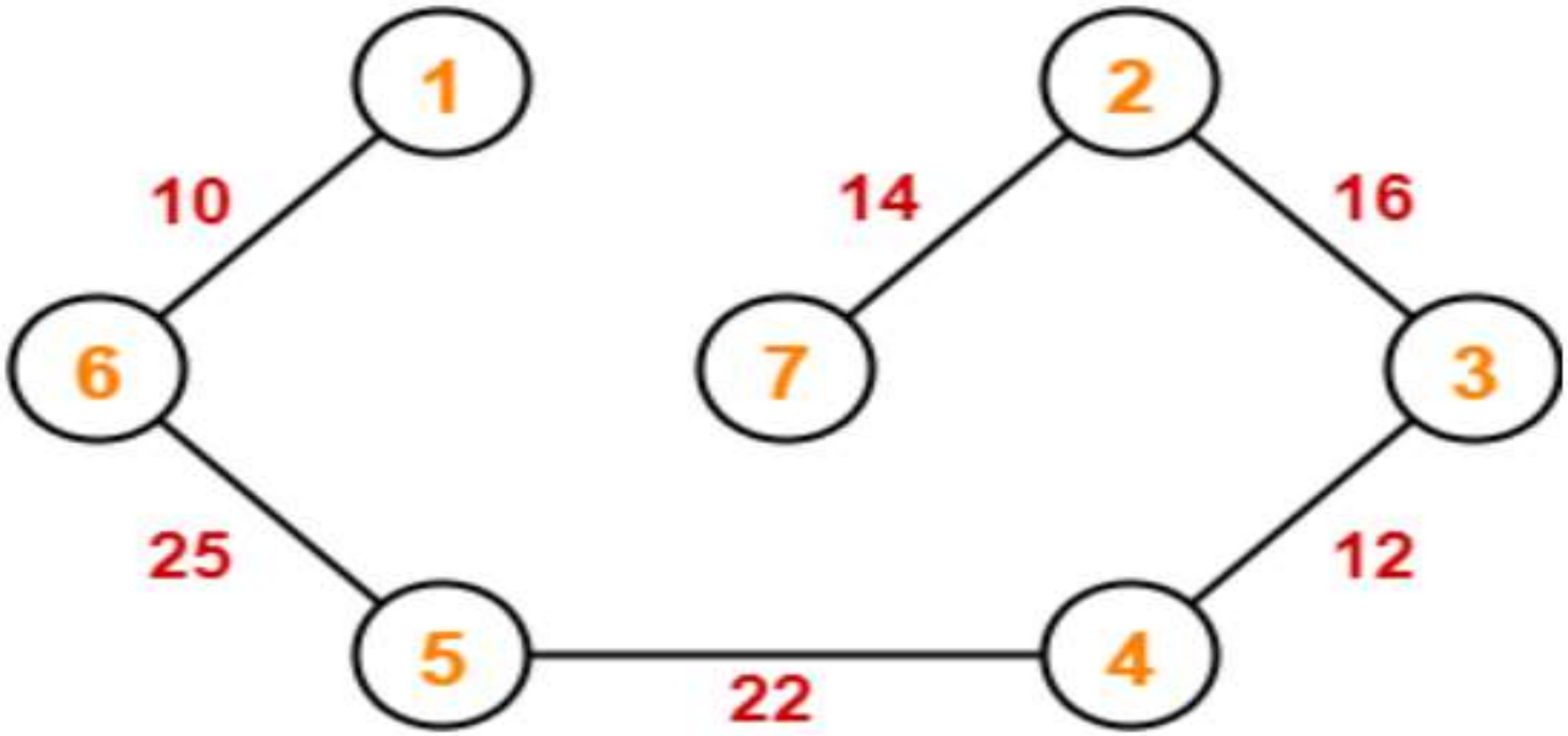


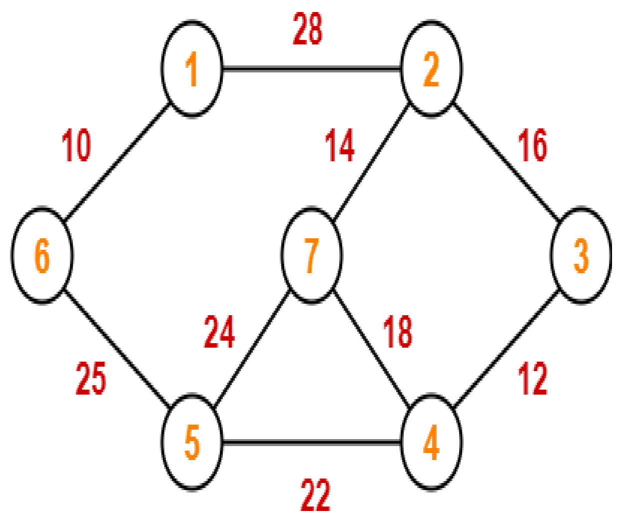
Example





Example





Since all the vertices have been connected / included in the MST, so we stop.

Weight of the MST = Sum of all edge weights

= 10 + 25 + 22 + 12 + 16 + 14

= 99 units