Spanning Tree

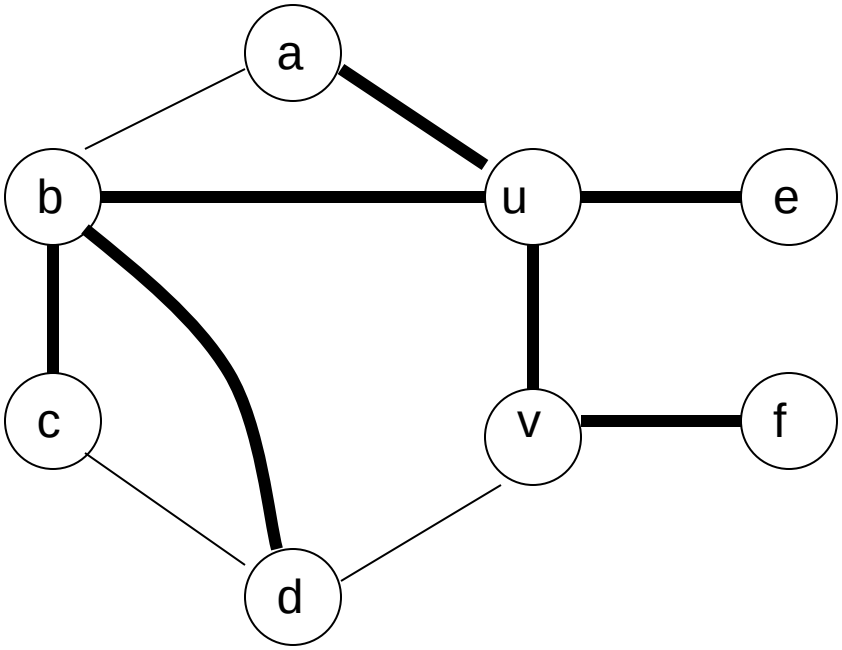
Dr. Sambit Bakshi

What is A Spanning Tree?

- A *spanning* tree for an undirected graph G=(V,E) is a subgraph of G that is a tree and contains all the vertices of G

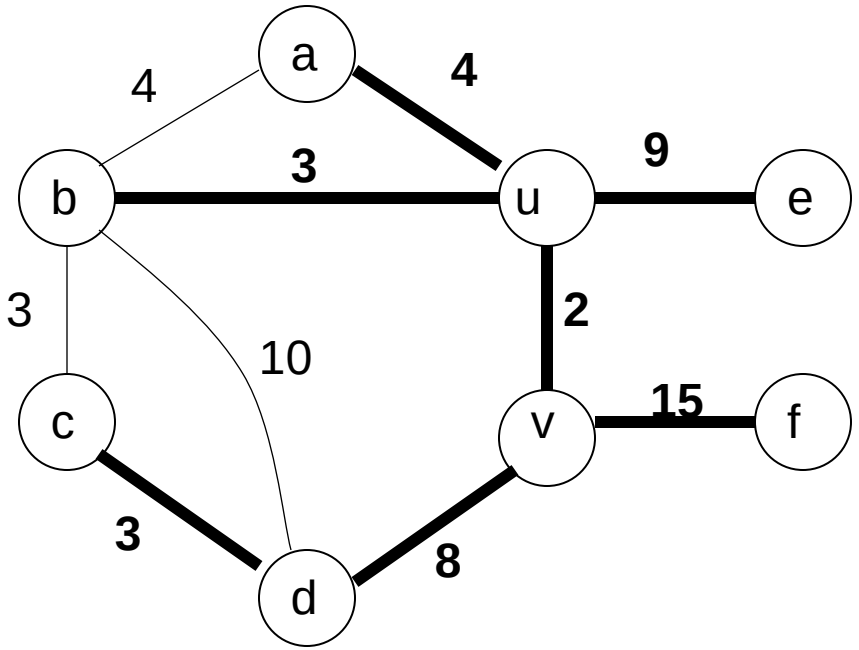
- Can a graph have more than one spanning tree? YES

- Can an unconnected graph have a spanning tree? **NO**



Minimal Spanning Tree.

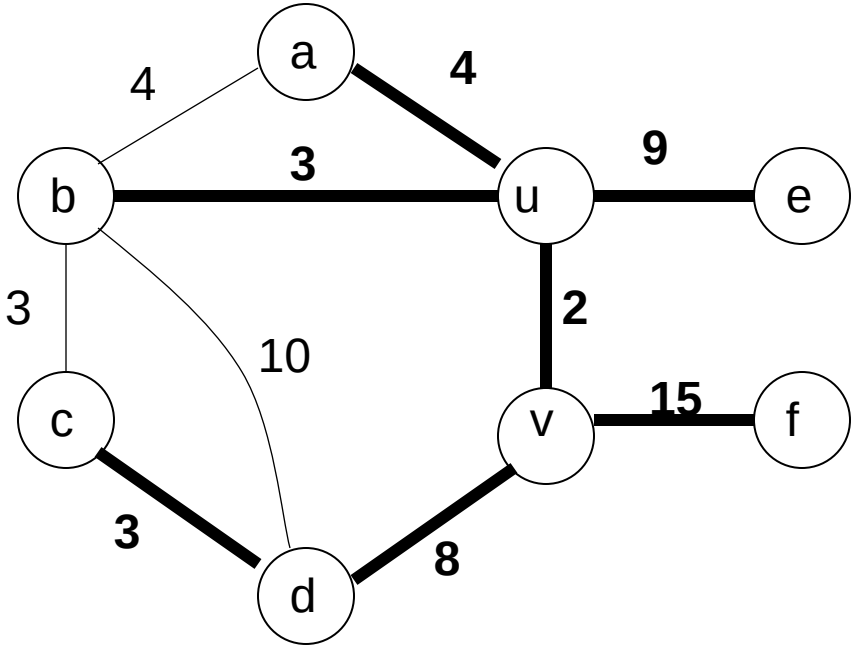
- The *weight* of a subgraph is the sum of the weights of it edges.
- A *minimum spanning tree* for a weighted graph is a spanning tree with minimum weight.

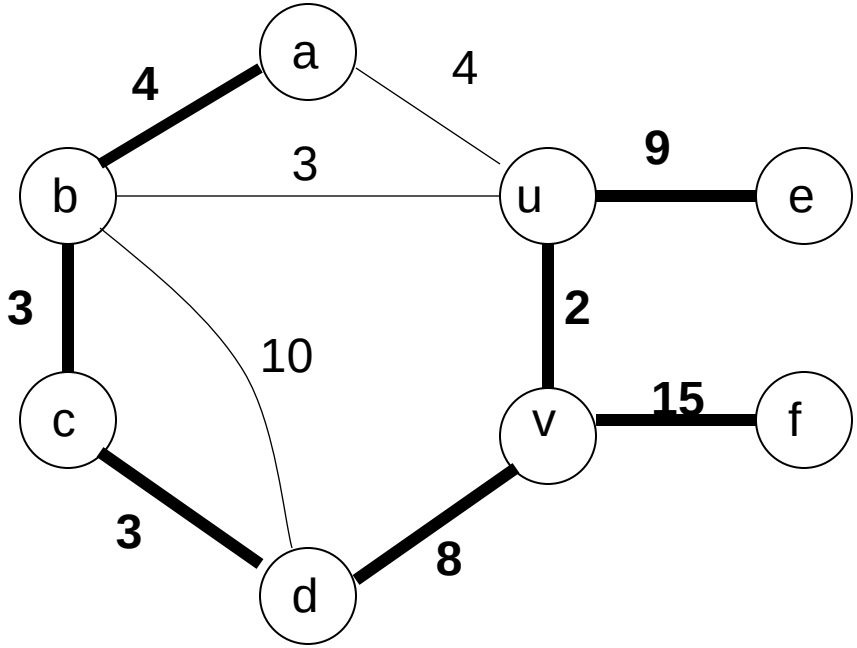


Mst T: $w(T) = \sum_{(u,v) \in T} w(u,v)$ is minimized

Minimal Spanning Tree.

- Can a graph have more then one minimum spanning tree? **YES**





MST Algorithms

We will show two ways to build a minimum spanning tree.

- A MST can be grown from the current spanning tree by adding the nearest vertex and the edge connecting the nearest vertex to the MST. (**Prim‘s algorithm**)

- A MST can be grown from a forest of spanning trees by adding the smallest edge connecting two spanning trees. (**Kruskal's algorithm**)

Notation

- Tree-vertices: in the tree constructed so far
- Non-tree vertices: rest of vertices

Prim's Selection rule

- Select the minimum weight edge between a tree-node and a non-tree node and add to the tree

Prim's algorithm : Main Idea

Select a vertex to be a tree-node

while (there are non-tree vertices) {

if there is no edge connecting a tree node with a non-tree node then

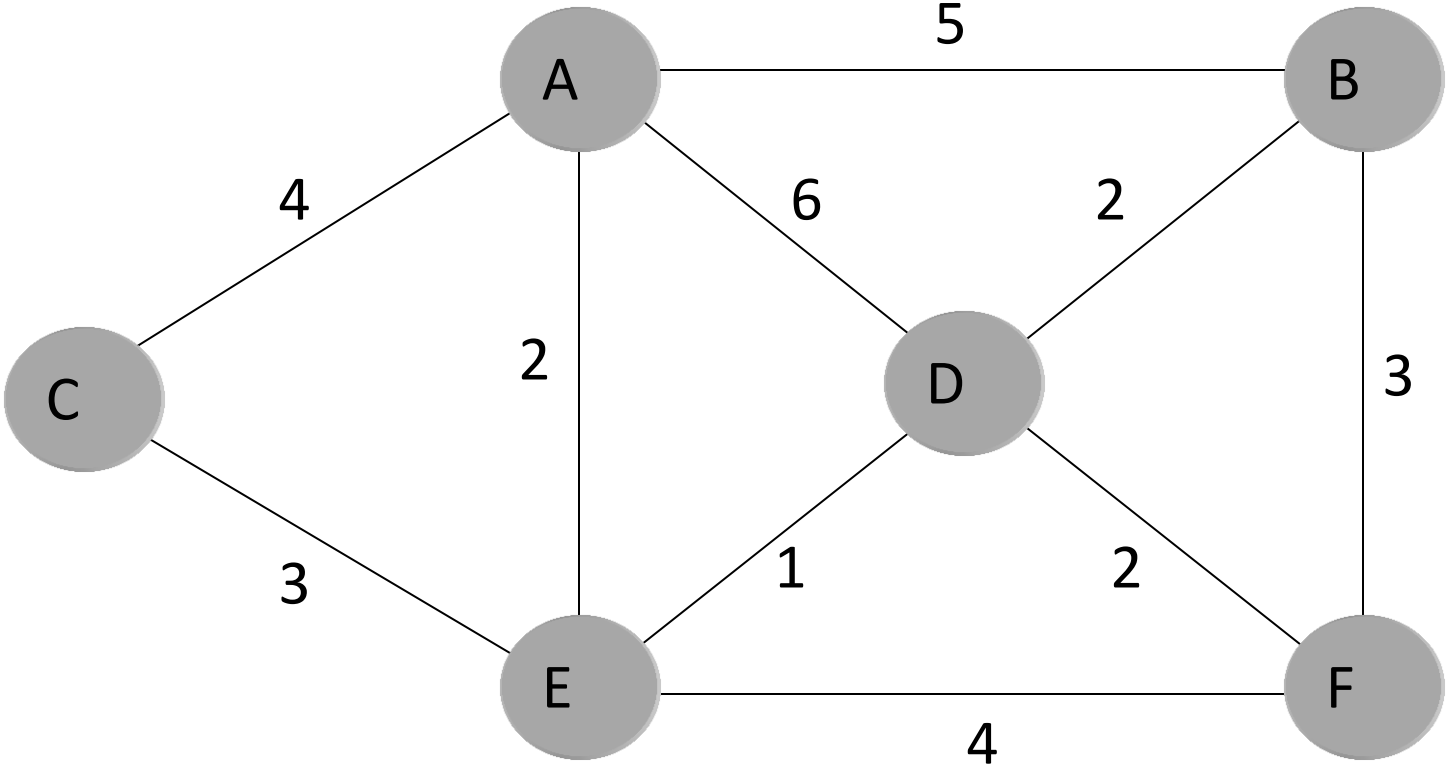
```
return "no spanning tree"
```

select an edge of minimum weight between a tree node and a non-tree node

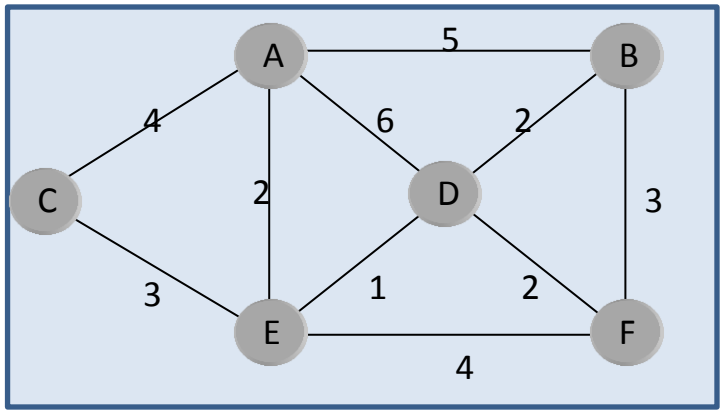
add the selected edge and its new vertex to the tree

}

return tree



Prim's Algorithm



A

B

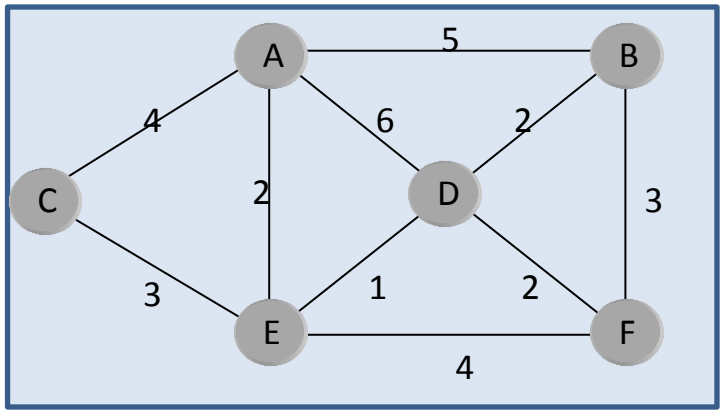
C

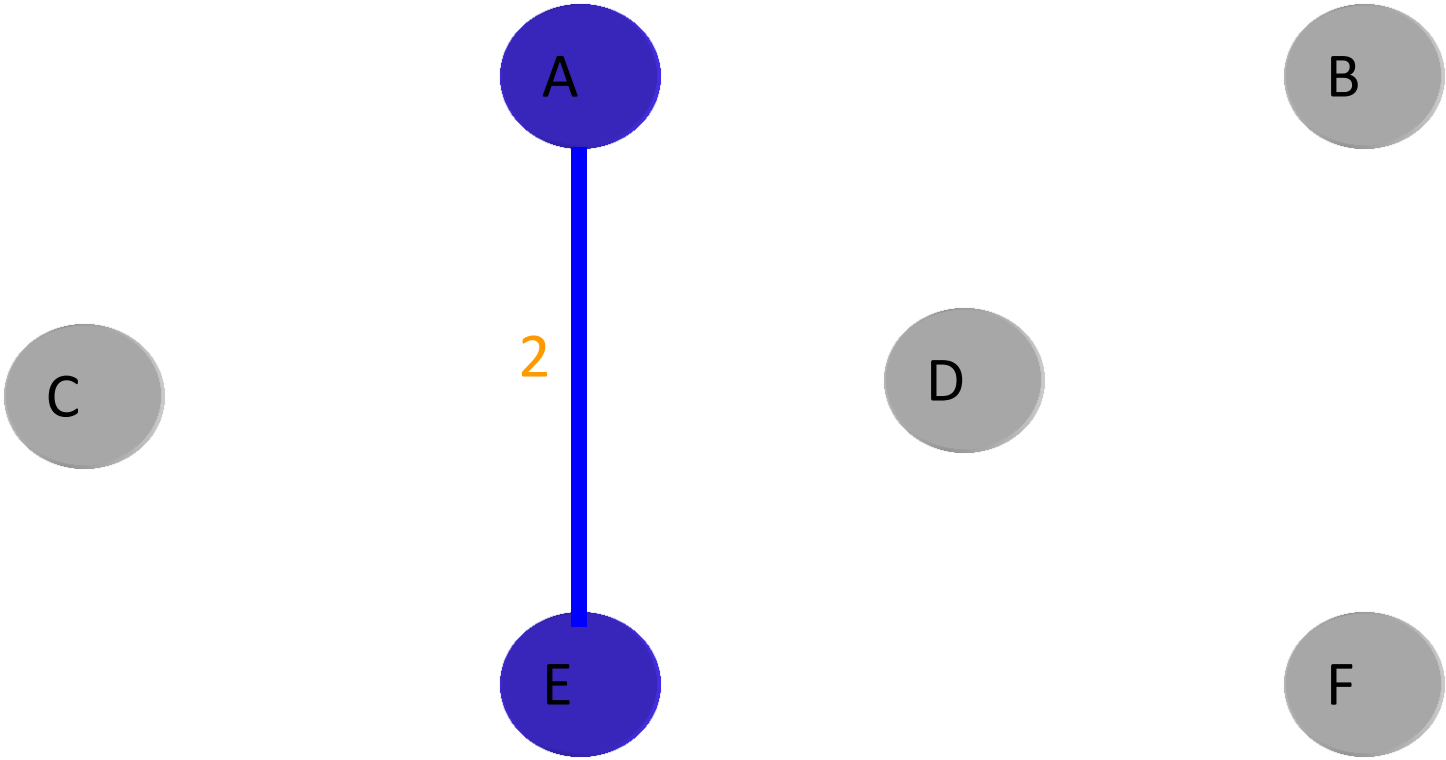
D

E

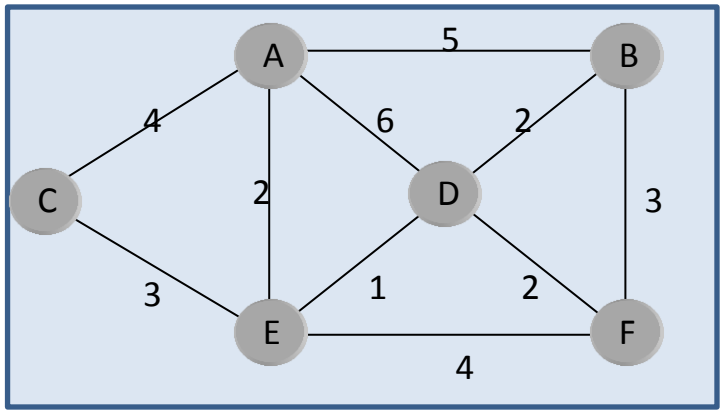
F

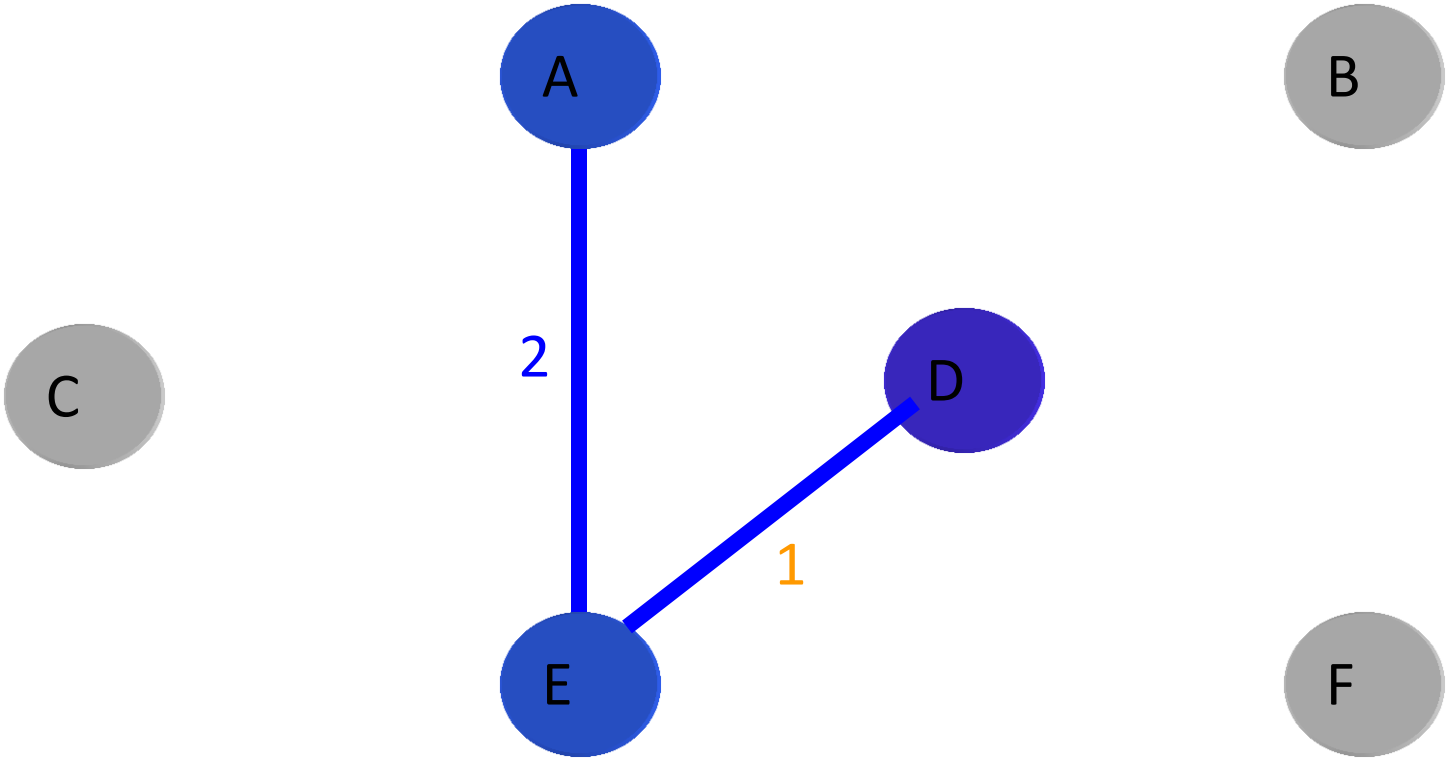
Prim's Algorithm



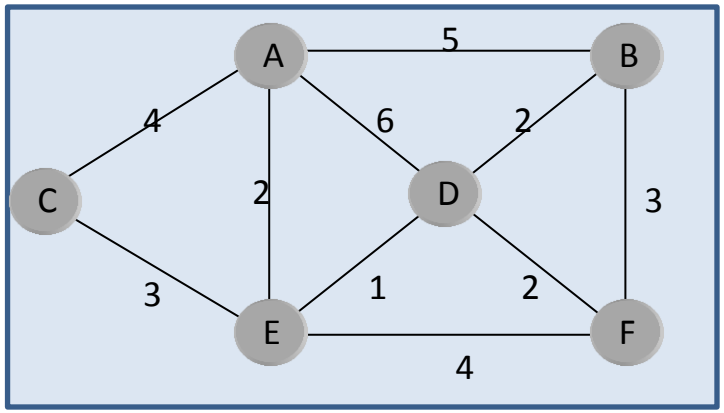


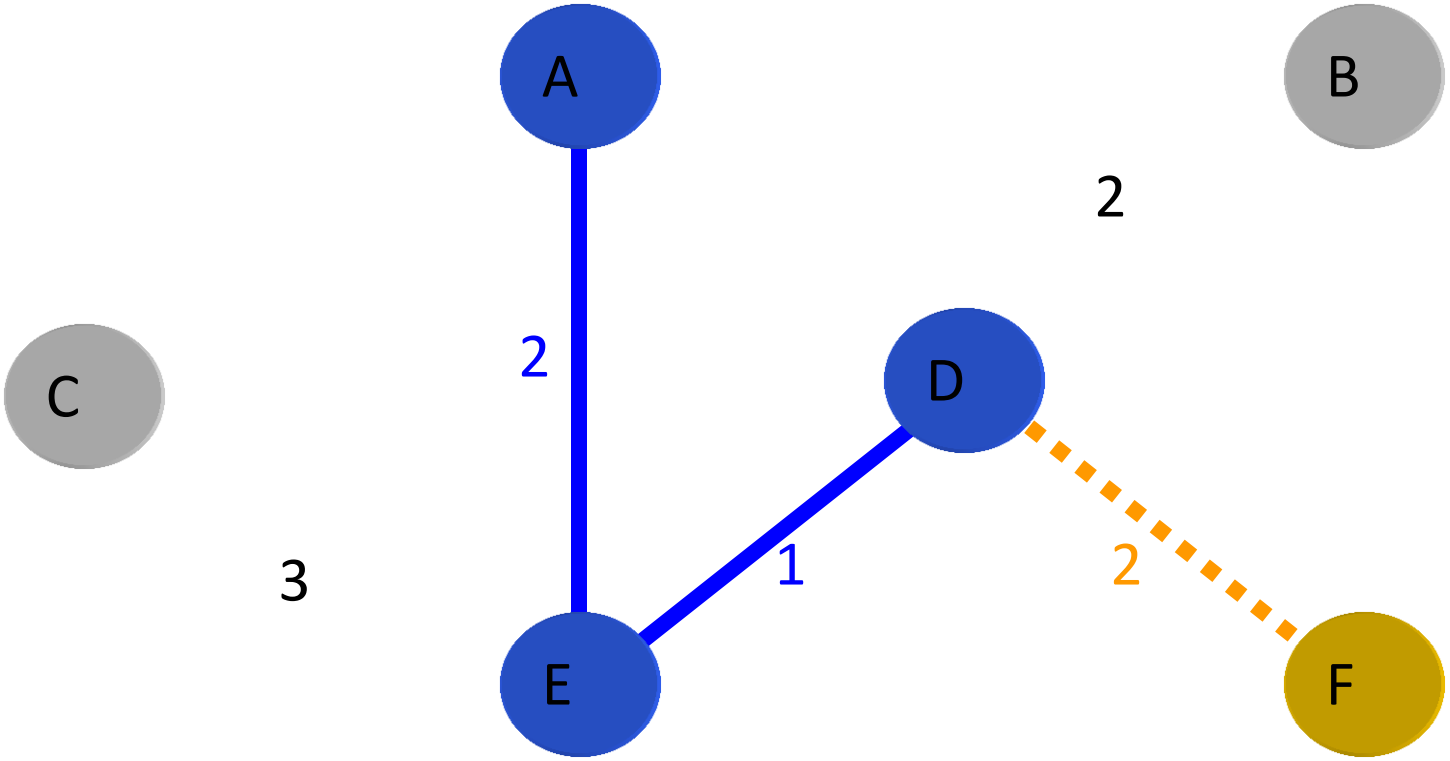
Prim's Algorithm

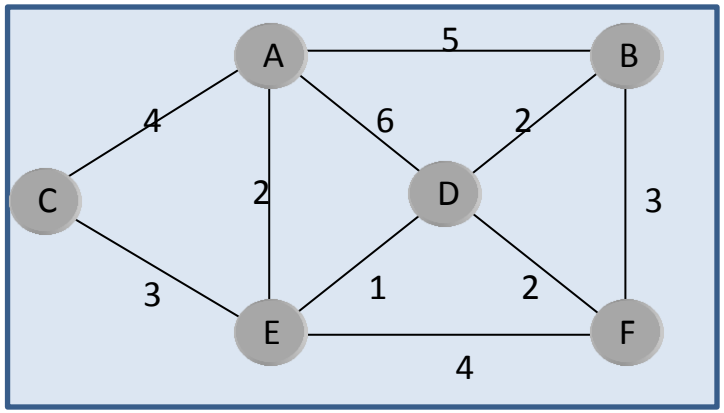


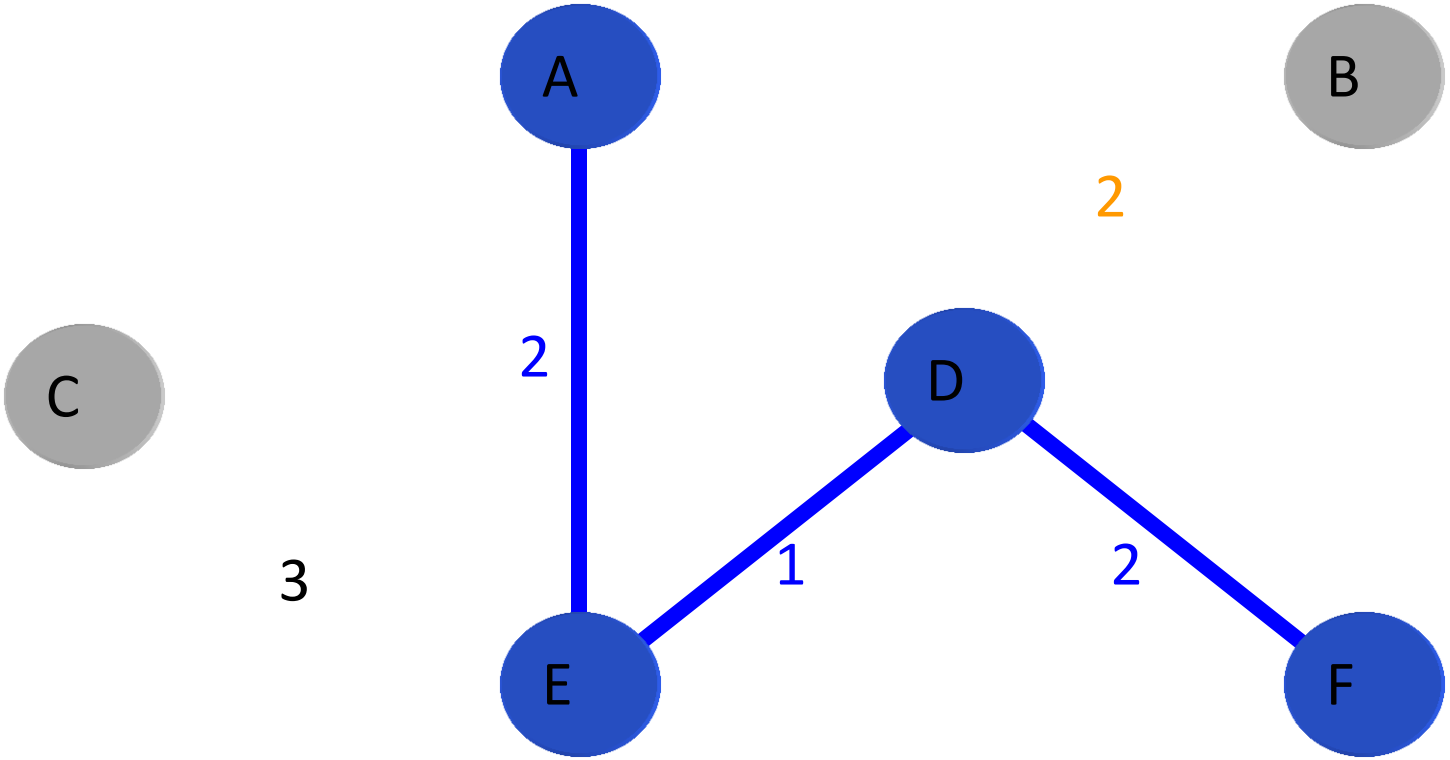


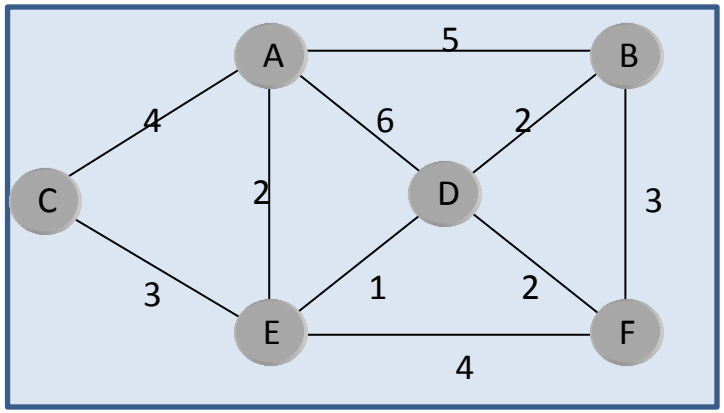
Prim's Algorithm

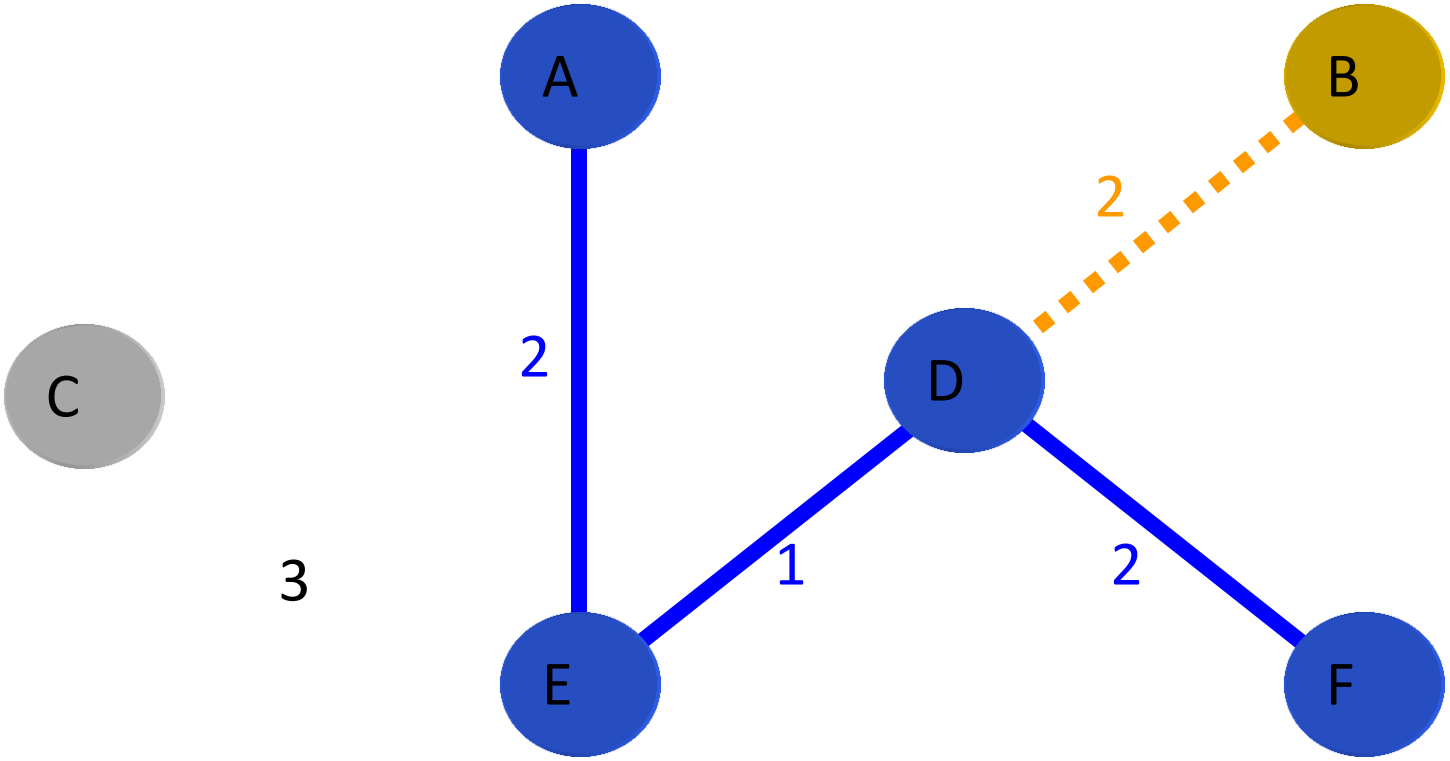

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

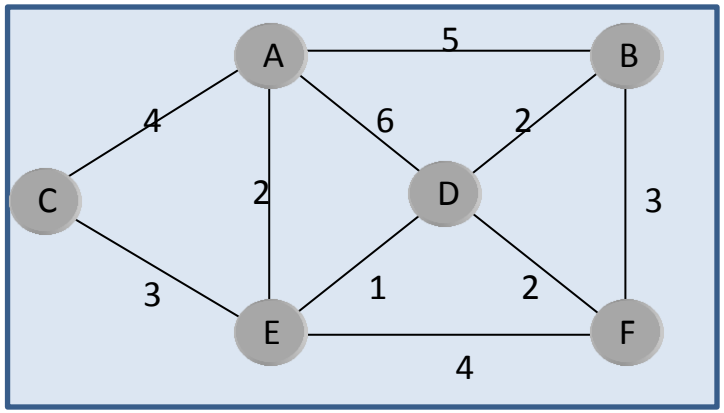

A
B
C
D
E
F
2
2
3
1
2

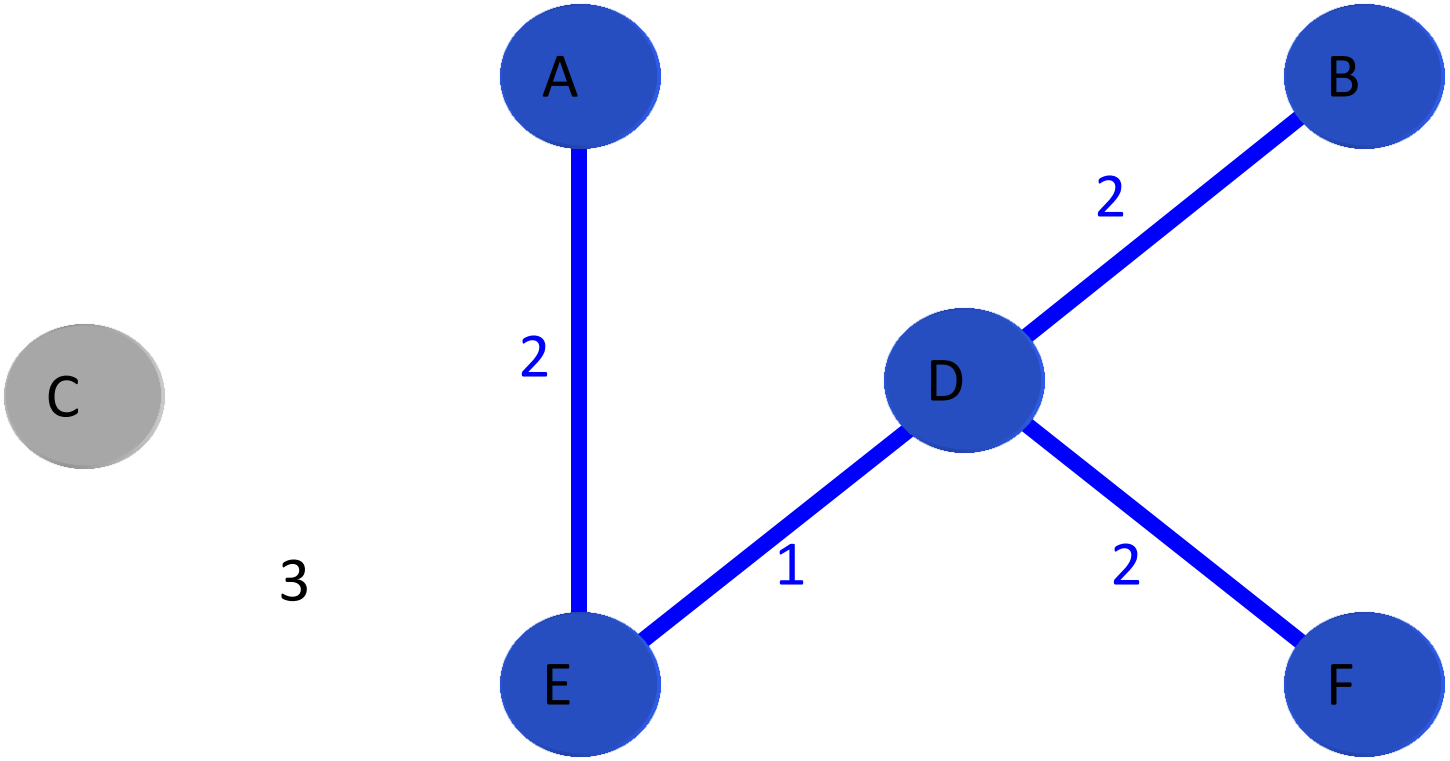

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

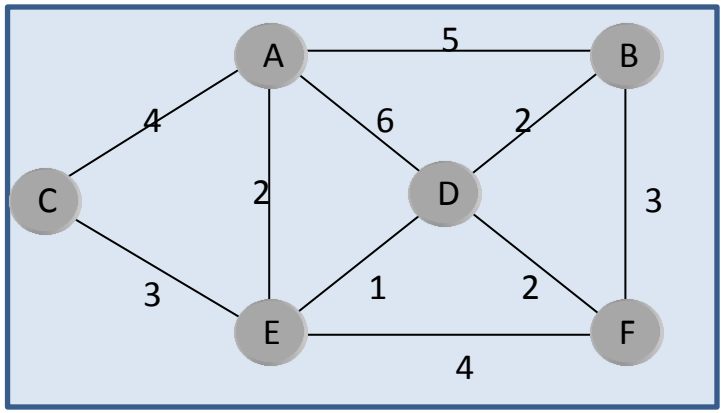

A
B
C
D
E
F
2
2
3
1
2

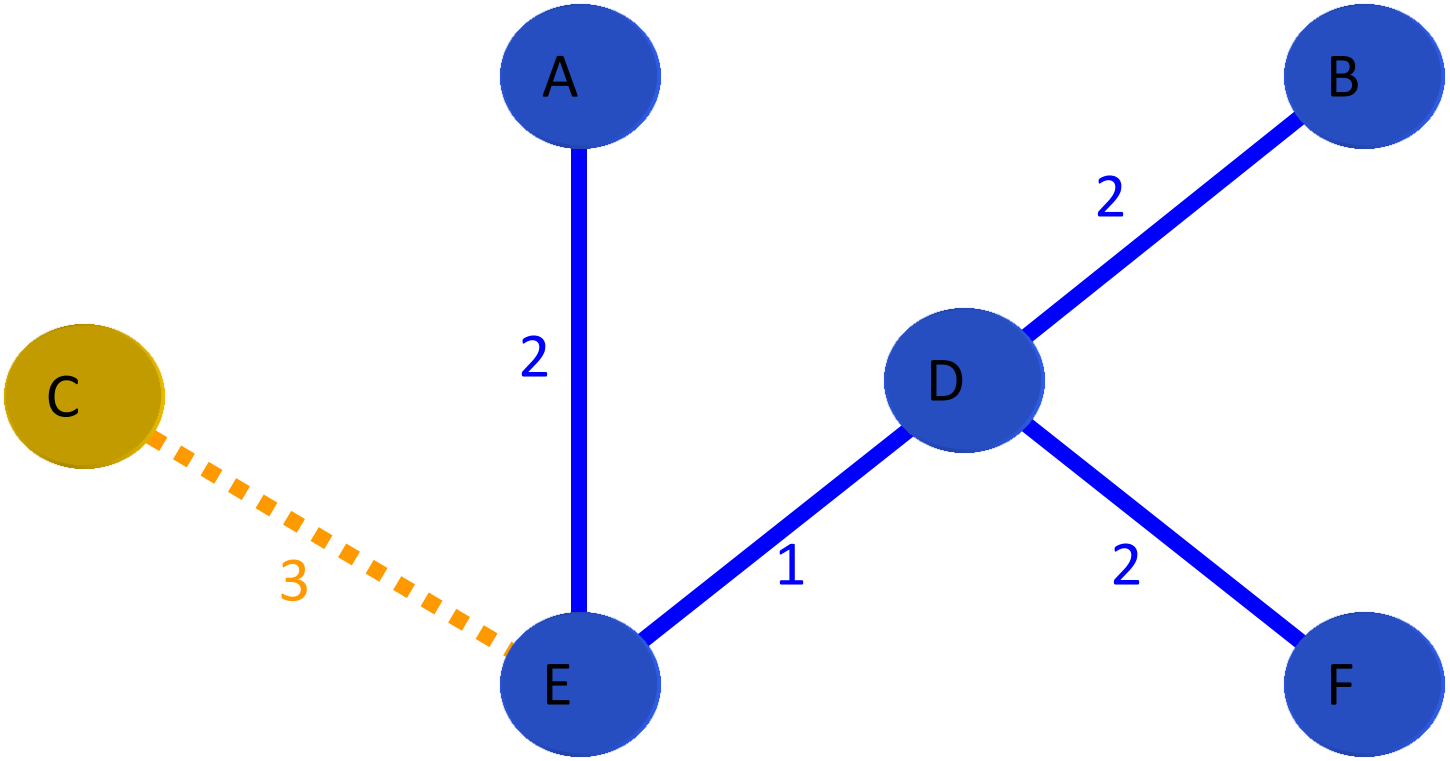

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
2
2
3
1
2

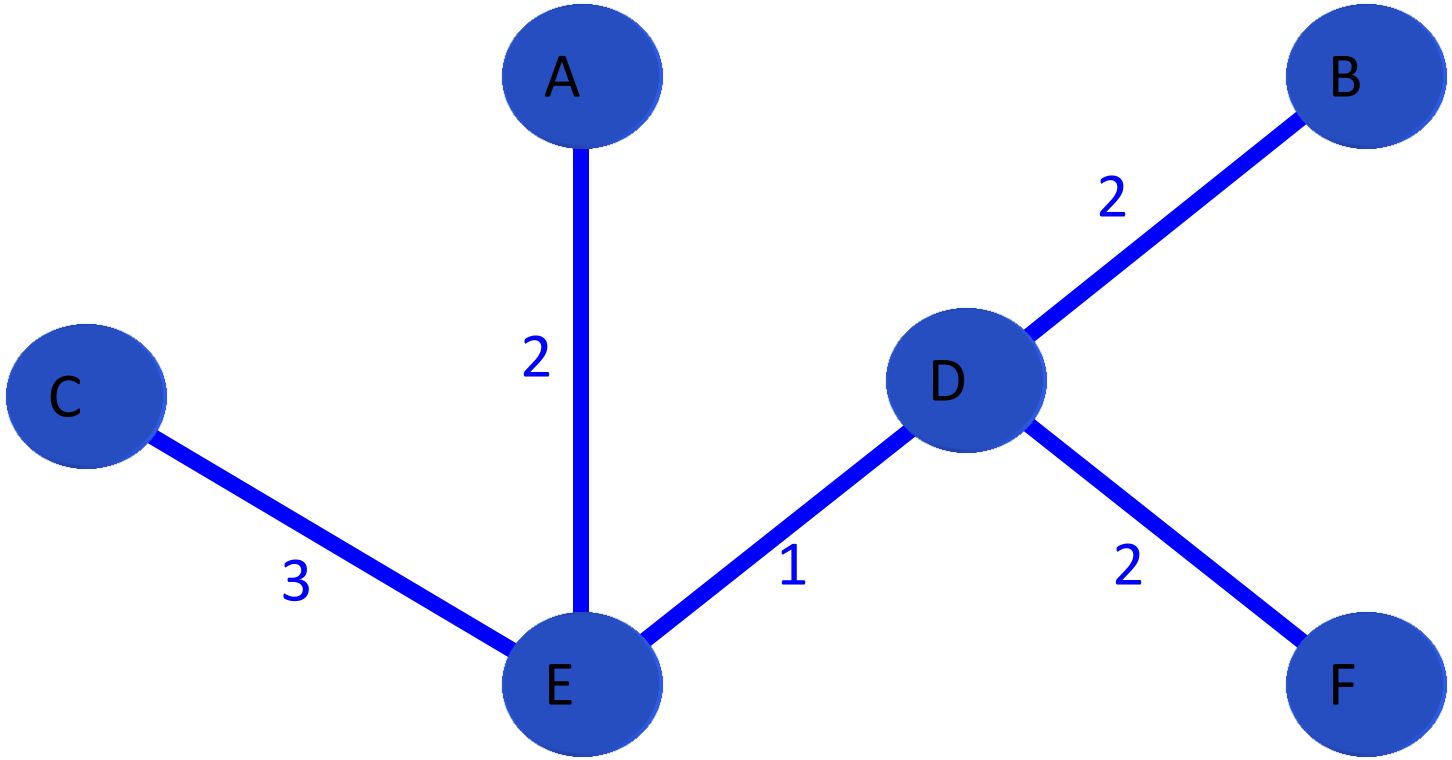

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
2
2
1
2
3


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
2
2
3
1
2

minimum- spanning tree

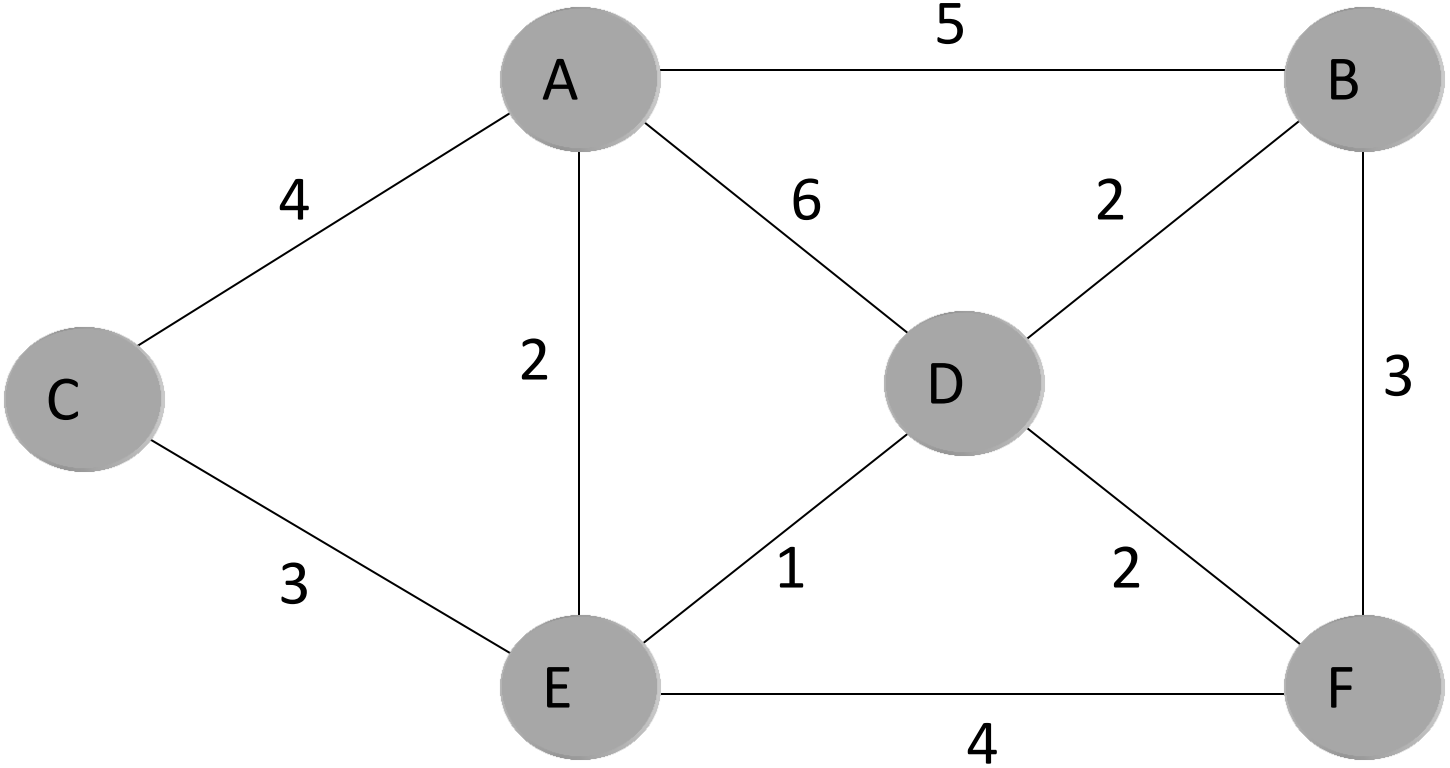


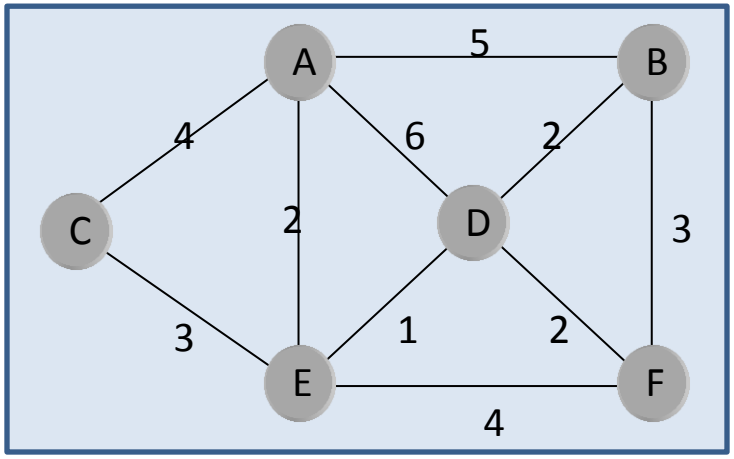
Kruskal's Algorithm

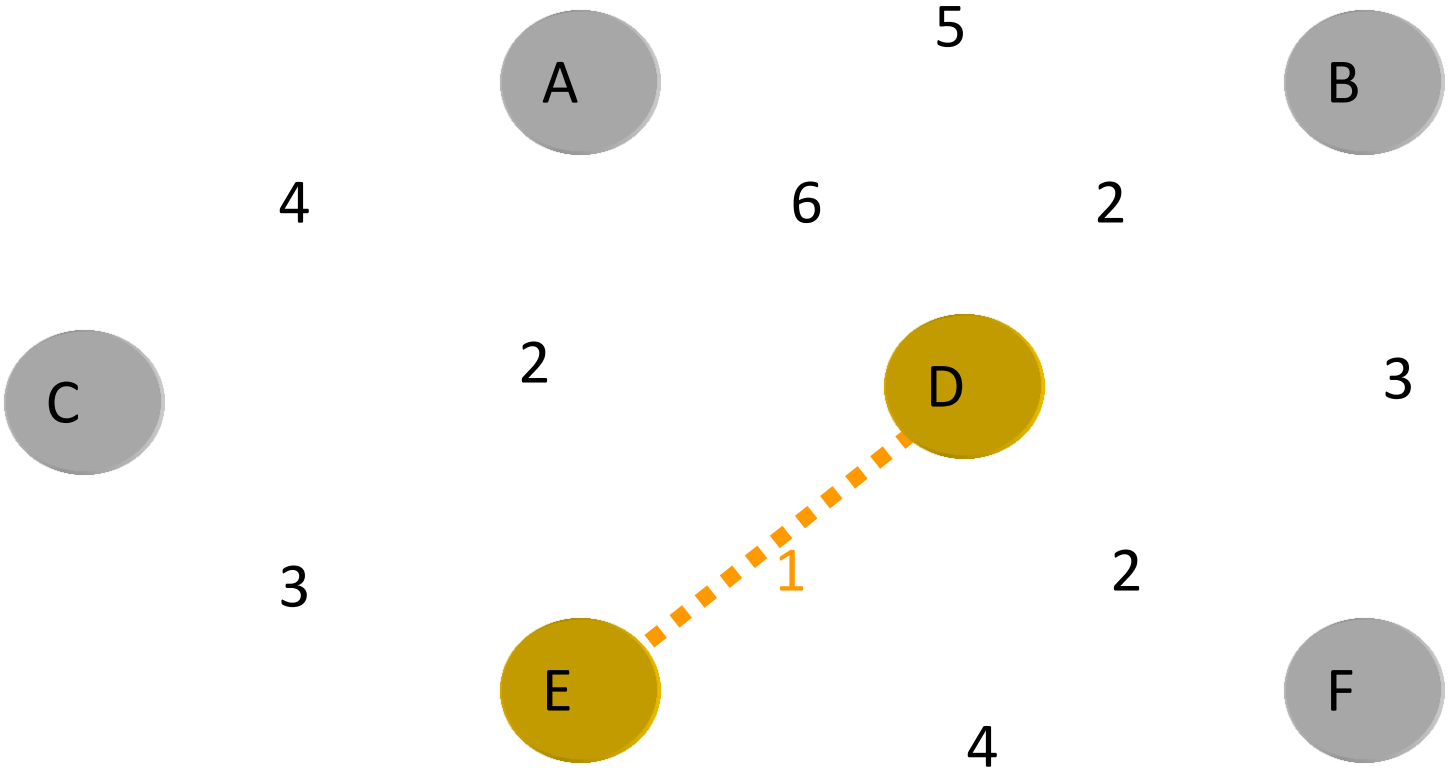
1. Each vertex is in its own cluster

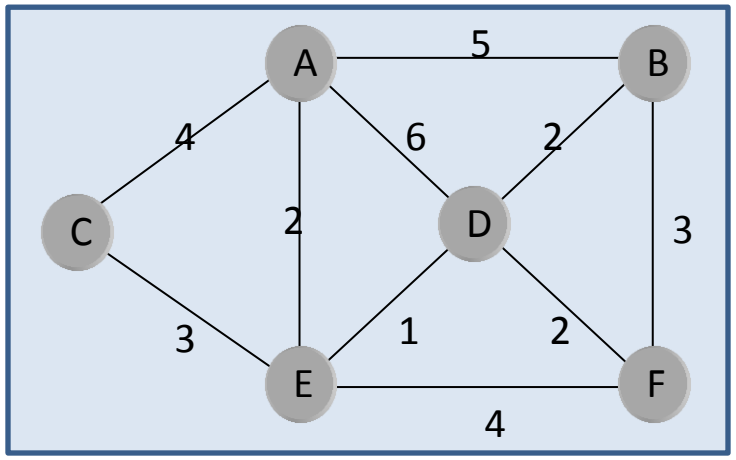
2. Take the edge *e* with the smallest weight
   - if *e* connects two vertices in different clusters, then *e* is added to the MST and the two clusters, which are connected by *e*, are merged into a single cluster
   - if *e* connects two vertices, which are already in the same cluster, ignore it

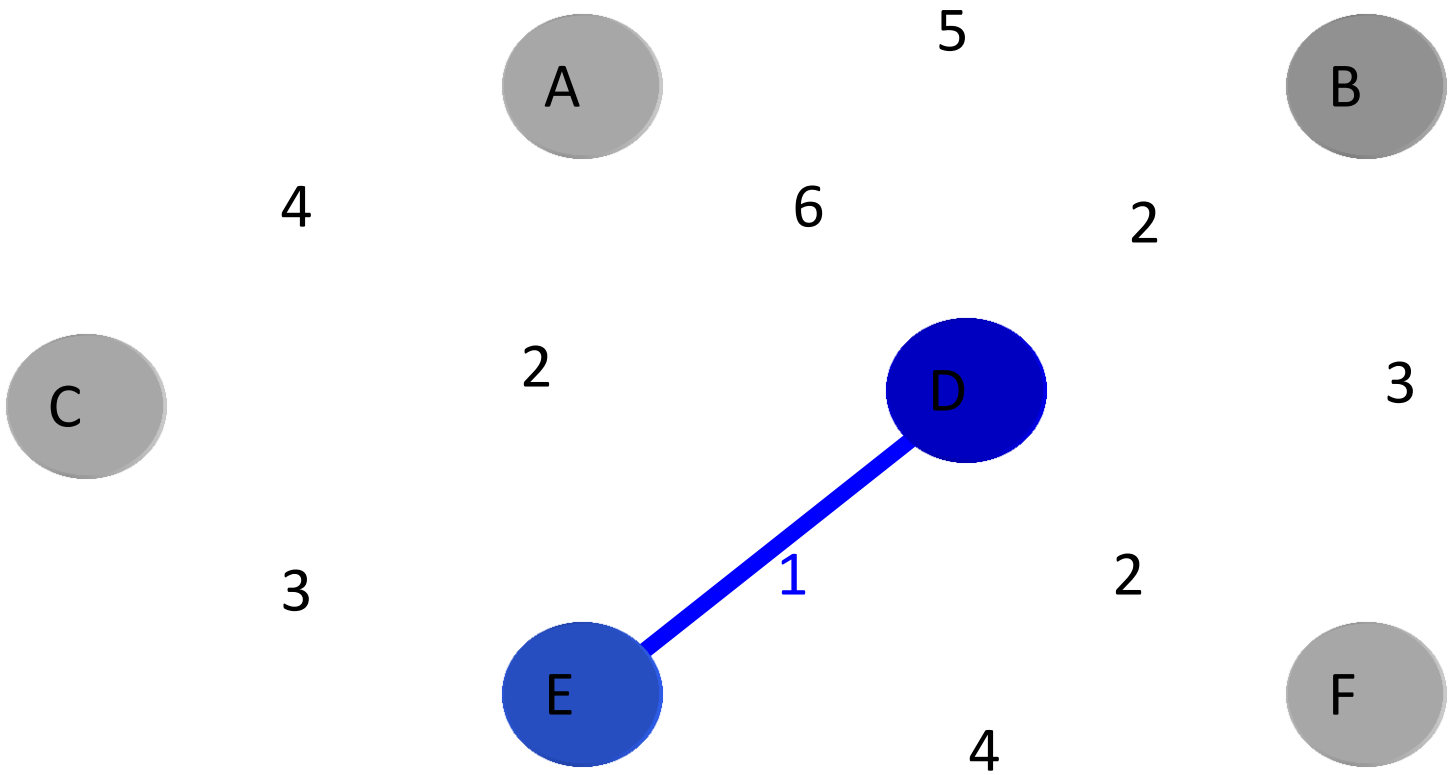
3. Continue until *n-1* edges were selected

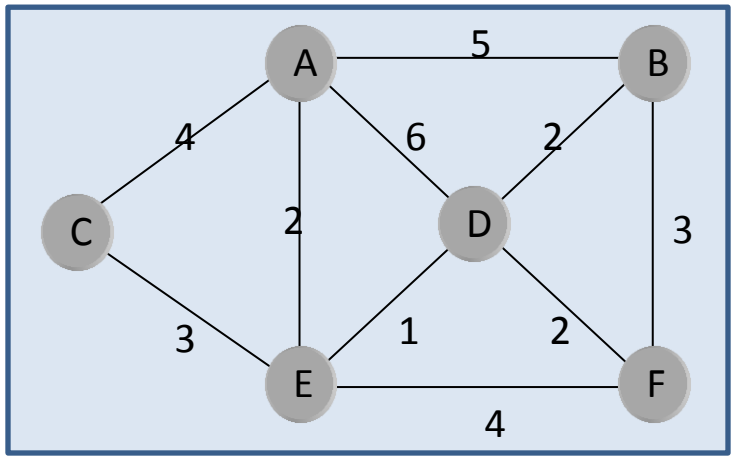

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

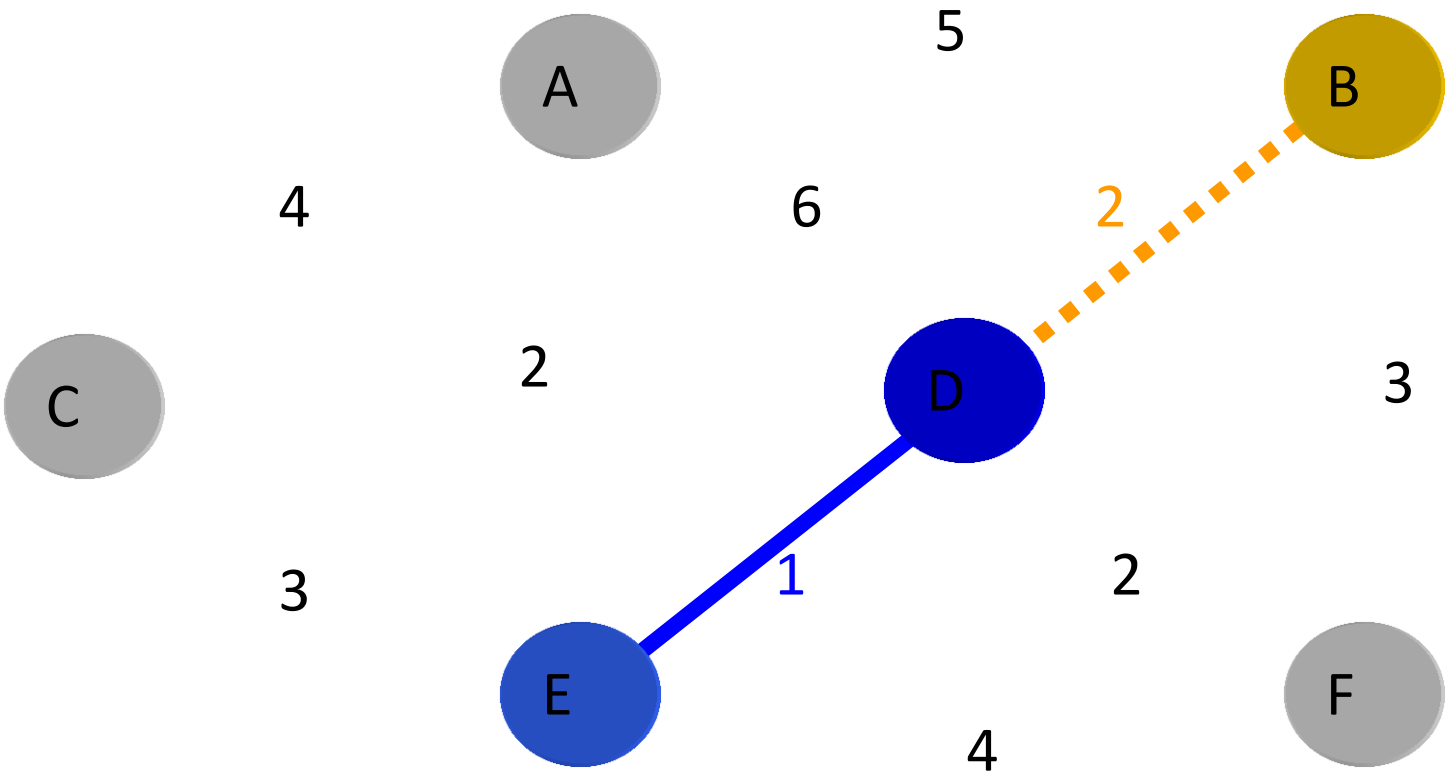

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

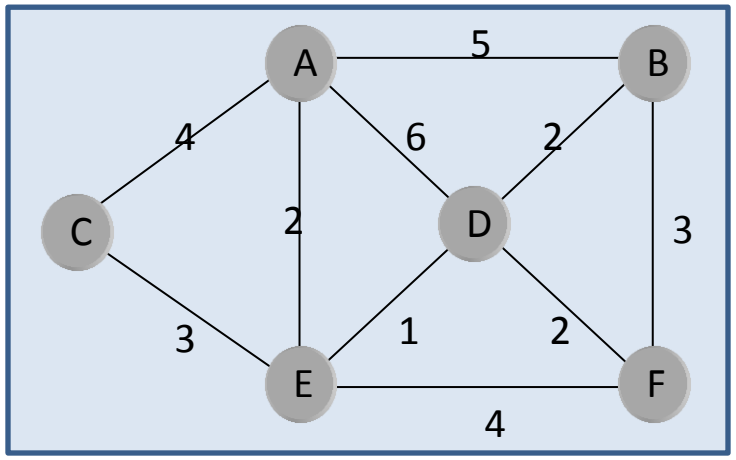

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

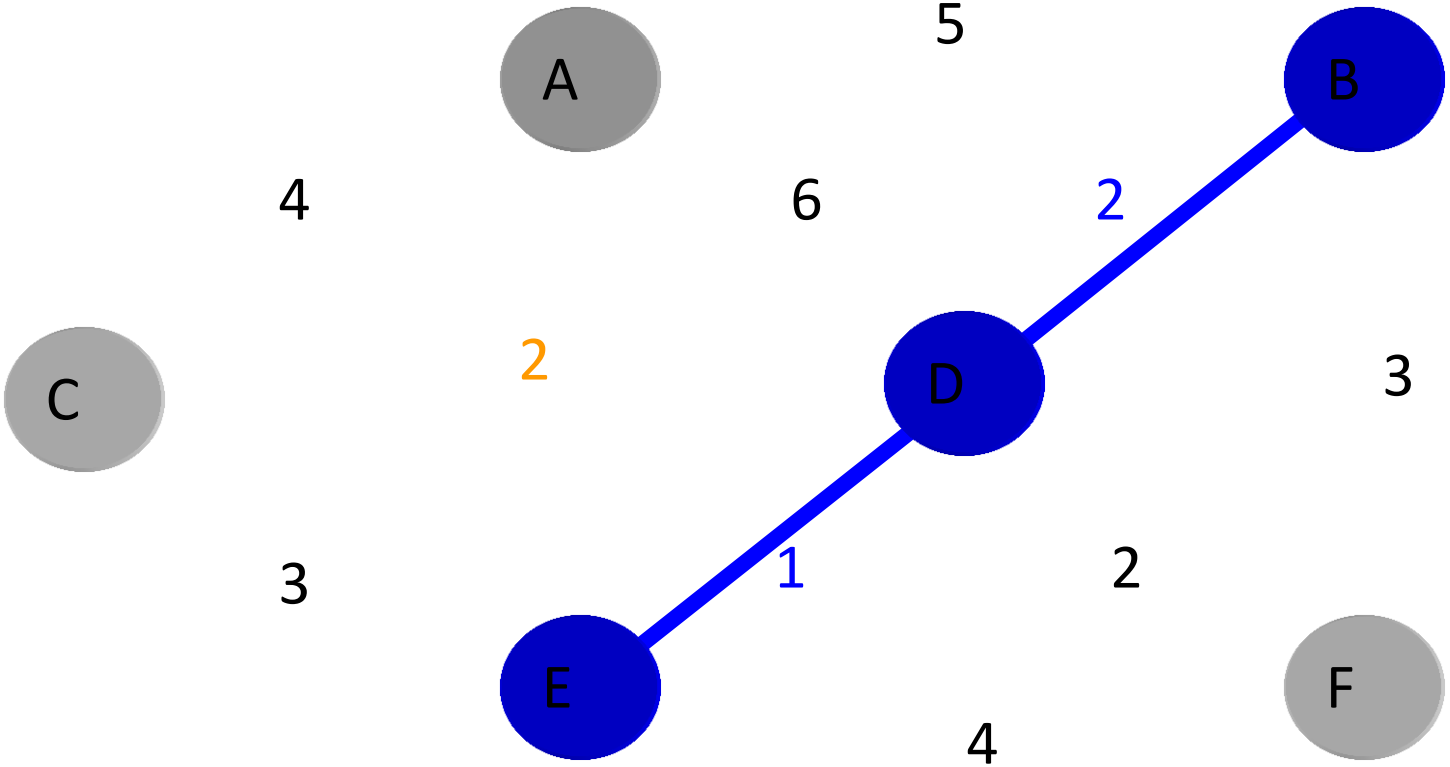

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

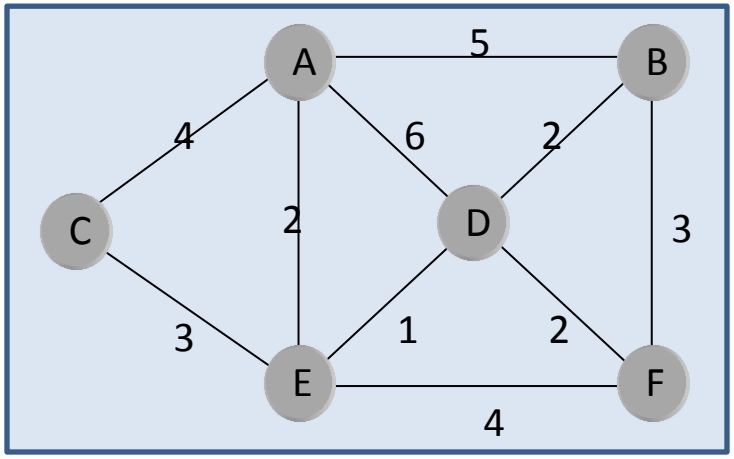

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

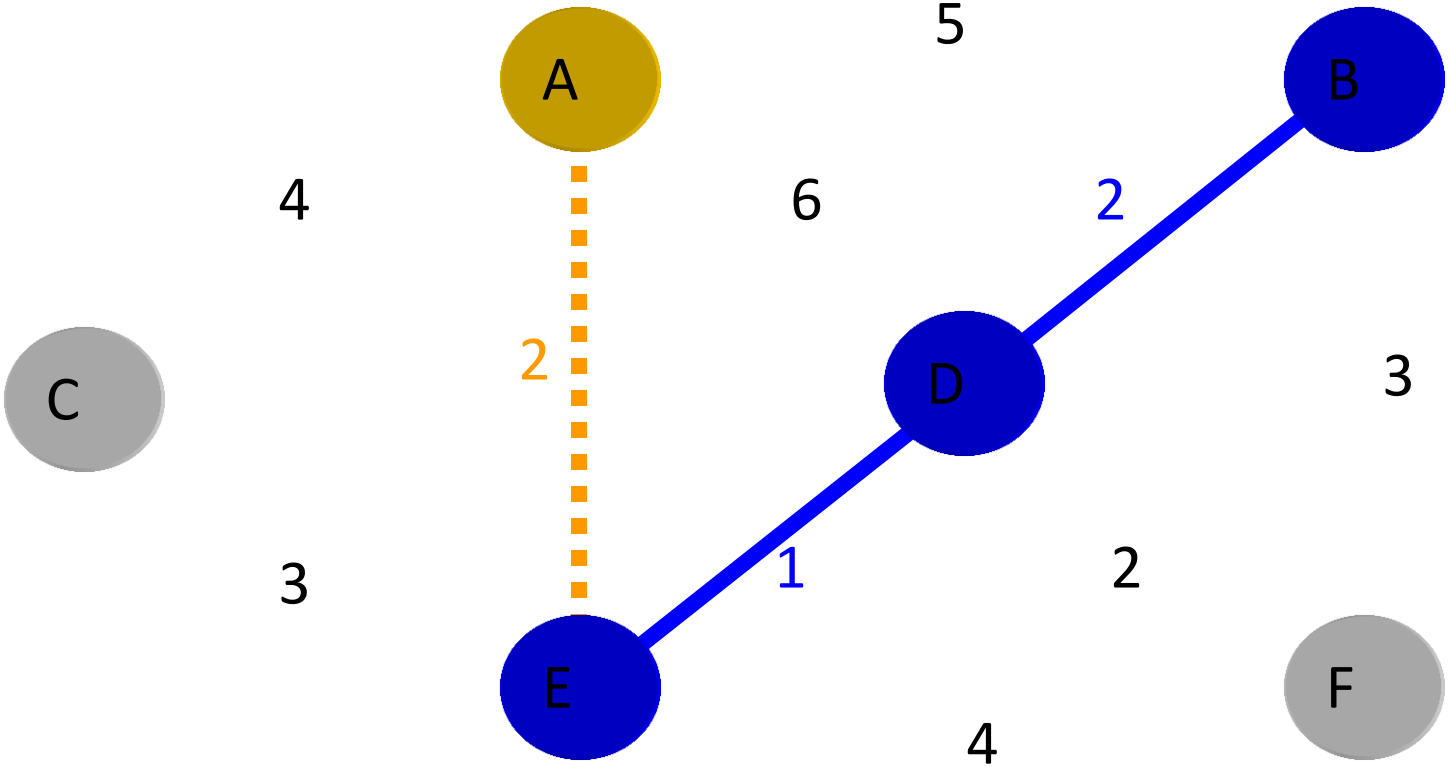

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

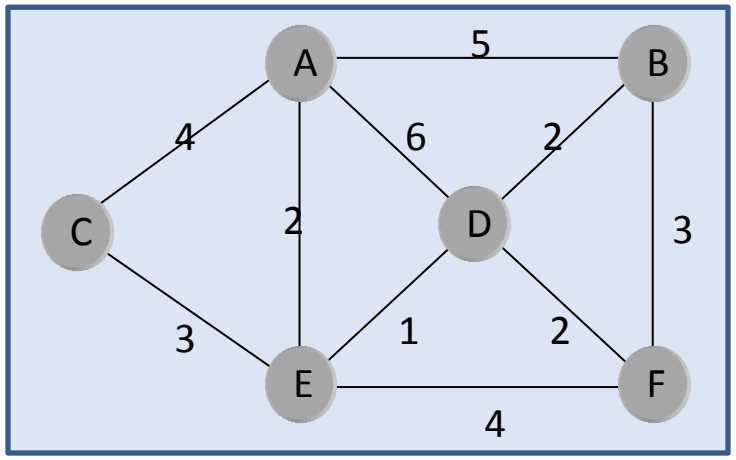

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

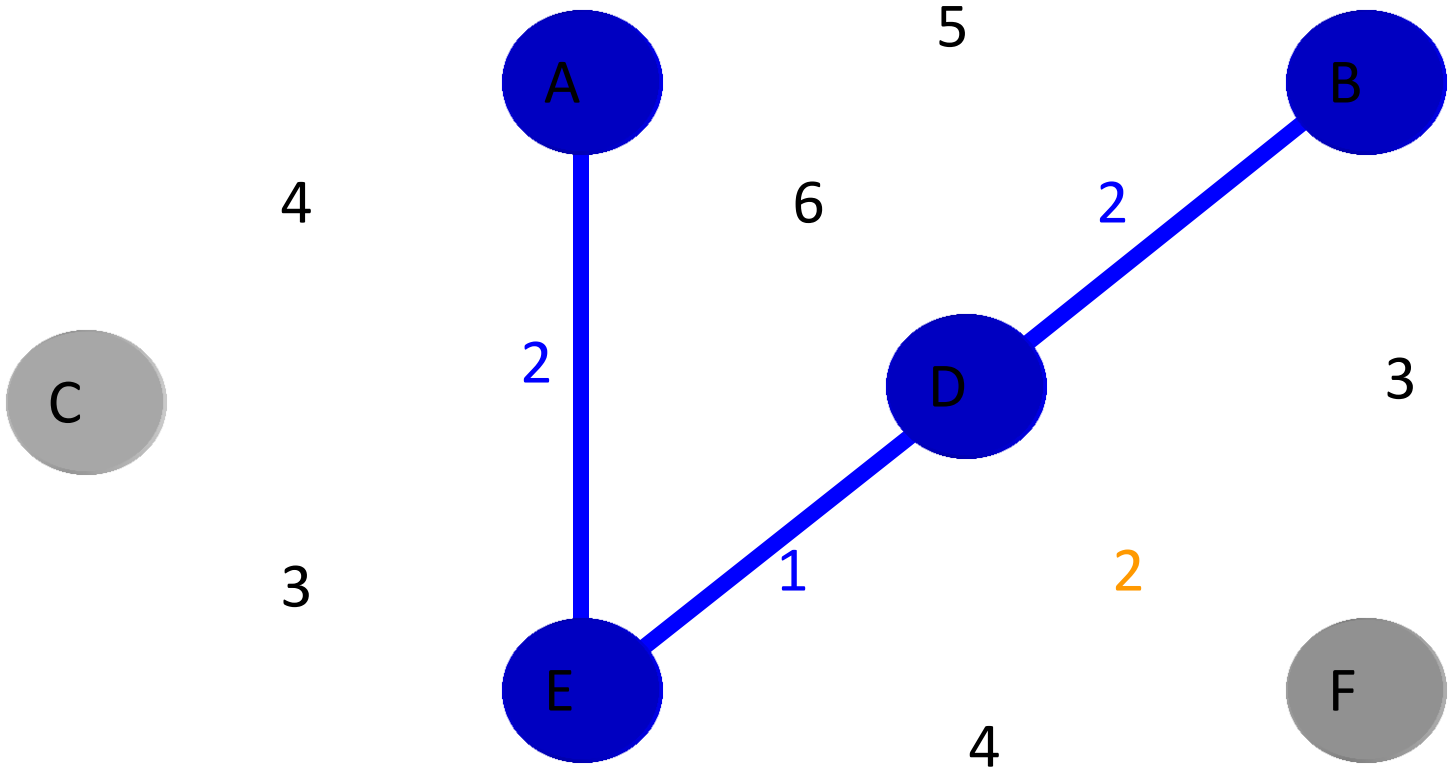

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

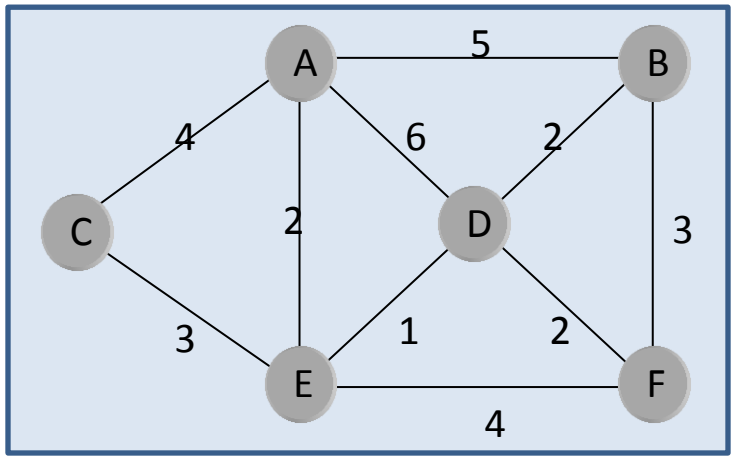

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

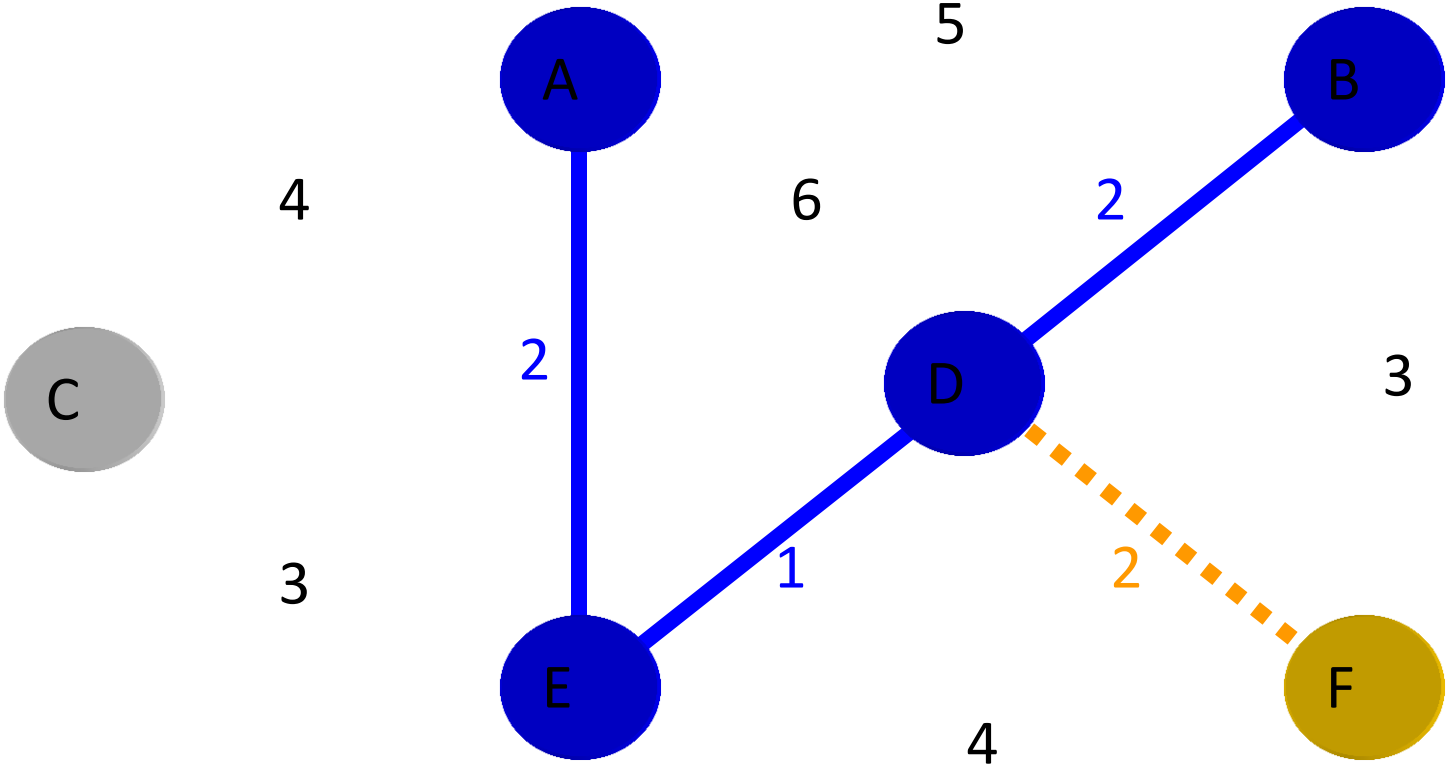

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

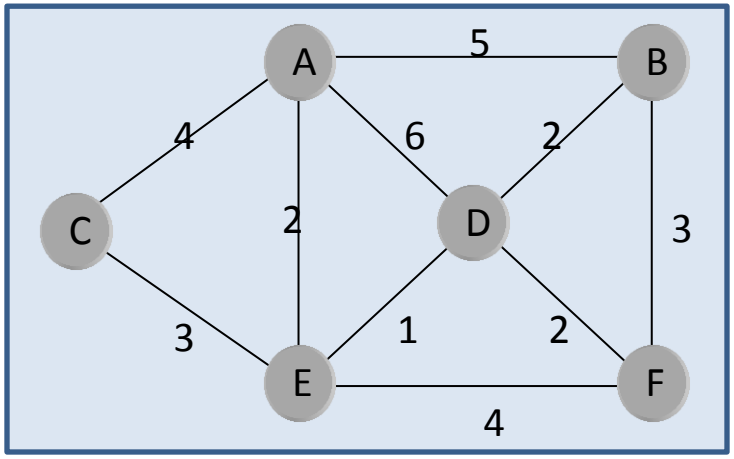

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

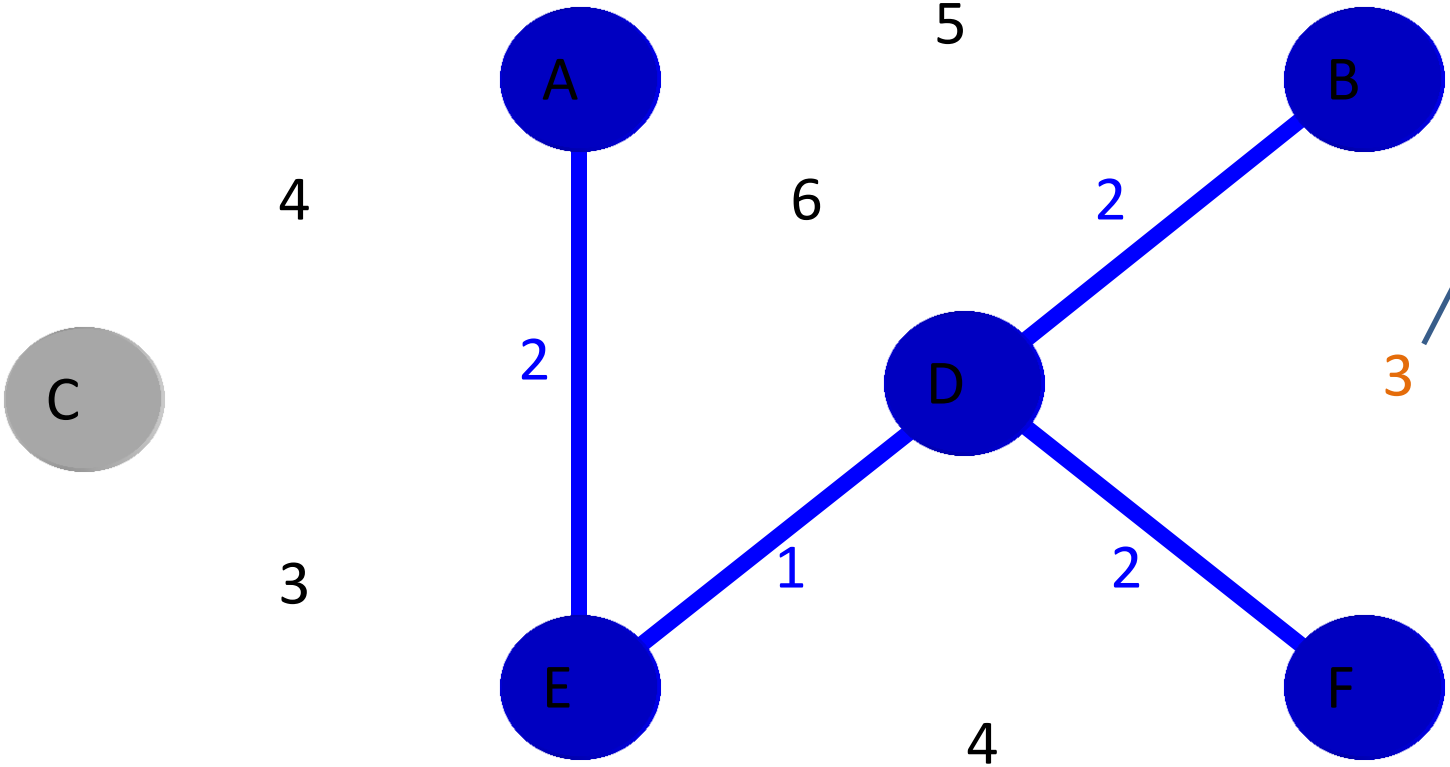

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

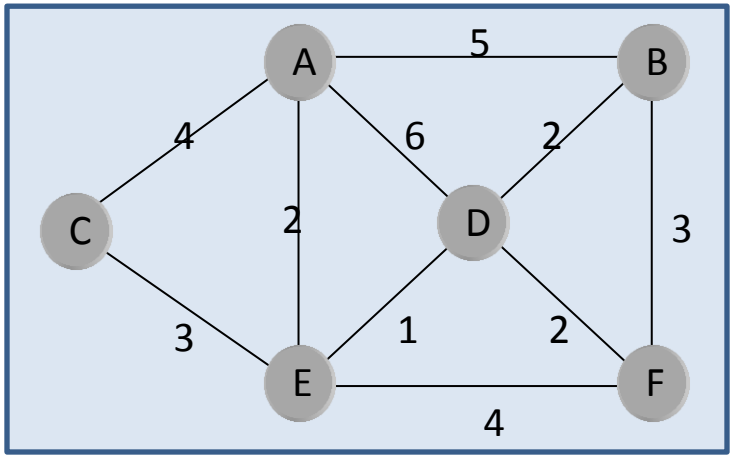

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

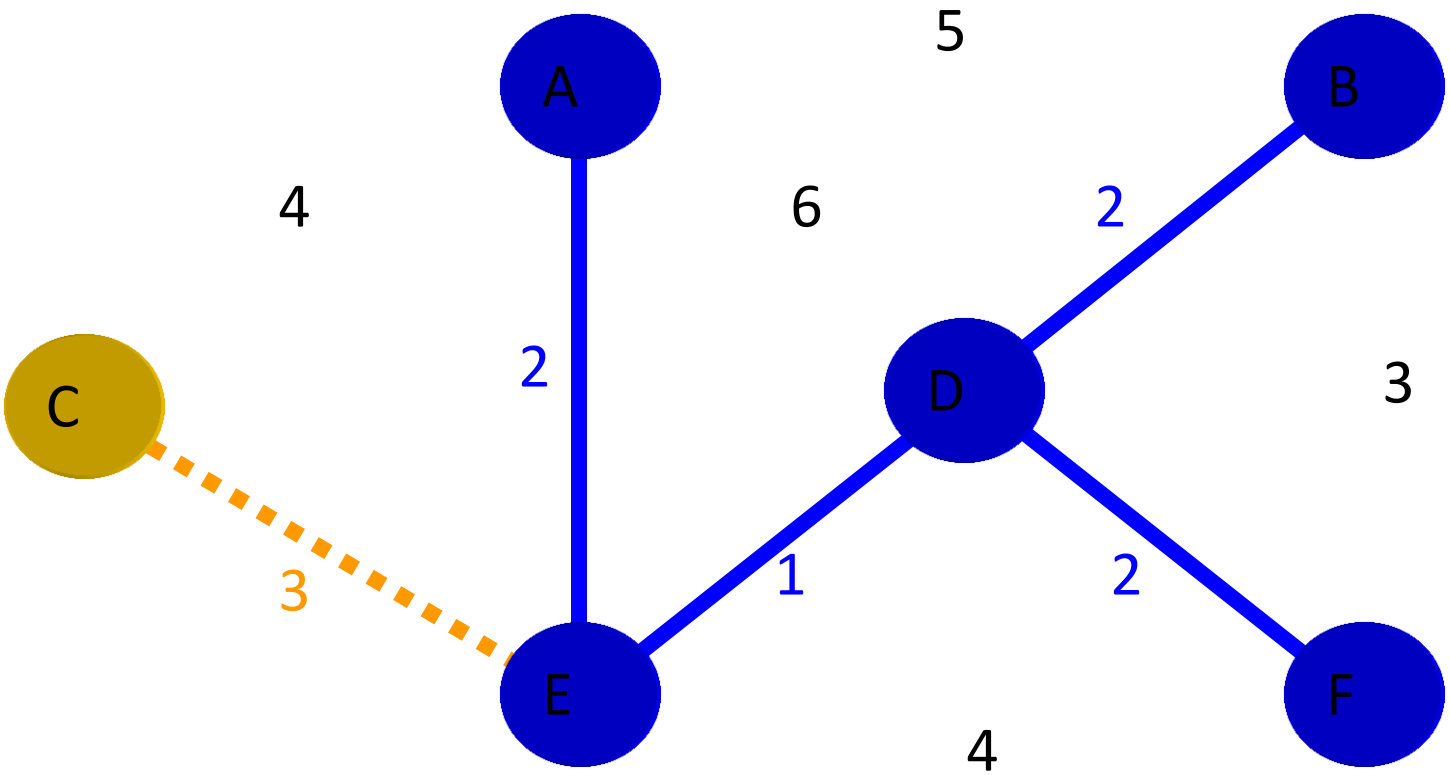

A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4



Can’t be chosen next
(B & F are in same cluster)




A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

minimum- spanning tree