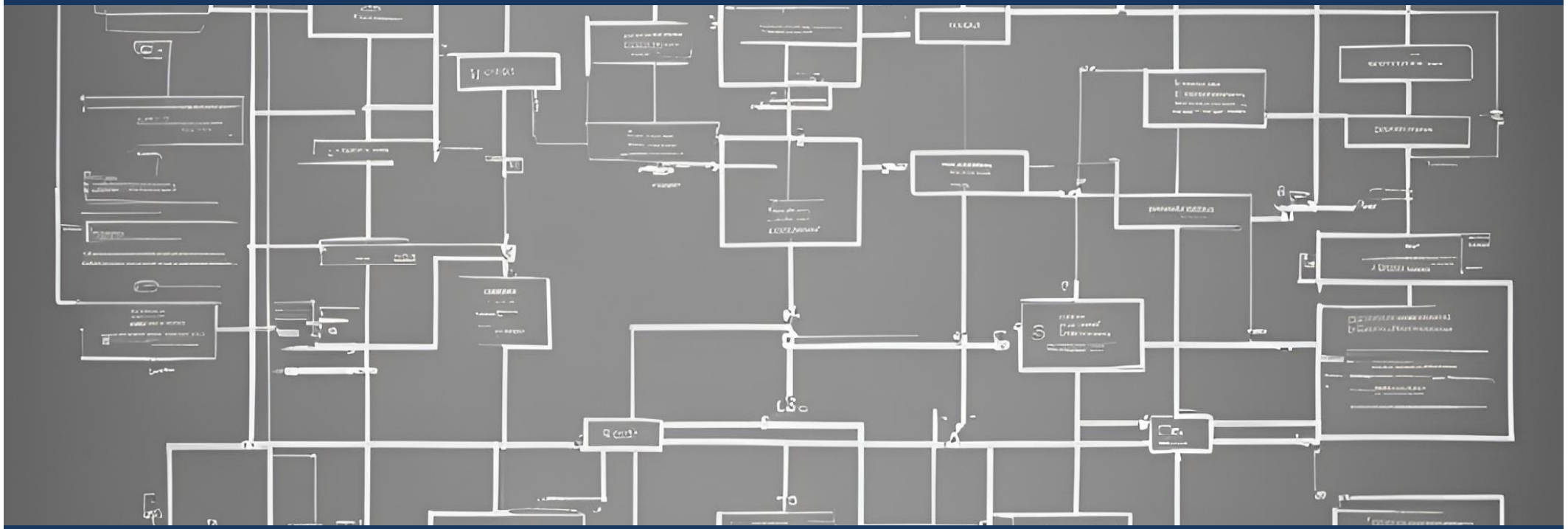
ITM 517 Algorithm

Ja-Hee Kim

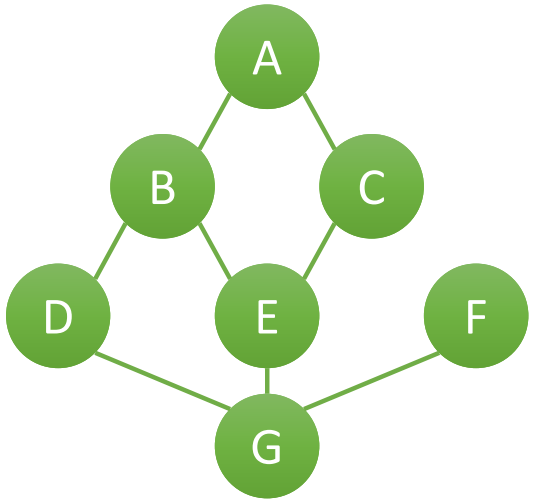
Graph

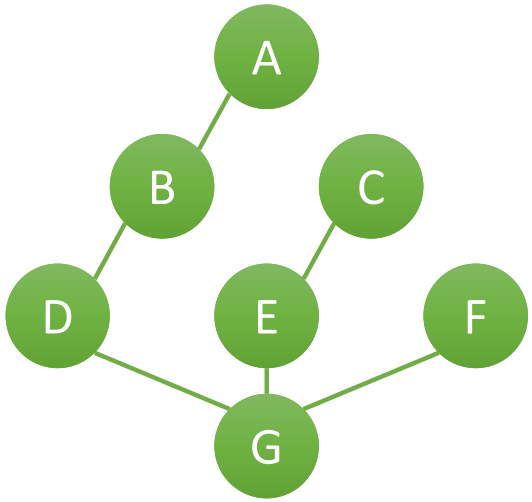


Spanning Tree

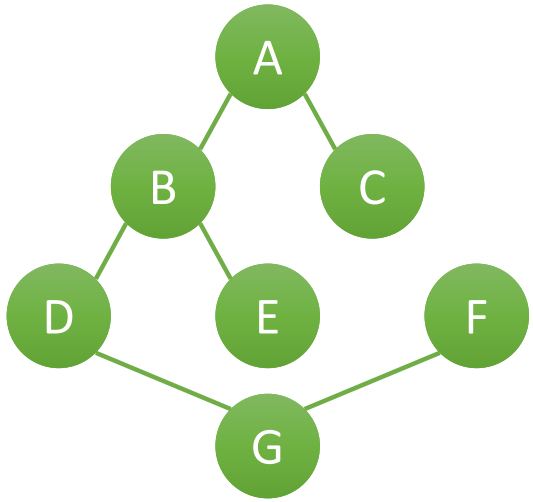
Spanning Tree

- Spanning tree T of an undirected graph G
  - a **subgraph** which includes all of the vertices of G, with **minimum possible number of edges**.
  - In general, a graph may have **several spanning trees**, but a graph that is not connected will not contain a spanning tree (but Spanning forests).





DF Spanning Tree



BF Spanning Tree

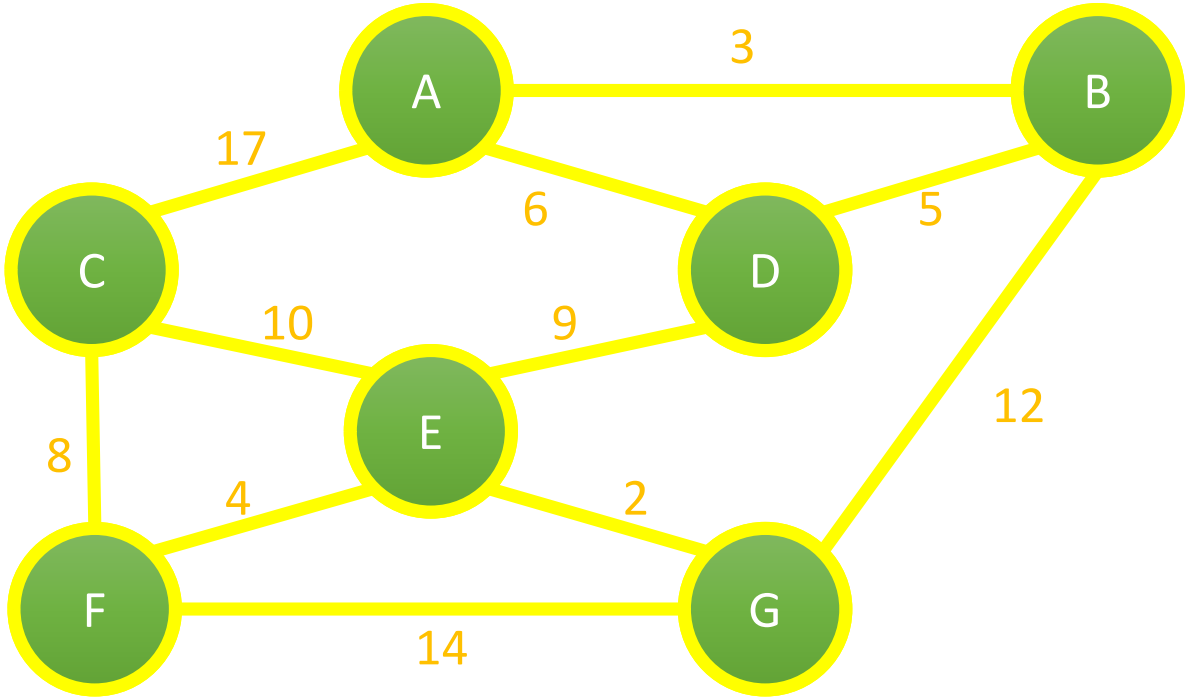
Minimum Spanning Tree

- aka
  - **MST**
  - **minimum weight spanning tree**
  - Minimum cost spanning tree
- A spanning tree with the minimum possible total edge weight.

- Algorithm
  - Kruskal's algorithm
  - Prim's algorithm

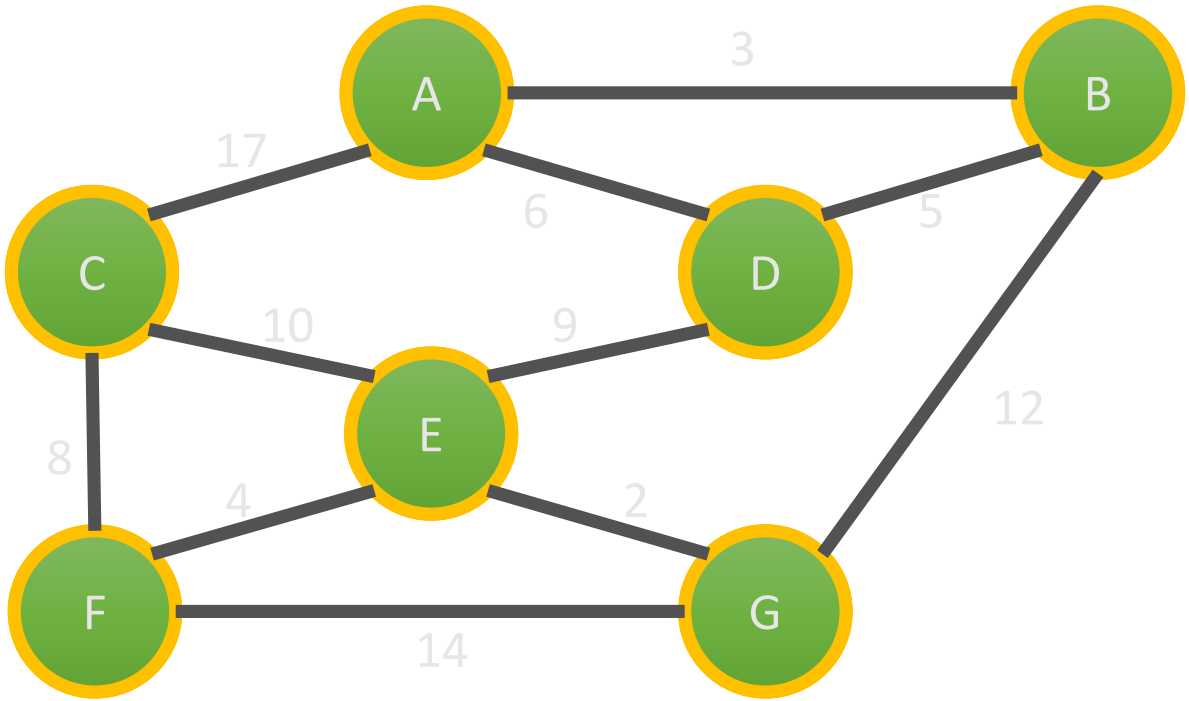
Kruskal's algorithm

```
KRUSKAL(G):
1. A = { }
2. Sort edges (u, v)ordered by weight(u, v), increasing:
3. foreach (u, v)
4.      if A = A U {(u, v)} does not make a cycle
5.              UNION(u, v)
6.      if |A| = n-1
7.              break;
8. return A
```



Kruskal's algorithm

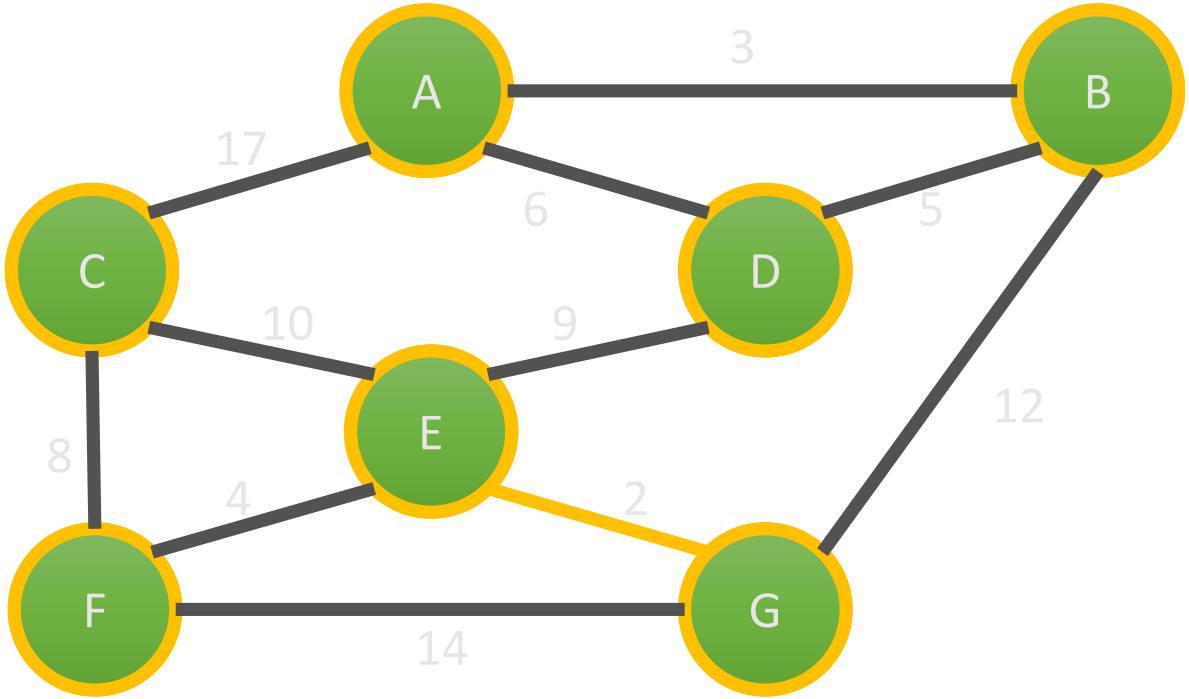
```
A = { }
Sort edges (u, v)ordered by weight(u, v), increasing:
```



Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

Kruskal's algorithm

```
foreach (u, v)
        if A = A ∪ {(u, v)} does not make a cycle
                UNION(u, v)
        if |A| = n-1
                break;
```



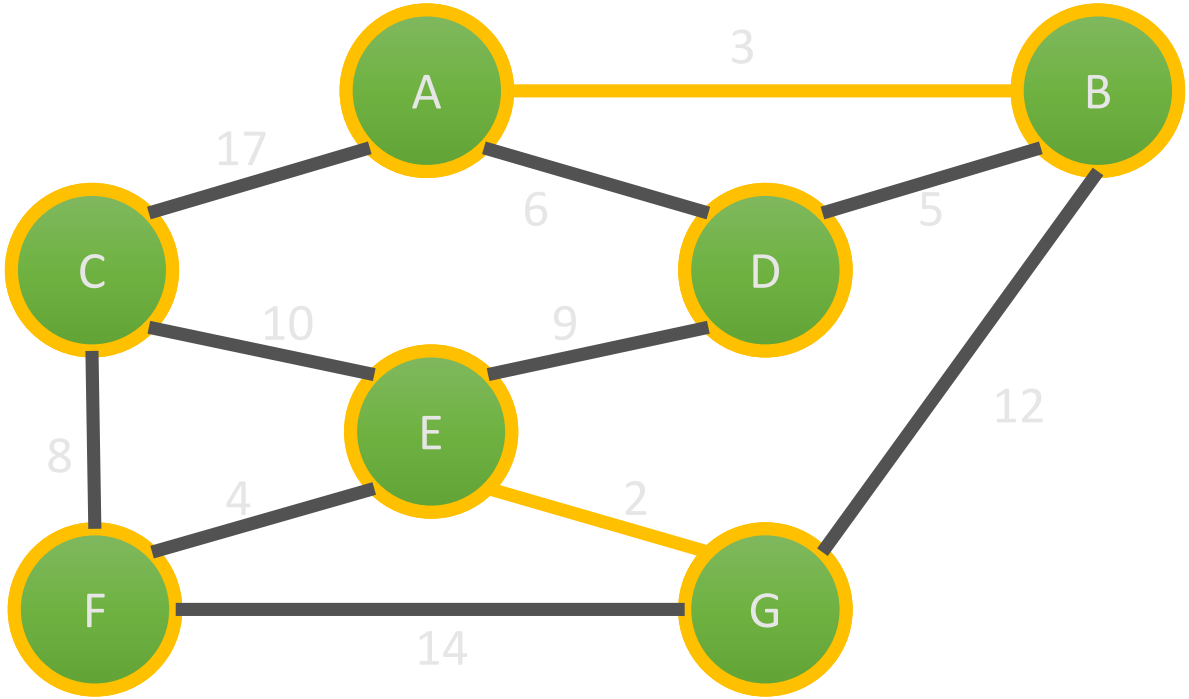
A = {(E,G)}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

2

Kruskal's algorithm

```
foreach (u, v)
        if A = A ∪ {(u, v)} does not make a cycle
                UNION(u, v)
        if |A| = n-1
                break;
```

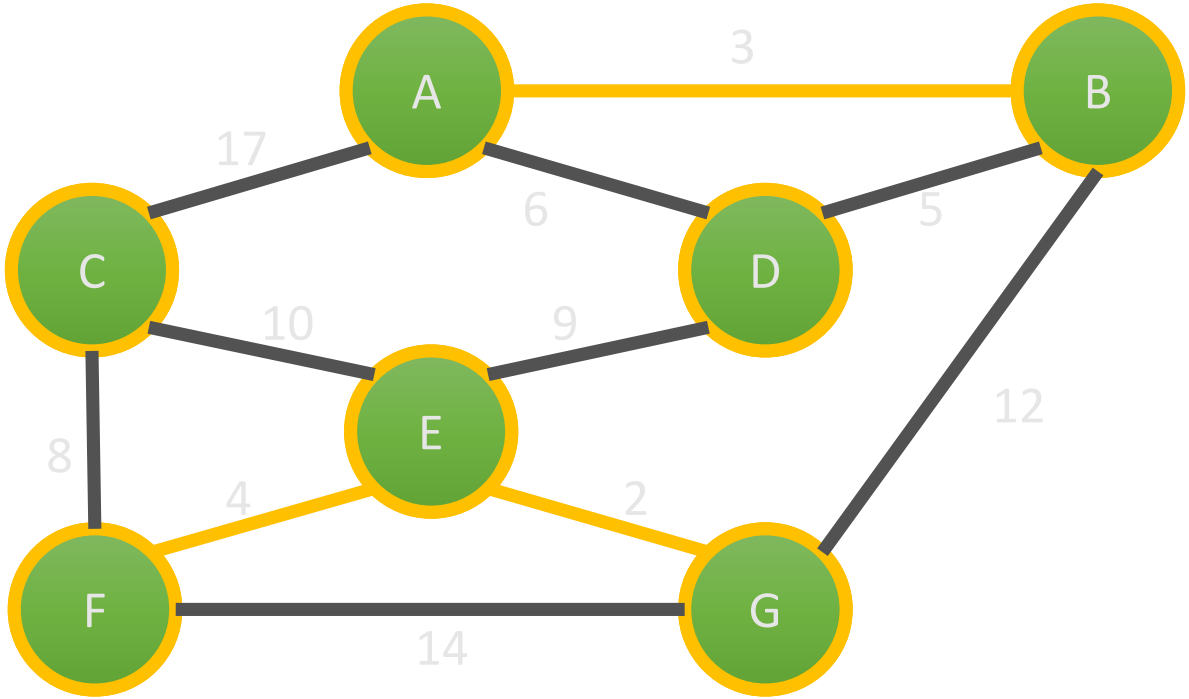


A = {(E,G), {A,B}}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

5

Kruskal's algorithm

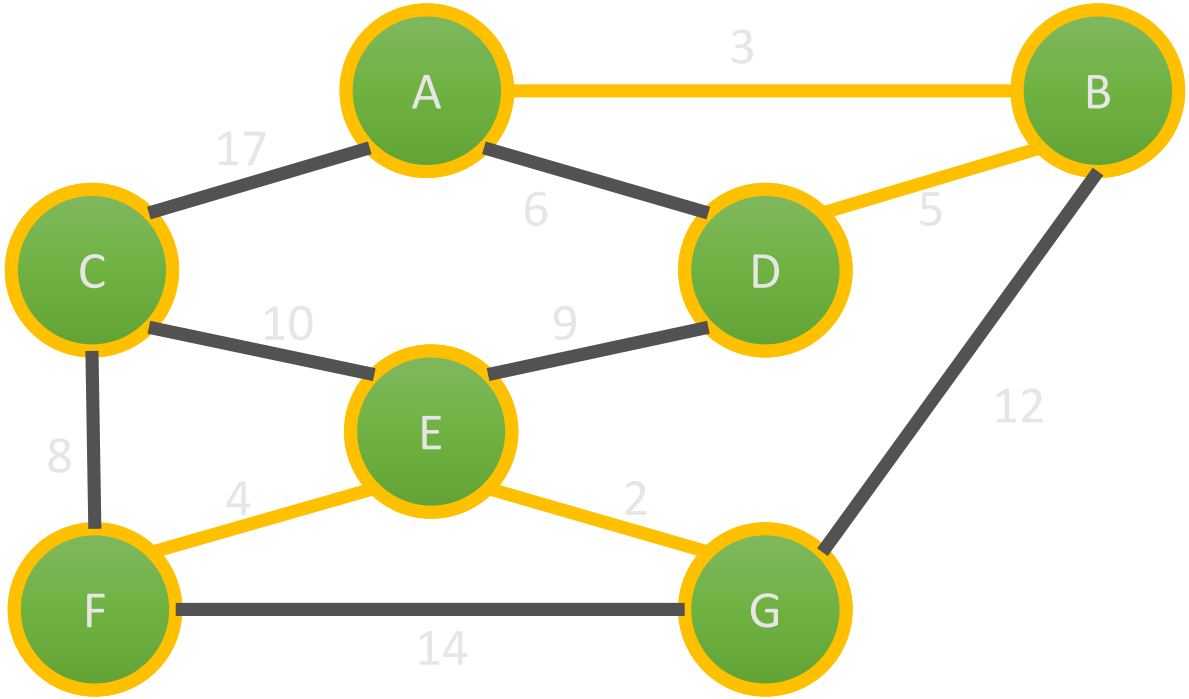


A = {(E,G), {A,B}, {E,F}}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

9

Kruskal's algorithm

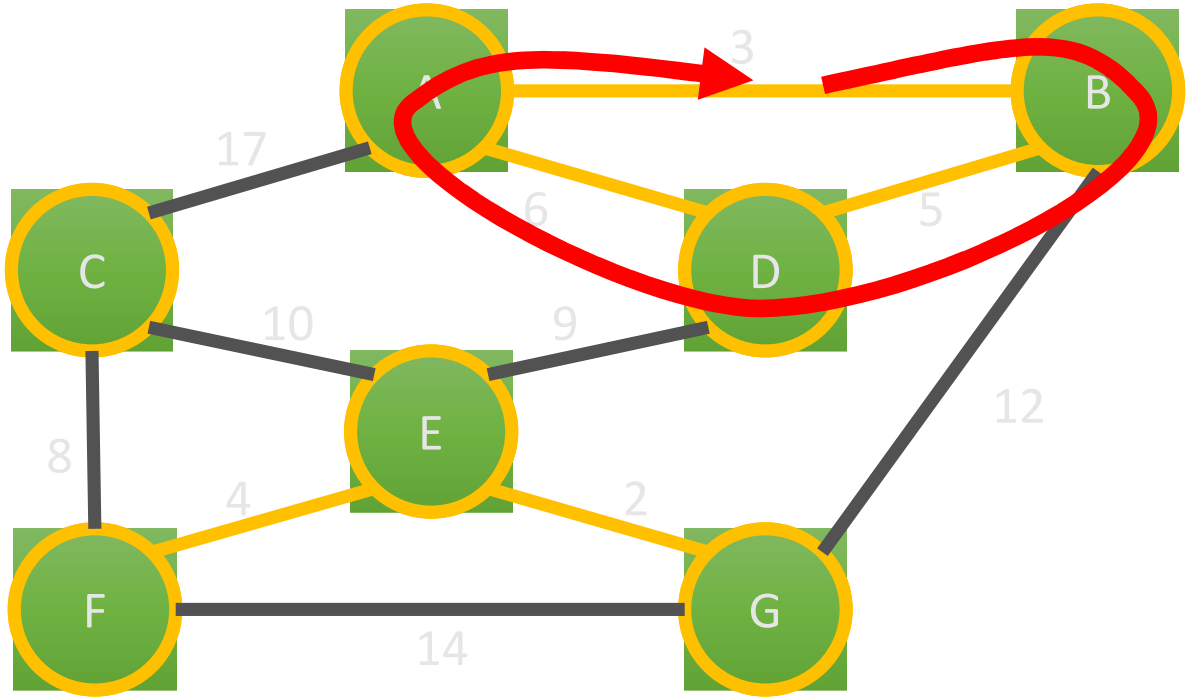


A = {(E,G), {A,B}, {E,F}, {B,D}}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

14

Kruskal's algorithm

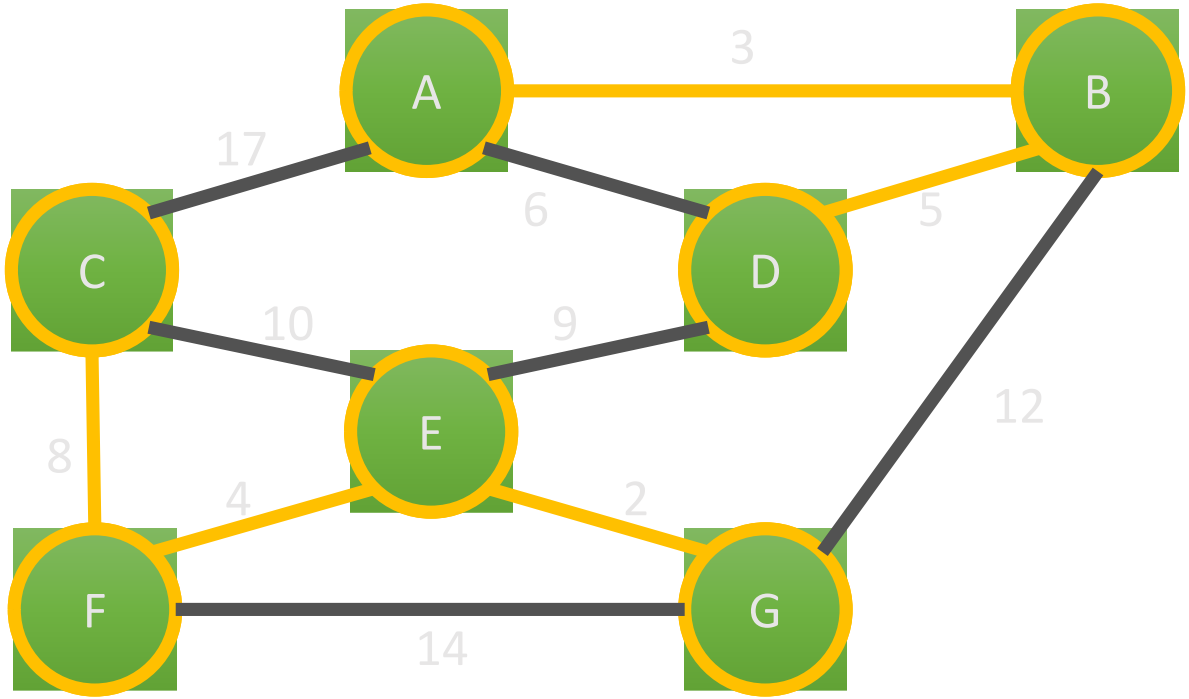


A = {(E,G), {A,B}, {E,F}, {B,D}}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	{A,D}
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

14

Kruskal's algorithm

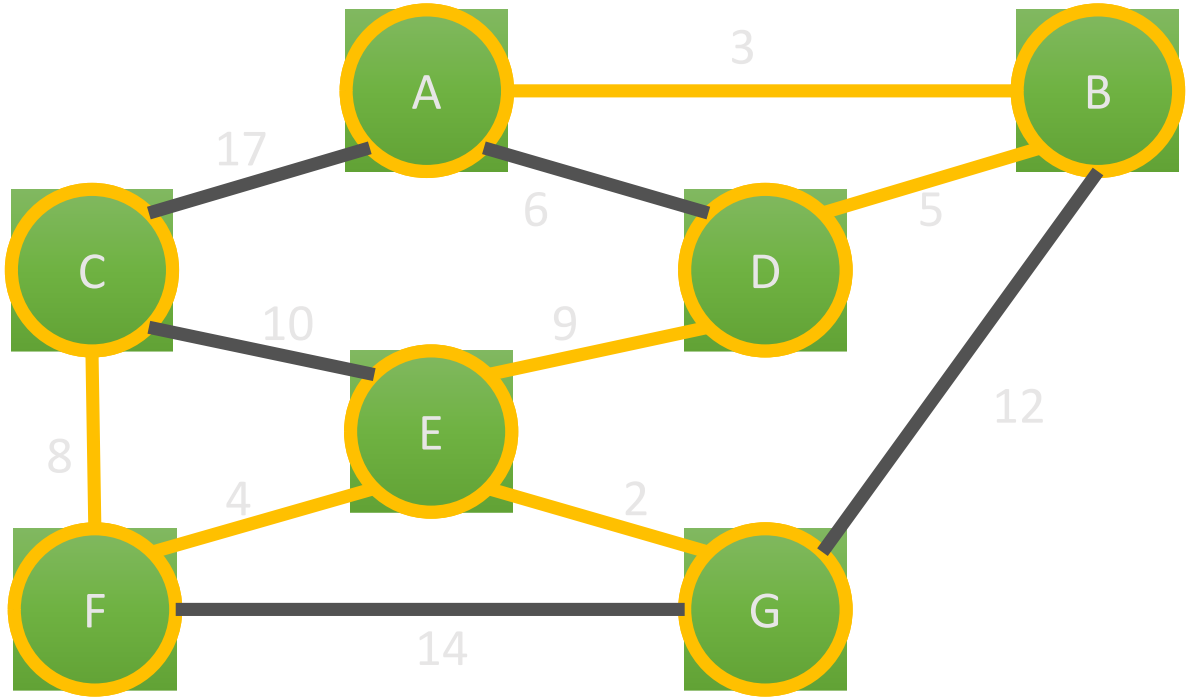


A = {(E,G), {A,B}, {E,F}, {B,D}, {C,F}}

Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	~~{A,D}~~
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

22

Kruskal's alorithm



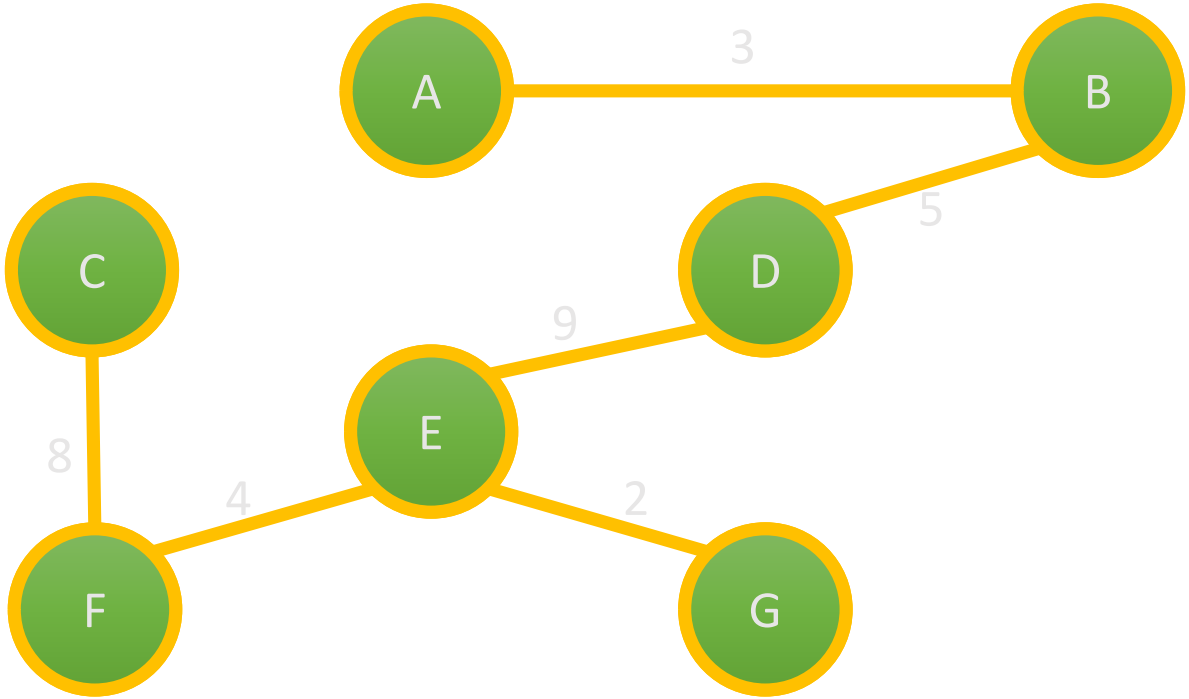
Weight	Edge
2	{E,G}
3	{A,B}
4	{E,F}
5	{B,D}
6	~~{A,D}~~
8	{C,F}
9	{D,E}
10	{C,E}
12	{B,G}
14	{F,G}
17	{A,C}

31

A = {(E,G), {A,B}, {E,F}, {B,D}, {C,F},{D,E}}

Kruskal's alorithm

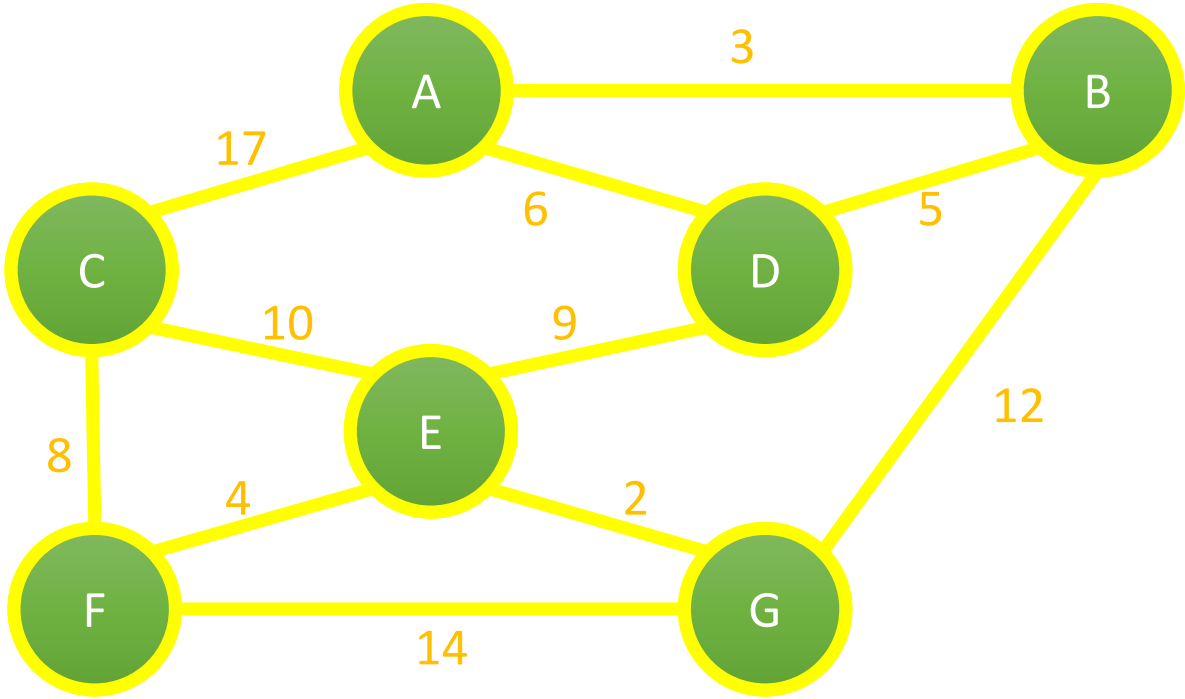
- Total weight: 31
- Return A = {(E,G), {A,B}, {E,F}, {B,D}, {C,F},{D,E}}



Prim's algorithm

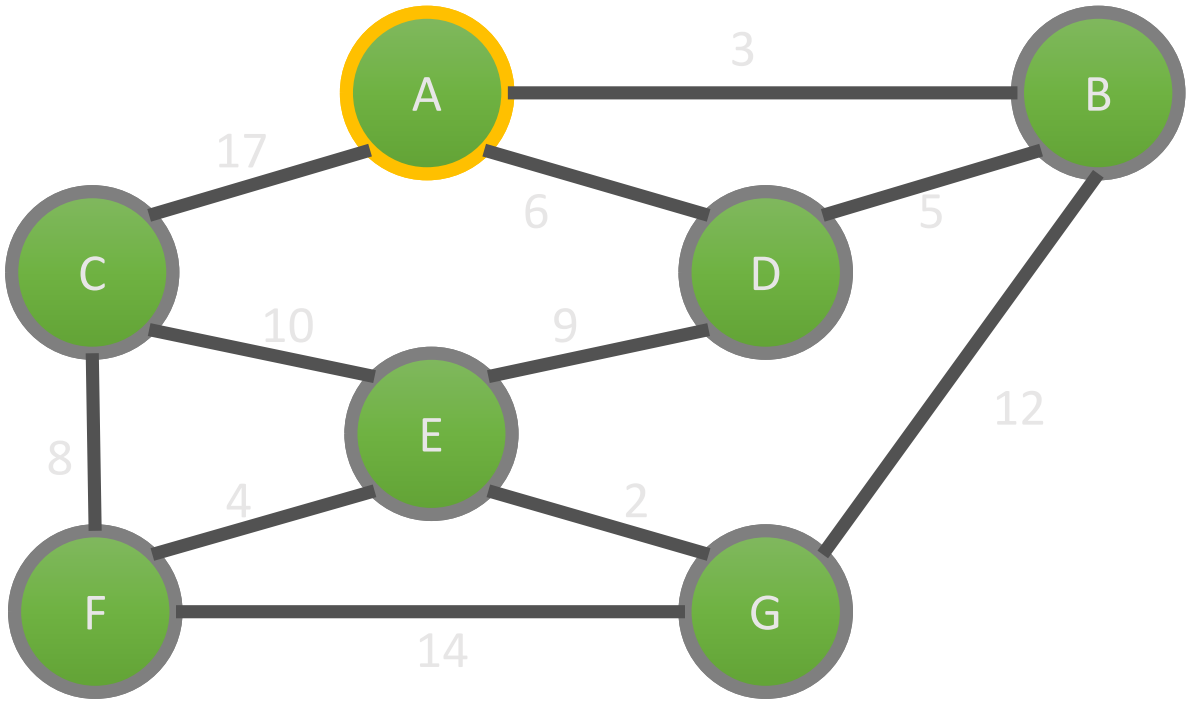
`Prim(G):`

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).



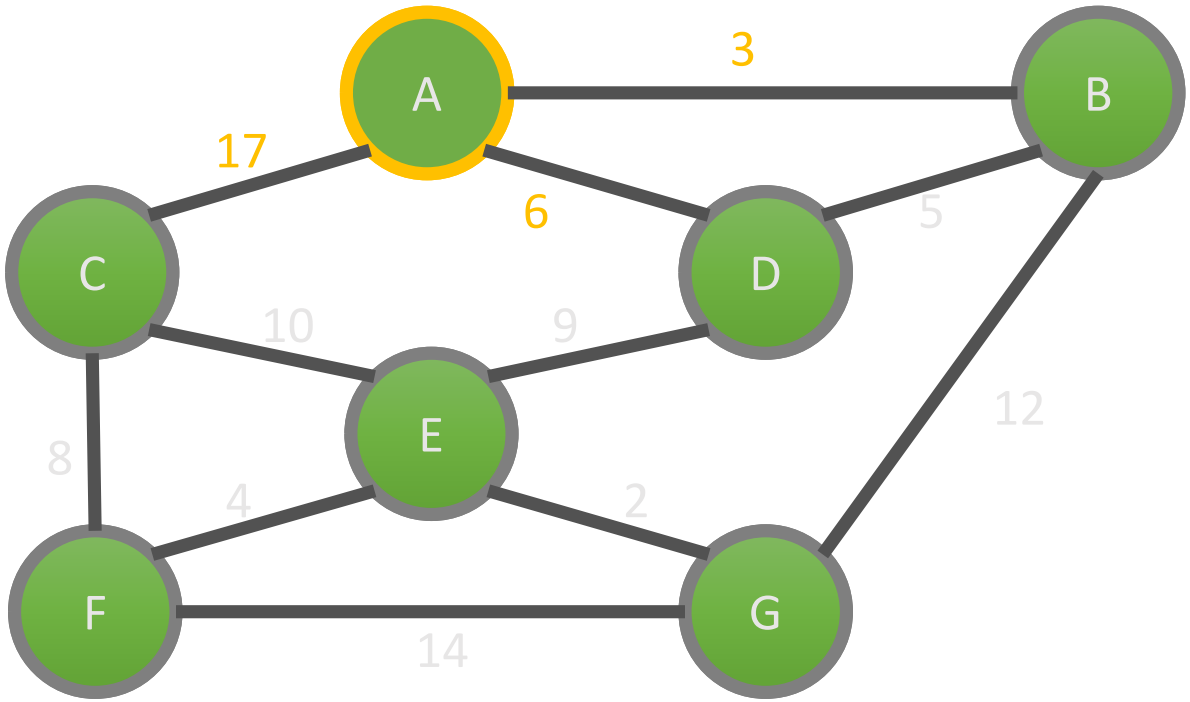
Prim's algorithm

Initialize a tree with a single vertex, chosen arbitrarily from the graph.



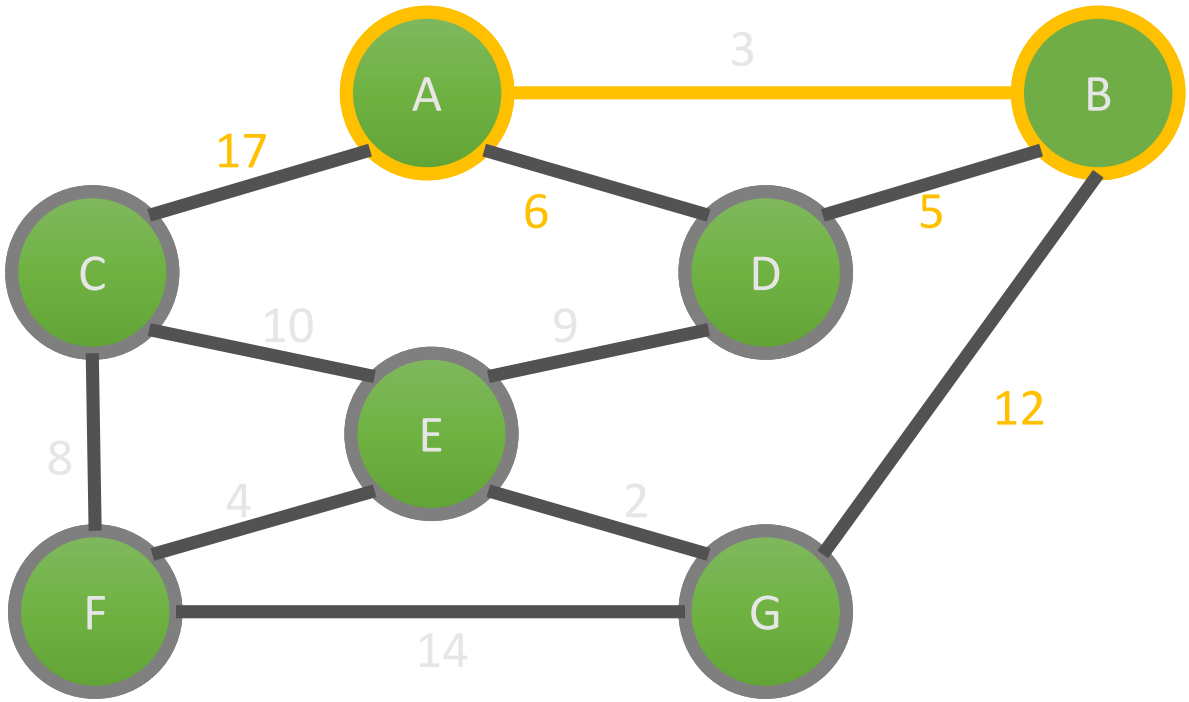
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



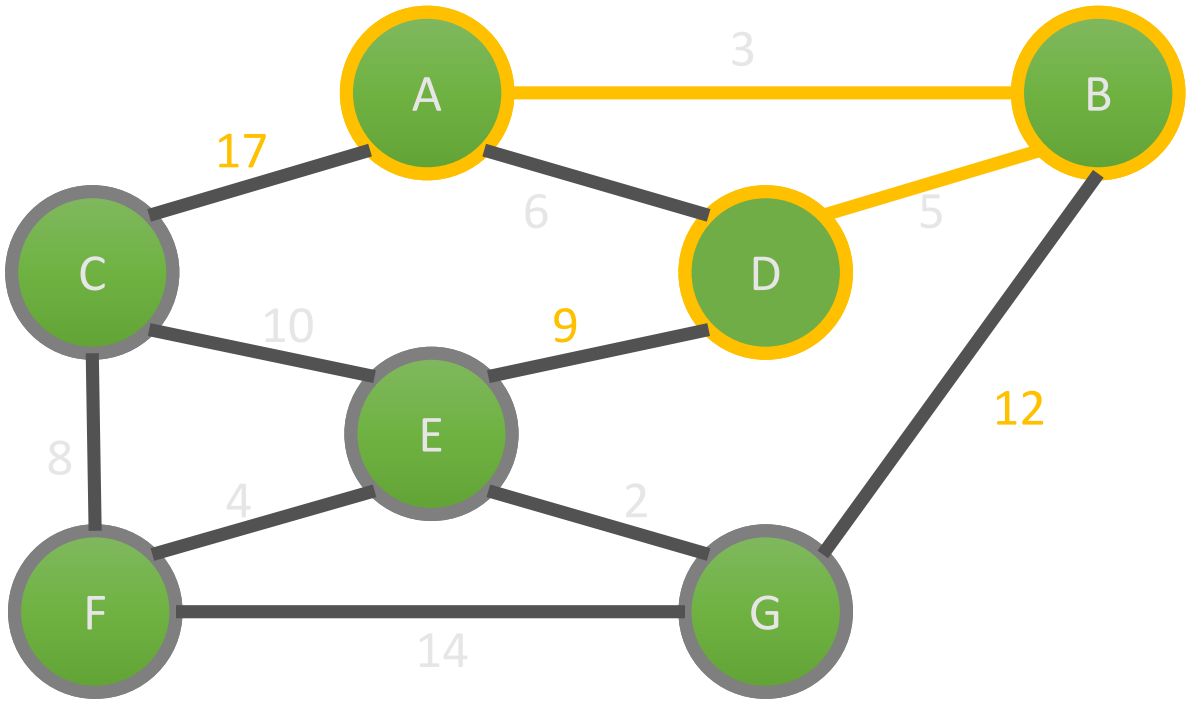
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



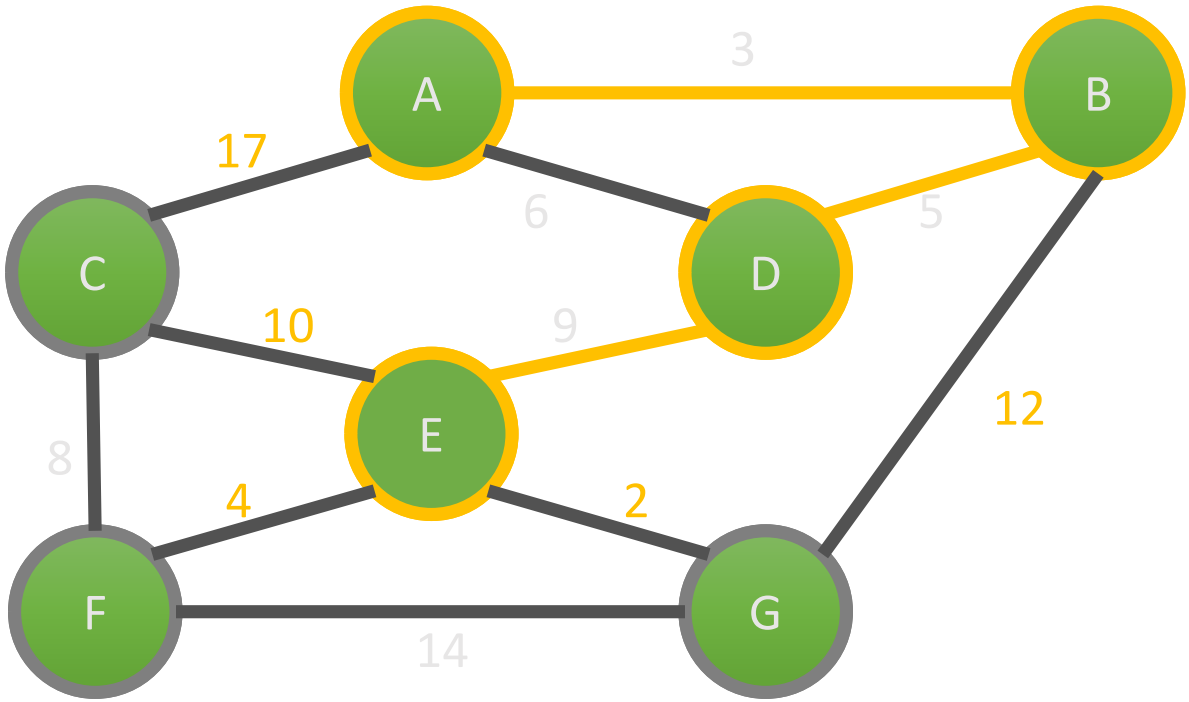
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



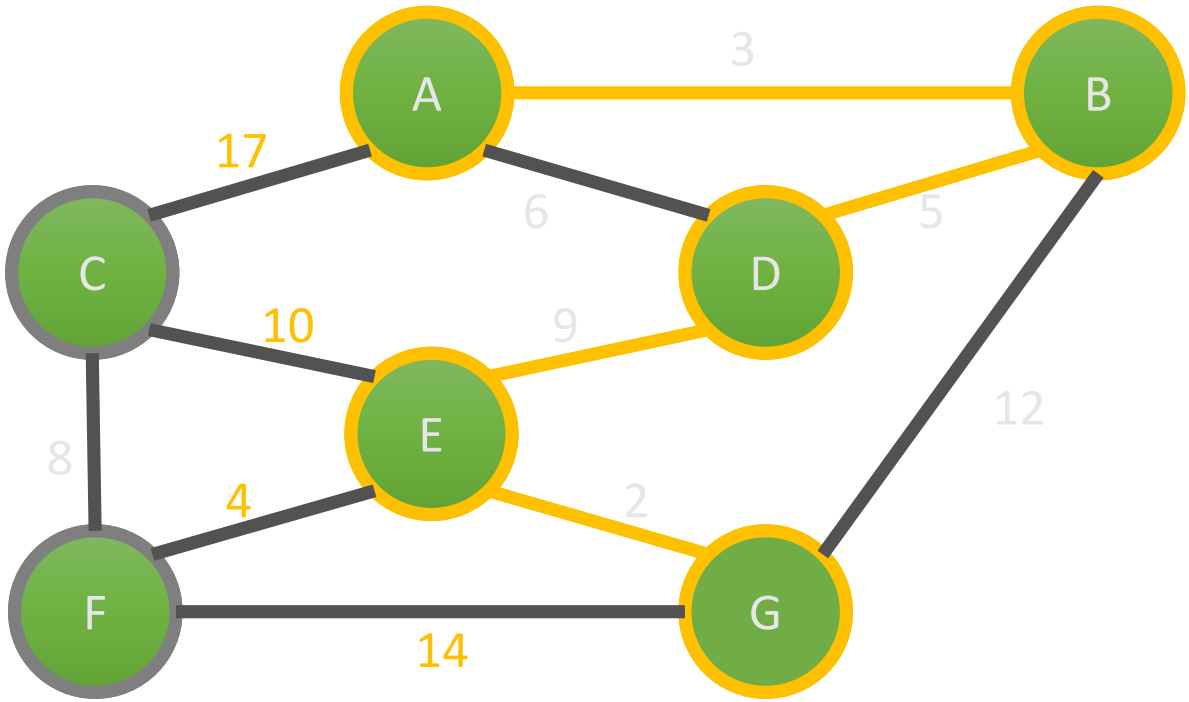
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



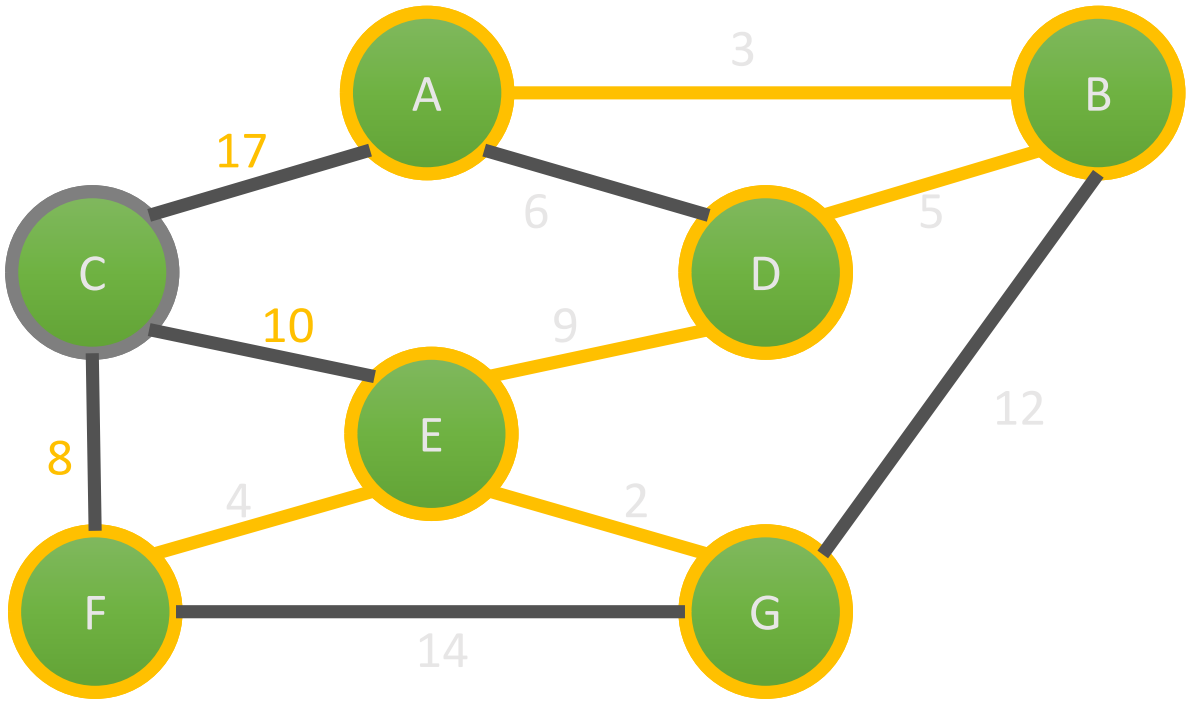
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



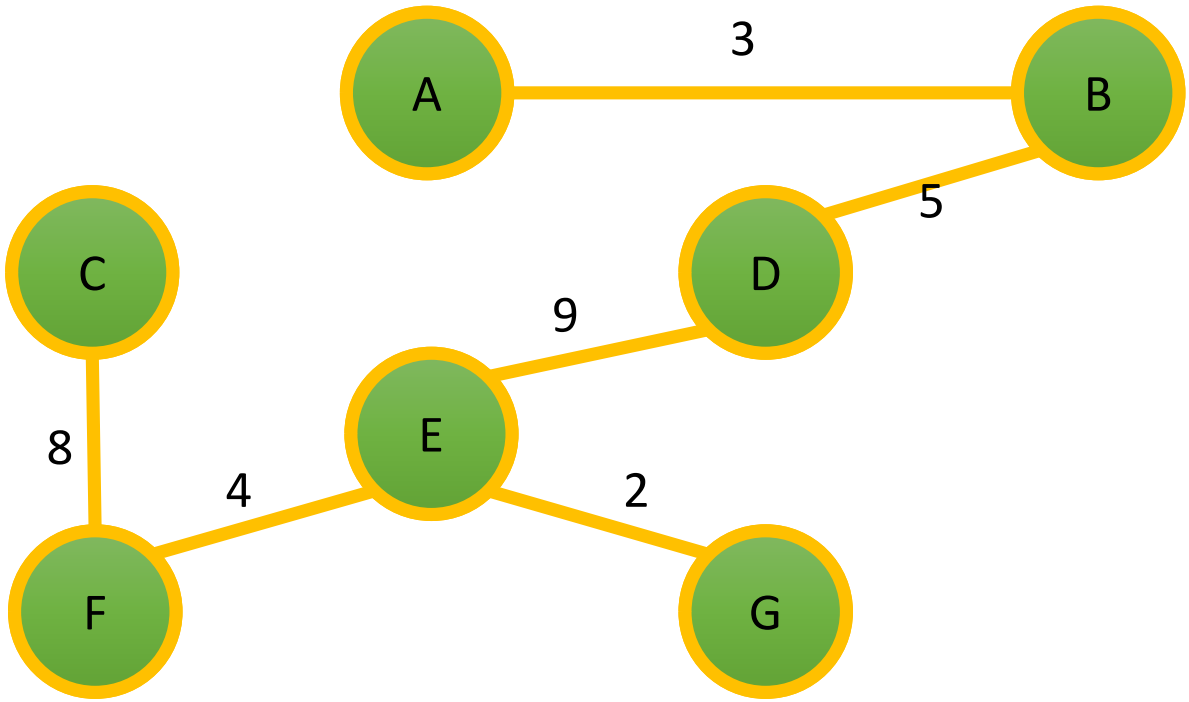
Prim's algorithm

Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.



Prim's algorithm

- Total weight: 31 (3+5+9+2+4+8)



Thanks