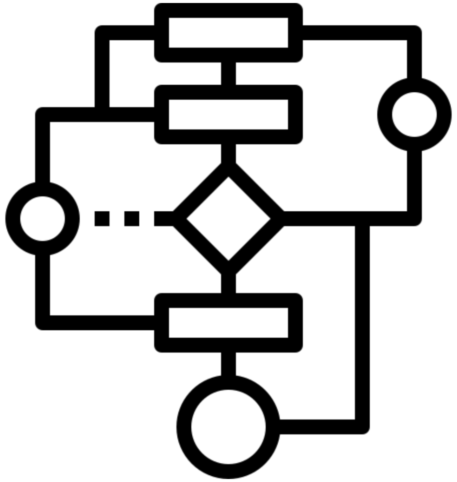
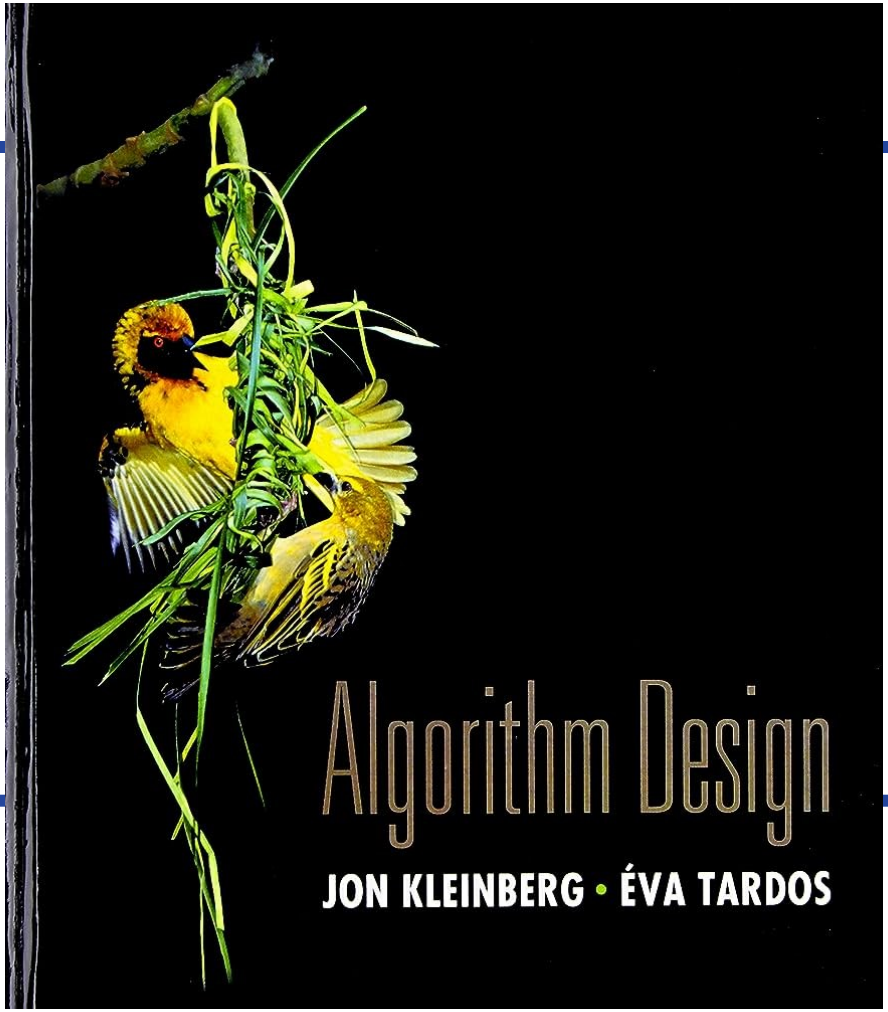


CS 310: Algorithms

Lecture 16

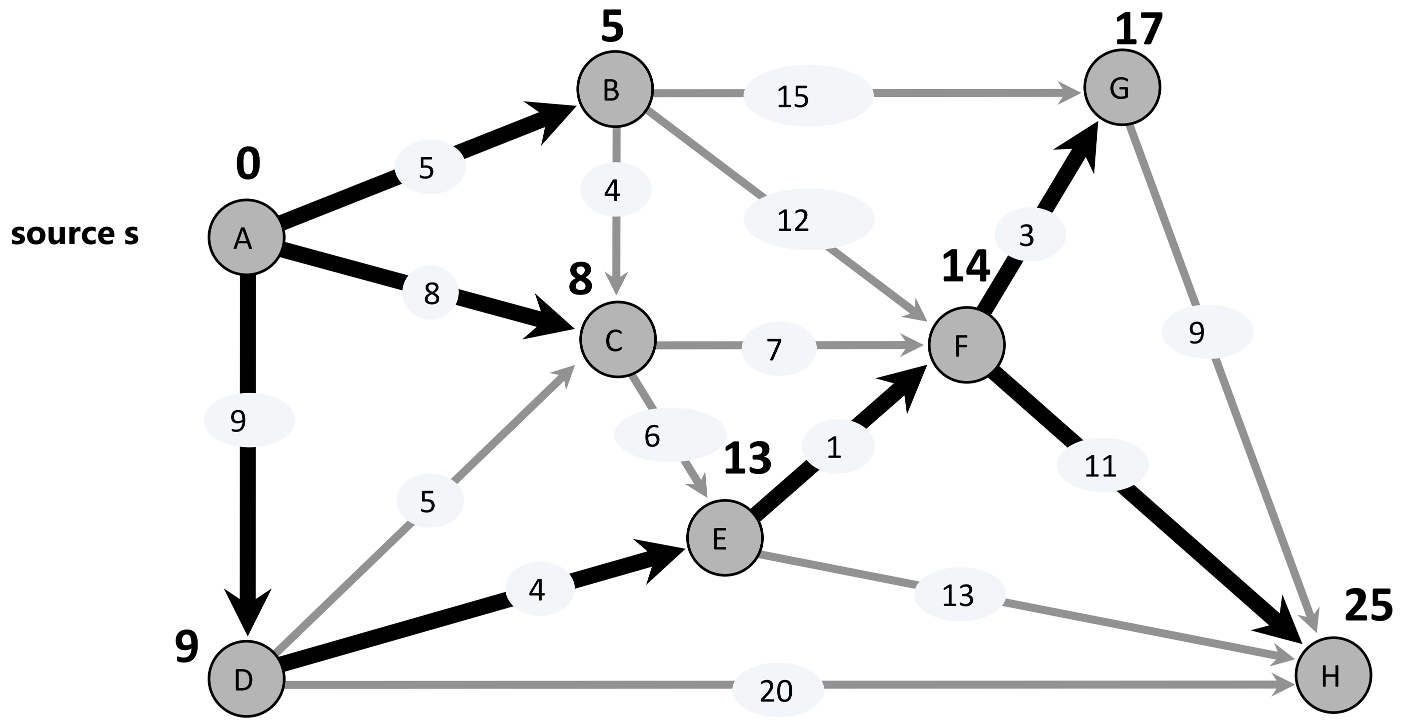
Instructor: Naveed Anwar Bhatti



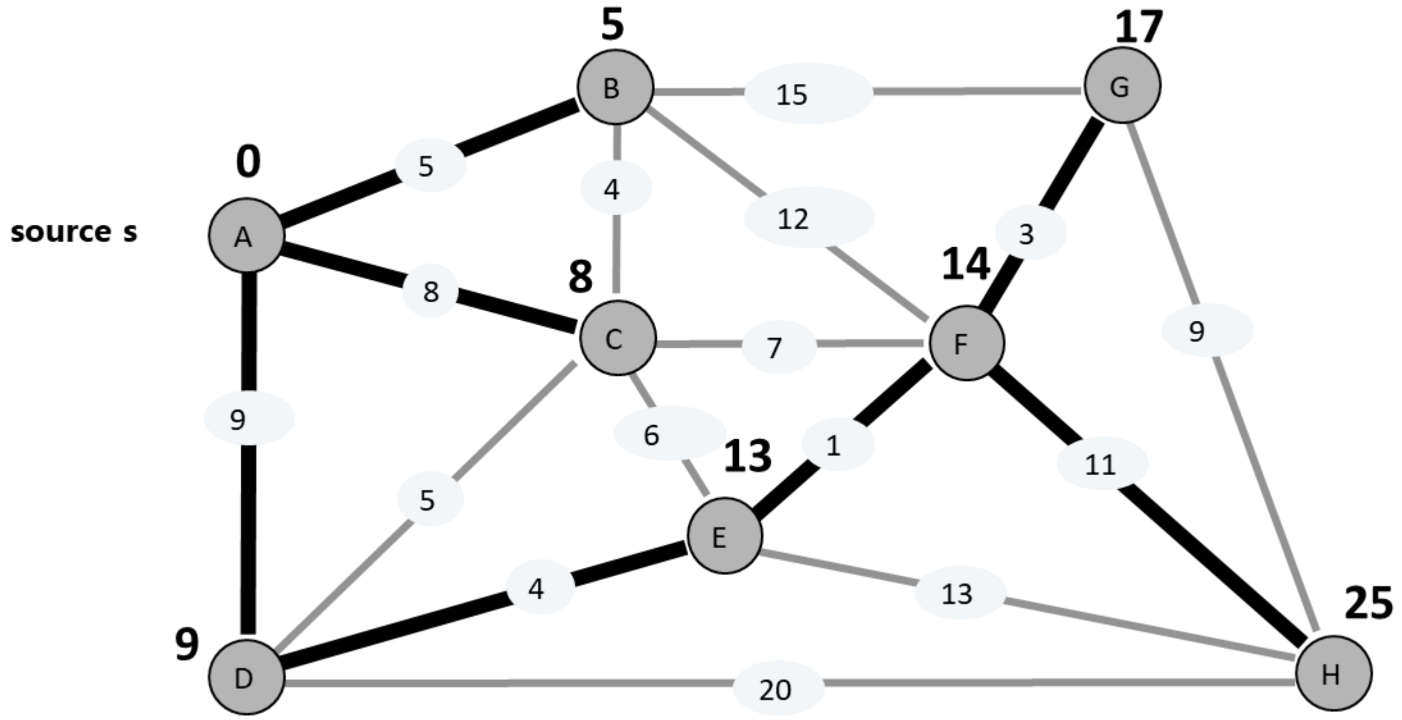
Chapter 4:
Greedy Algorithms

Section 4.5:
Minimums Spanning Tree (MST)

Single-source shortest paths problem (Dijkstra Algorithm)



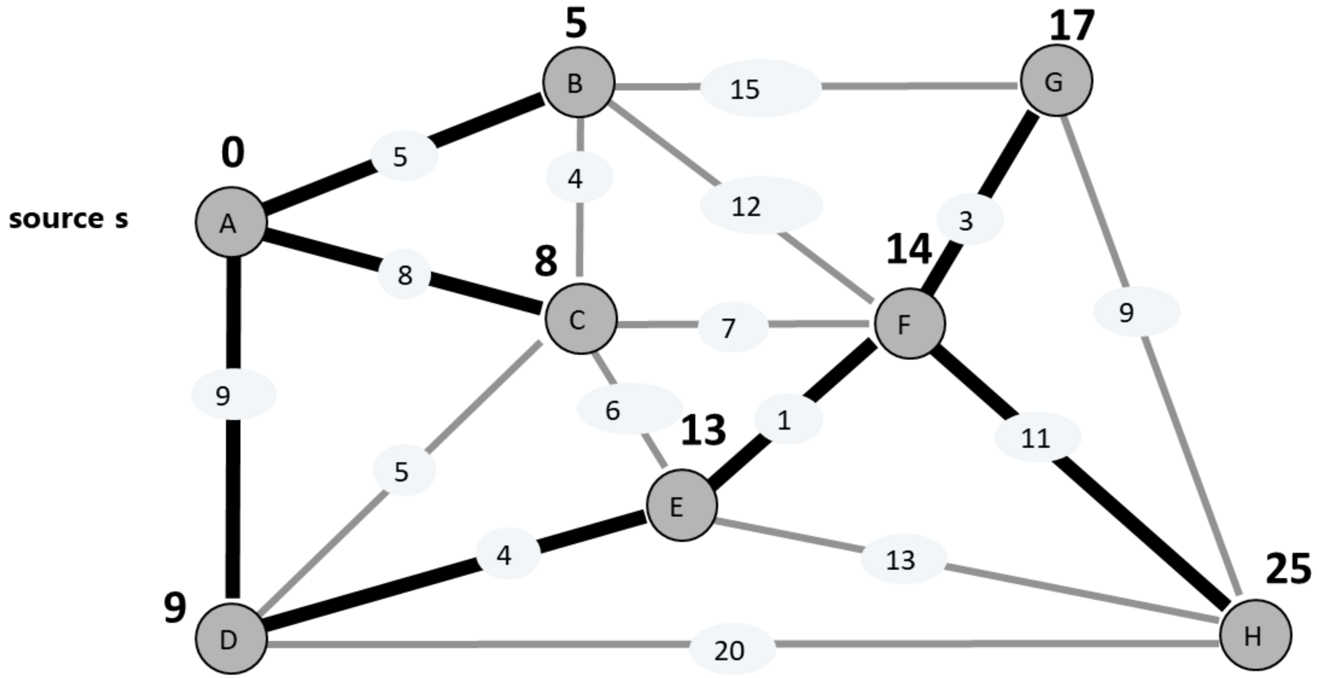
Single-source shortest paths problem (Dijkstra Algorithm)



Spanning tree definition

Def. Subgraph $H = (V, T)$ is a ***spanning tree*** of an undirected graph $G = (V, E)$ that is:

- **A tree:** *connected* and *acyclic*.
- **Spanning:** includes all of the vertices.



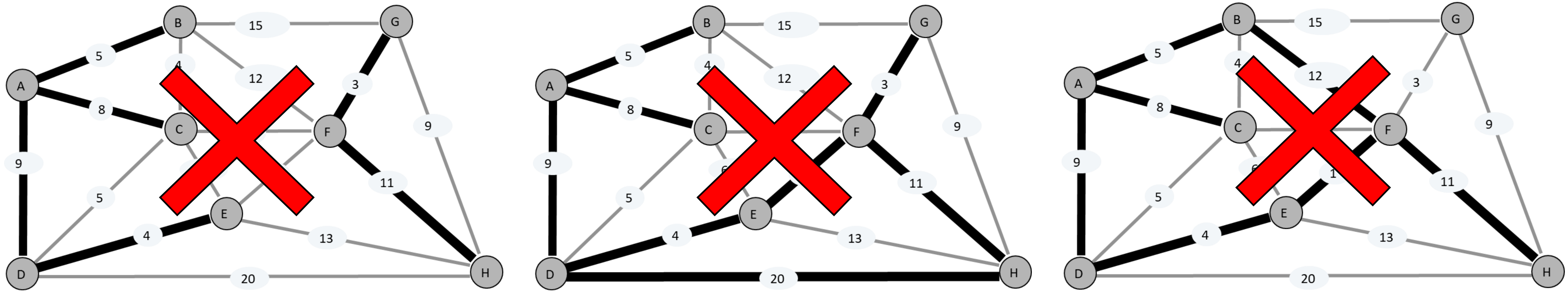
graph G = (V, E)

spanning tree H = (V, T)

Spanning tree definition

Def. Subgraph $H = (V, T)$ is a ***spanning tree*** of an undirected graph $G = (V, E)$ that is:

- **A tree:** *connected* and *acyclic*.
- **Spanning:** includes all of the vertices.



graph G = (V, E)

spanning tree H = (V, T)

Scan the QR or use link to join

https://forms.office.com/r/ZV0ZXt62Jy

Copy link

5 responses submitted

Let G be a connected edge-weighted graph with V vertices and E edges. How many edges will be in spanning tree of G?

V – 1	100%
V	0%
E – 1	0%
E	0%
I don't know.	0%

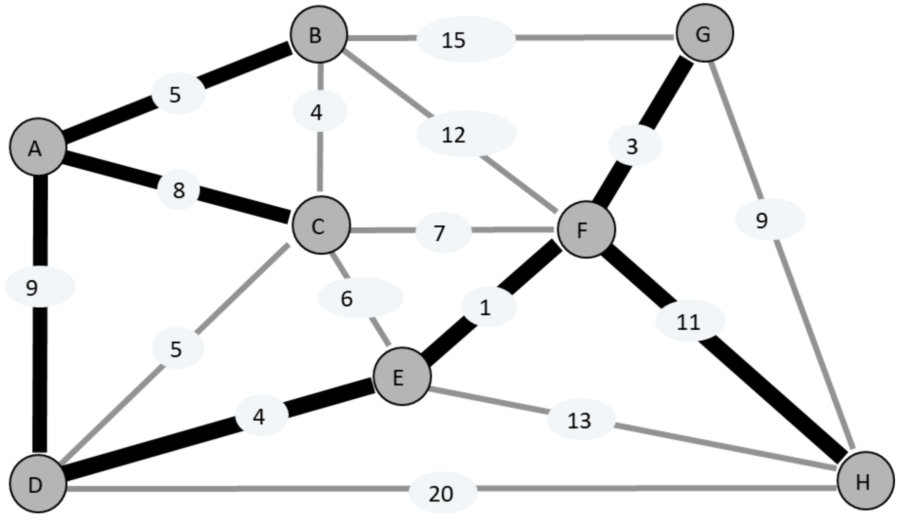
Treemap Bar

1 of 1

Spanning tree properties

- Proposition. Let $H = (V, T)$ be a **spanning tree** of an undirected graph $G = (V, E)$. Then, the following are true:
  - H is acyclic and connected.
  - H is connected and has $|V| - 1$ edges.
  - H is acyclic and has $|V| - 1$ edges.
  - H is minimally connected: removal of any edge disconnects it.
  - H is maximally acyclic: addition of any edge creates a cycle.

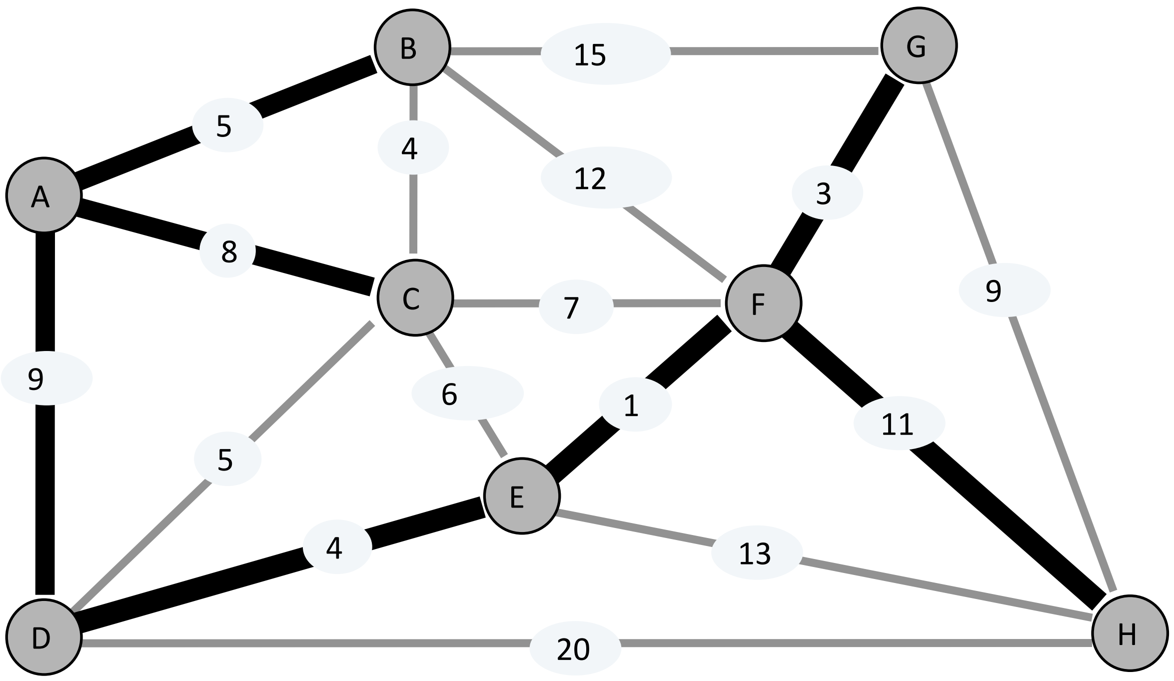
graph G = (V, E)
spanning tree H = (V, T)



Minimum Spanning Tree (MST)

- **Input.** Connected, undirected graph ***G*** with positive edge weights.
- **Output.** A spanning tree of minimum weight.

5+8+9+4+1+3+11 = 41



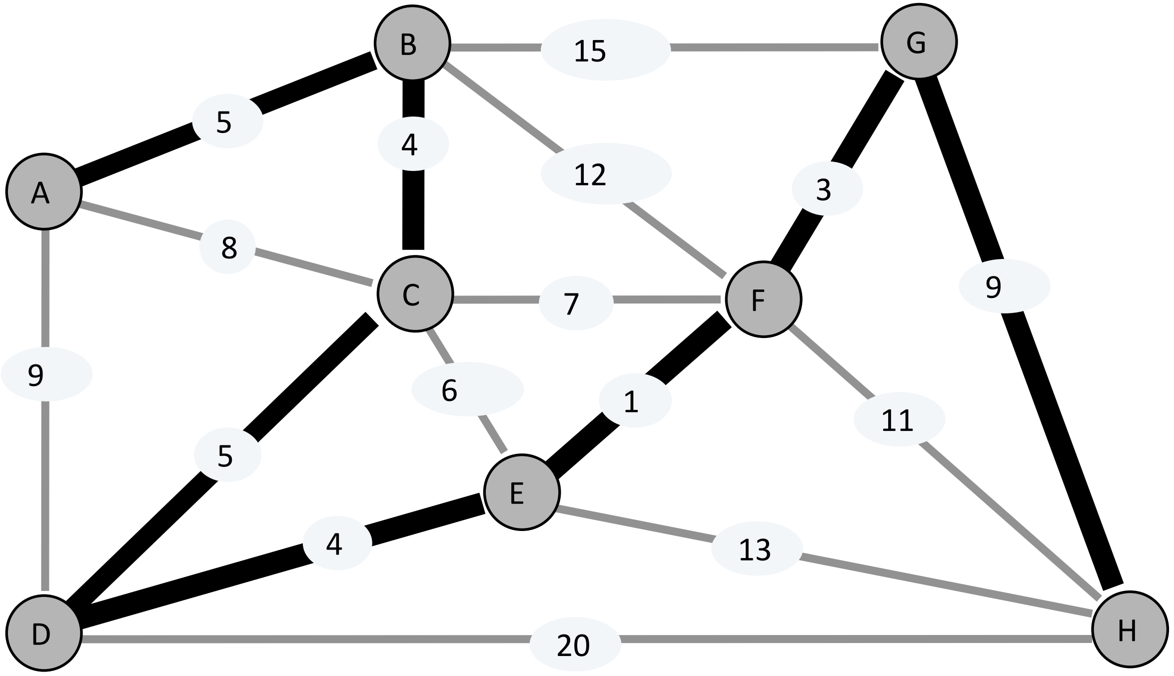
graph G = (V, E)

spanning tree H = (V, T)

Minimum Spanning Tree (MST)

- **Input.** Connected, undirected graph ***G*** with positive edge weights.
- **Output.** A spanning tree of minimum weight.

5+4+5+4+1+3+9 = 31

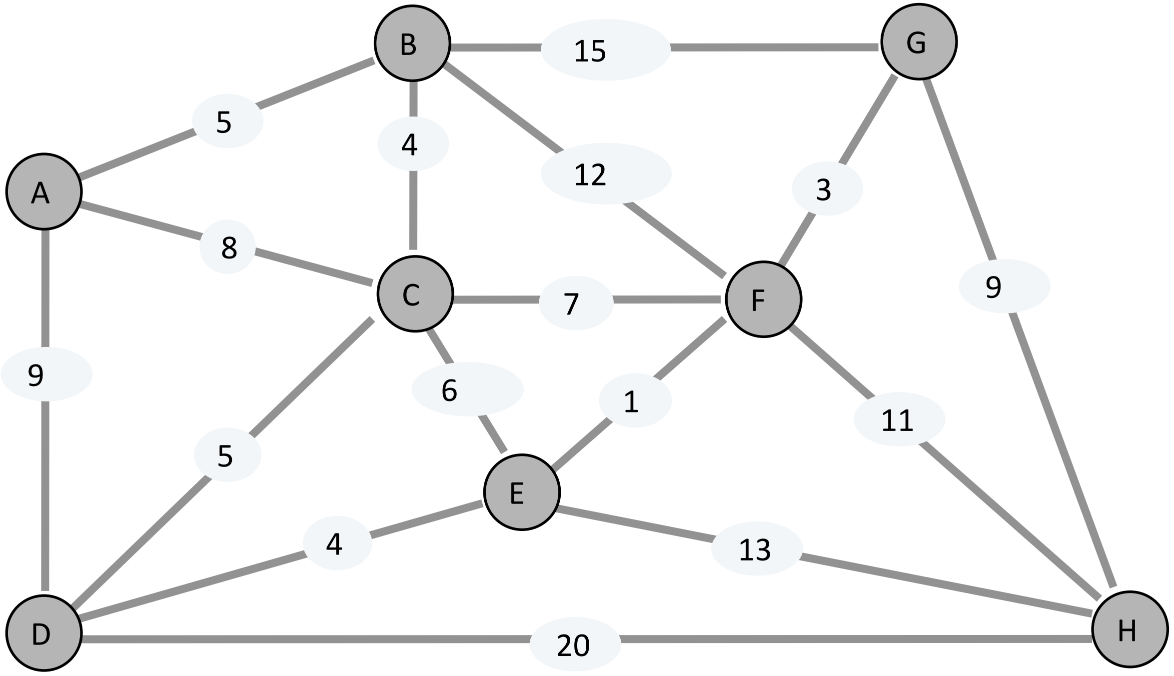


graph G = (V, E)

Minimum spanning tree H = (V, T)

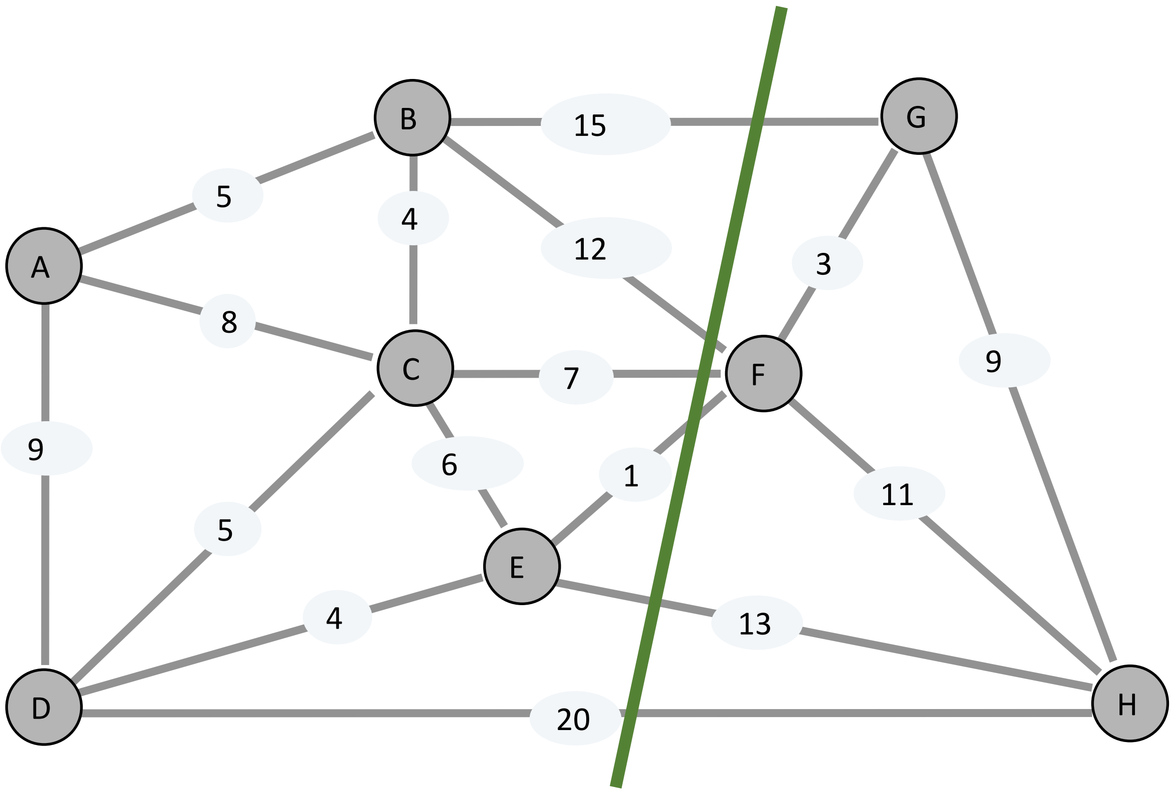
Cut Property

Def: A **cut** in a graph is a partition of its vertices into two (nonempty) sets



Cut Property

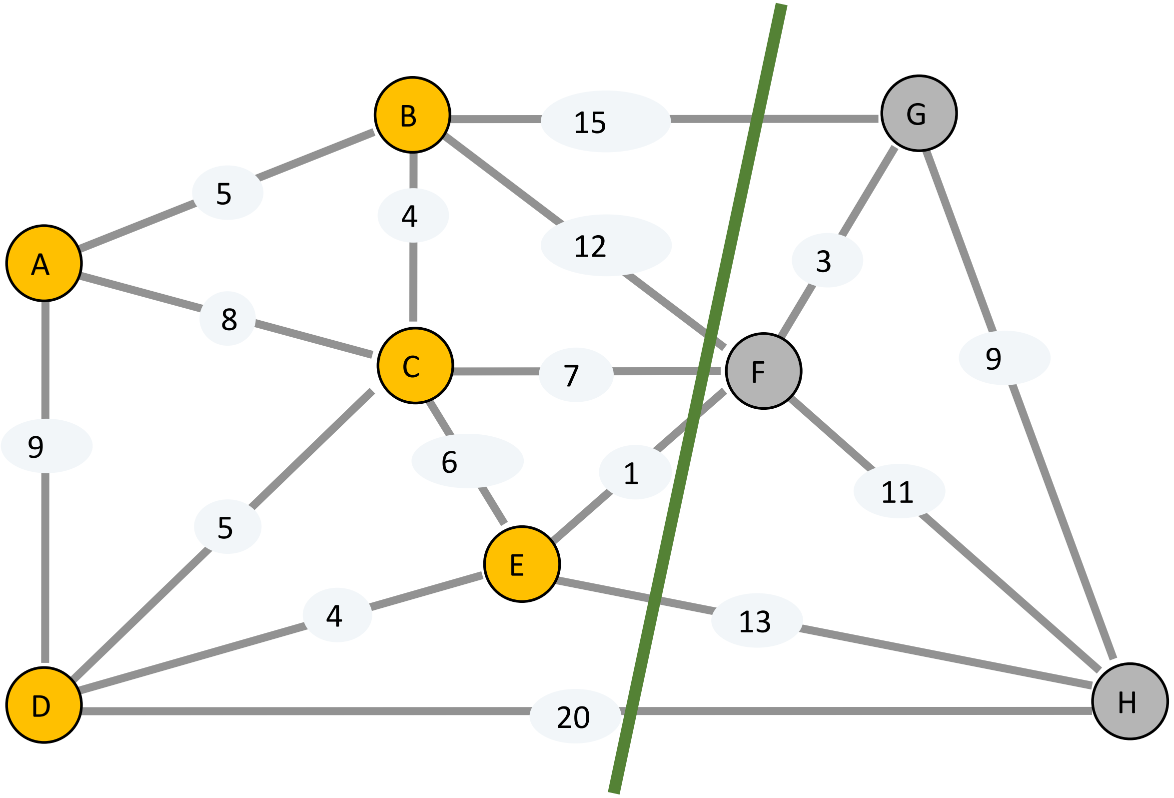
Def: A **cut** in a graph is a partition of its vertices into two (nonempty) sets



Cut Property

Def: A **cut** in a graph is a partition of its vertices into two (nonempty) sets

{A,B,C,D,E}

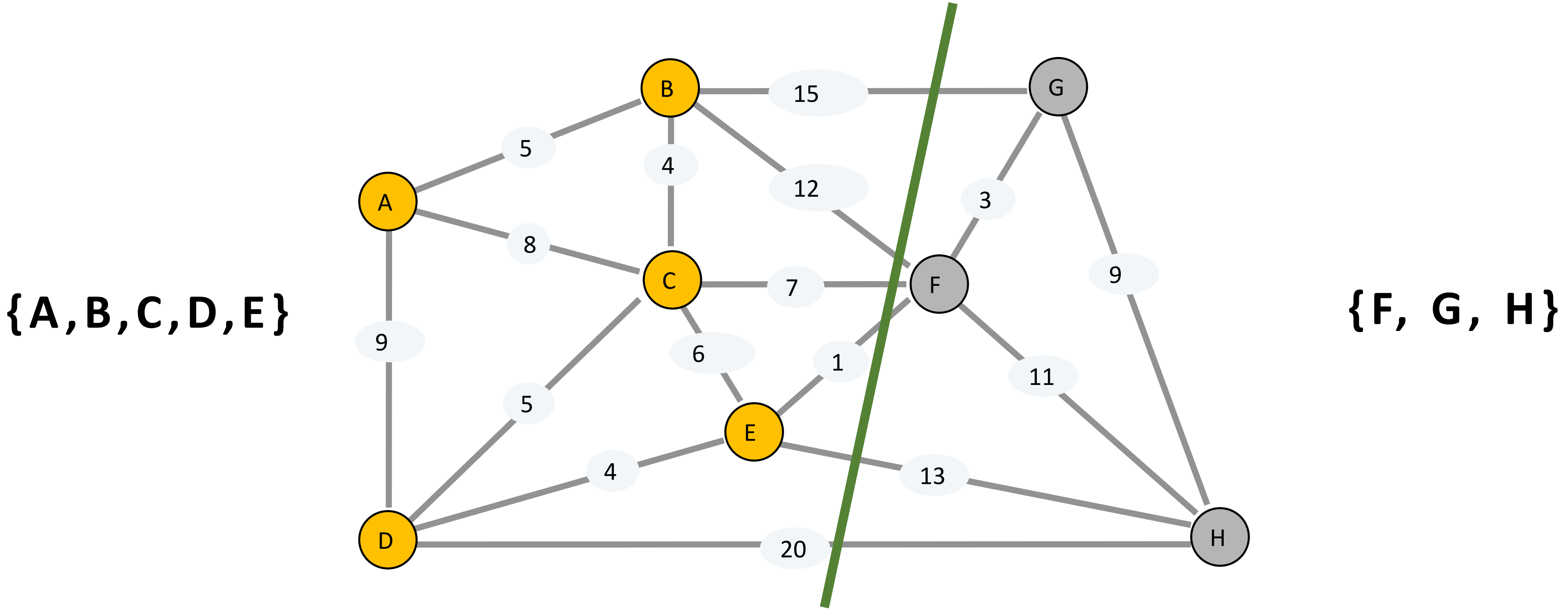


{F, G, H}

Cut Property

Def: A **cut** in a graph is a partition of its vertices into two (nonempty) sets

Def: A **crossing edge** connects a vertex in one set with a vertex in the other.

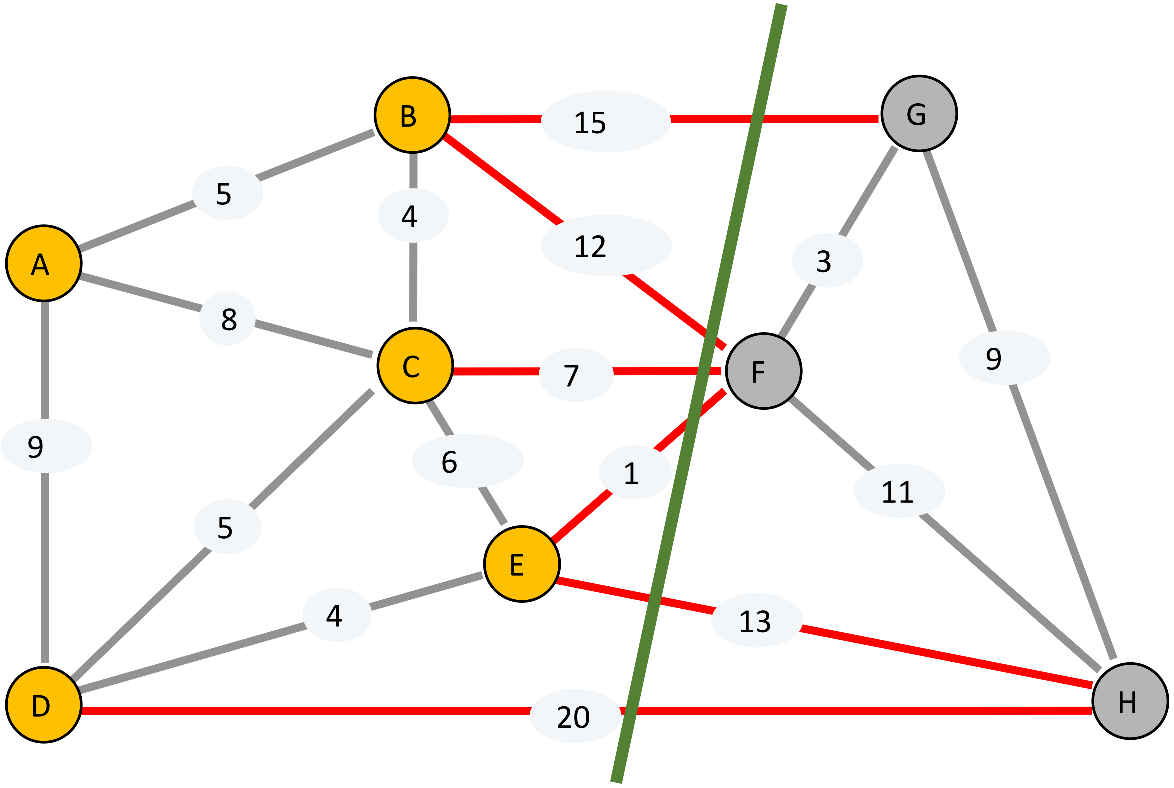


Cut Property

Def: A **cut** in a graph is a partition of its vertices into two (nonempty)

Def: A **crossing edge** connects a vertex in one set with a vertex in the other.

{A,B,C,D,E}



{F, G, H}

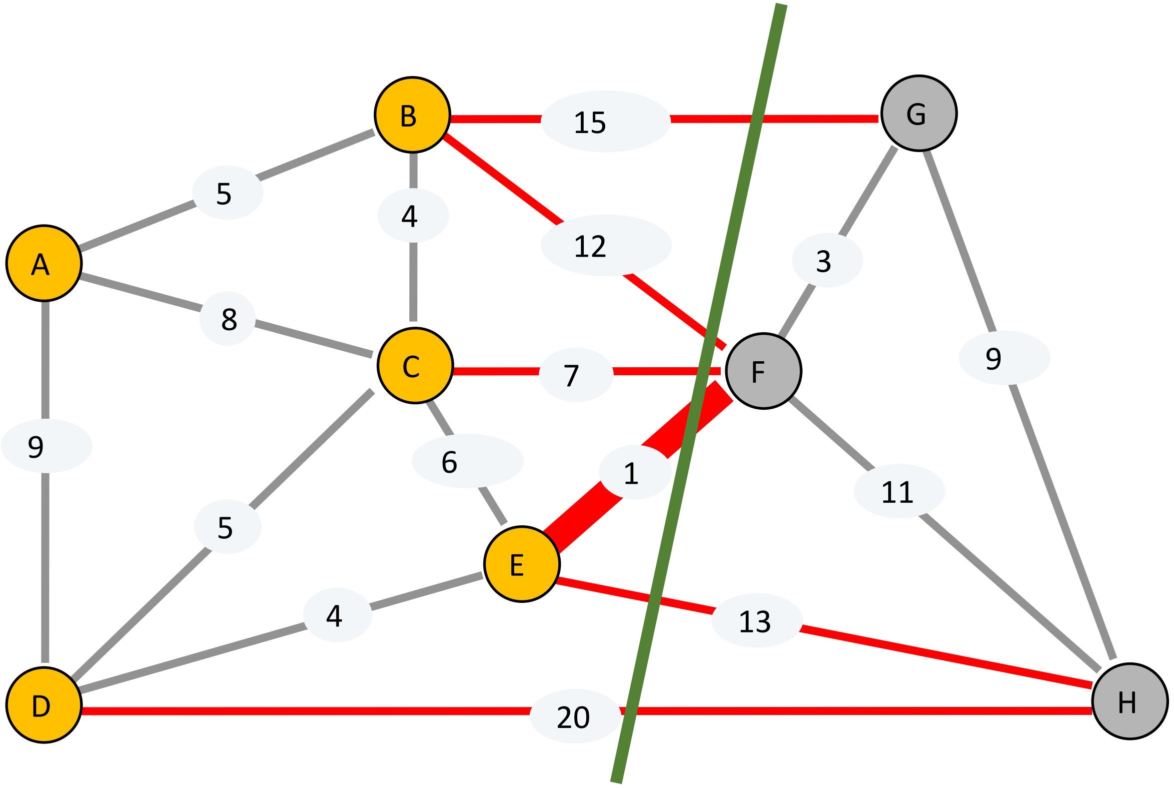
Cut property: Given any cut, the crossing edge of min weight is in the MST.

Cut Property

Def: A **cut** in a graph is a partition of its vertices into two (nonempty)

Def: A **crossing edge** connects a vertex in one set with a vertex in the other.

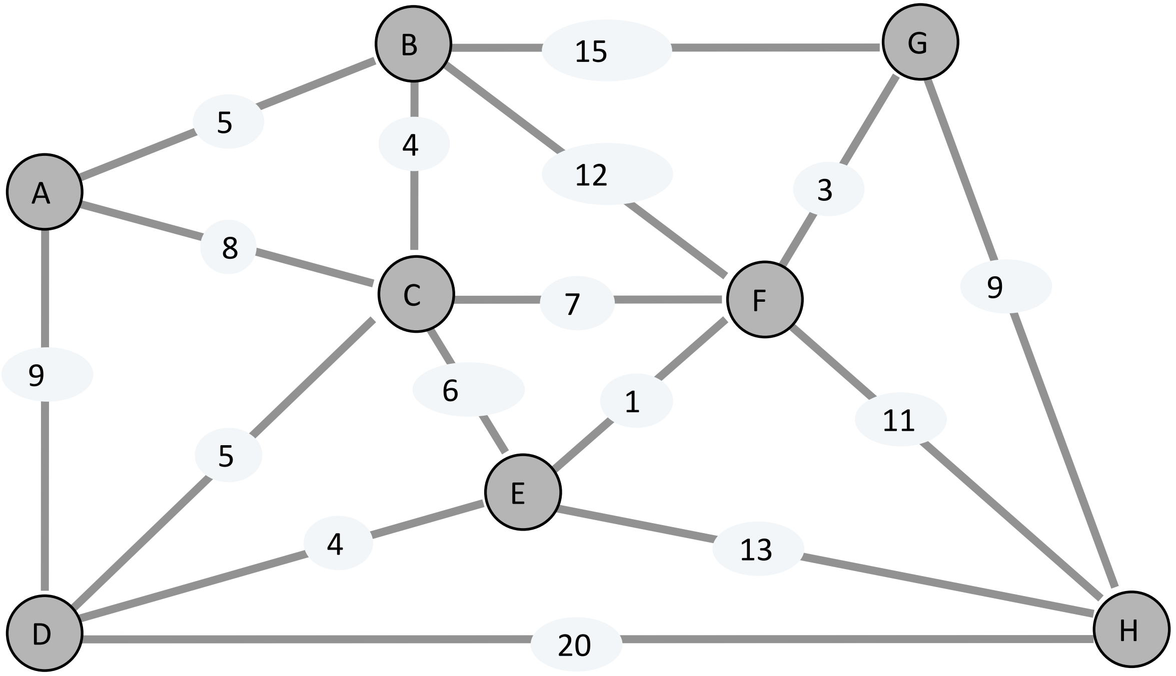
{A,B,C,D,E}



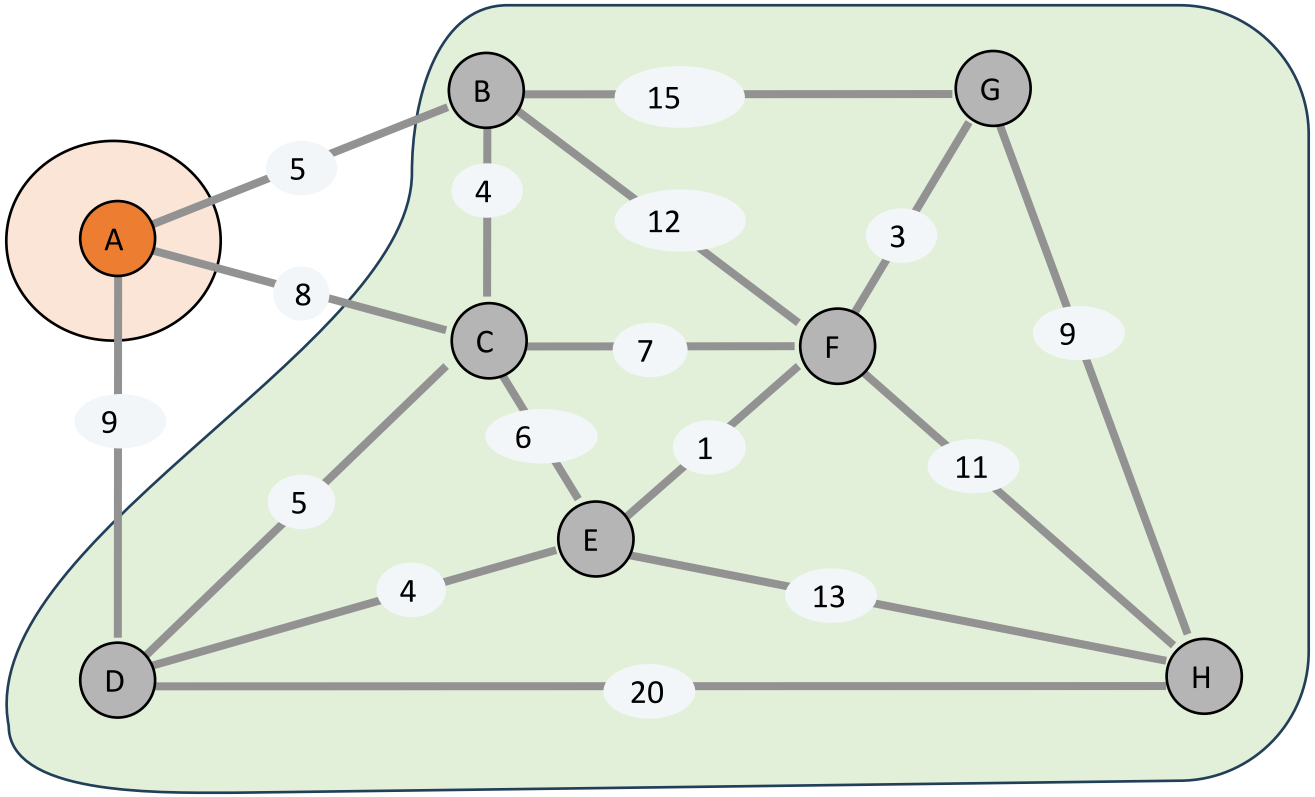
{F, G, H}

Cut property: Given any cut, the crossing edge of min weight is in the MST.

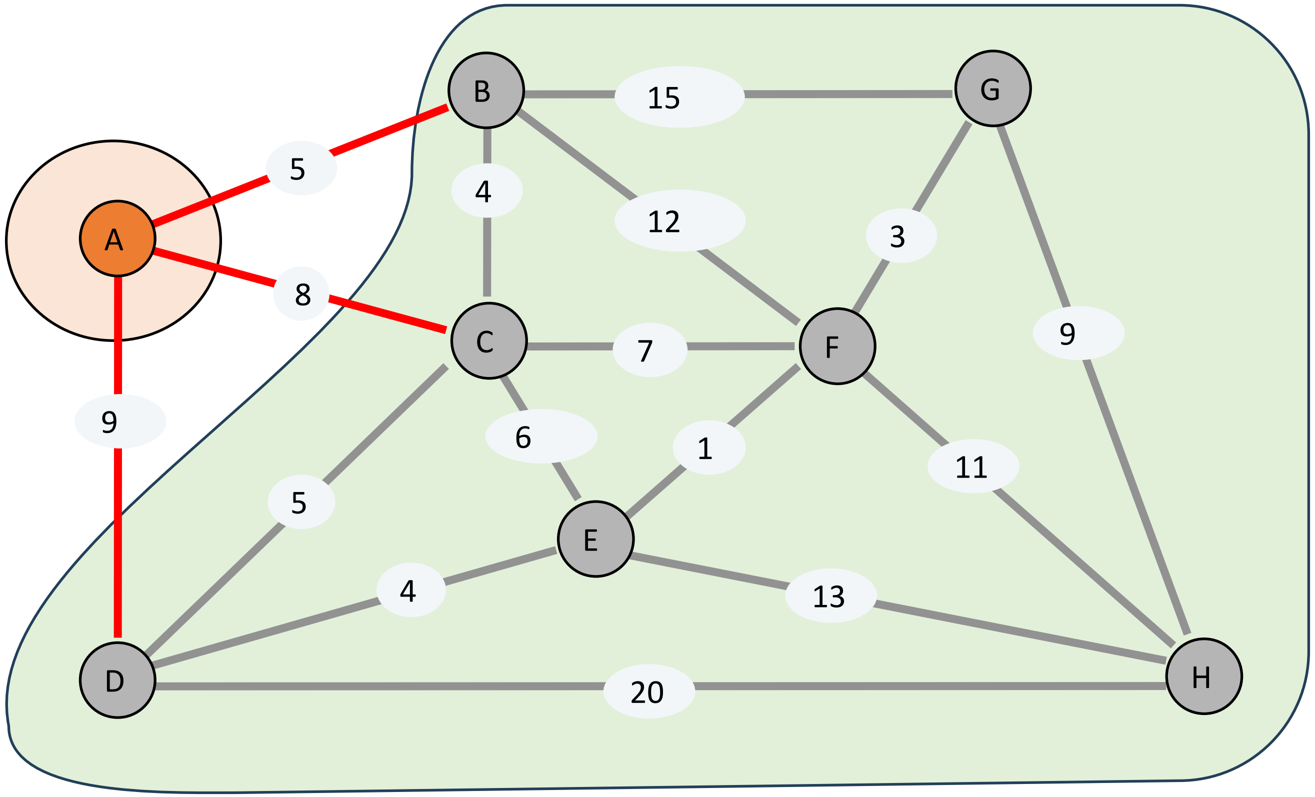
Prim's algorithm to find MST



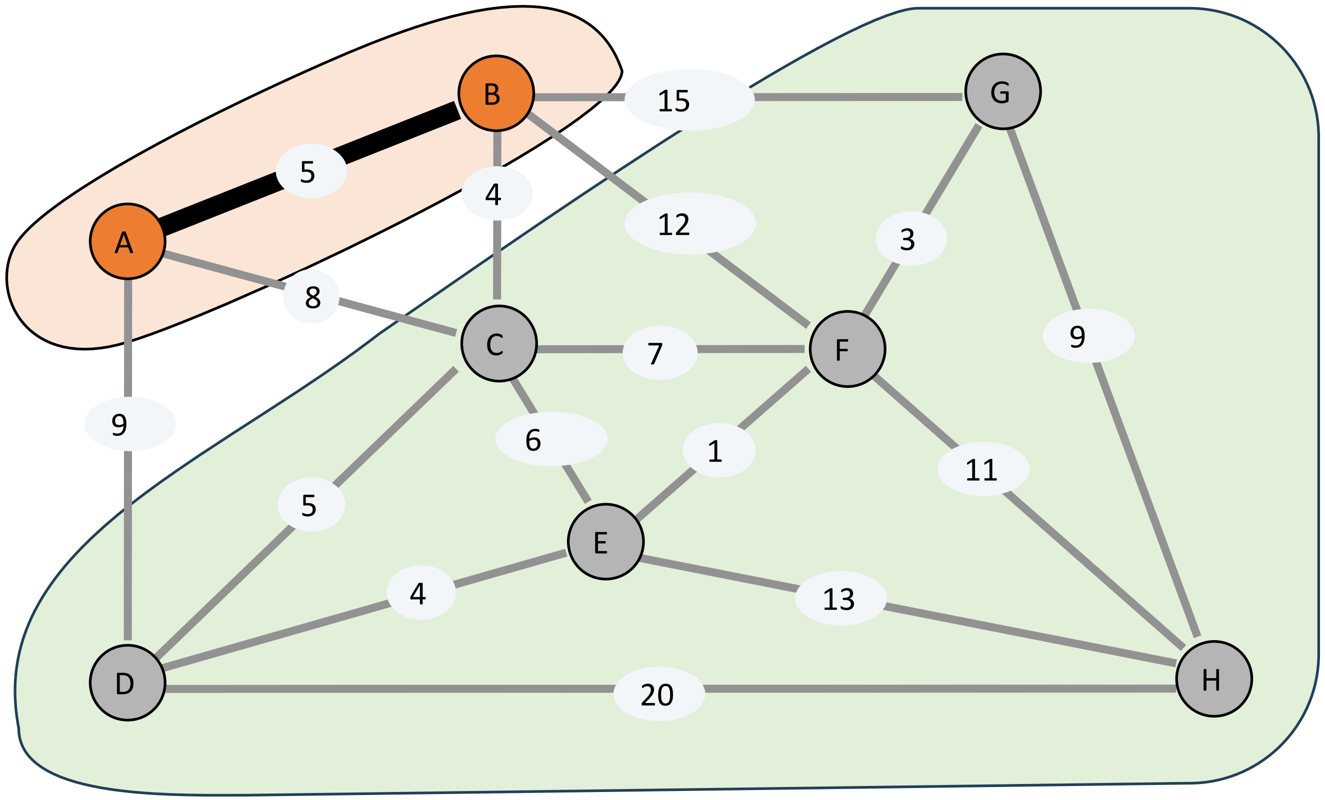
Prim’s algorithm to find MST



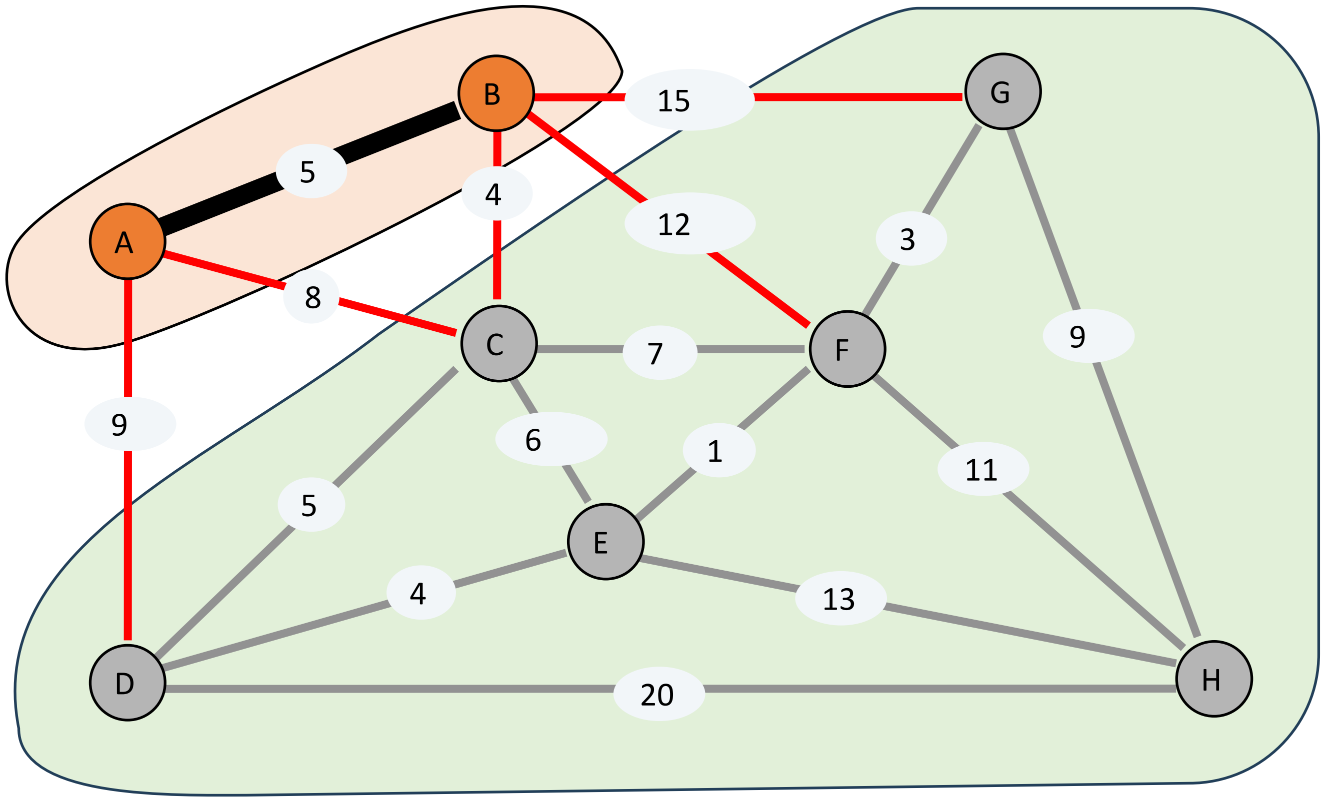
Prim’s algorithm to find MST



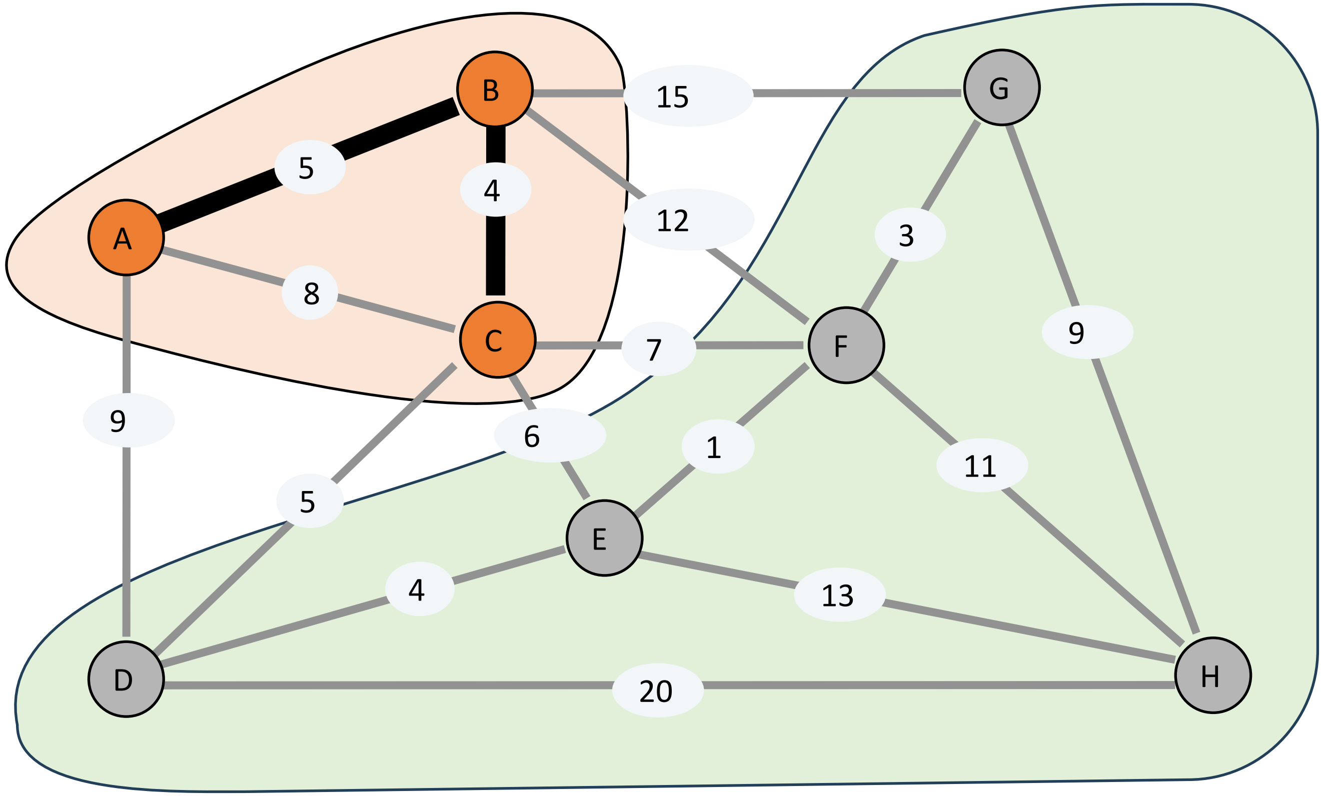
Prim's algorithm to find MST



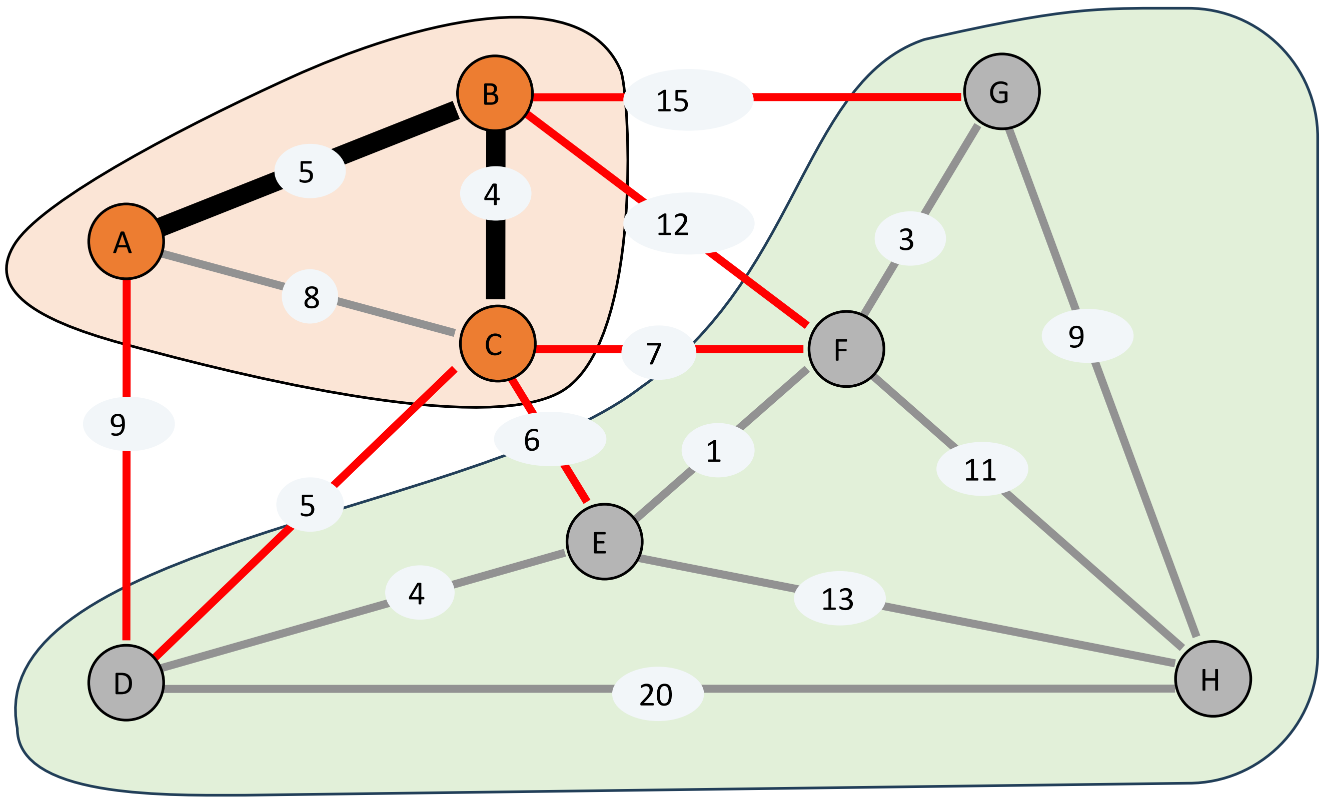
Prim’s algorithm to find MST



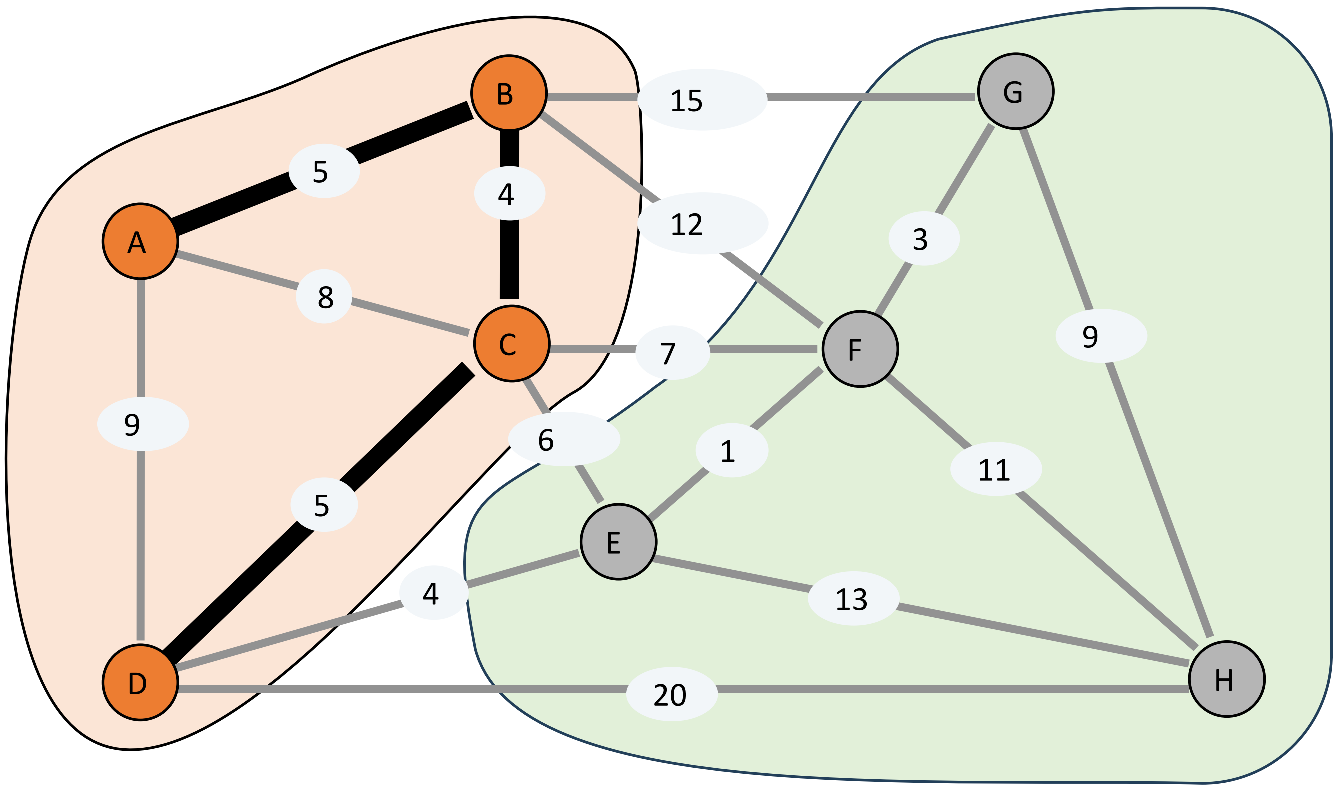
Prim’s algorithm to find MST



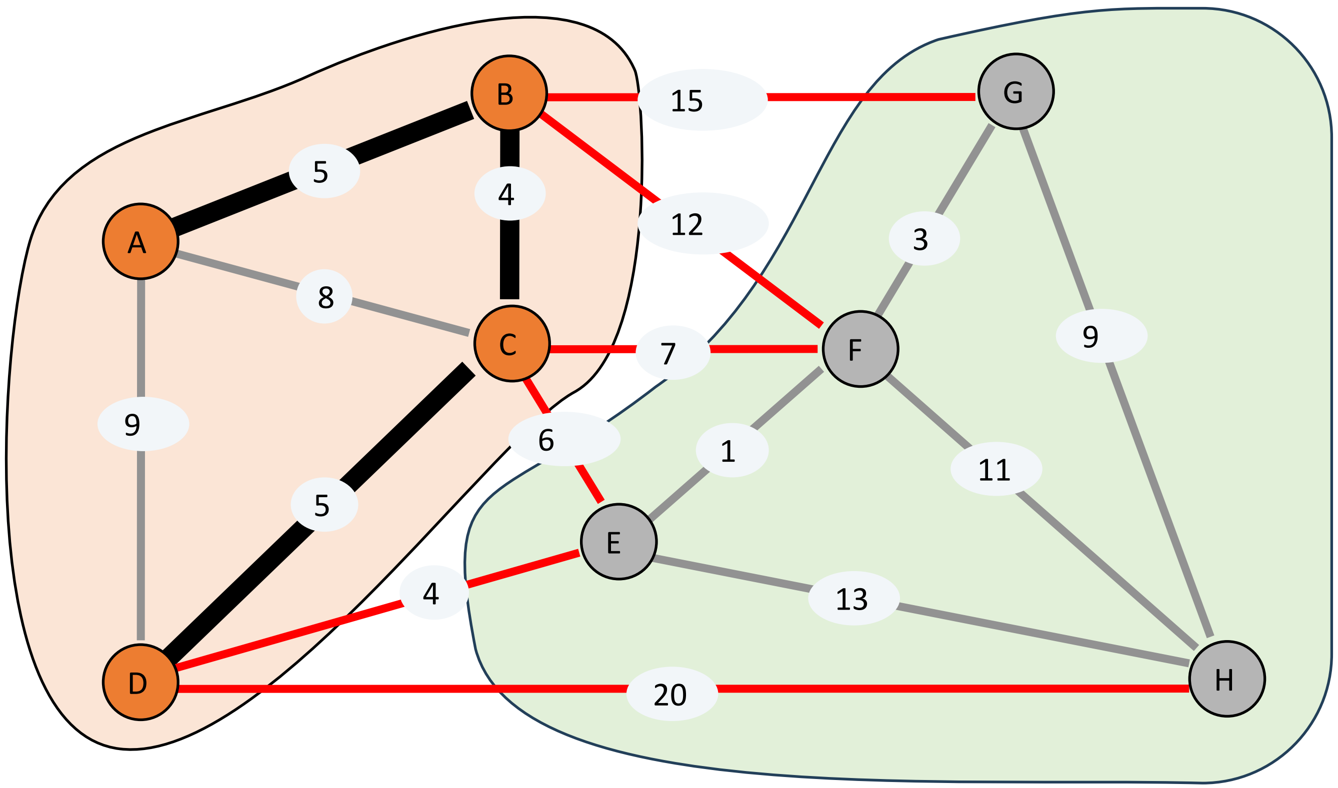
Prim’s algorithm to find MST



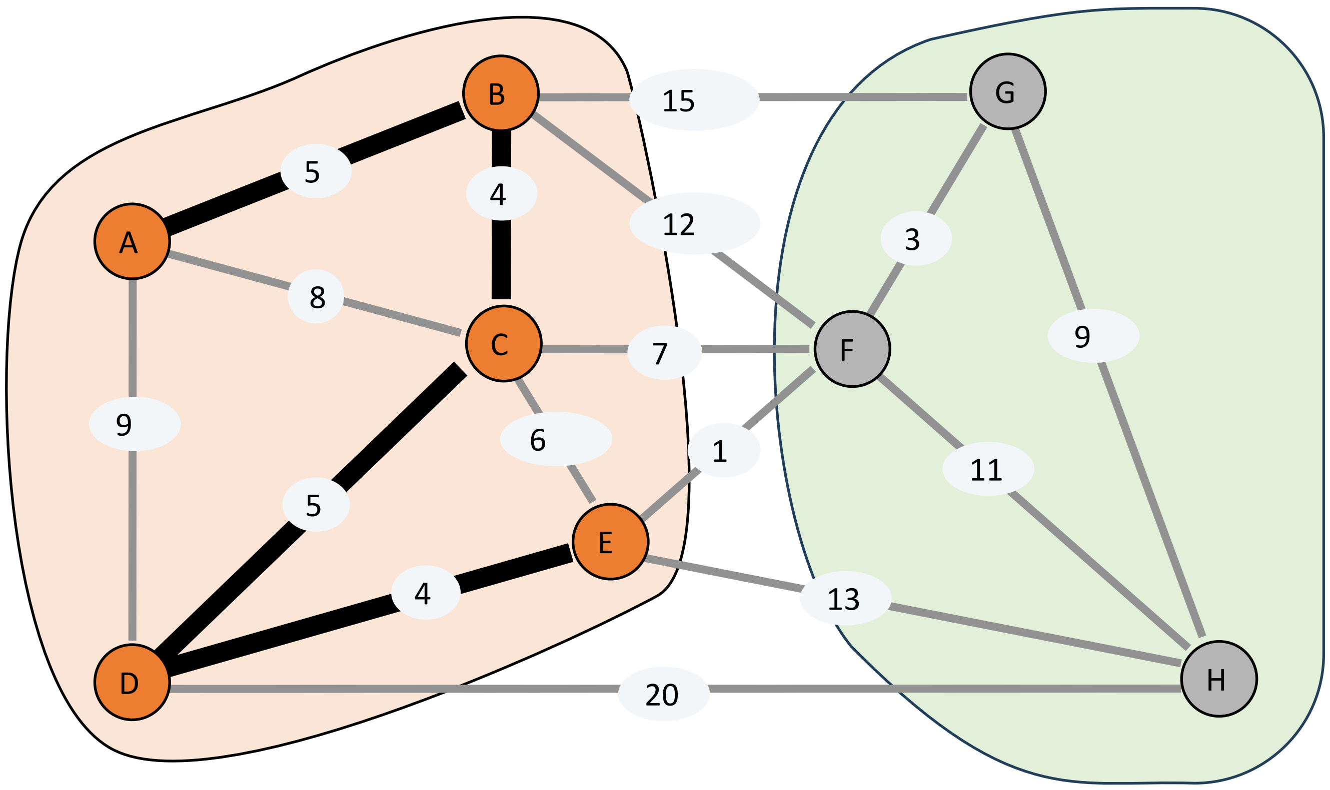
Prim’s algorithm to find MST



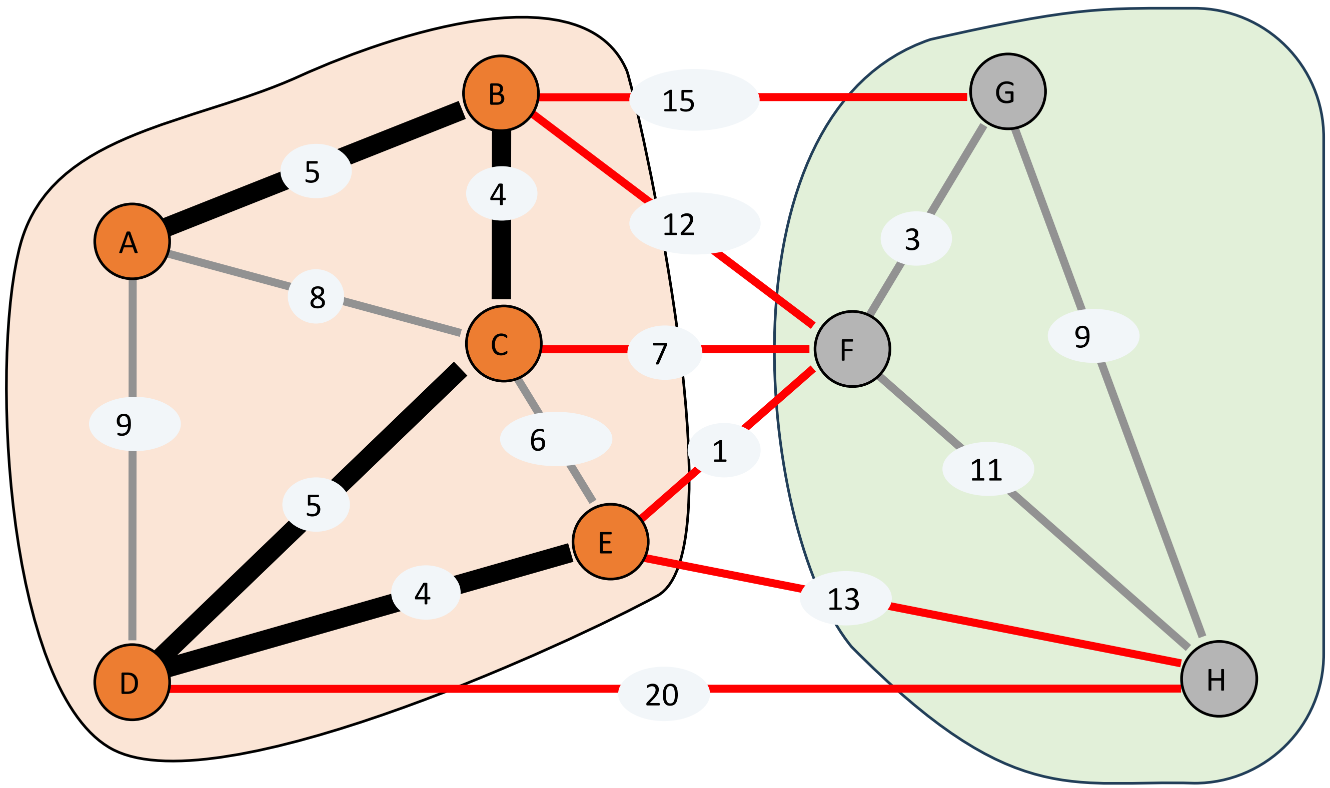
Prim’s algorithm to find MST



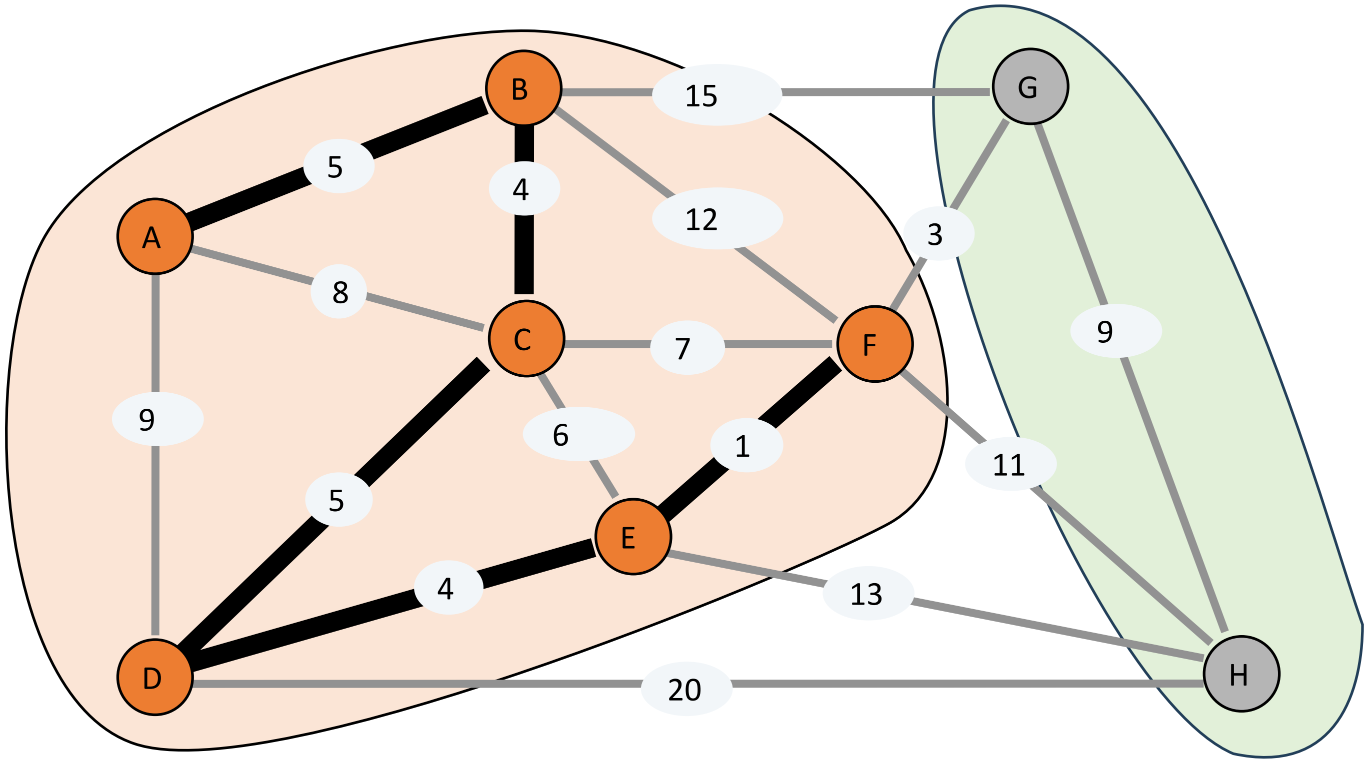
Prim’s algorithm to find MST



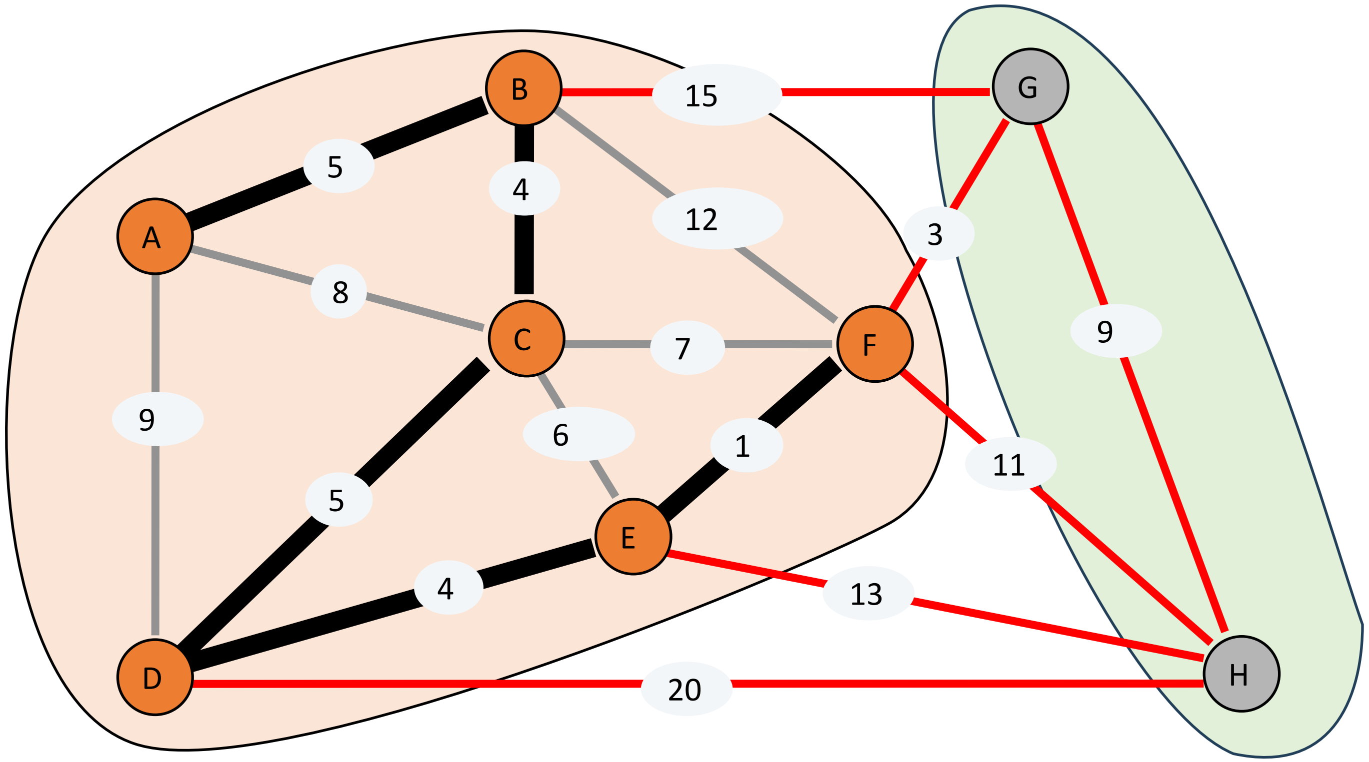
Prim's algorithm to find MST



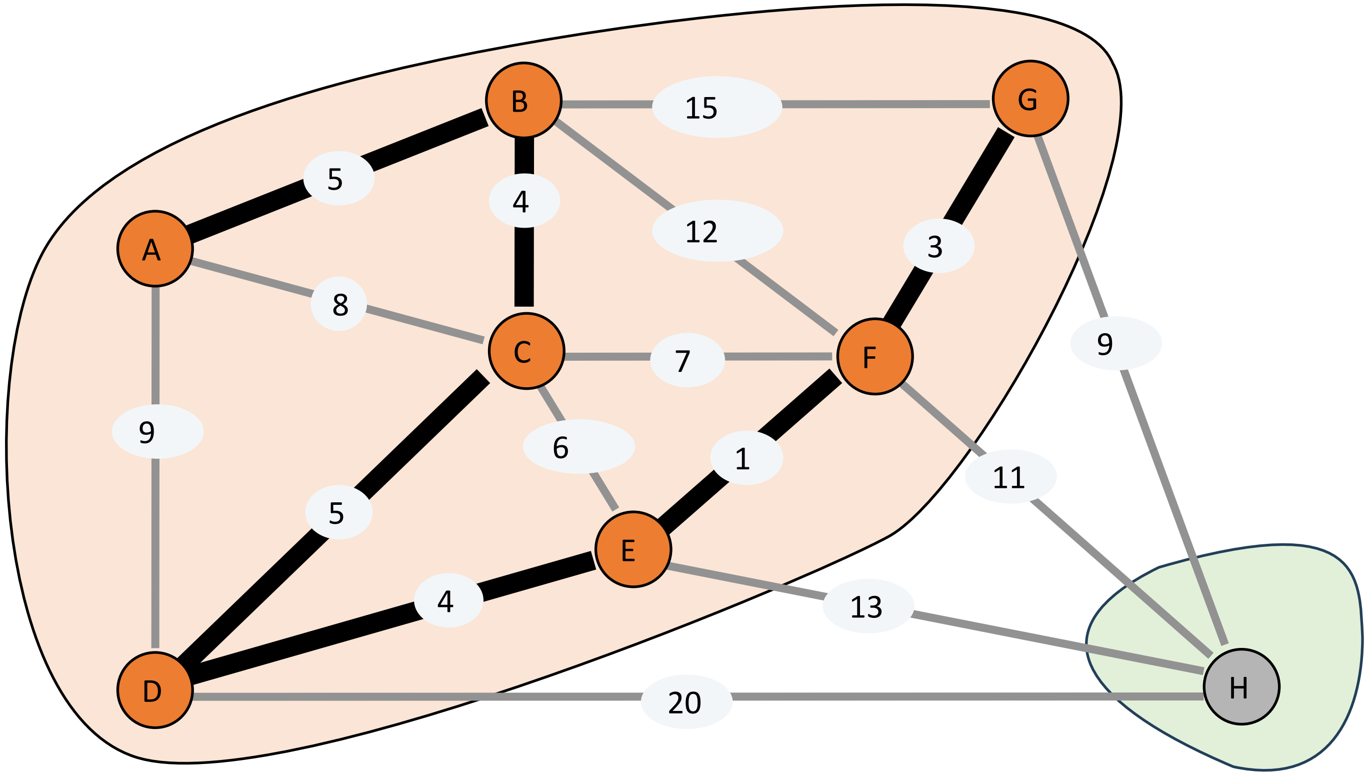
Prim’s algorithm to find MST



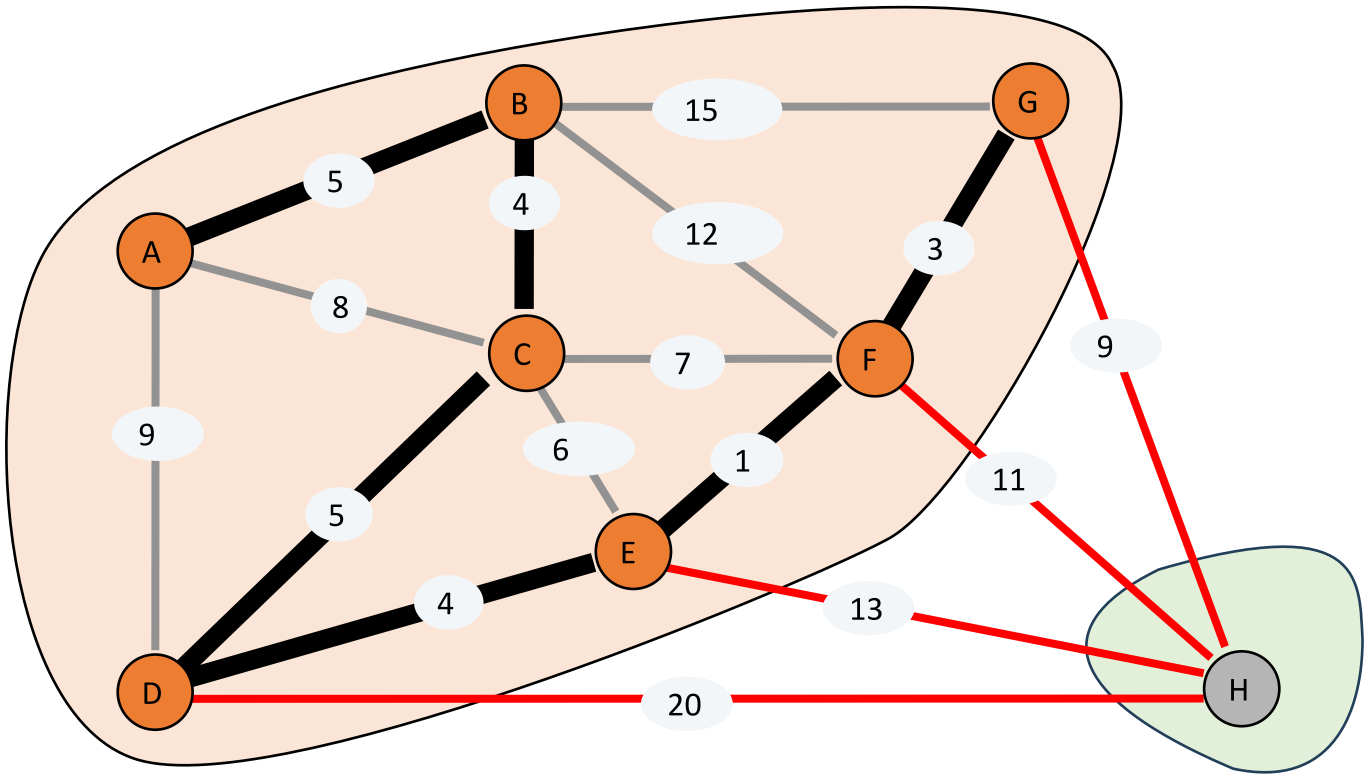
Prim’s algorithm to find MST



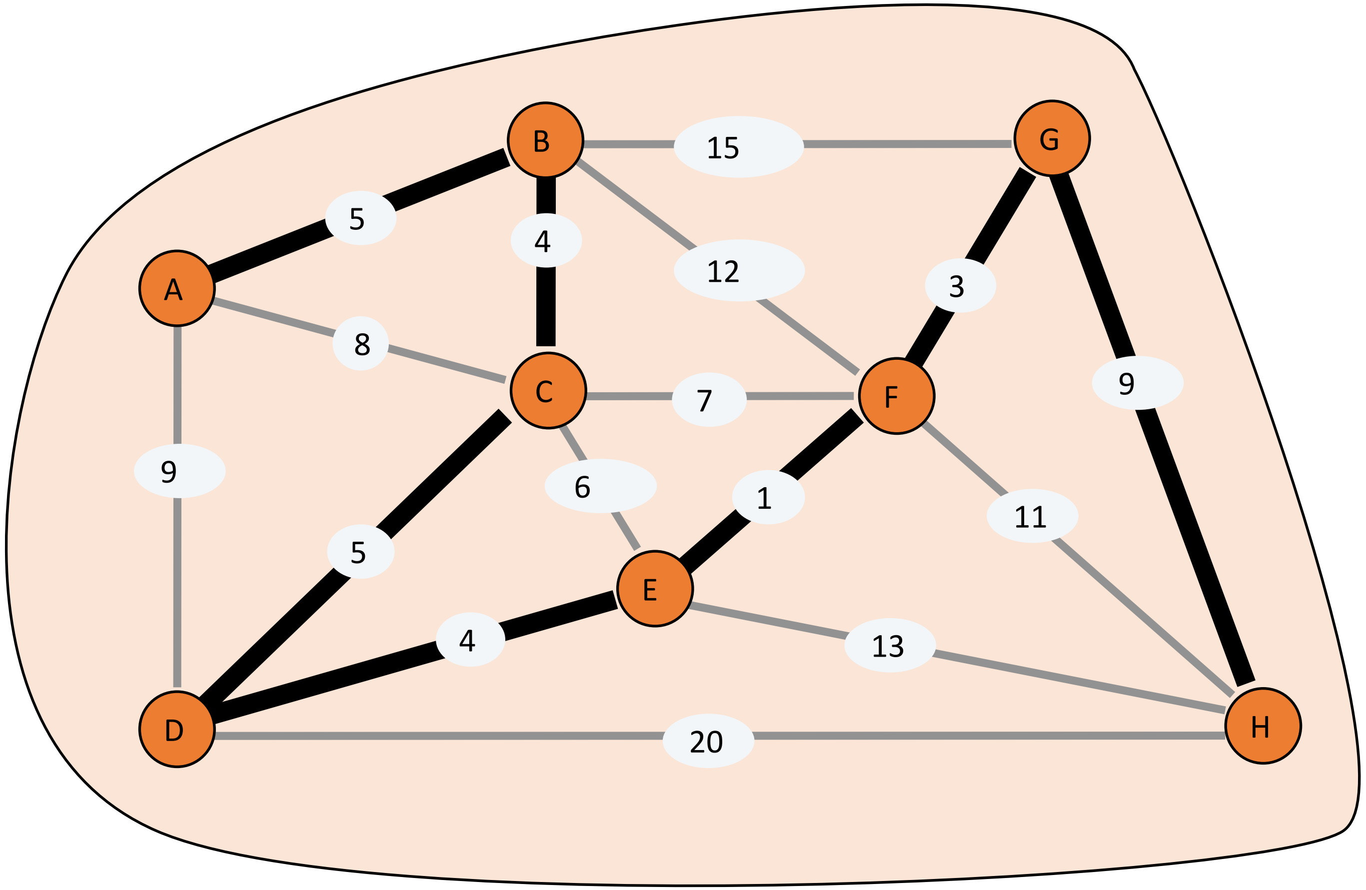
Prim’s algorithm to find MST



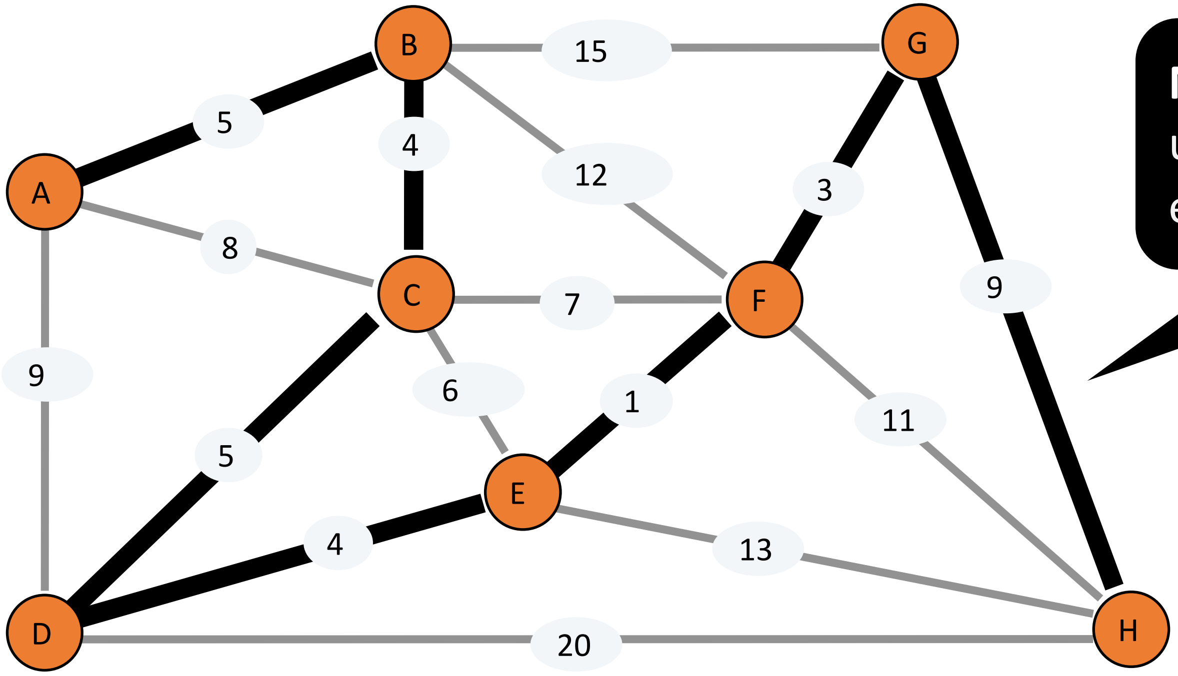
Prim’s algorithm to find MST



Prim’s algorithm to find MST



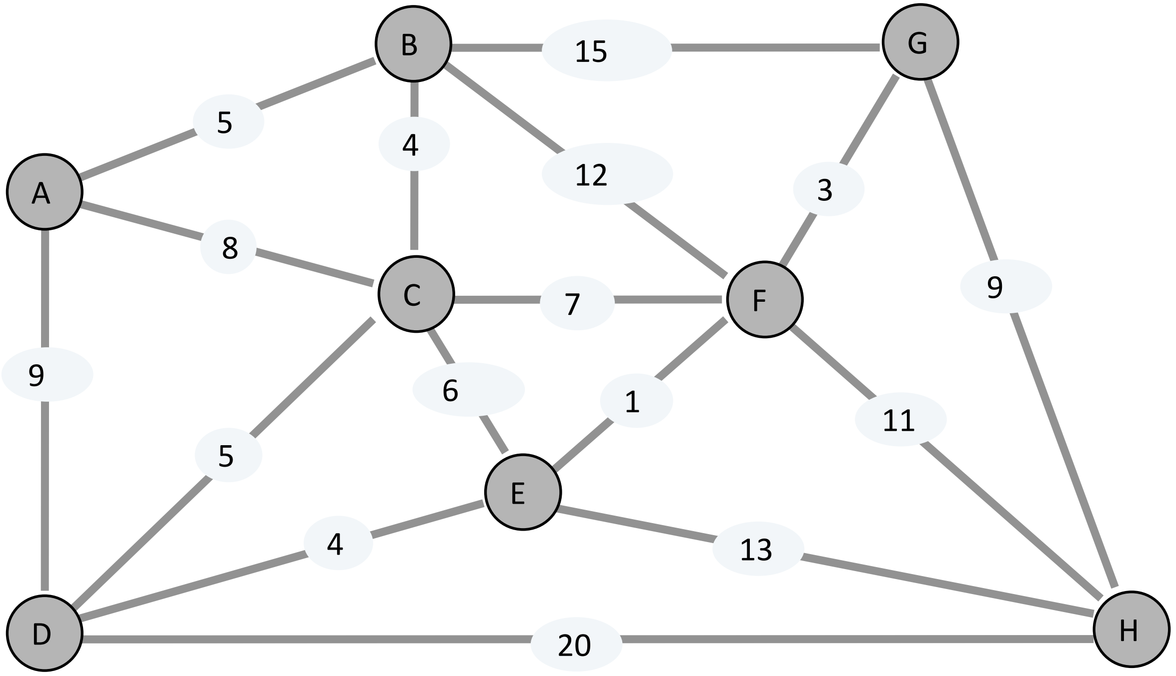
Prim’s algorithm Time Complexity (Same as Dijkstra)



Note: MST will always be unique for each graph given edge **weights are unique**

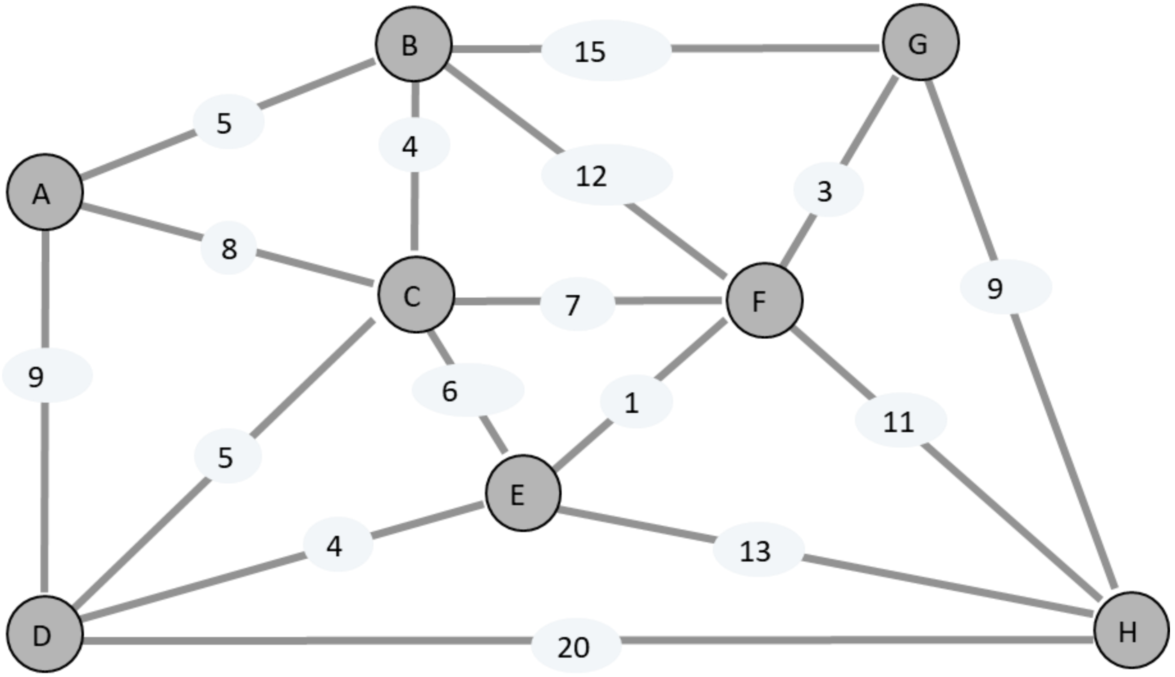
Kruskal’s algorithm to find MST

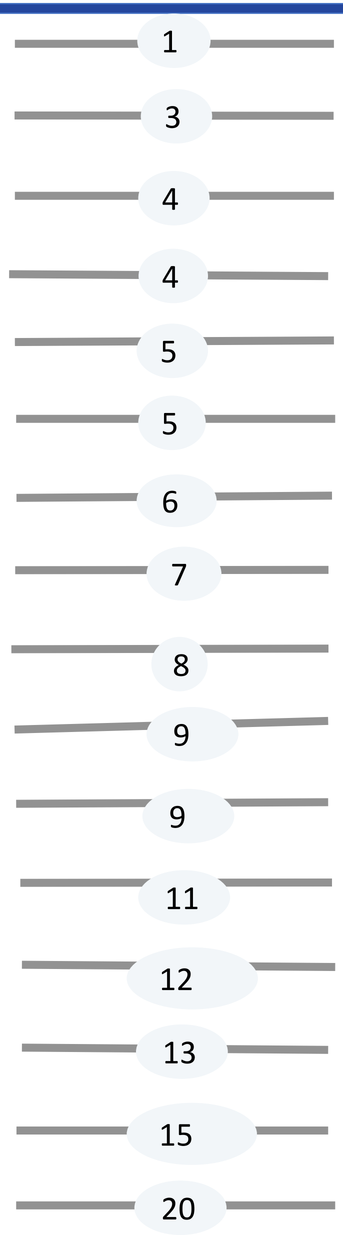
- Consider edges in ascending order of weight



Kruskal’s algorithm to find MST

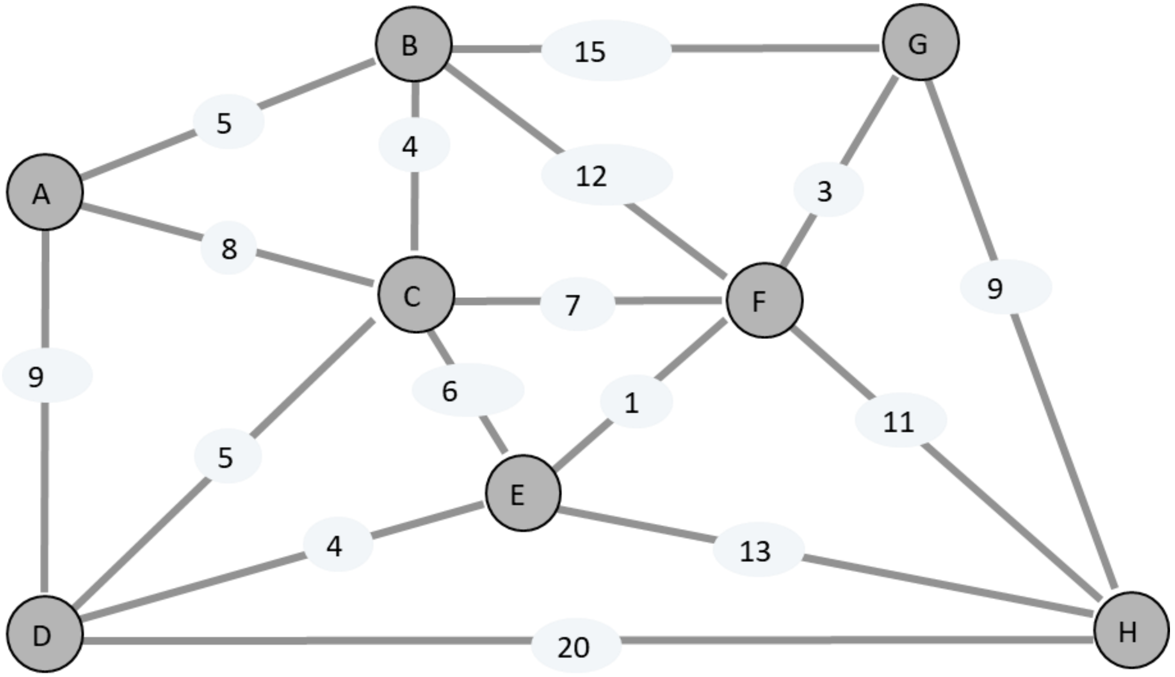
- Consider edges in ascending order of weight

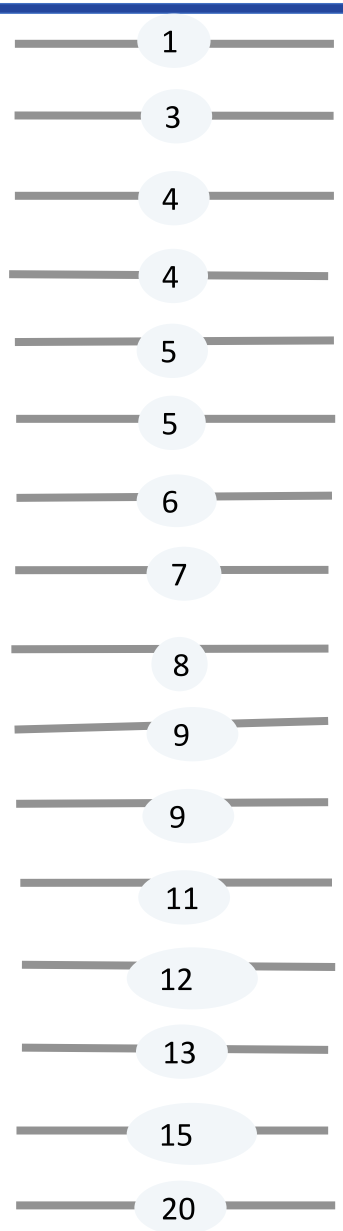




Kruskal's algorithm to find MST

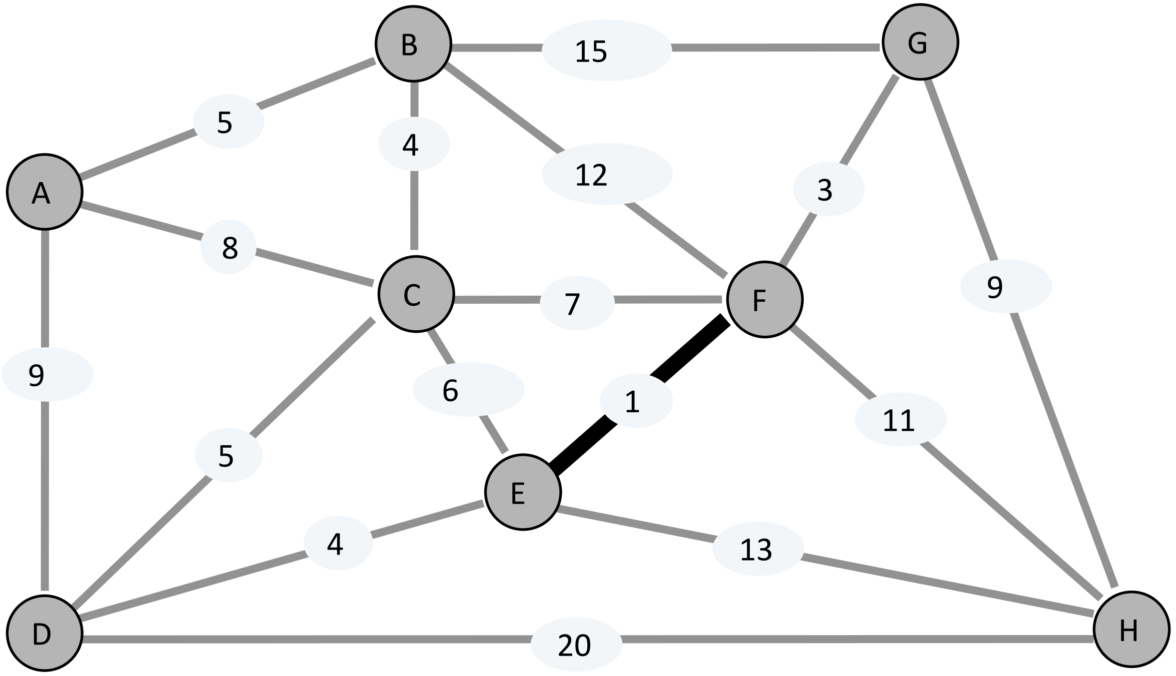
- Consider edges in ascending order of weight
- Add to e to MST unless it would create a cycle

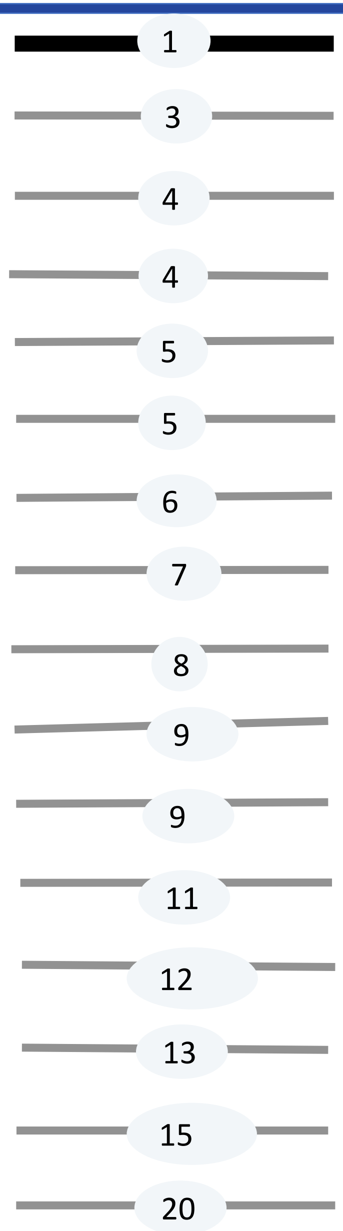




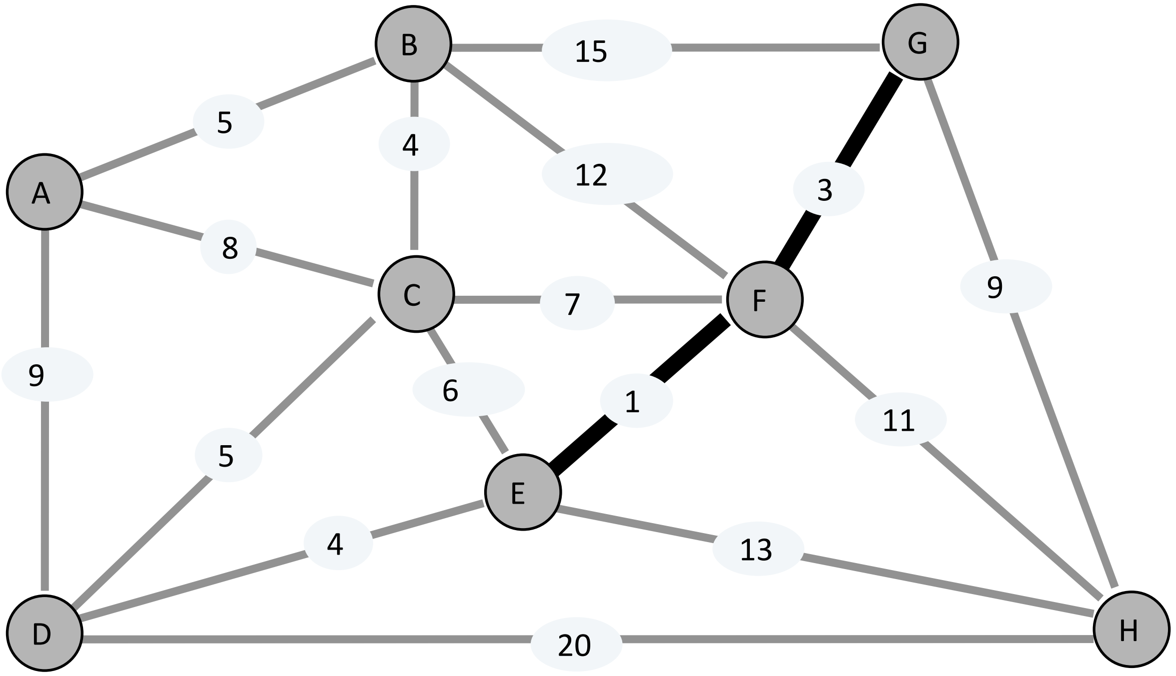
Kruskal's algorithm to find MST

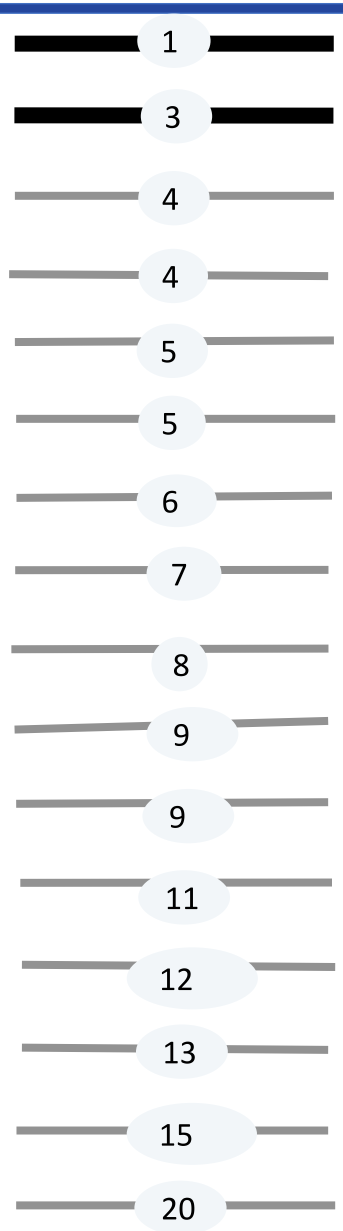
- Consider edges in ascending order of weight
- Add to e to MST unless it would create a cycle



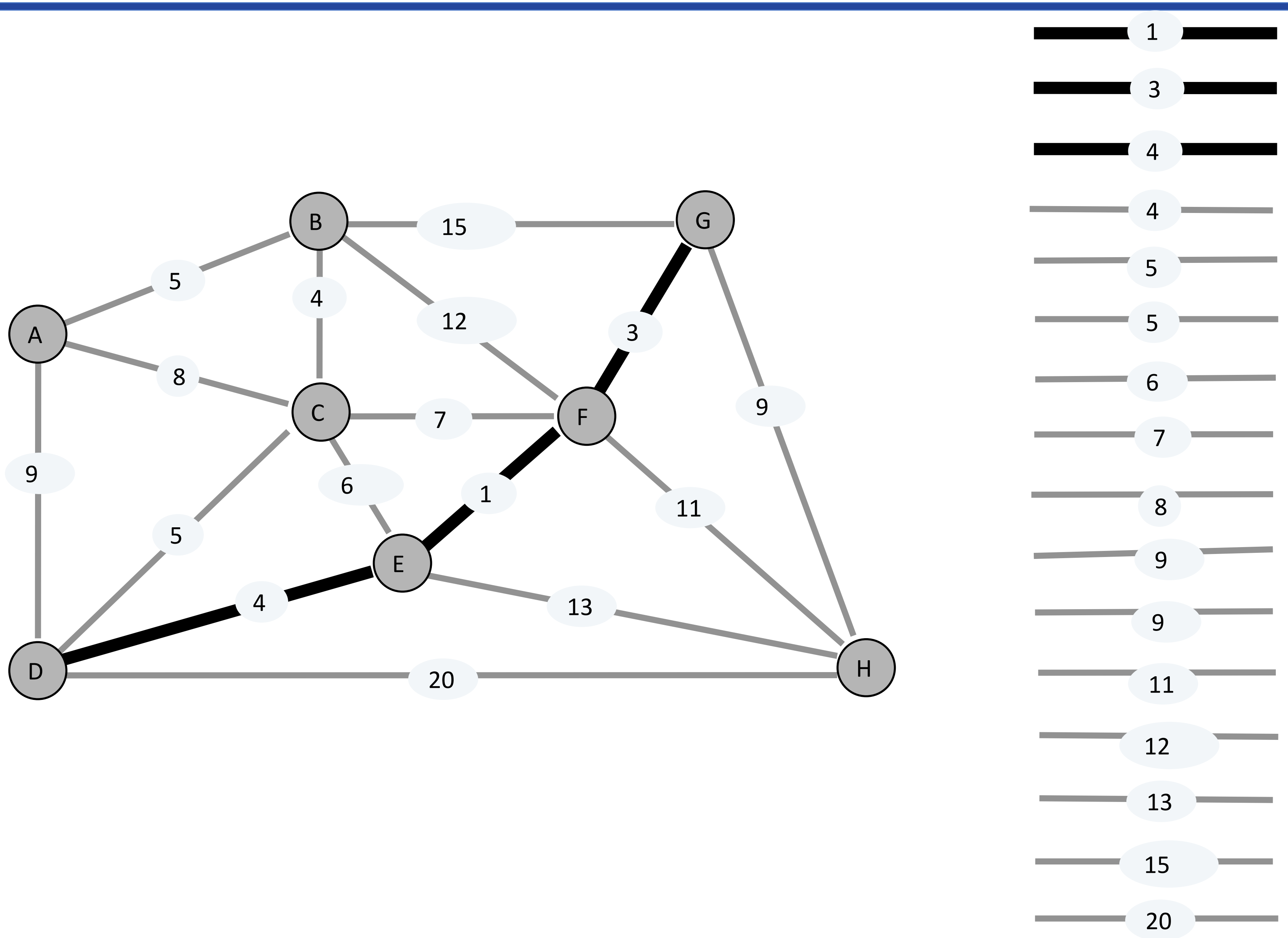


Kruskal’s algorithm to find MST

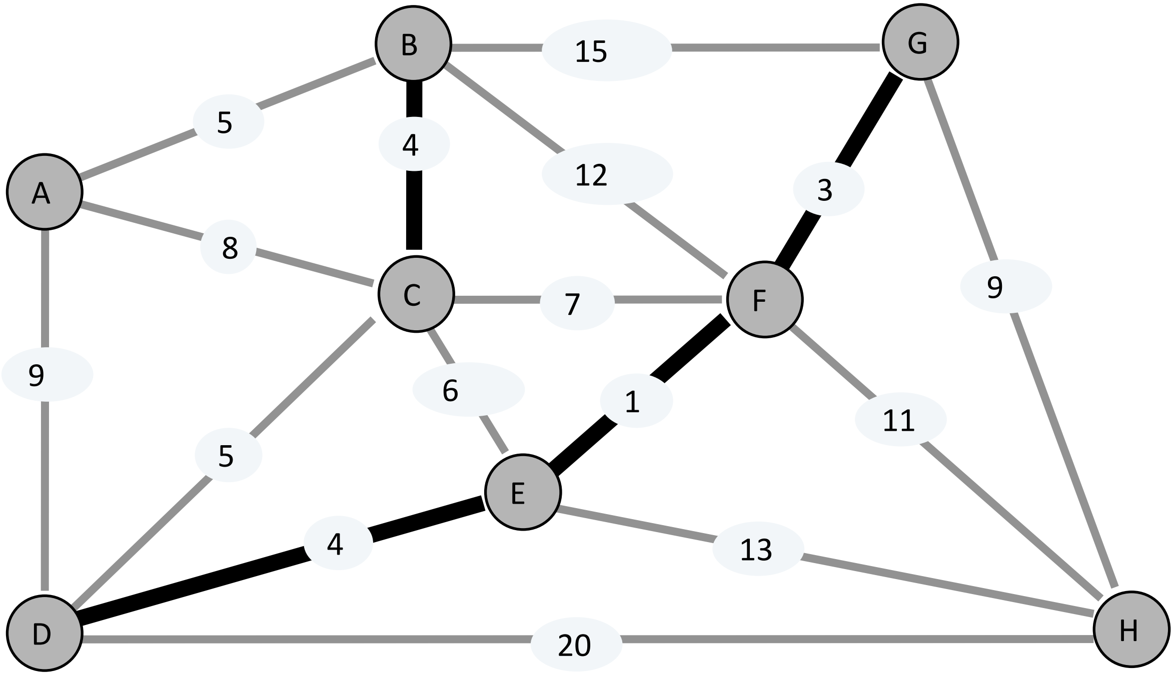


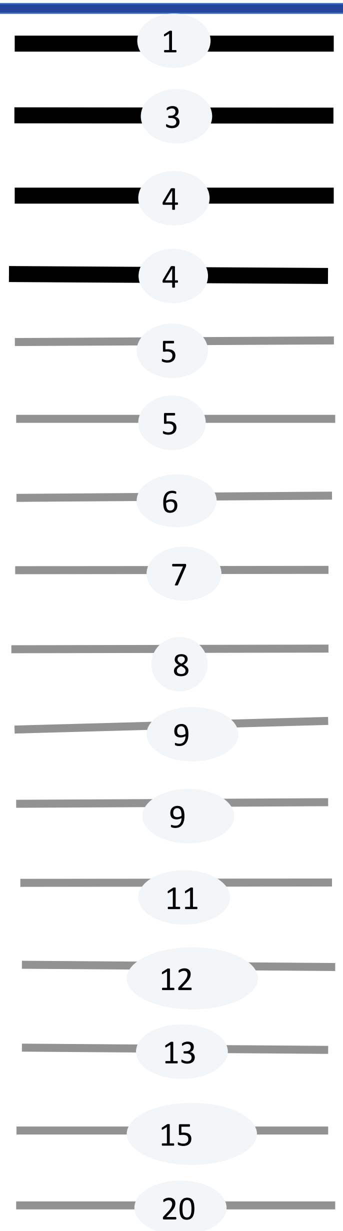


Kruskal’s algorithm to find MST

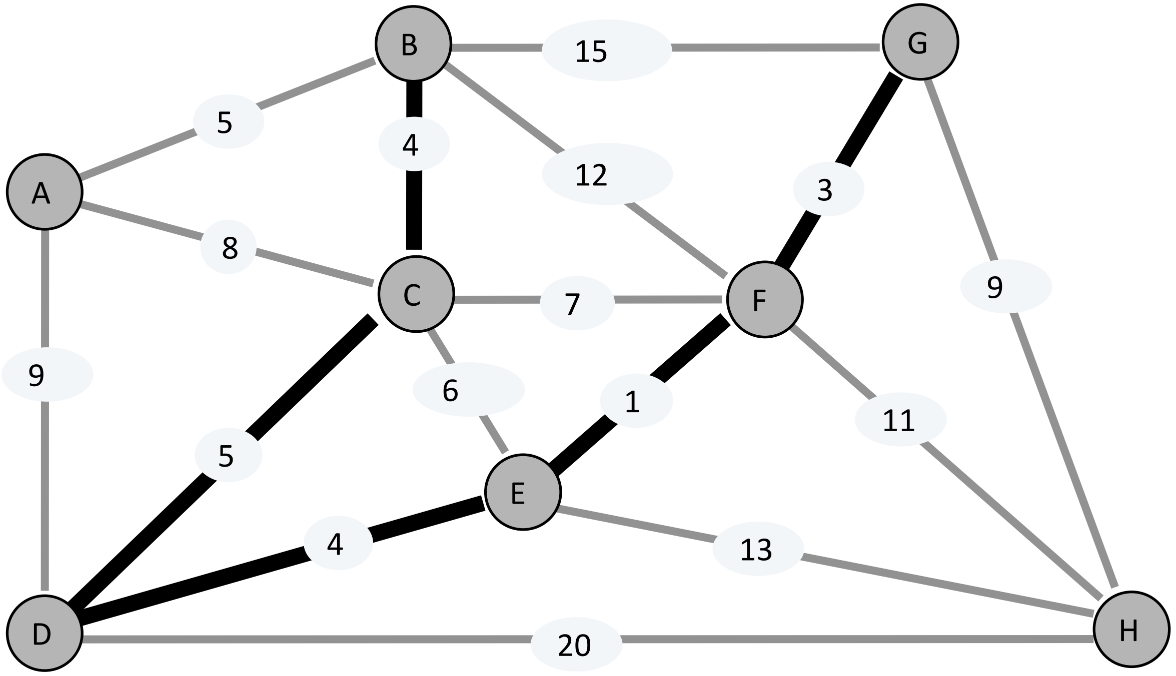


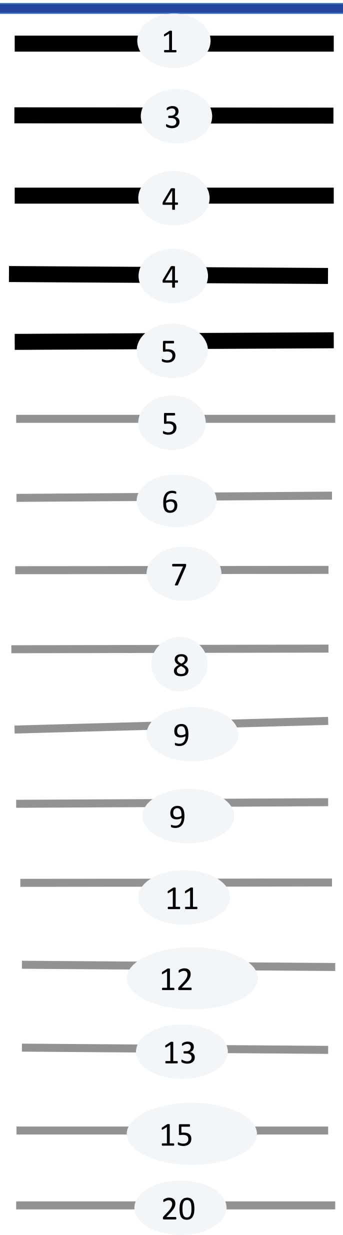
Kruskal's algorithm to find MST



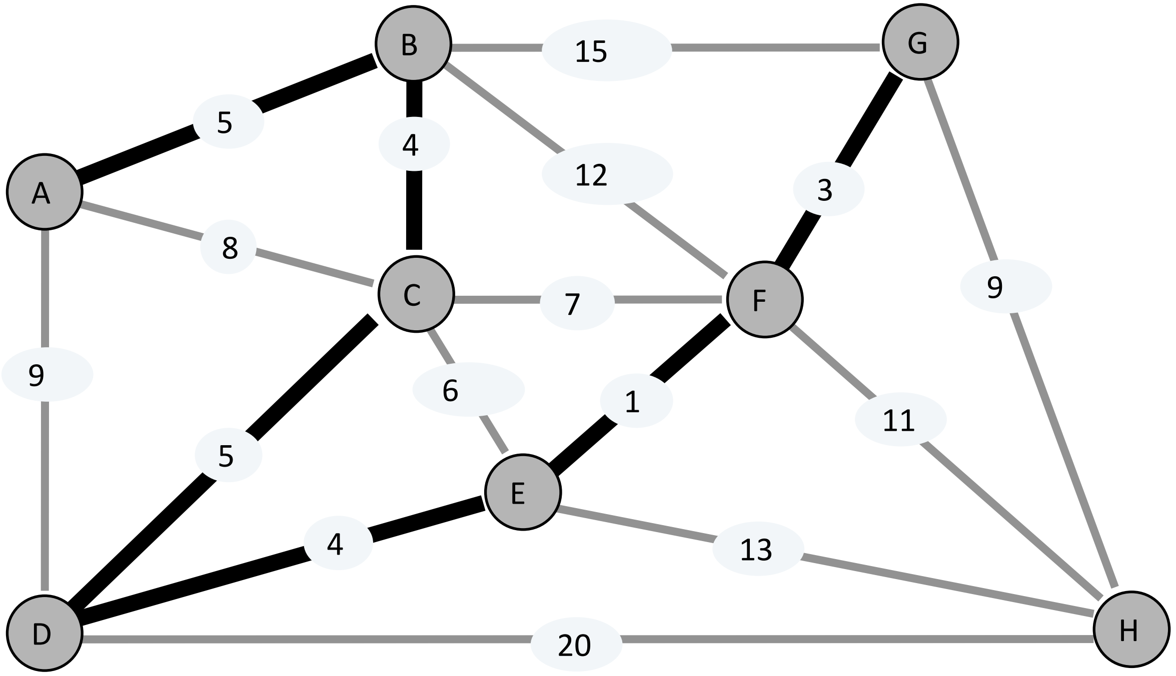


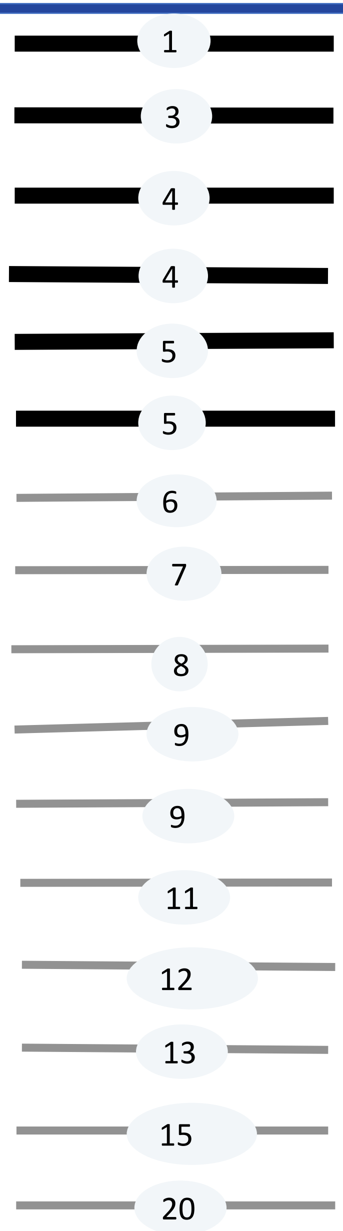
Kruskal’s algorithm to find MST



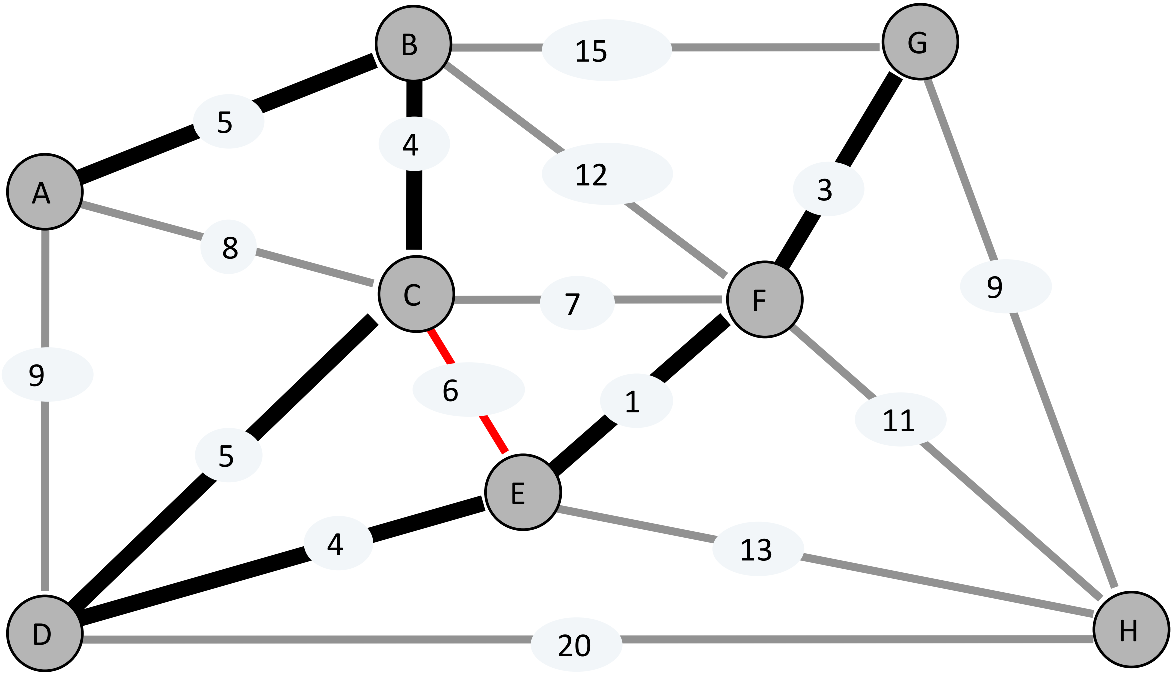


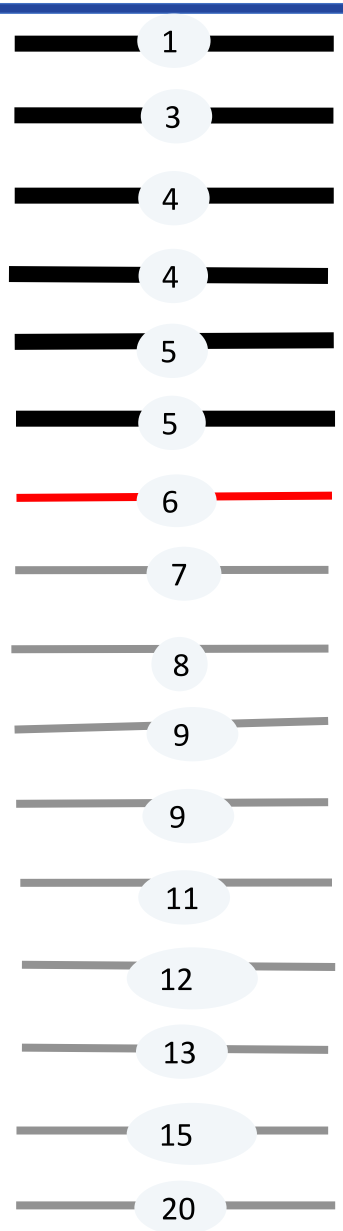
Kruskal's algorithm to find MST



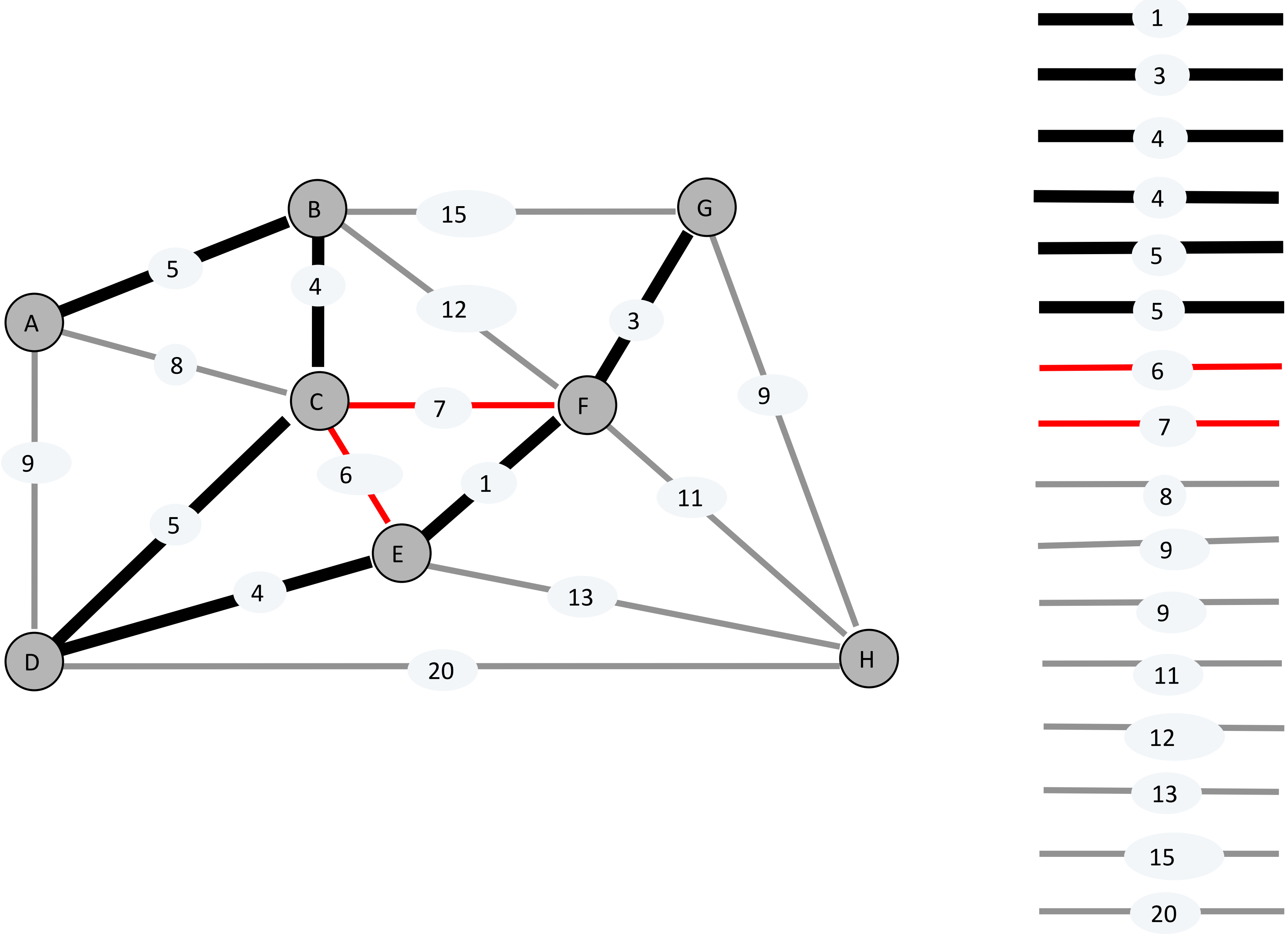


Kruskal's algorithm to find MST

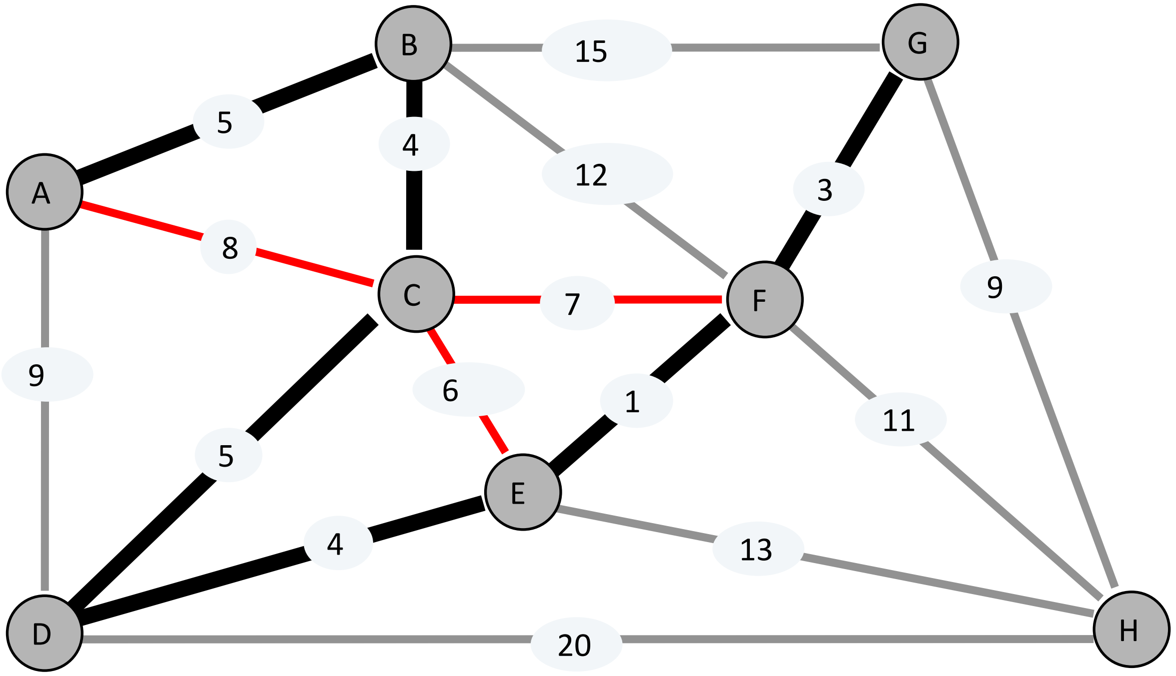


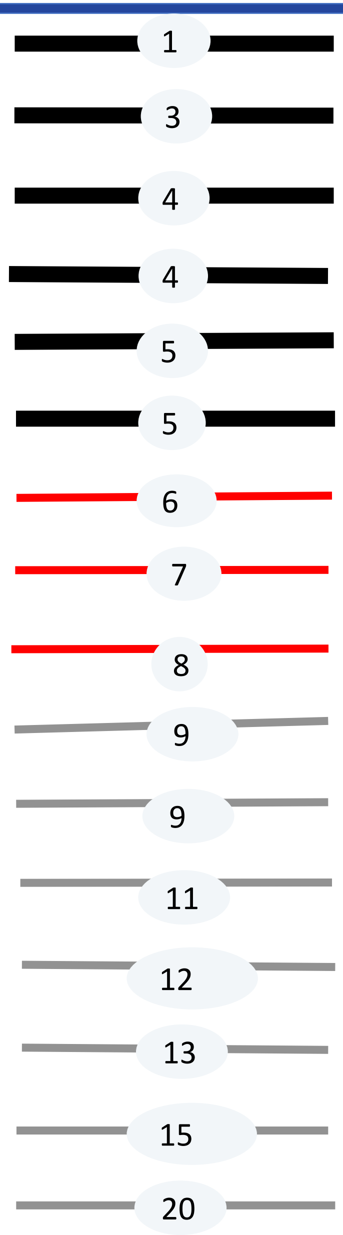


Kruskal’s algorithm to find MST

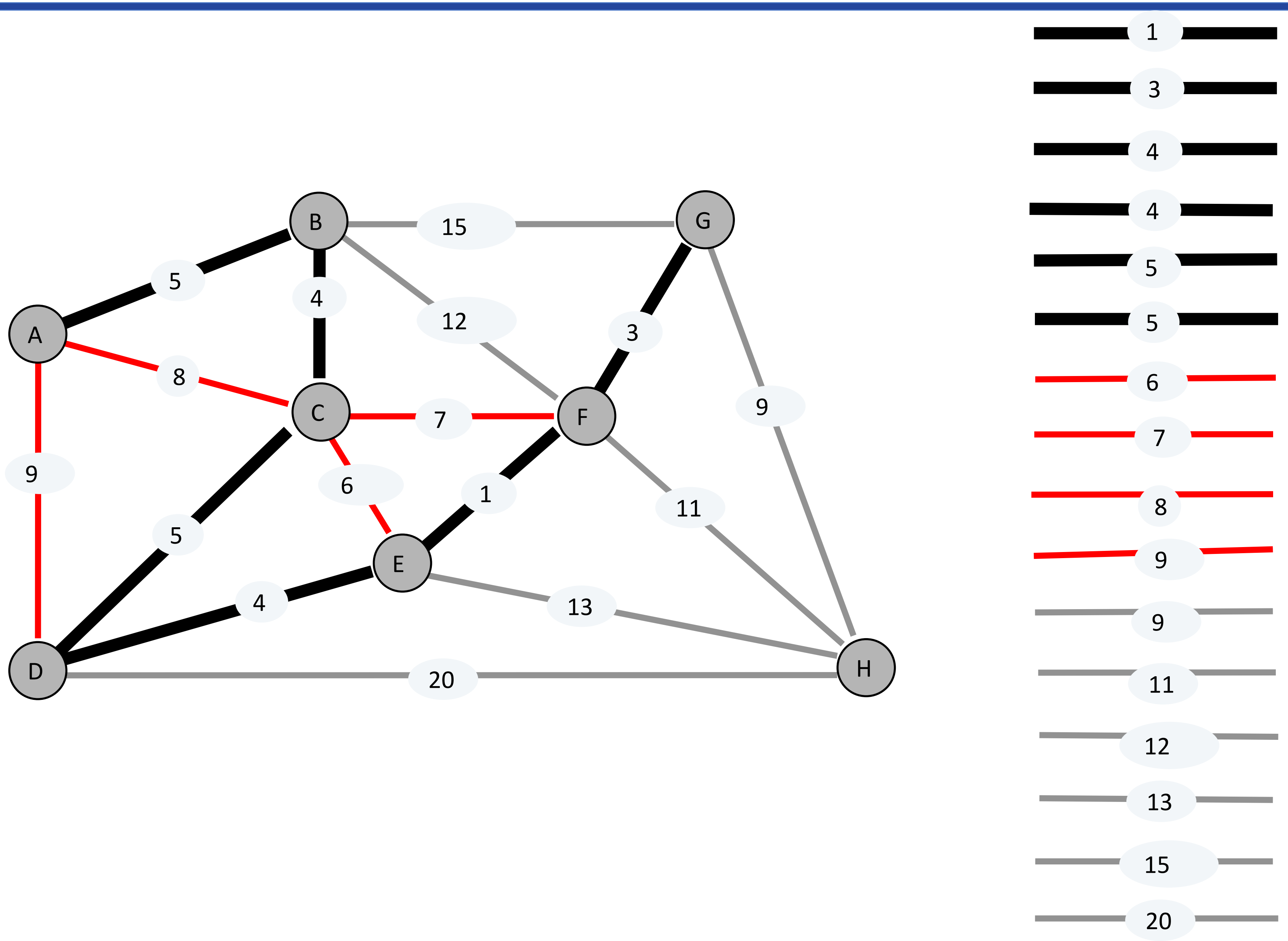


Kruskal's algorithm to find MST

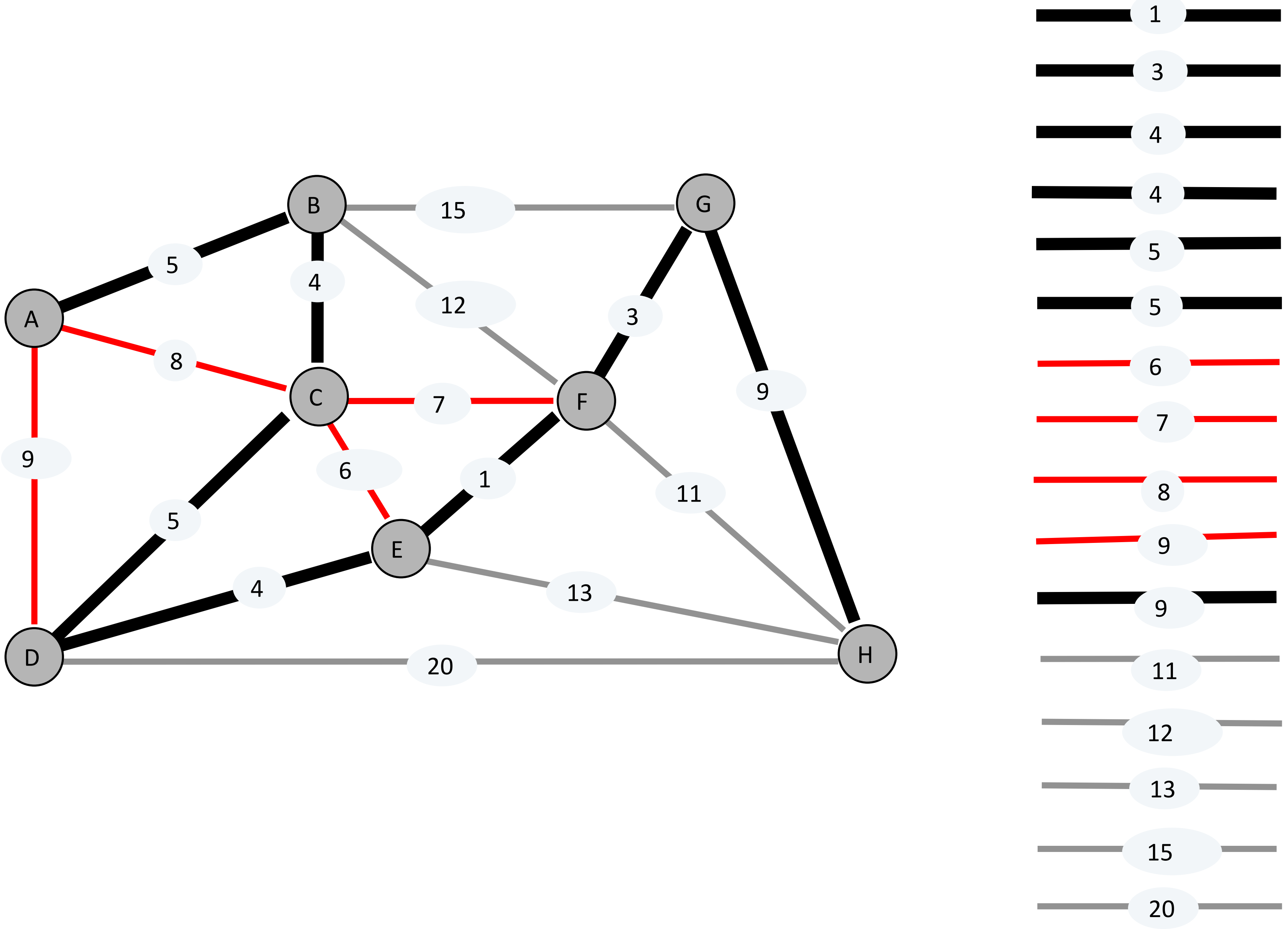




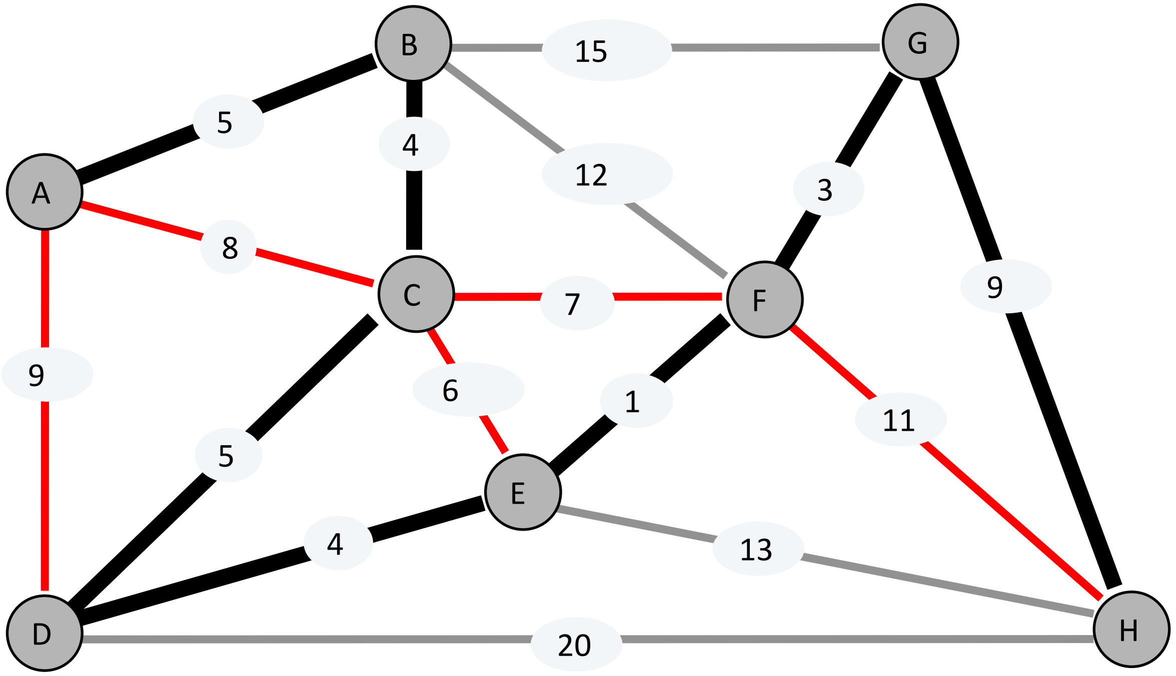
Kruskal’s algorithm to find MST

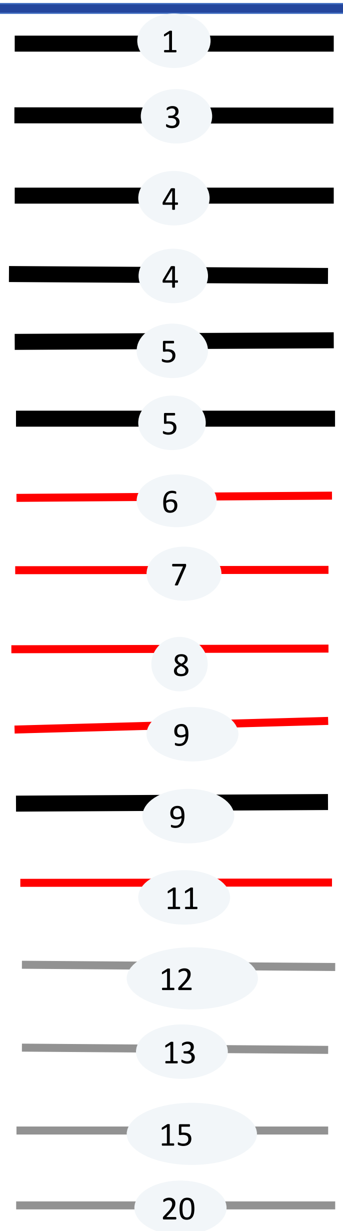


Kruskal’s algorithm to find MST

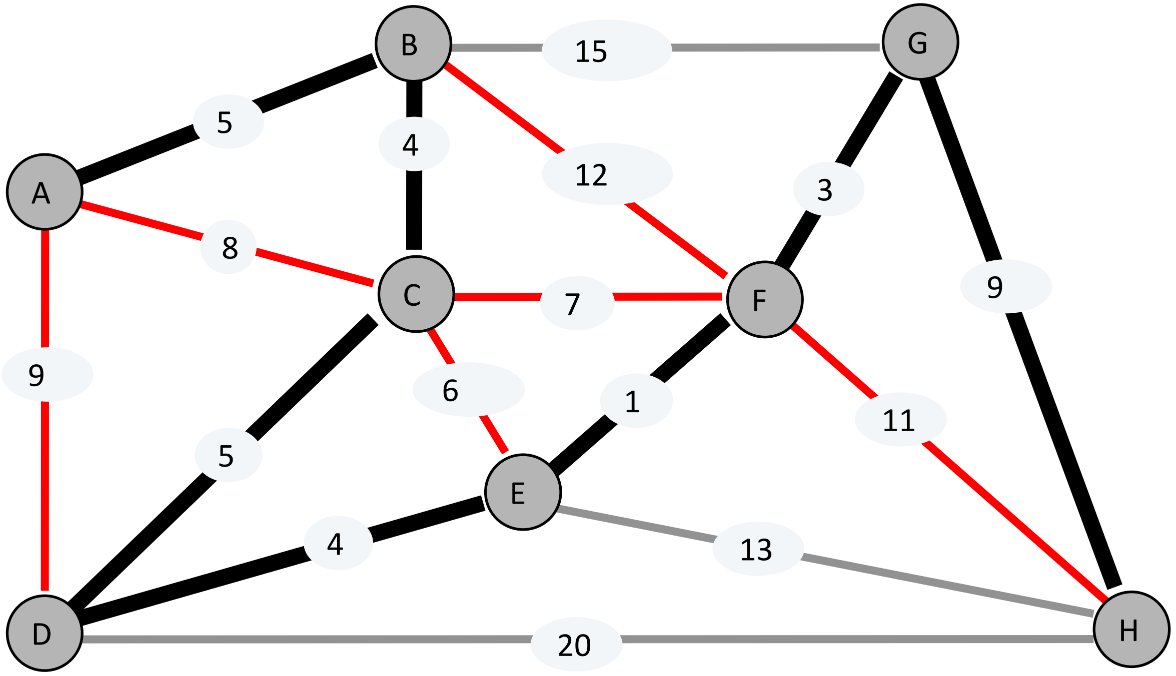


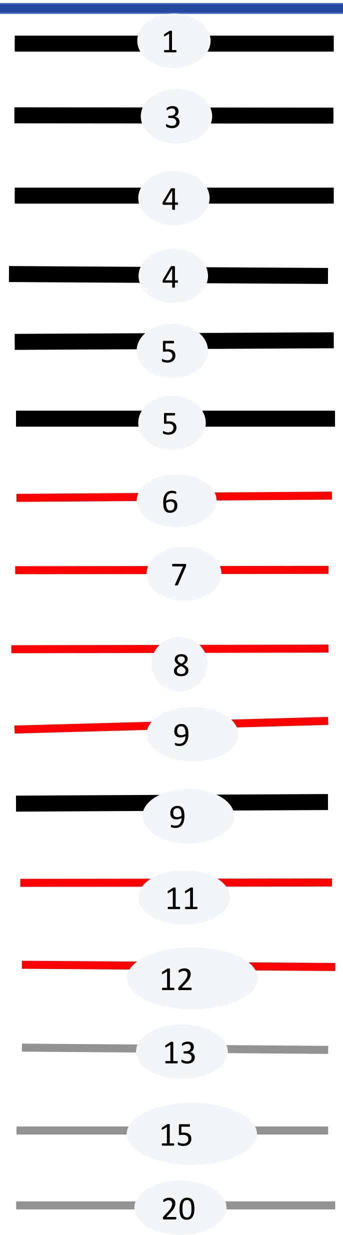
Kruskal's algorithm to find MST



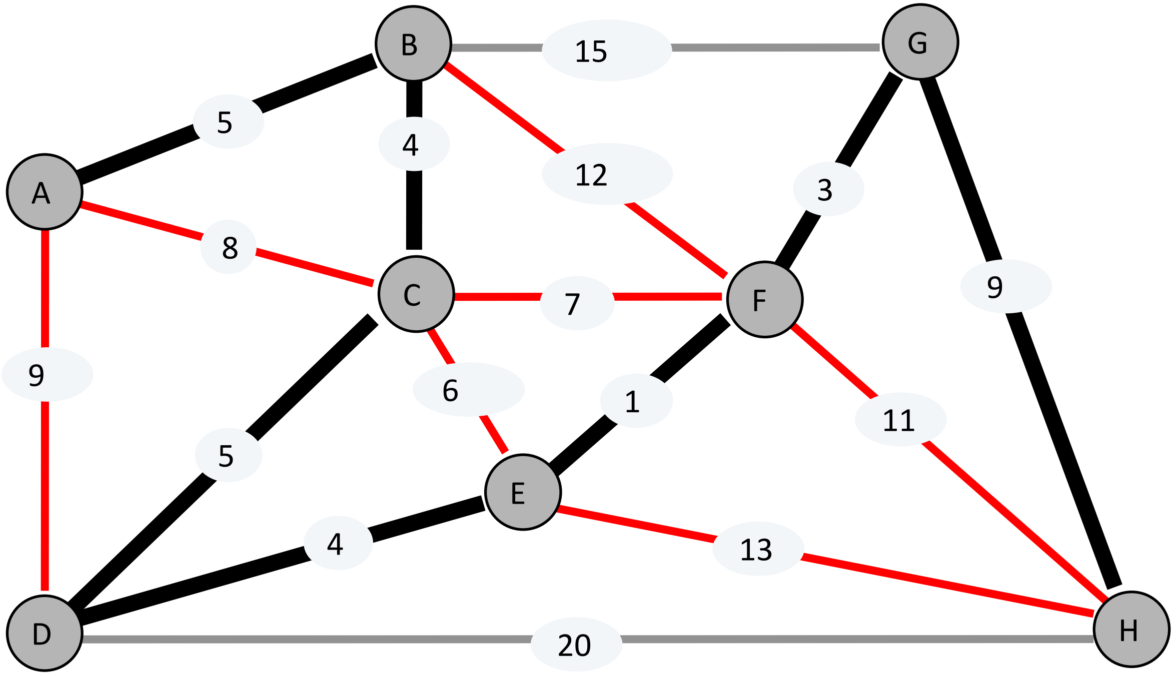


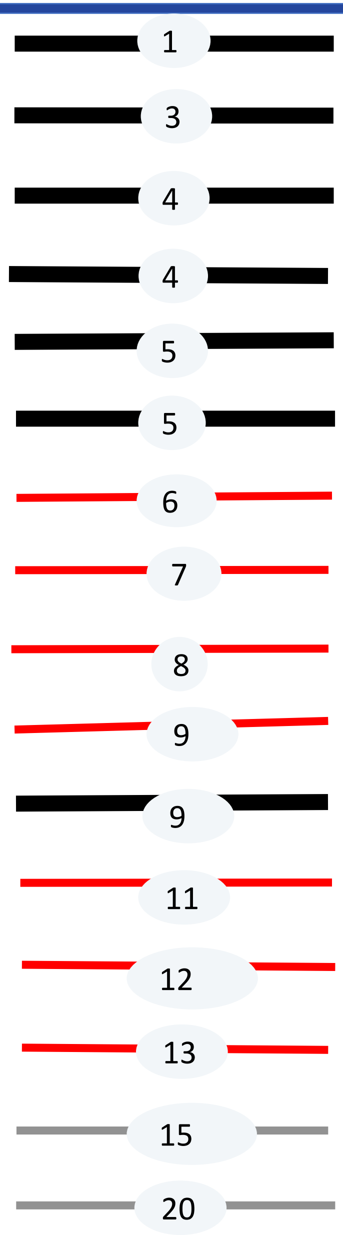
Kruskal's algorithm to find MST



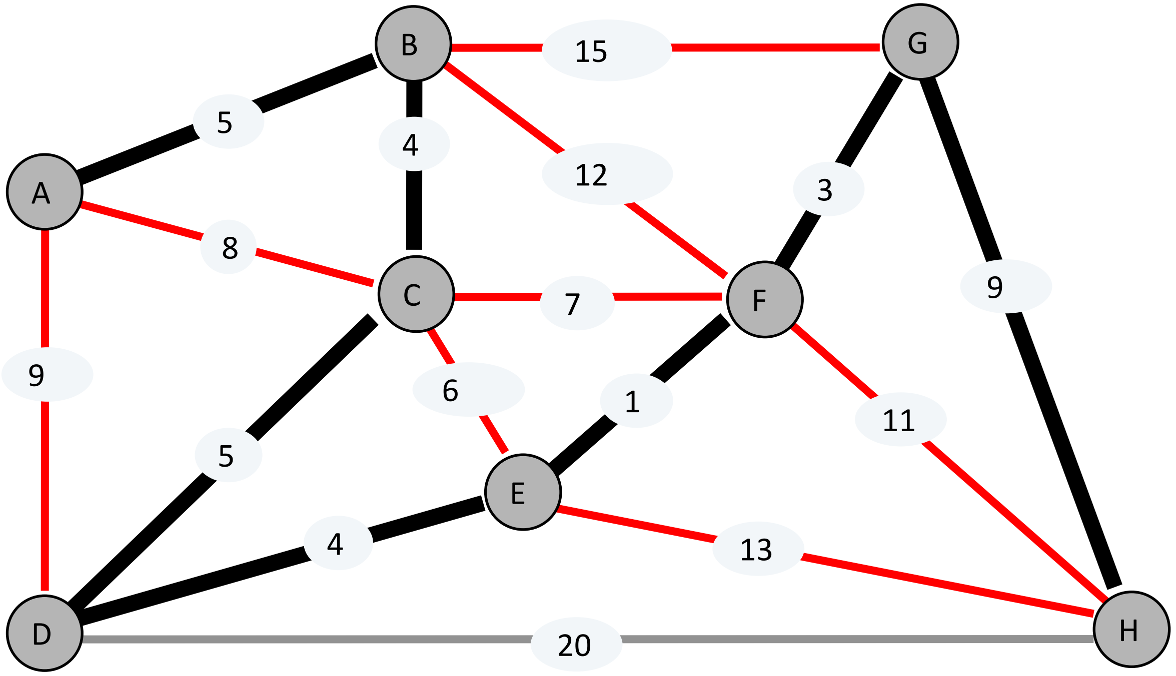


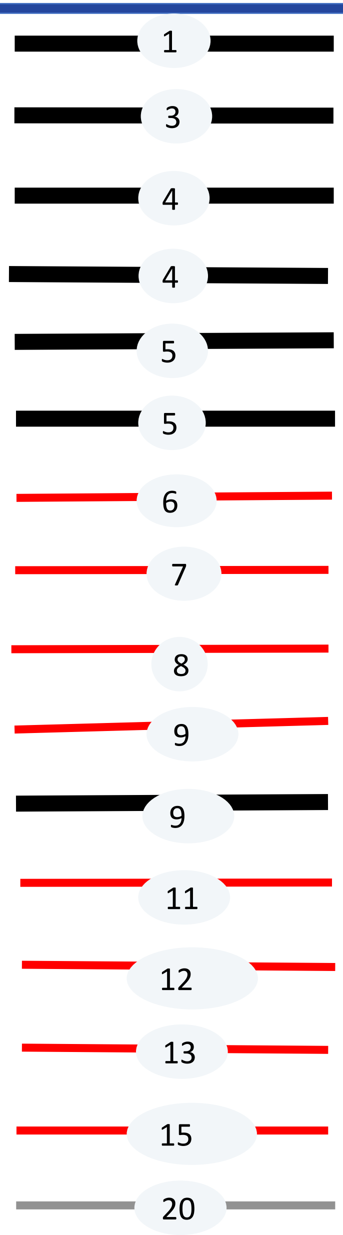
Kruskal’s algorithm to find MST



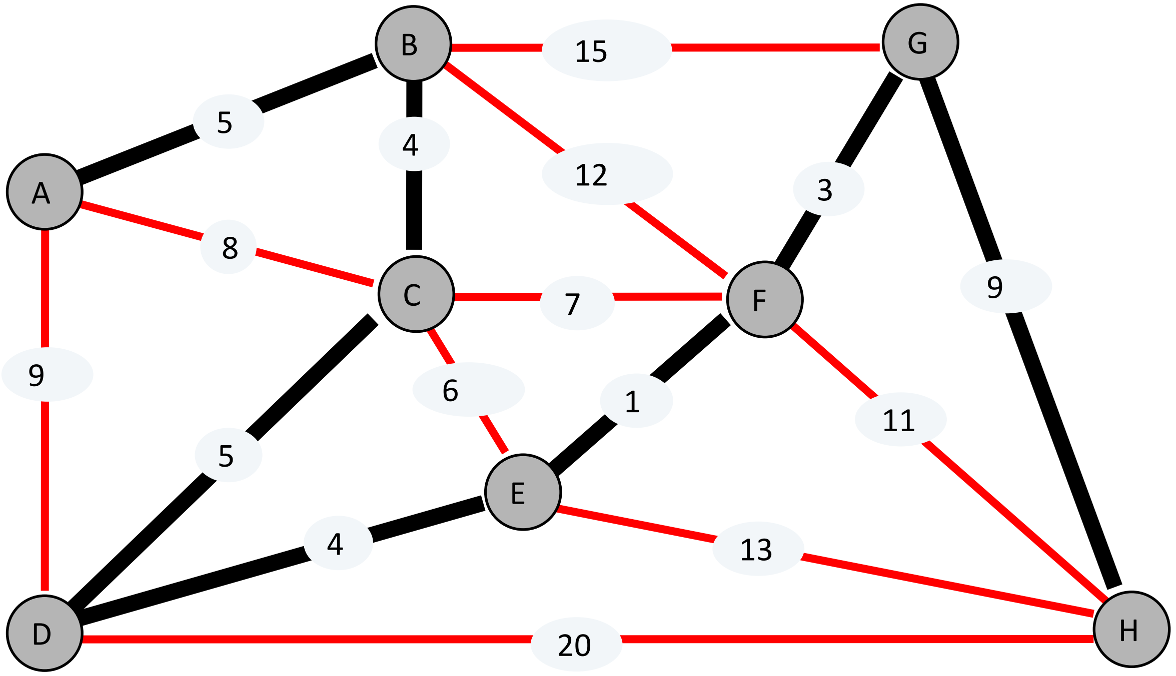


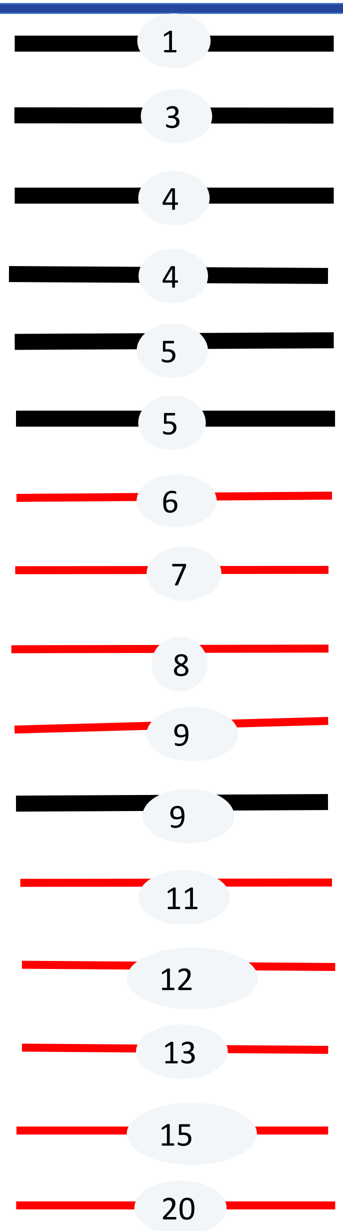
Kruskal's algorithm to find MST



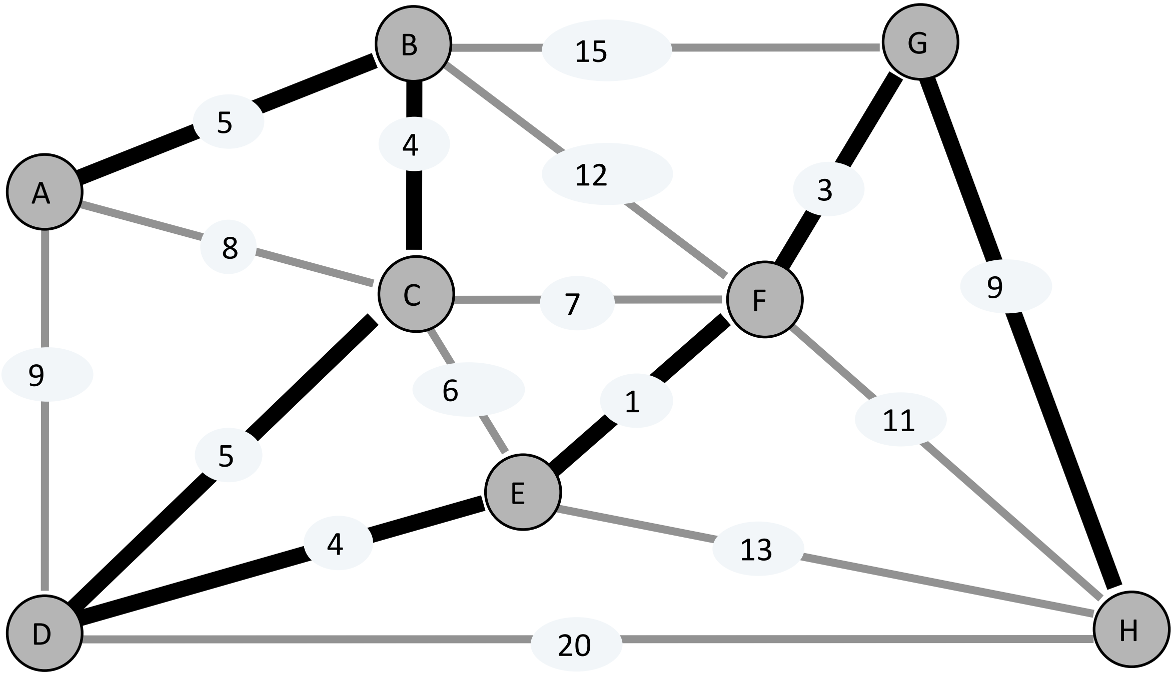


Kruskal’s algorithm to find MST

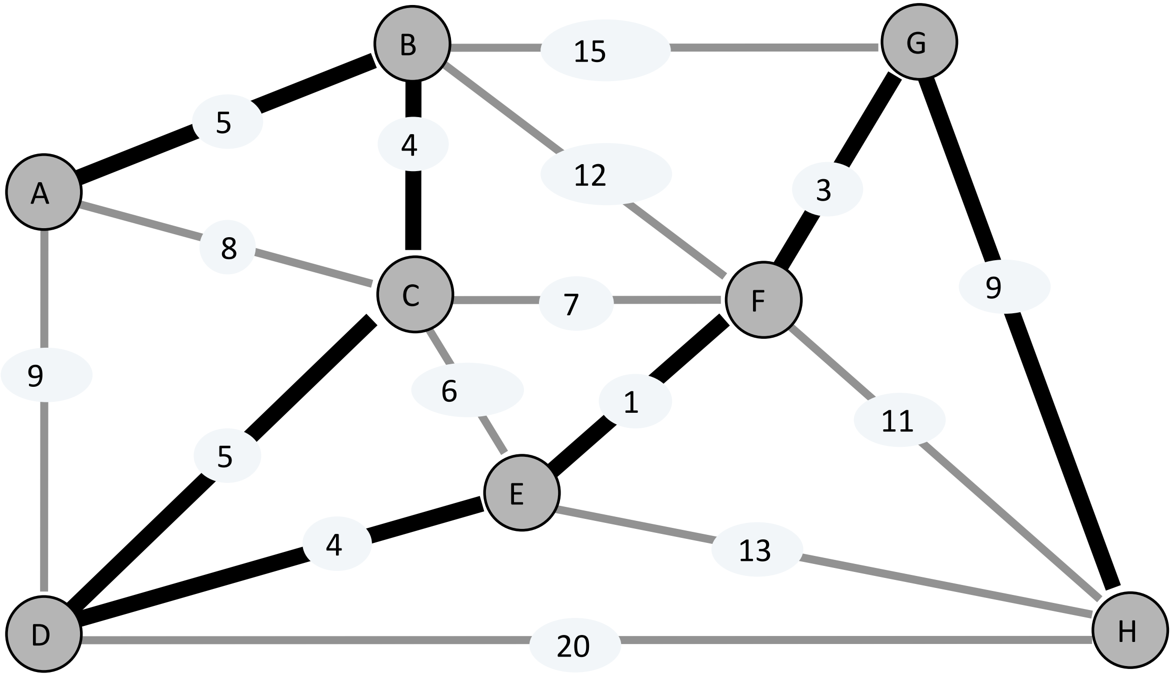




Kruskal's algorithm to find MST

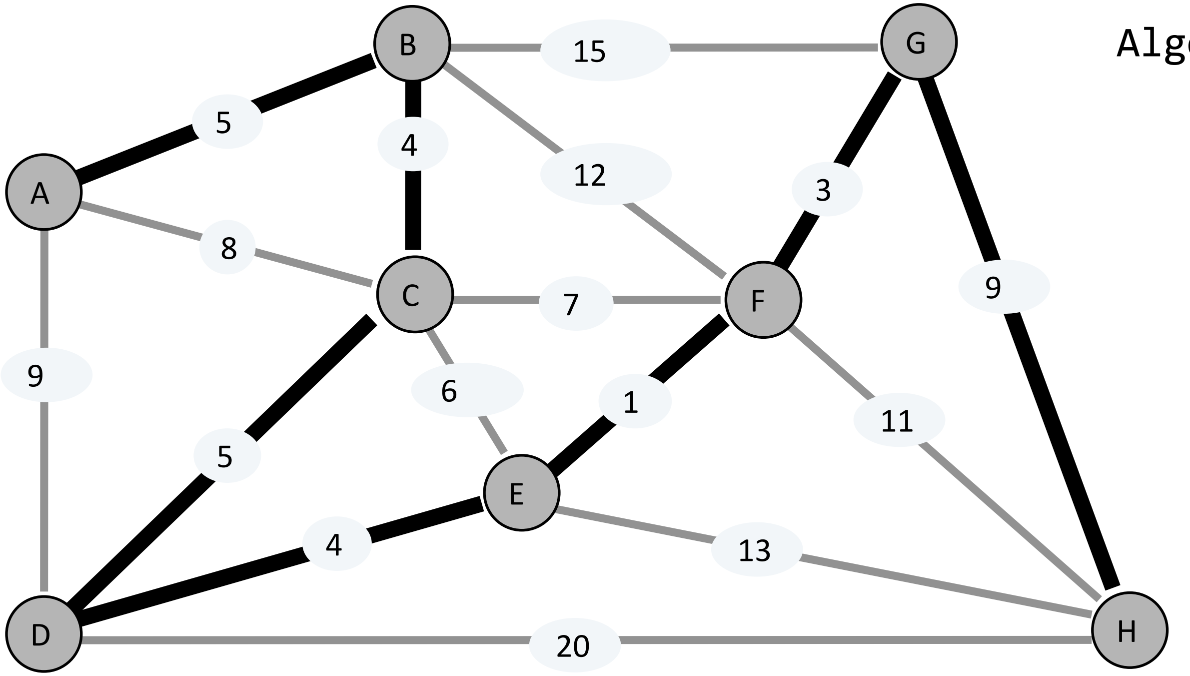


Kruskal’s algorithm to find MST



- Sort edges in ascending order of weight.
- Check for each edge
    - **Case 1:** If adding ***e*** to ***MST*** creates a cycle, discard e according to cycle property.
    - **Case 2:** Otherwise, insert ***e = (u, v)*** into ***MST***

Kruskal's algorithm (using BFS/DFS)



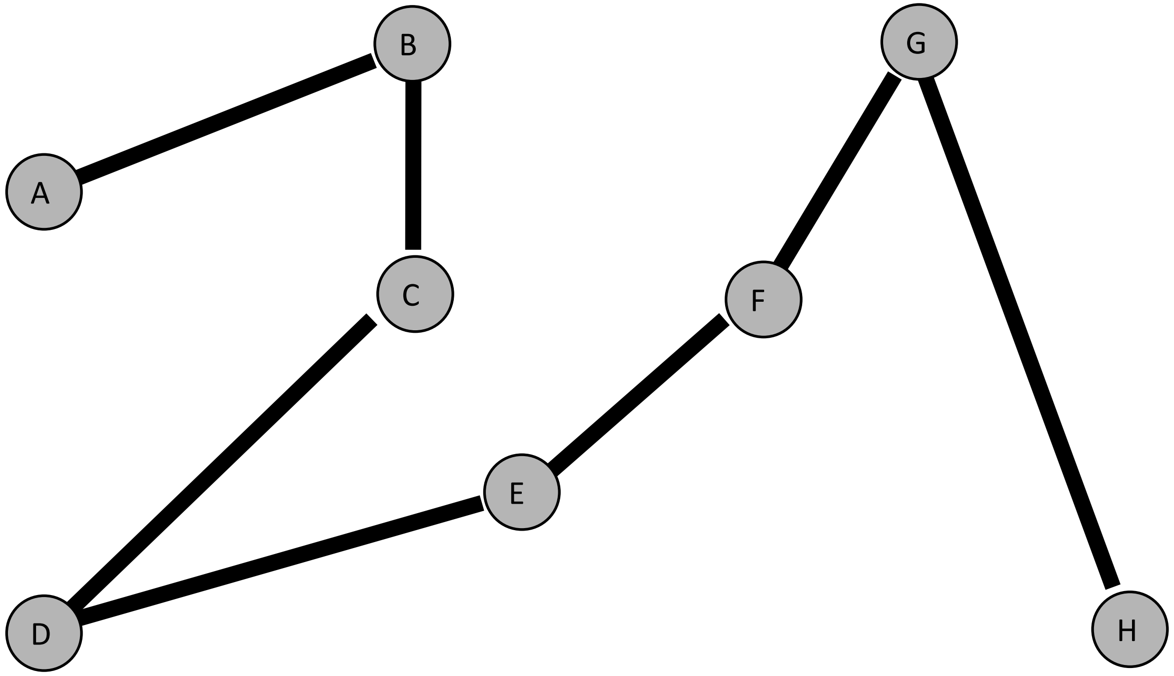

```
Algorithm Kruskal(G):

    MST ← empty set
    Sort edges E by weight in non-decreasing order

    for each edge e = (u, v) in E do
        if not BFS(u, v, MST) then
            Add e to MST
        end if
    end for

    return MST
```

$\boldsymbol{O(ElogE)}$

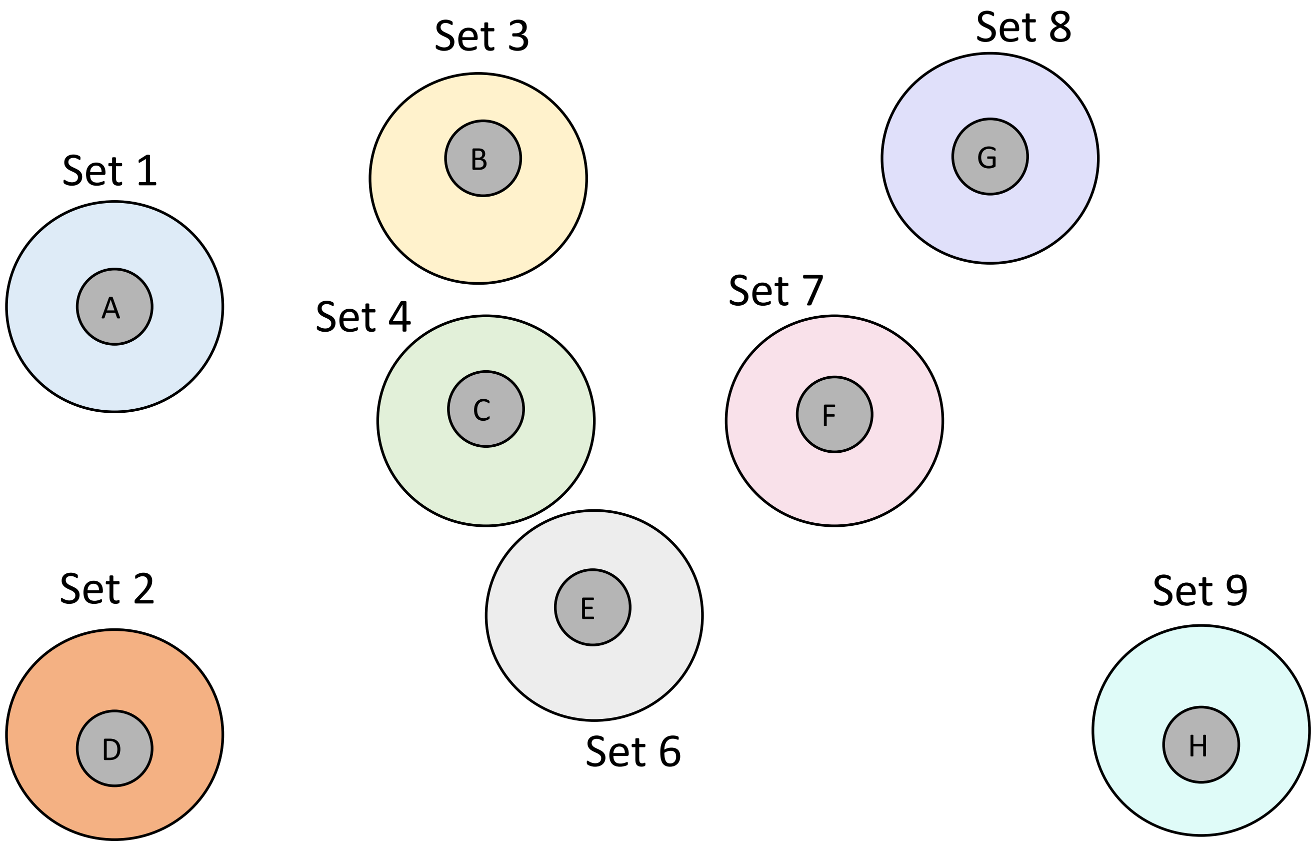
$\boldsymbol{O(V+E)}$

$\boldsymbol{O(E)}$

Kruskal's algorithm (Union-Find Method)

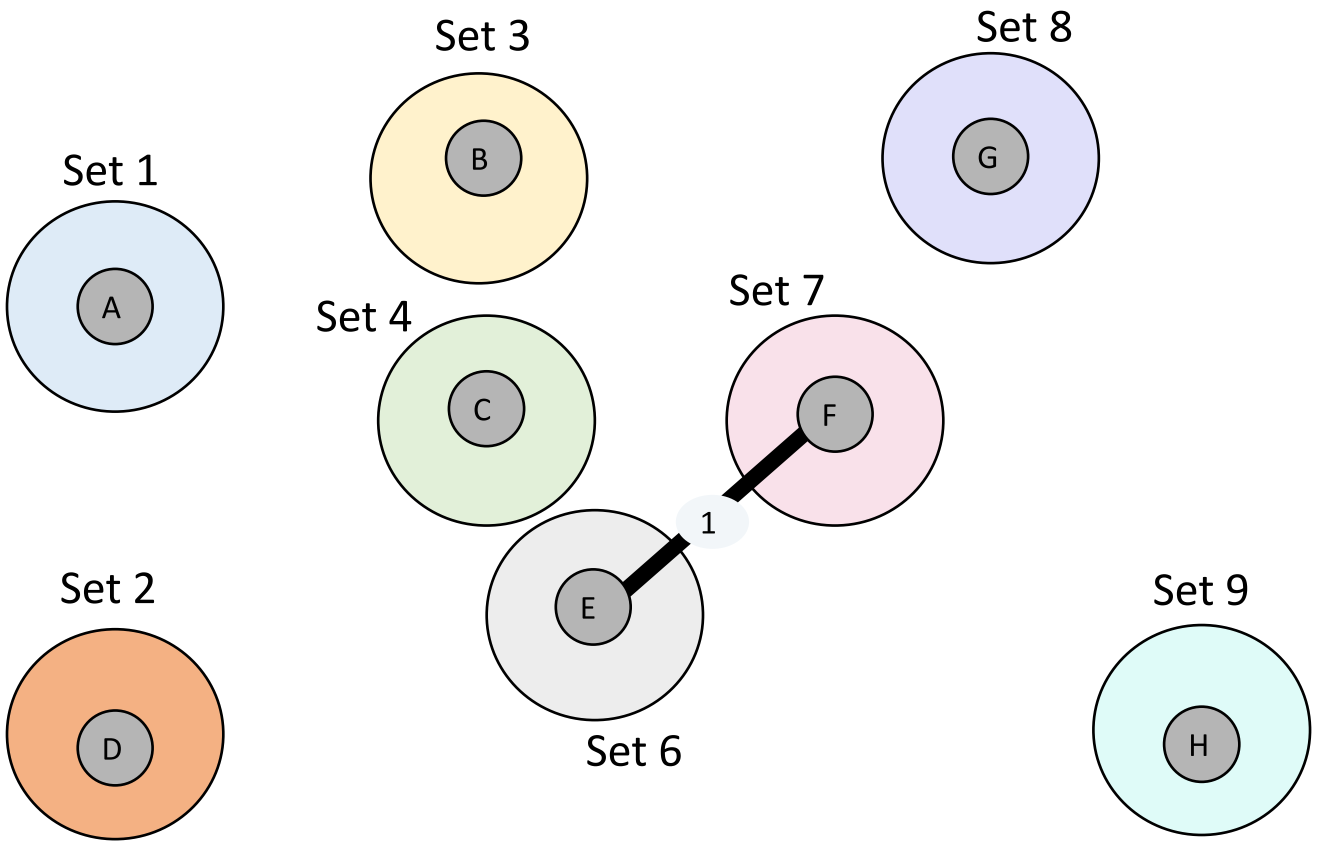


Kruskal's algorithm (Union-Find Method)



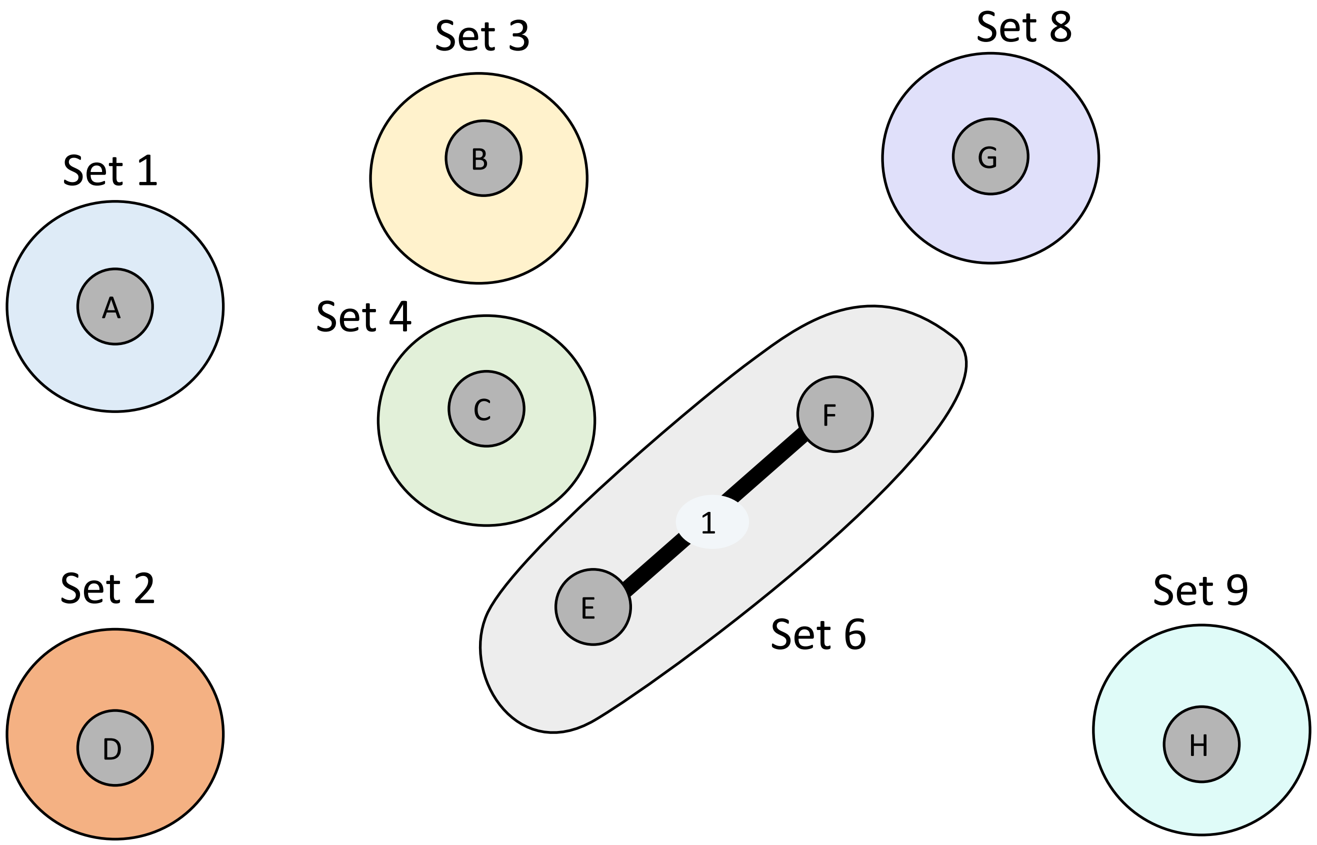
Start with a graph where each vertex is in its own separate set

Kruskal's algorithm (Union-Find Method)



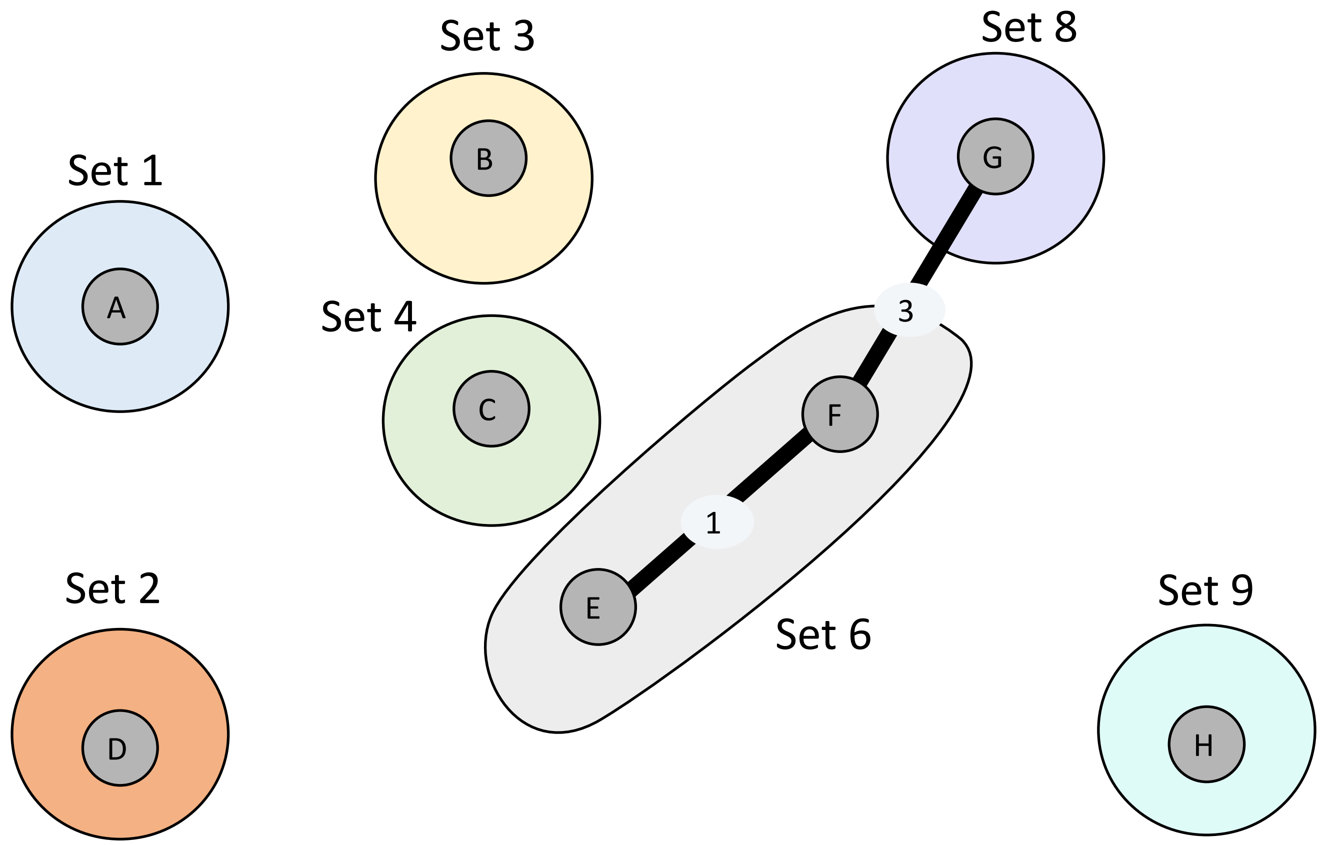
For each edge (which connects two vertices), check if the two vertices belong to the same set or not.

Kruskal's algorithm (Union-Find Method)

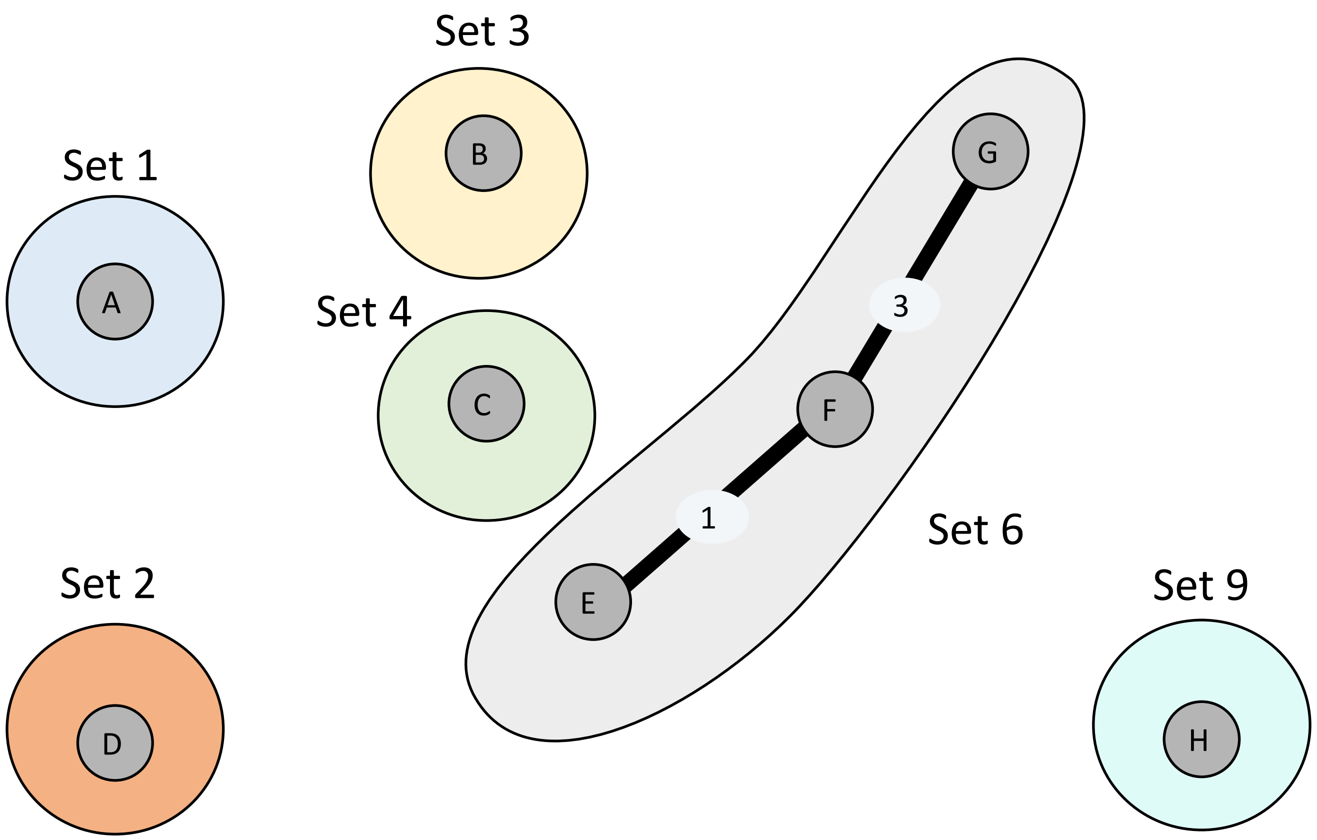


If they belong to different sets, it means the edge connects two distinct components of the graph. Perform a union operation to merge the sets of the two vertices

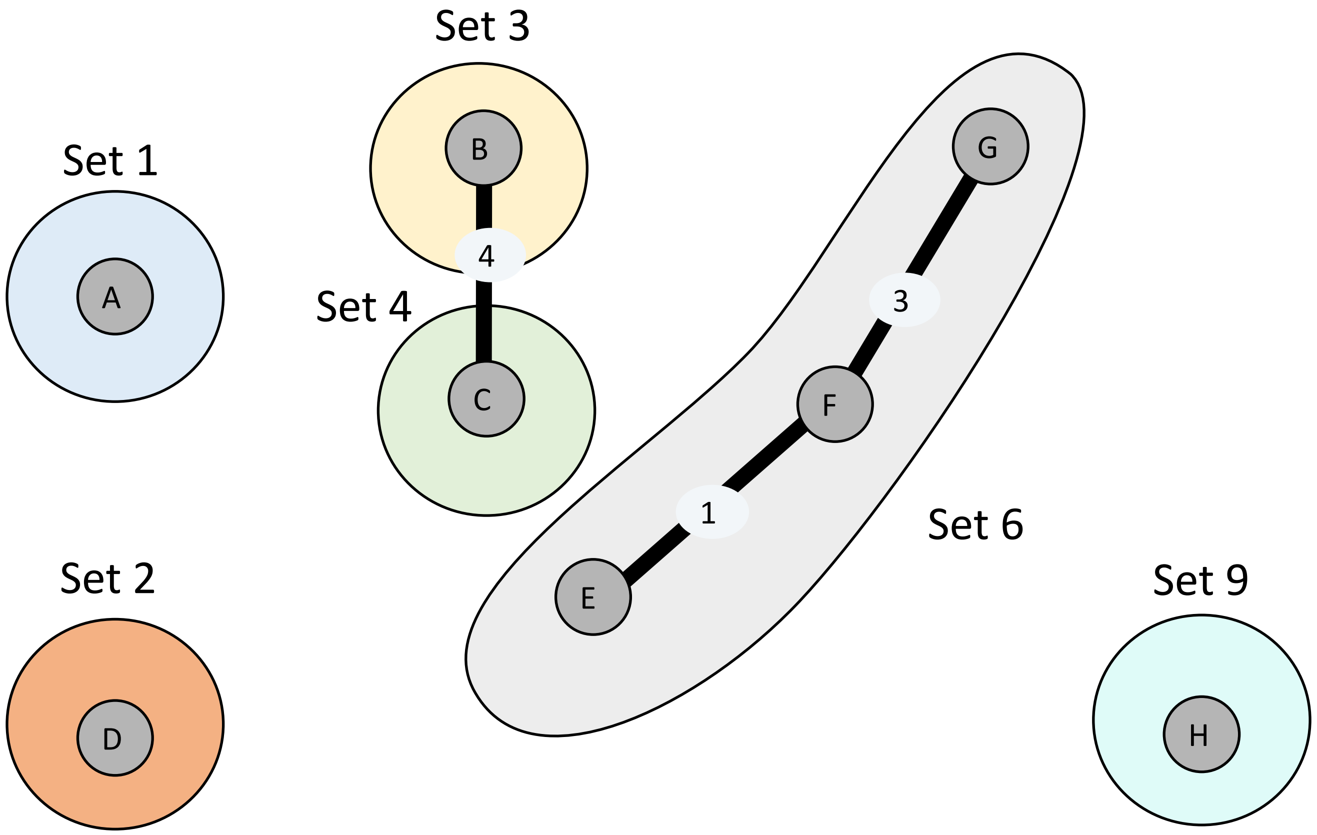
Kruskal's algorithm (Union-Find Method)



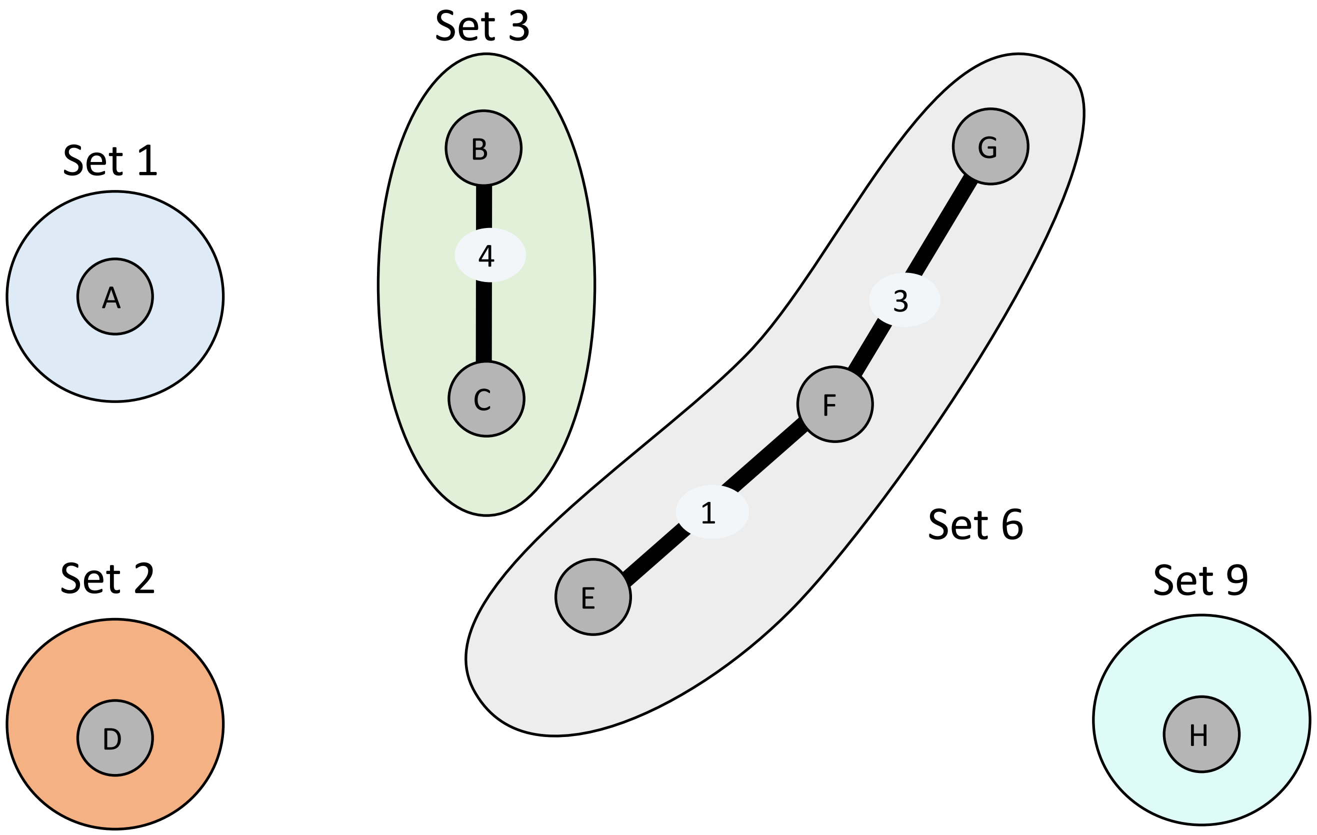
Kruskal’s algorithm (Union-Find Method)



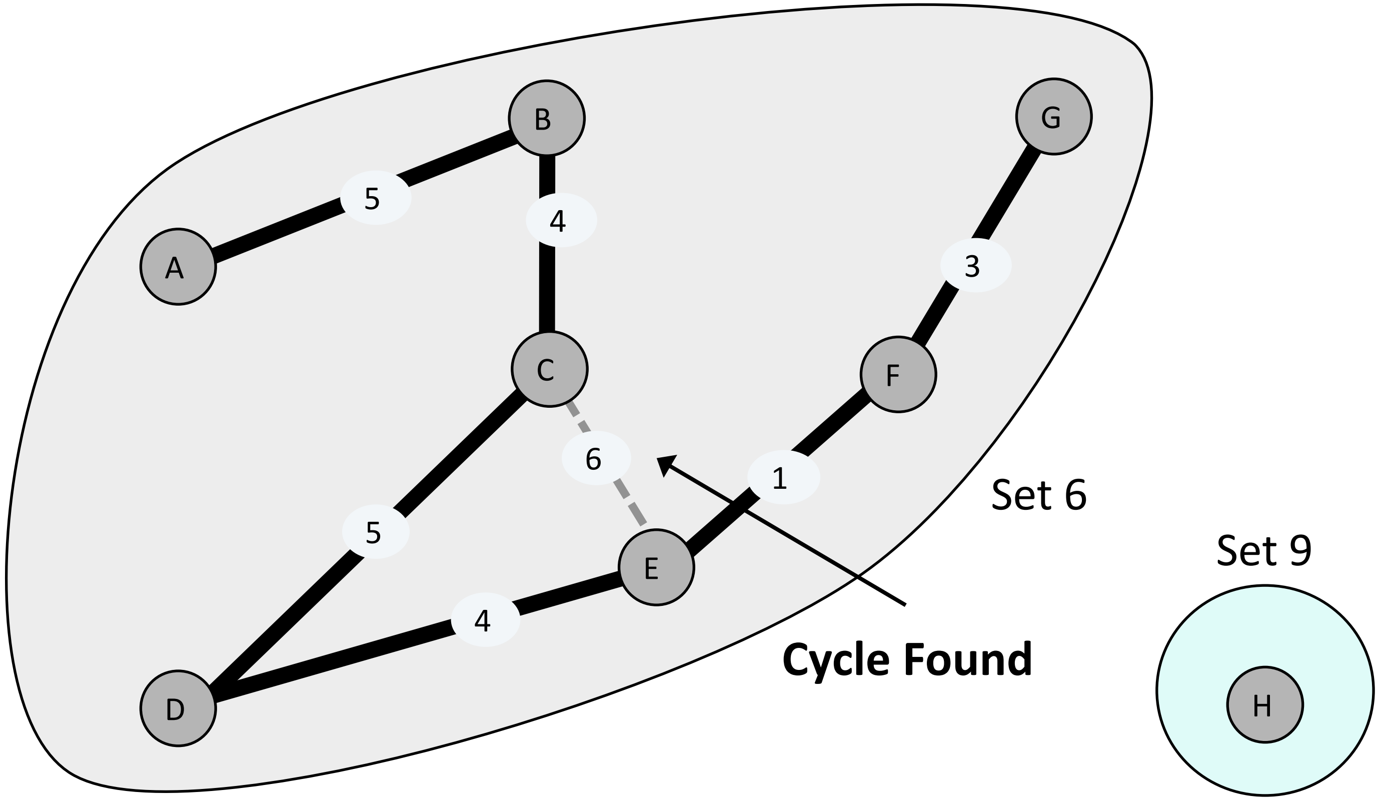
Kruskal’s algorithm (Union-Find Method)



Kruskal’s algorithm (Union-Find Method)



Kruskal's algorithm (Union-Find Method)



If they belong to the same set, a cycle is detected because adding this edge will create a loop.

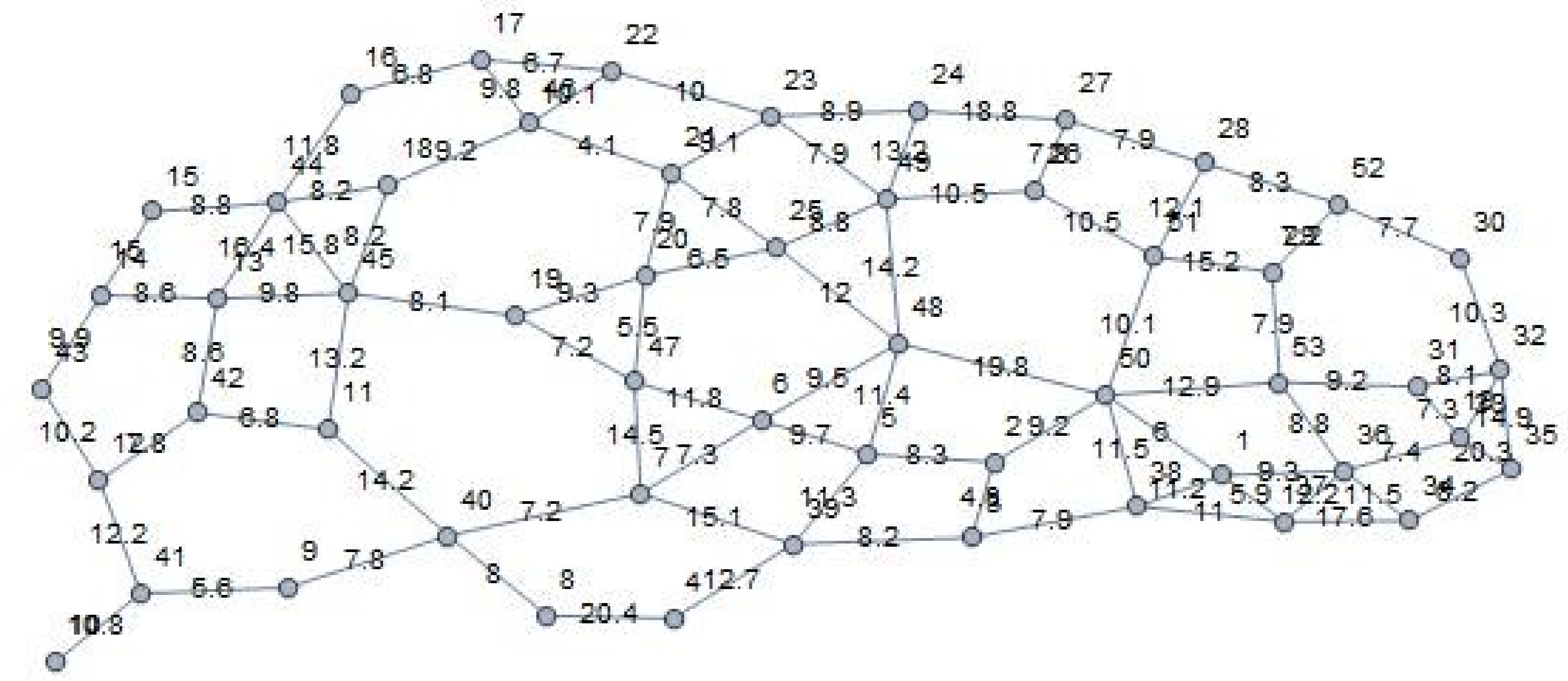
Kruskal's algorithm (Union-Find Method)

```
Kruskal(G) {
    MST ← empty set
    Sort edges E by weight in non-decreasing order   } O(ElogE)

    foreach (u ∈ V) make a set containing singleton u } O(V)

    for i = 1 to m
        (u,v) = e_i
        if (u and v are in different sets) {         } O(αV)
            MST ← MST ∪ {e_i}
            merge the sets containing u and v         } O(1)
        }
    return MST                                         (for loop) } O(E)
}
```

Inverse Ackermann function which grows extremely slow. We can consider it as constant

MST: Live Poll 2

Find the minimum spanning tree for the given weighted graph

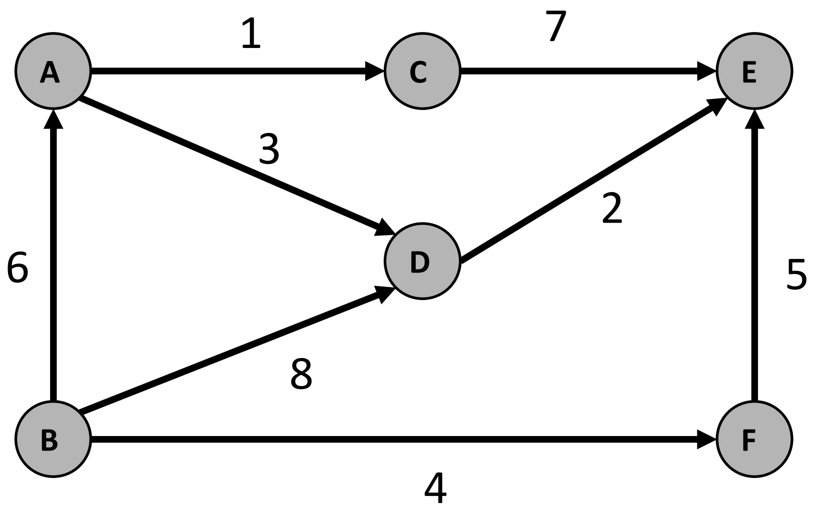


Scan the QR or use link to join

https://forms.office.com/r/MvHNues4DL

MST: Live Poll 2

Find the minimum spanning tree for the given weighted directed graph



Scan the QR or use link to join

https://forms.office.com/r/MvHNues4DL

Scan the QR or use link to join

https://forms.office.com/r/MvHNues4DL

Copy link

30 responses submitted

Find the minimum spanning tree for the given directed graph

{(A,C), (A,D), (D,E), (B,F), (F,E)}	63%
{(C,E), (B,A), (D,E), (B,D), (F,E)}	3%
{(A,C), (A,D), (B,F), (B,D), (F,E)}	0%
I don't know	6%
I don't have any energy left	3%
End the lecture...Please	23%

Treemap Bar

1 of 1

Thanks a lot

Mathew Perry

"Chandler"

1969 – 2023

If you are taking a Nap, **wake up**........Lecture Over