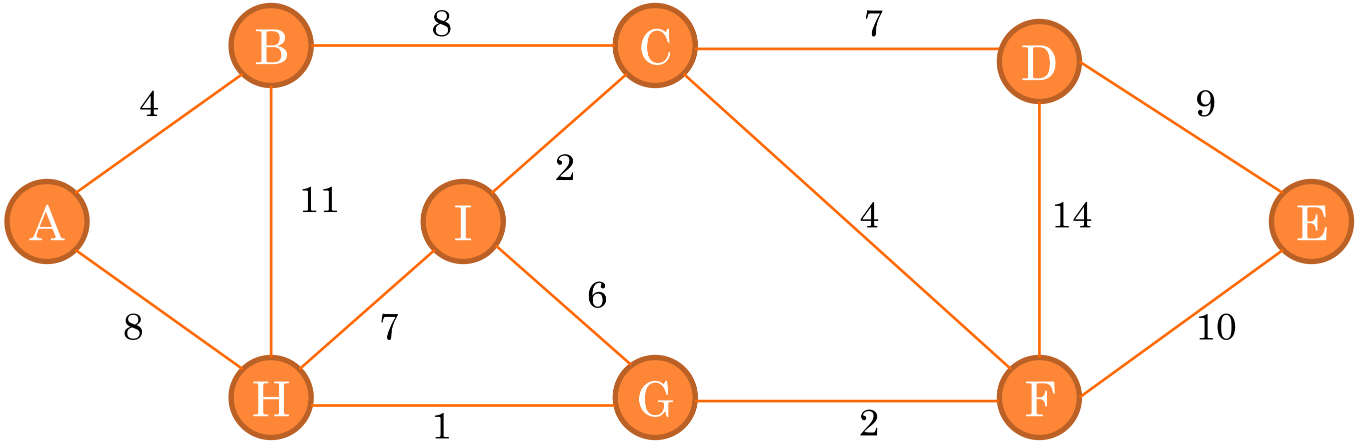
Minimum Spanning Tree

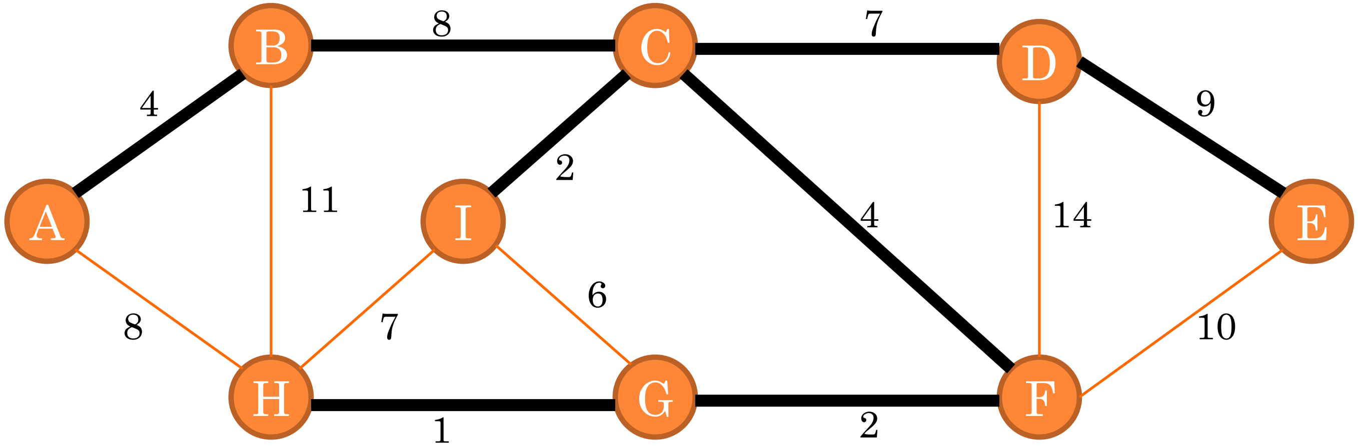
Tanjina Helaly

Minimum Spanning Tree

- Tree: Connected Acyclic Graph
- Spanning Tree: A tree that contains all vertices.
  - Formal definition: Given an connected weighted graph G, a Spanning tree is a subgraph that connects all vertices and acyclic.
- Minimum Spanning tree:
  - Given a graph(V,E) and edge weights function w: E -> R, find the spanning tree of minimum weight.

Example


A
B
C
D
E
F
G
H
I
8
7
4
11
2
4
14
9
8
7
6
10
1
2


A
B
C
D
E
F
G
H
I
8
7
4
11
2
4
14
9
8
7
6
10
1
2

APPLICATION

- Cluster analysis
- Face verification
- Avoid cycle in network
- Network design(communication, electrical, computer, road)
- Reducing data storage in sequencing amino acid in protein.

Brute Force

- Find all spanning tree and select the one with minimum total weight.
- Complexity: Exponential

Greedy Algorithm

- It grows one edge at a time
- Assume, prior to each iteration, A is a subset of some minimum spanning tree.
- At each step, we determine an edge (u, v) that we can add to A [A U (u,v)] and A still remain a subset of minimum spanning tree
  - This edge is called safe edge.

- Algorithm

GENERIC-MST(G, w)

1 A =Ǿ ;

2 **while A does not form a spanning tree**

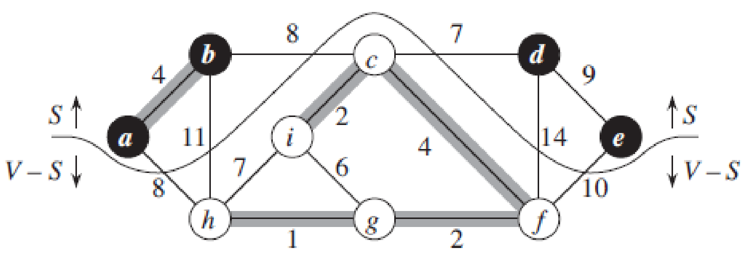
3 find an edge .u; / that is safe for A

4 A = A U {(u, v)}

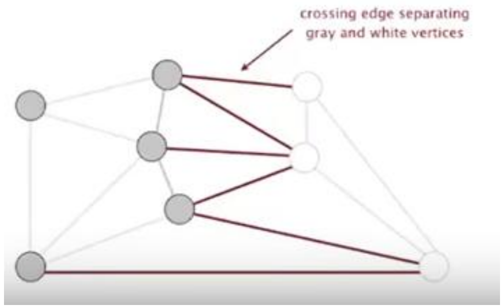
5 **return A**

Cut Property

- A **Cut** in a graph is a partition of its vertices into 2 non-empty set.
- A **crossing edge** connects a vertex in one set with a vertex in the other set.
- **Cut property**: Given any cut, the crossing edge of min weight is in MST.

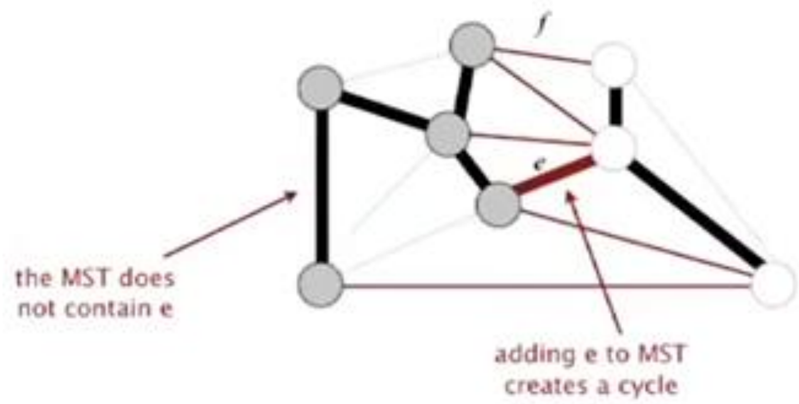


(a)



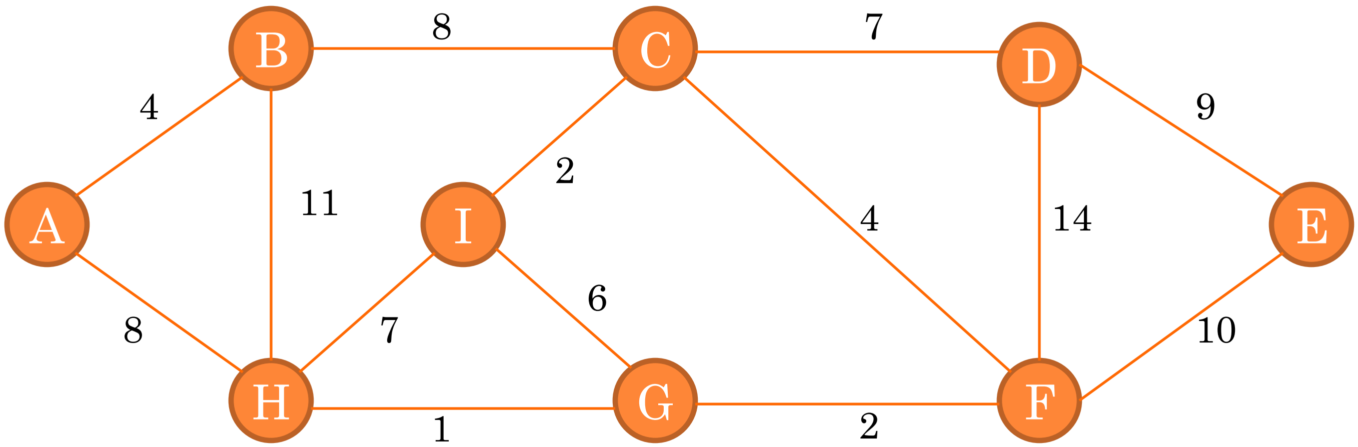
Proof

- Suppose min-weight crossing edge e is not in MST
- Some other crossing edge f should be in MST otherwise the graph will not be connected.
  - So, adding e to MST creates a cycle
  - Removing f and adding e is also a spanning tree
  - Since weight of e<weight of f
    - ST with e will have lower weight than ST with f.
      - This is a contradiction. So, f can't be in MST



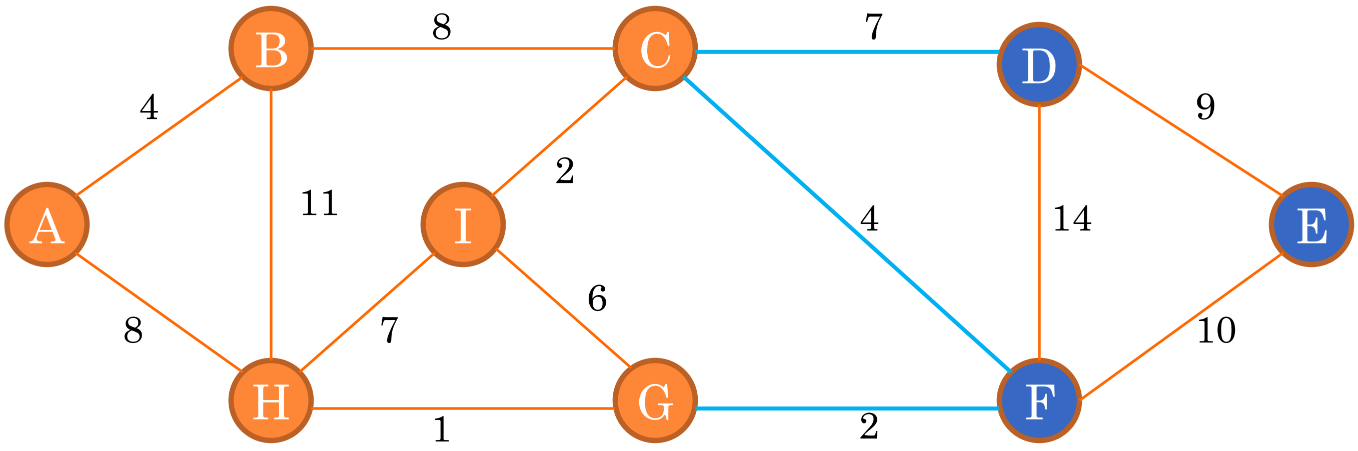
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



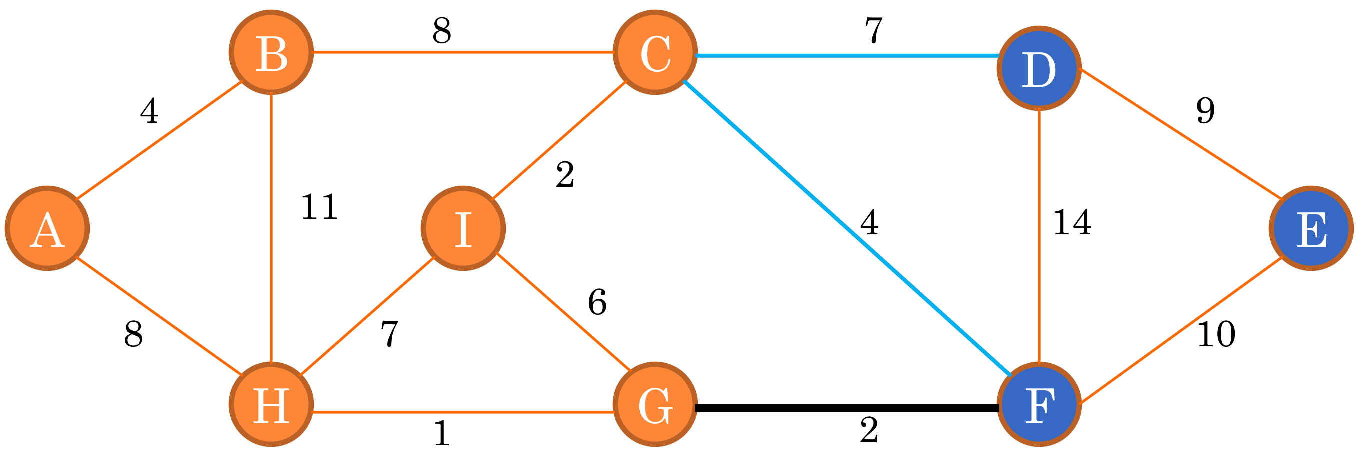
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



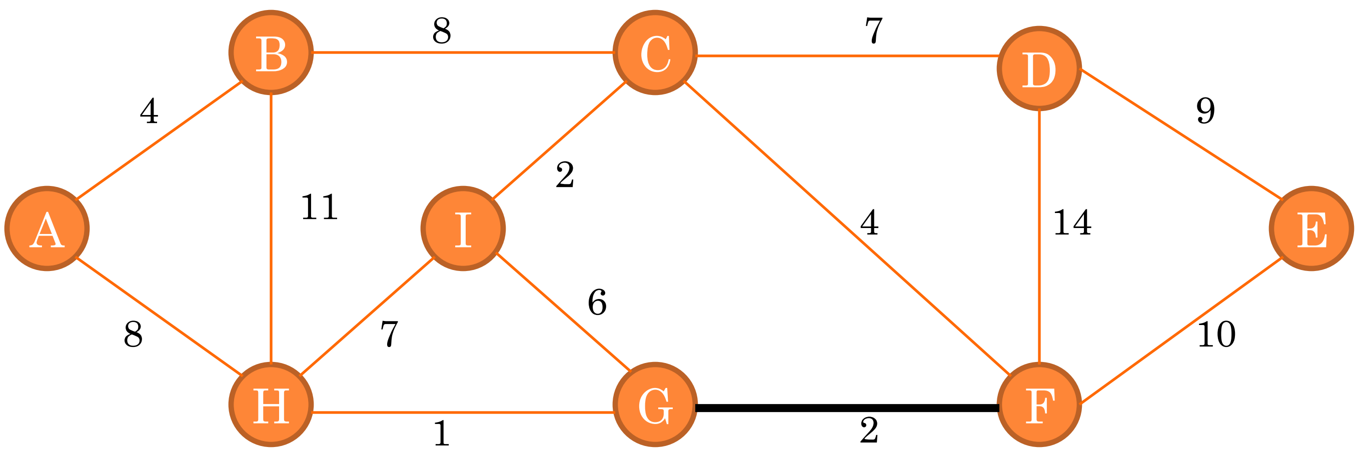
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



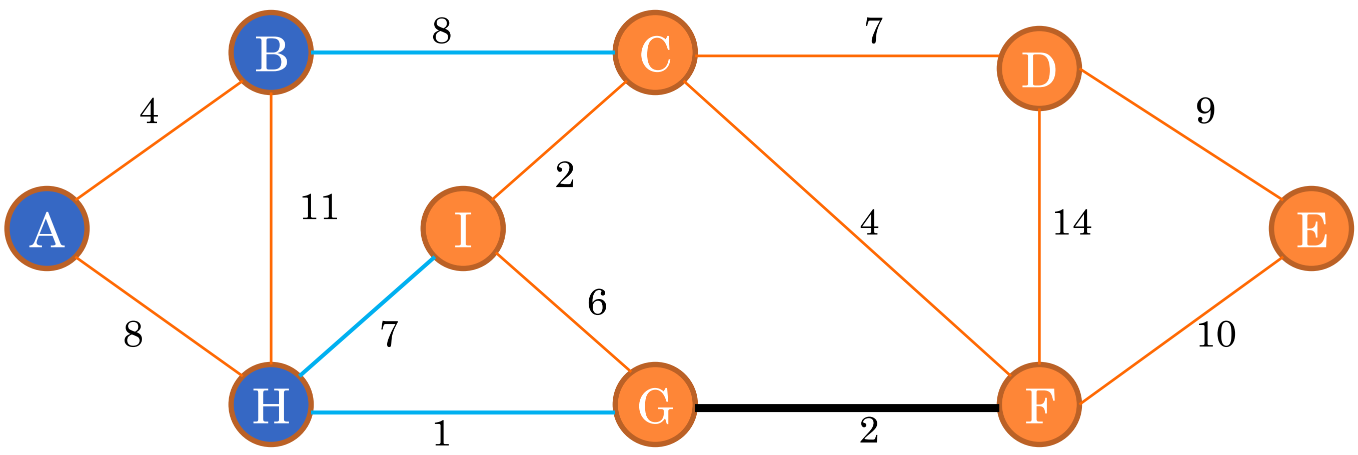
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



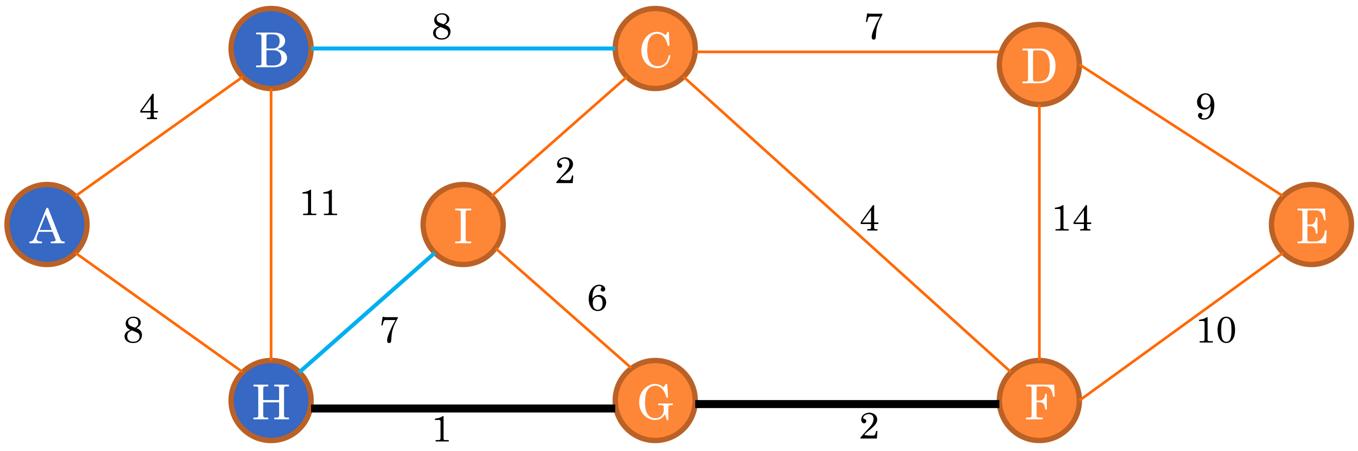
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



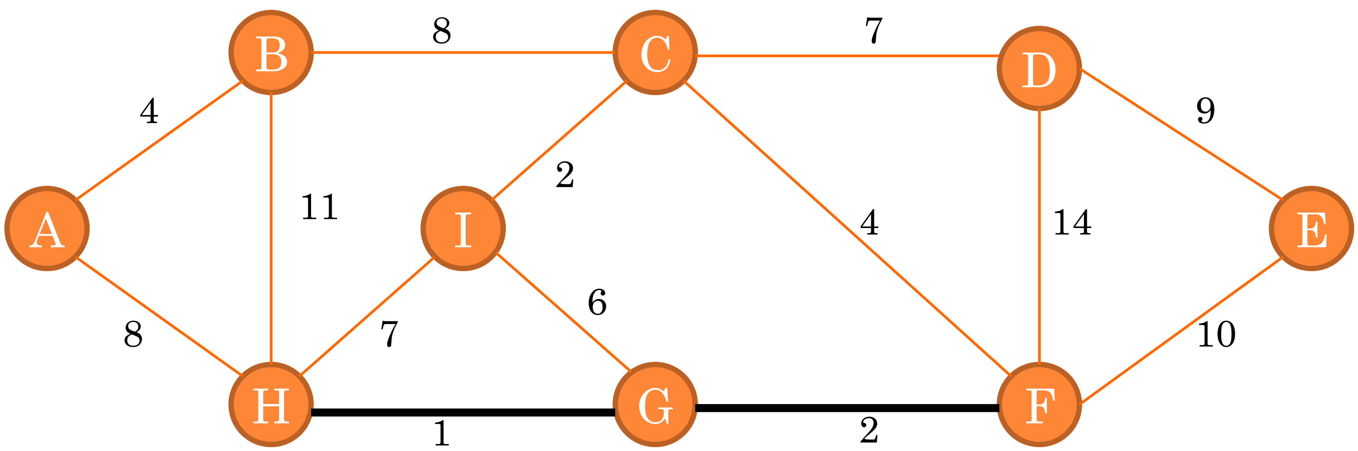
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



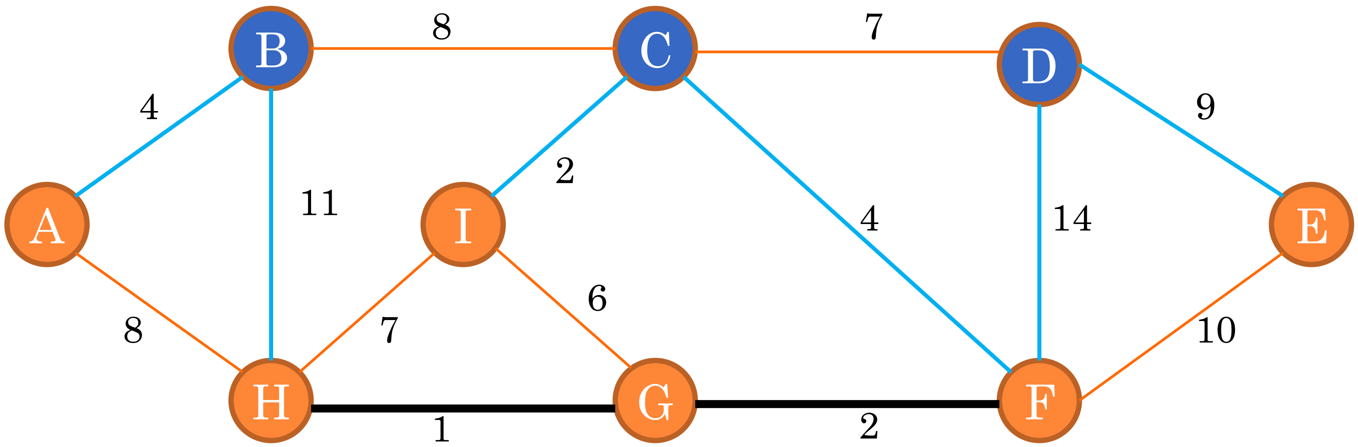
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



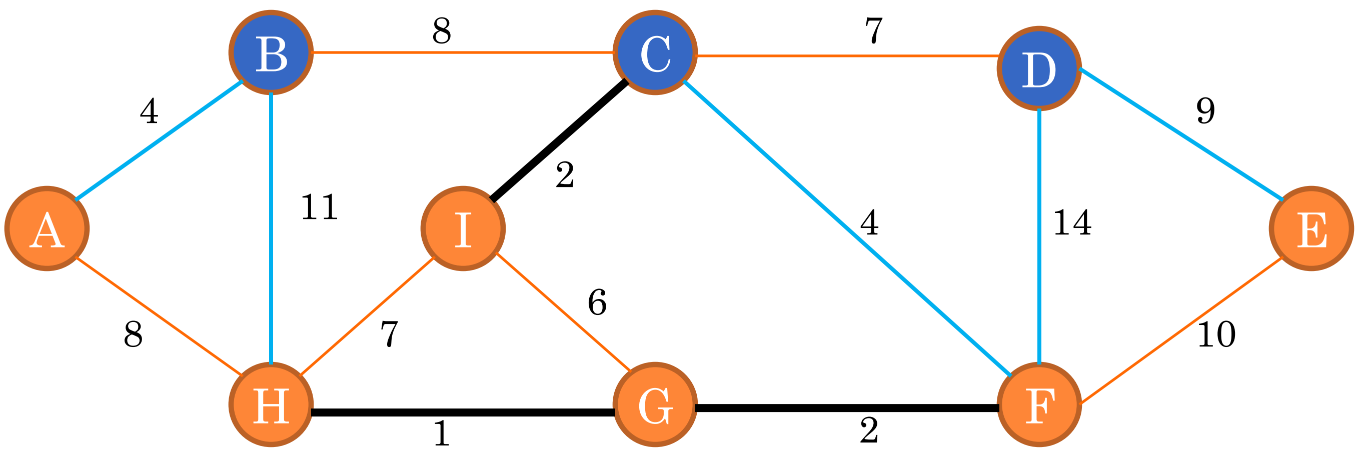
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



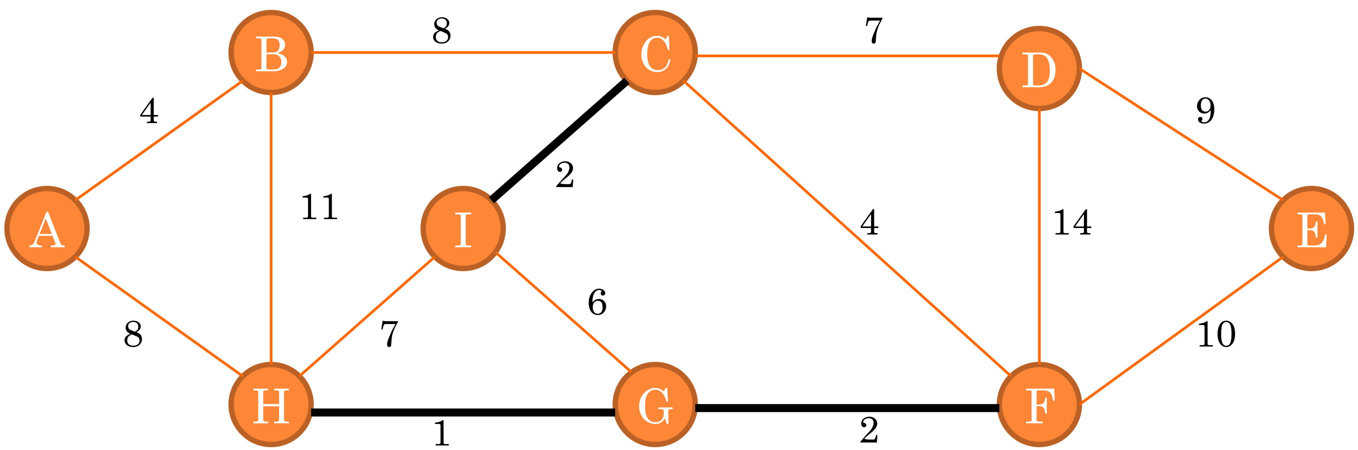
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



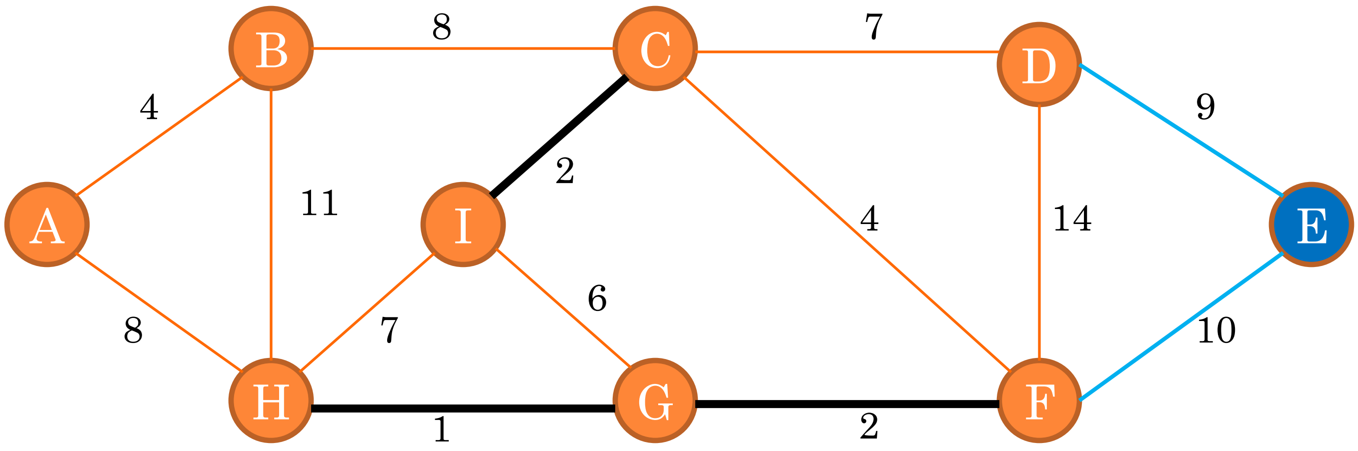
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



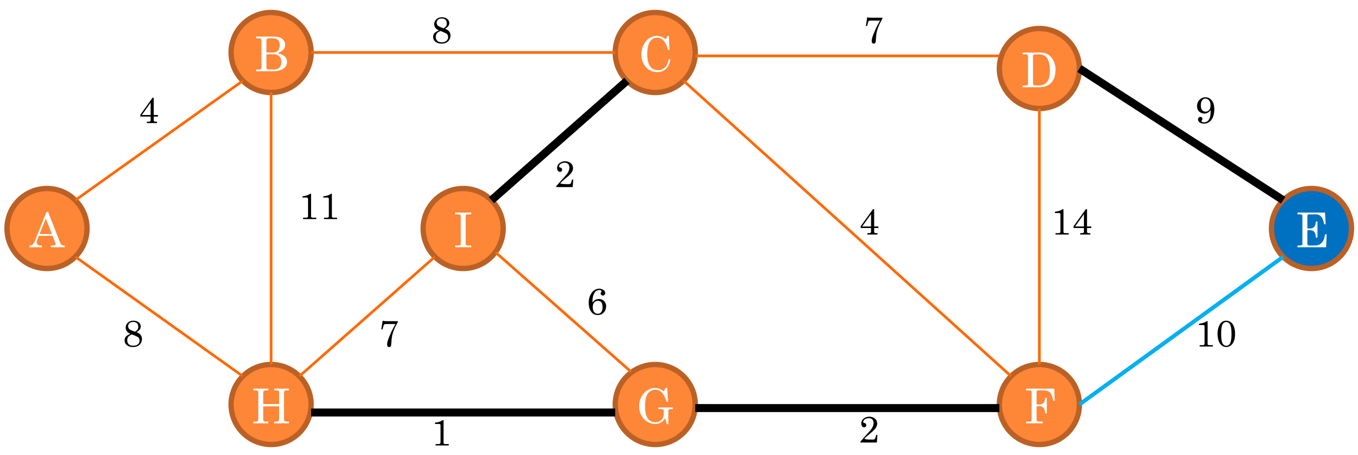
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



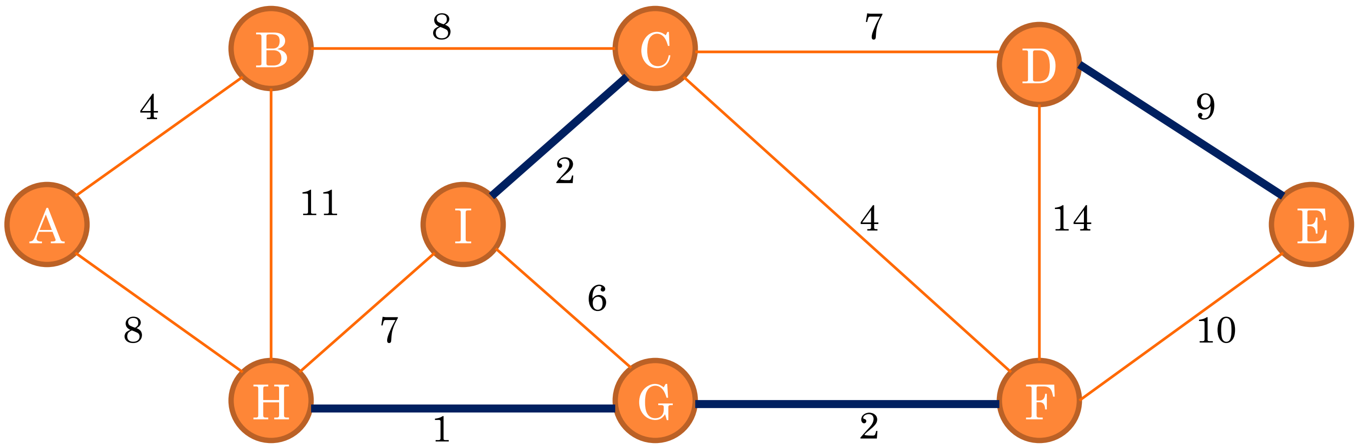
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



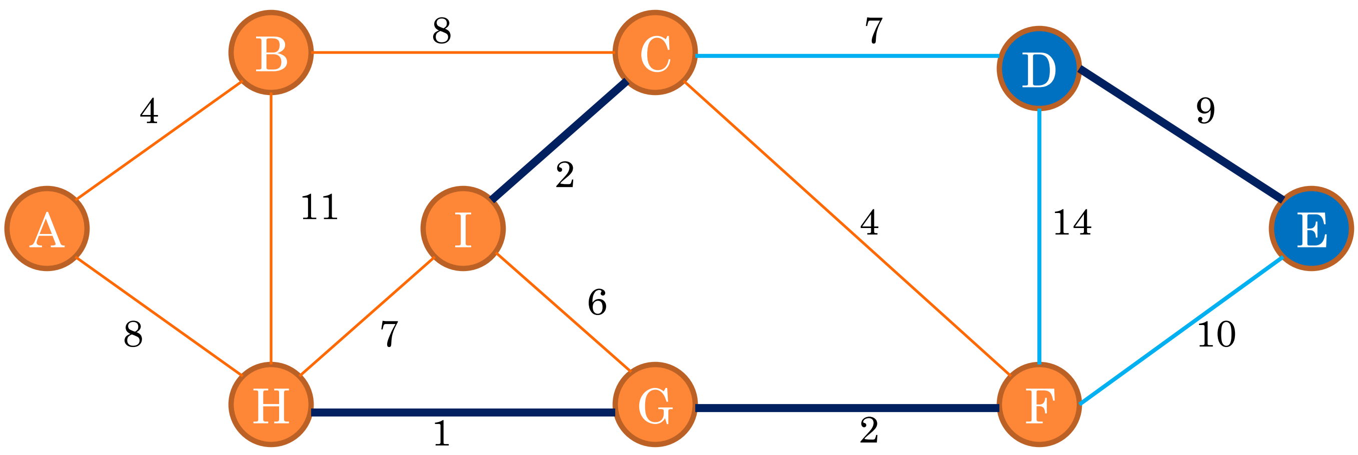
Greedy MST Algorithm

- Start with all edge colored non-black. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



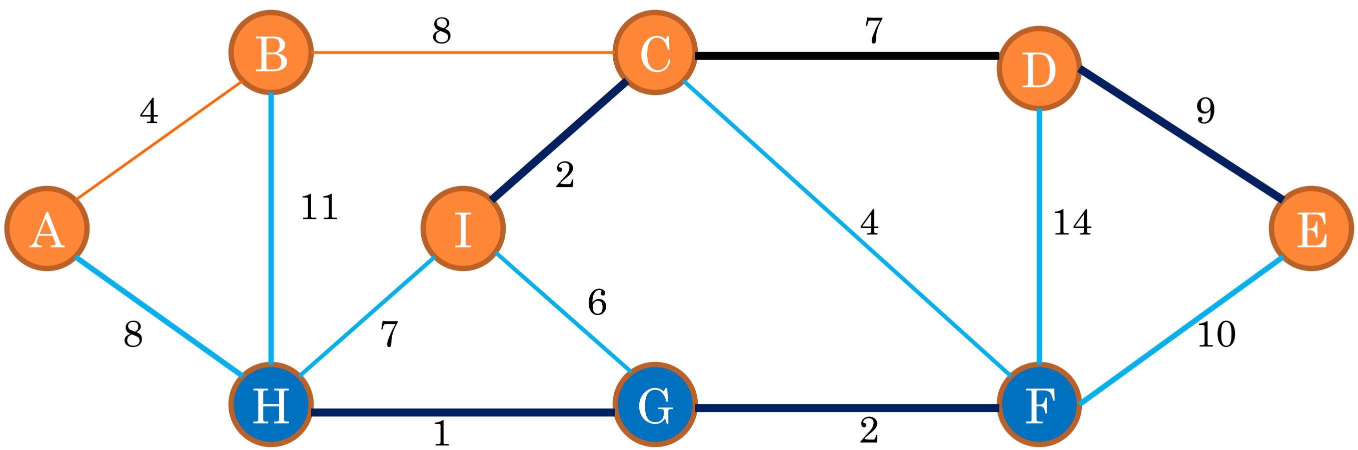
Greedy MST Algorithm

- Start with all edge colored non-black. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



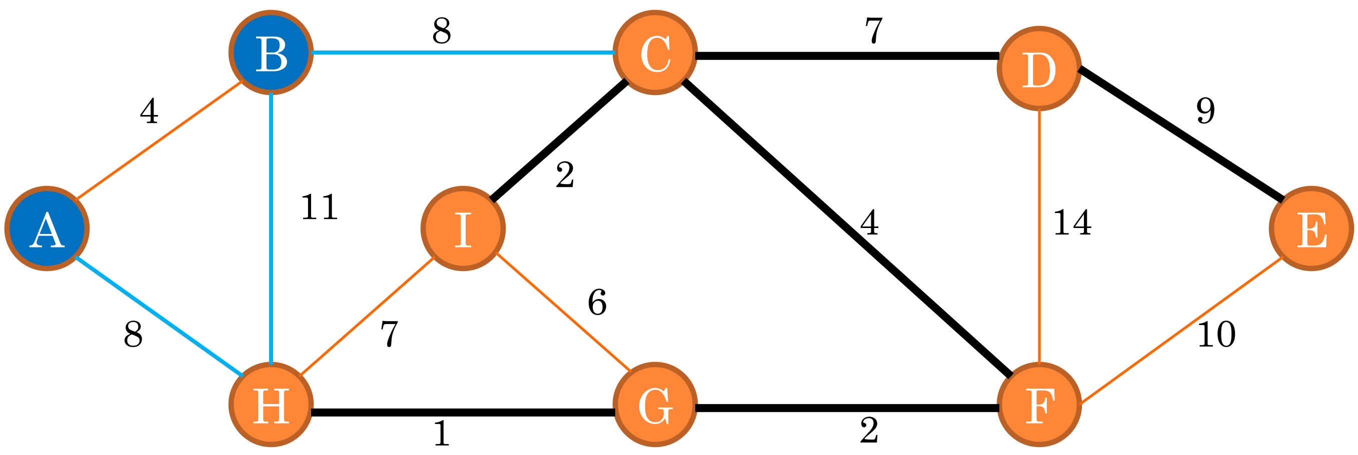
Greedy MST Algorithm

- Start with all edge colored non-black. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



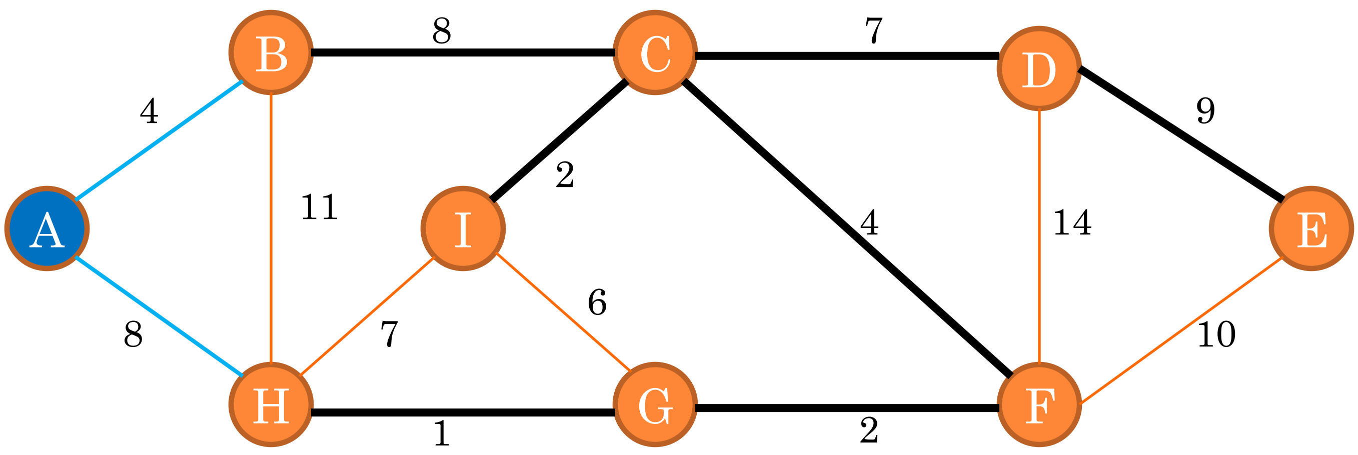
GREEDY MST ALGORITHM

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



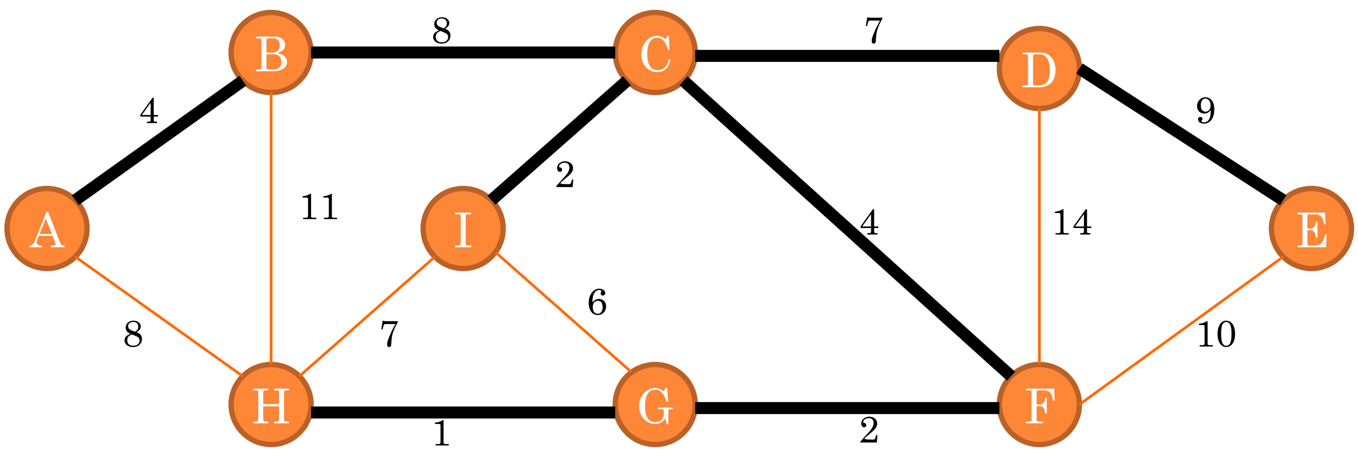
Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



Greedy MST Algorithm

- Start with all edge colored gray. [No edge is selected to be in MST]
- Find a cut with no black crossing edge; color the min-weight crossing edge black.
- Repeat until v-1 edges are colored black



Kruskal's ALgorithm

Kruskal's Algorithm

- Select the shortest edge in a graph
- Select the next shortest edge which does not create a cycle
- Repeat step 2 until all vertices have been connected.

Kruskal's Algorithm - pseudocode

- Algorithm

$Kruskal(G)$:

$A = \emptyset$

$foreach\ v \in G.V$

$\quad Make - Set(v)$

$sort\ G.E\ in\ ascending\ order$

$foreach\ (u, v) in\ G.E$

$\quad if\ Find - Set(u) \neq Find - Set(v)$

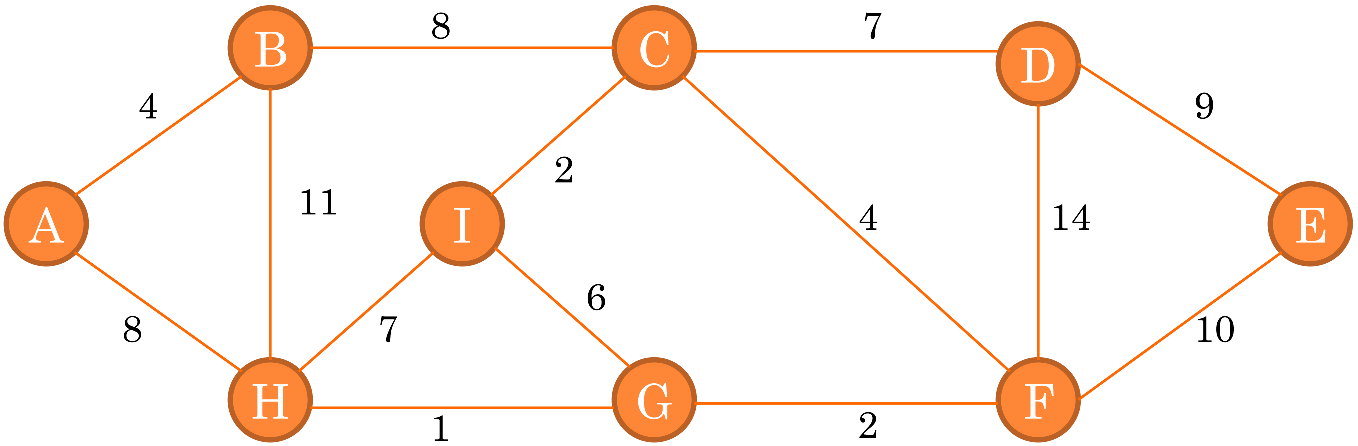
$\qquad A = A \cup \{(u, v)\}$

$\qquad Union(u, v)$

$return$

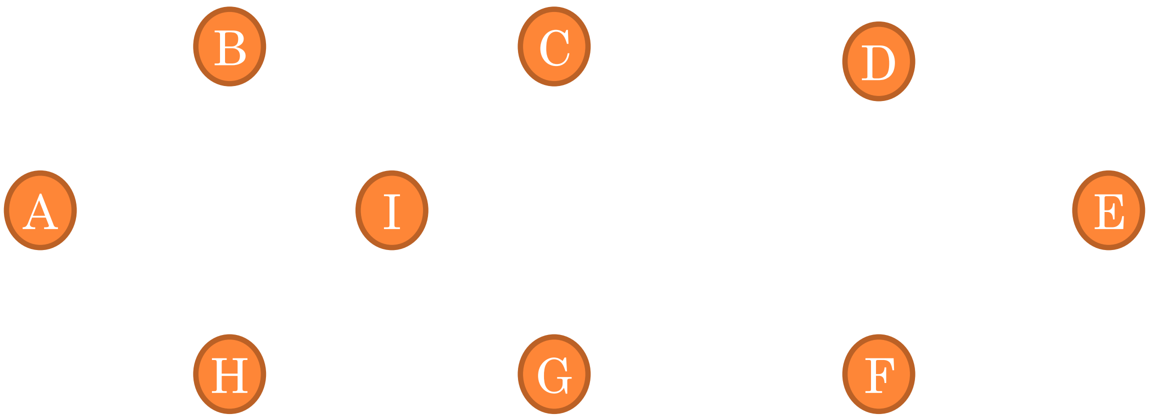
Kruskal's Algorithm - Example

- Given the graph below, create the MST using Kruskal's algorithm.



Kruskal's Algorithm - Example

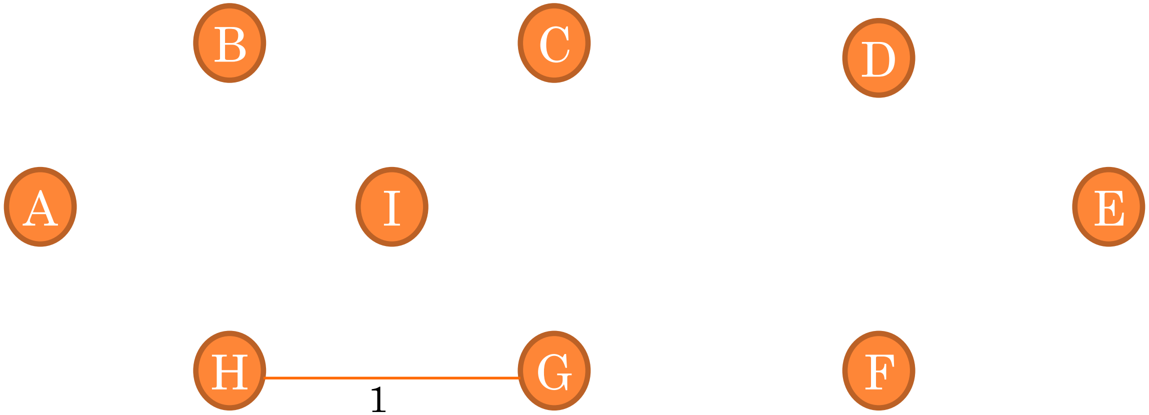
- Initial Set.
- One vertex in each node
- Edges are arranged in ascending order of cost.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

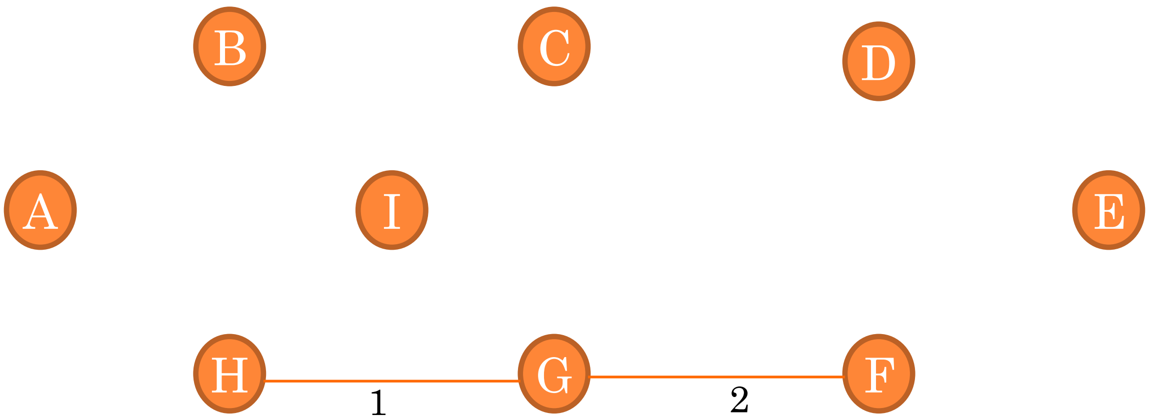
- Check G-H.
- As G and H are not in same component, connect those.



Edge	Cost
G-H	**1**
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

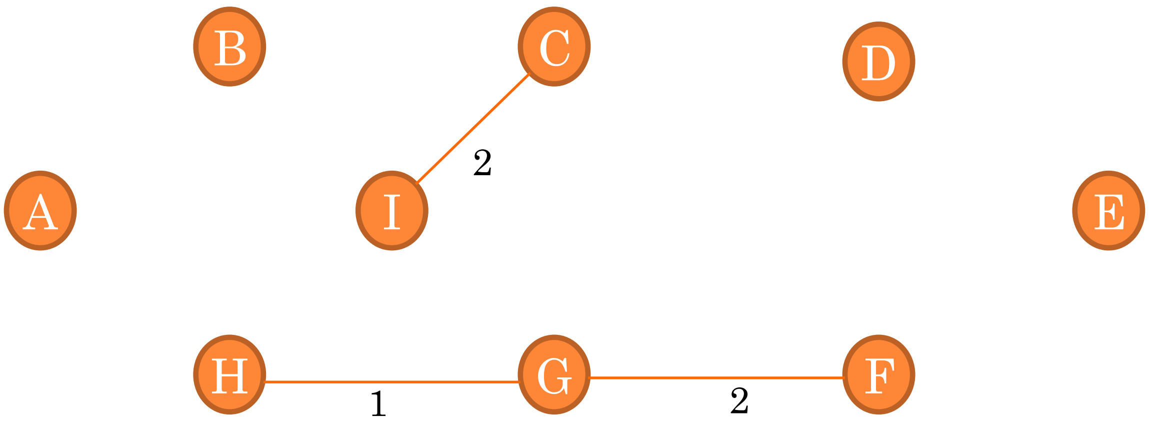
- Check G-F.
- As G and F are not in same component, connect those.



Edge	Cost
G-H	1
G-F	**2**
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

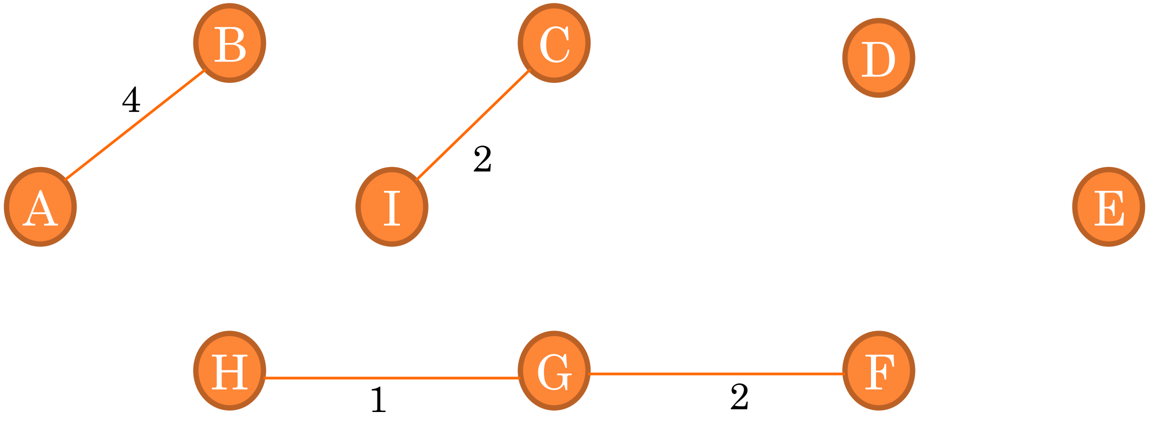
- Check C-I.
- As C and I are not in same component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	**2**
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

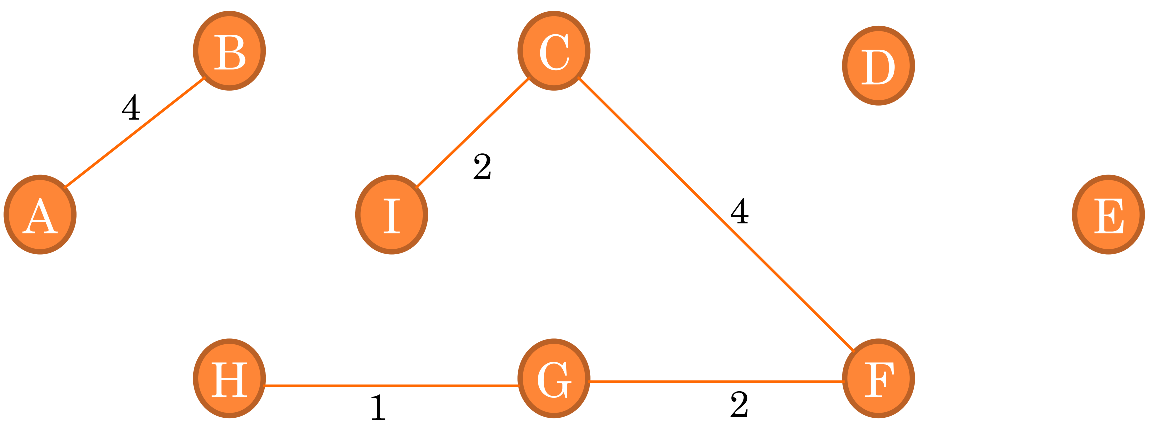
- Check A-B.
- As A and B are not in same component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	**4**
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

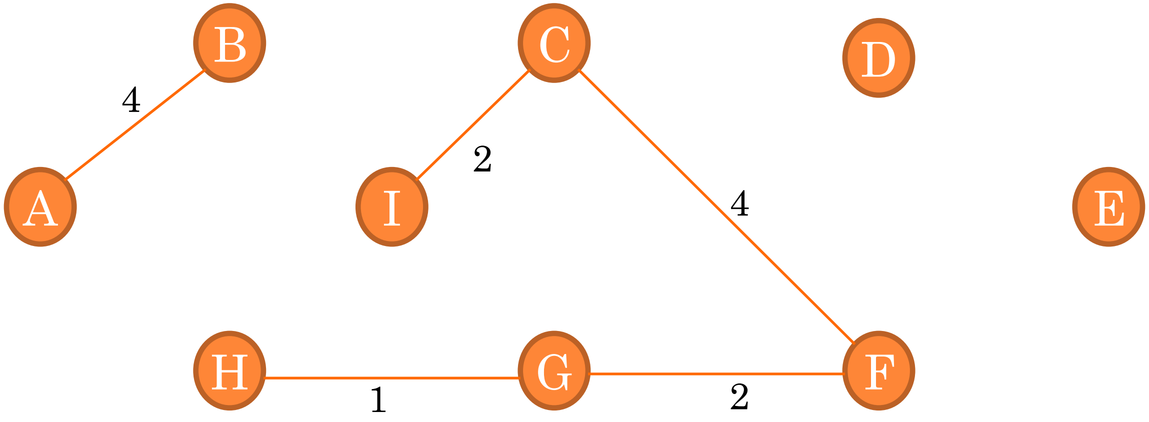
- Check C-F.
- As C and F are not in same component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	**4**
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

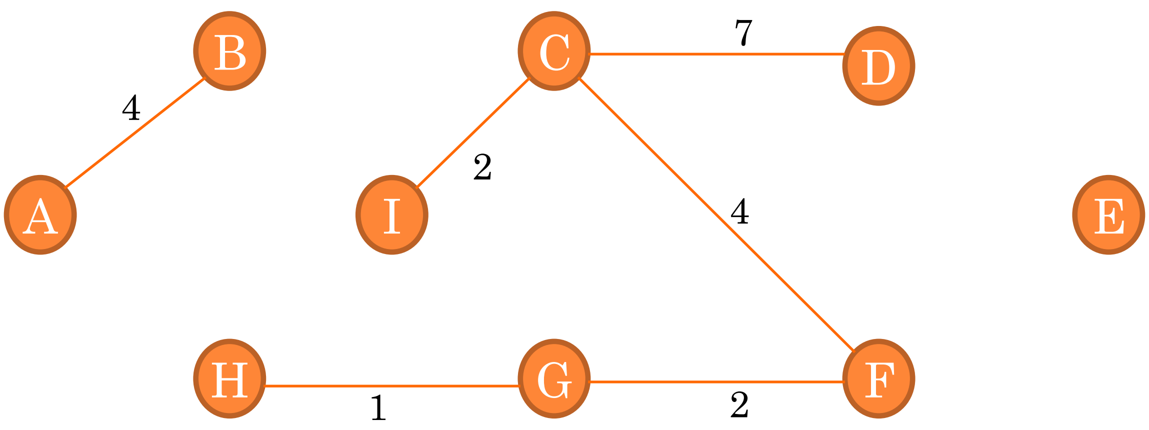
- Check G-I.
- As G and I **are in same component**, **do not** connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	**6**
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

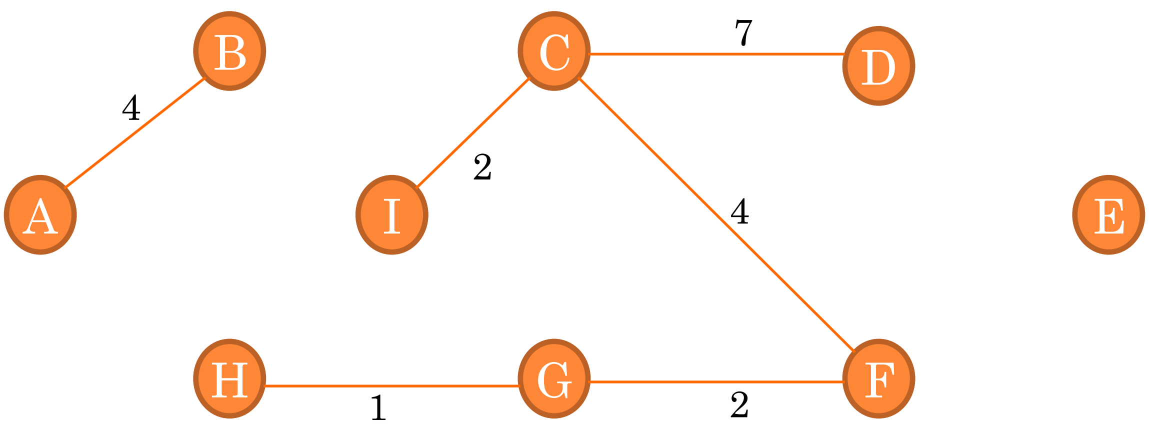
- Check C-D.
- As C and D **are not in same** component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	**7**
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

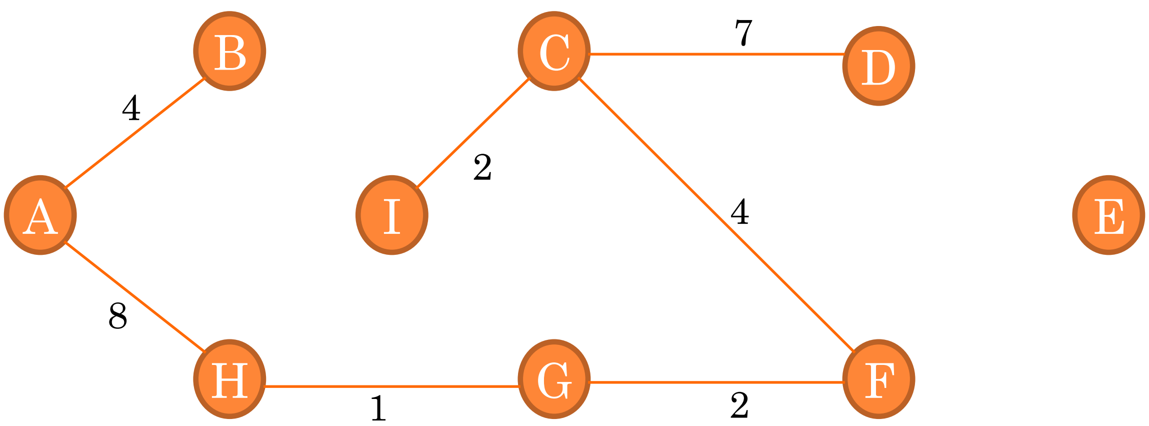
- Check H-I.
- As H and I **are in same component**, **do not** connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

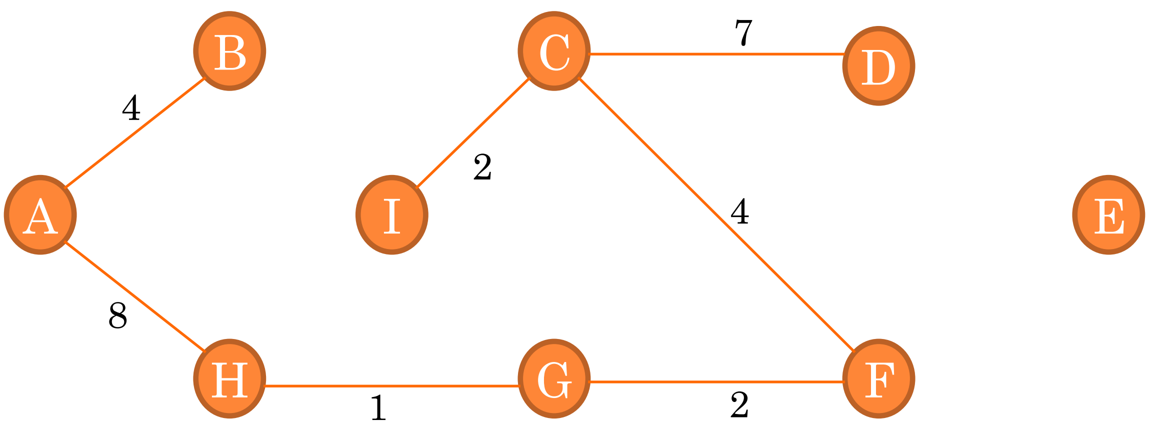
- Check A-H.
- As A and H **are not in same** component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	**8**
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

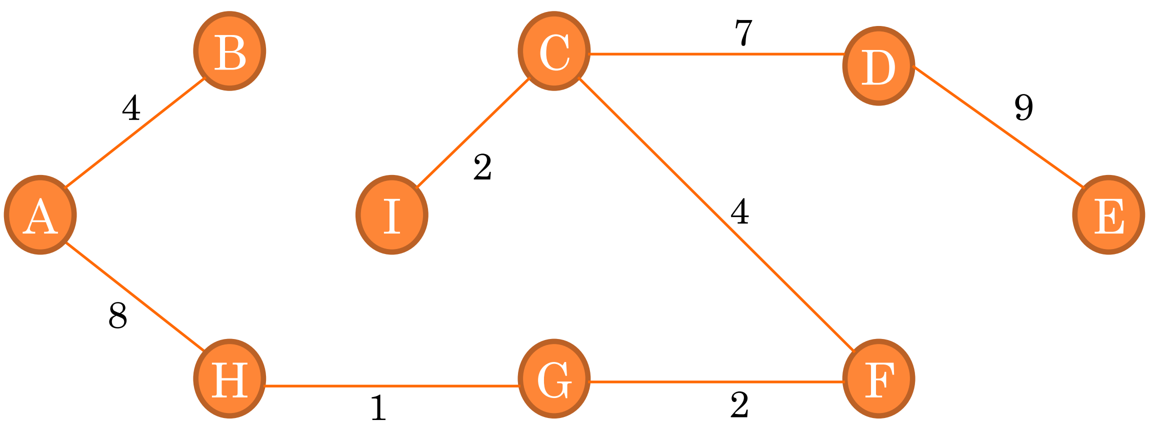
- Check B-C.
- As B and C **are in same component**, **do not** connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

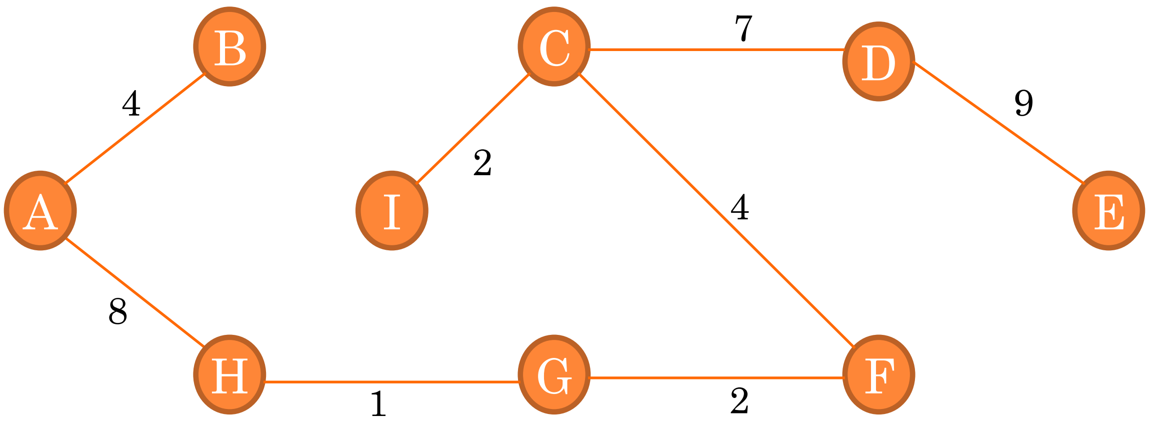
- Check D-E.
- As D and E **are in not same** component, connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	**9**
E-F	10
B-H	11
D-F	14

Kruskal's Algorithm - Example

- Check the last 3 edges.
- E & F, B & H, D & F **are in same** component, no need to connect those.



Edge	Cost
G-H	1
G-F	2
C-I	2
A-B	4
C-F	4
G-I	6
C-D	7
H-I	7
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

KRUSKAL'S ALGORITHM - COMPLEXITY

- Algorithm

```
Kruskal(G):
A = ∅
foreach v ∈ G.V                                   V
    Make − Set(v)
sort G.E in ascending order of weight             E logE
foreach (u, v) in G.E                             E
    if Find − Set(u) ≠ Find − Set(v)              E logV
        A = A ∪ {(u, v)}                                      E logV
        Union(u, v)                               V logV
return
```

- Complexity

  $= O(E \log E + E \log V) = O(E \log E)$

  As $E_{max} = V^2$, we can write $O(\log E) = O(\log V)$

- So, complexity = $O(E \log E)$ or $O(E \log V)$

Try at home

- Is MST always unique for a graph?
  - Hints: 2 components can be connected via multiple edges.
- Is minimum weight edge always included in MST?
  - Hints: think about non-unique weight edges.
- Negate the weights
- Increase all weights by a fixed number.

Reference

- Chapter 23 (Cormen) -> 23.1, 23.2