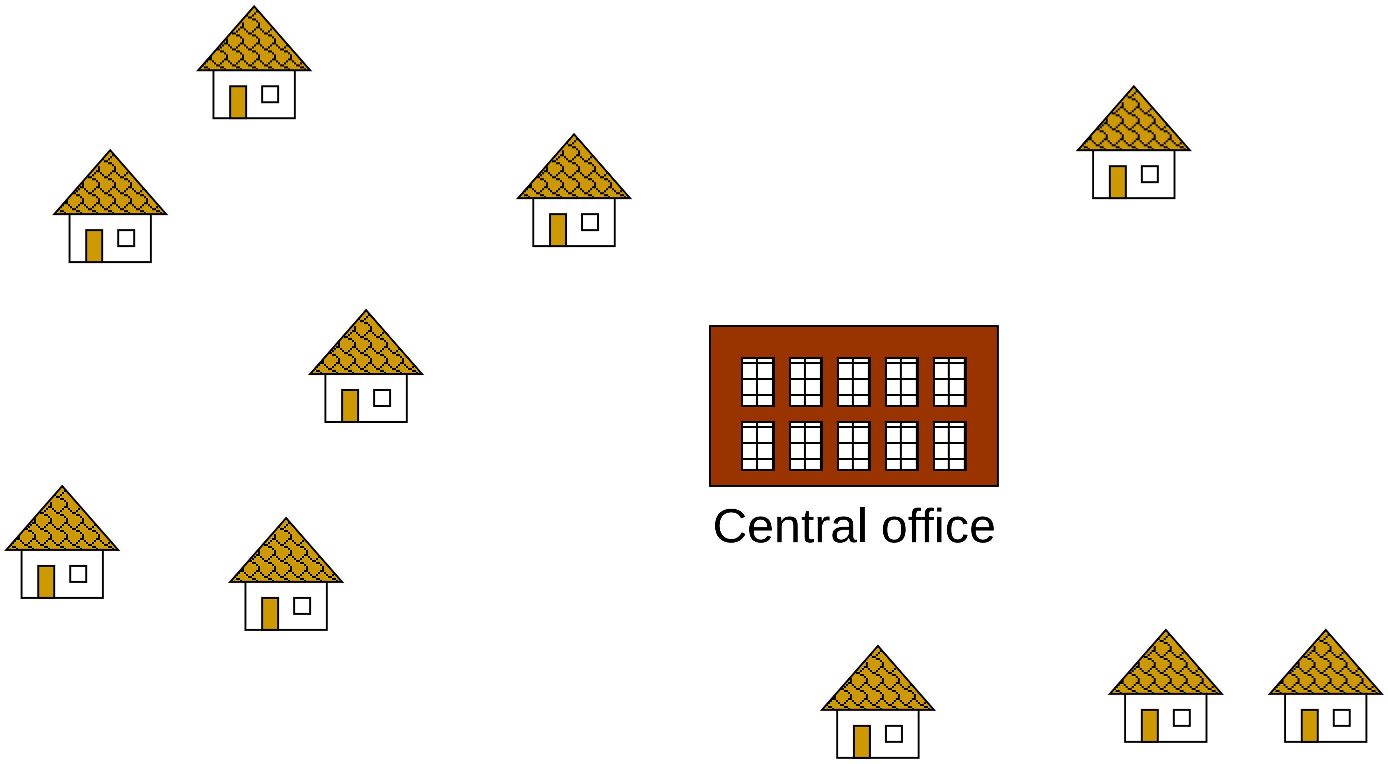
Greedy Algorithms: Prim's Algorithm

Textbook: Chapter 9.1

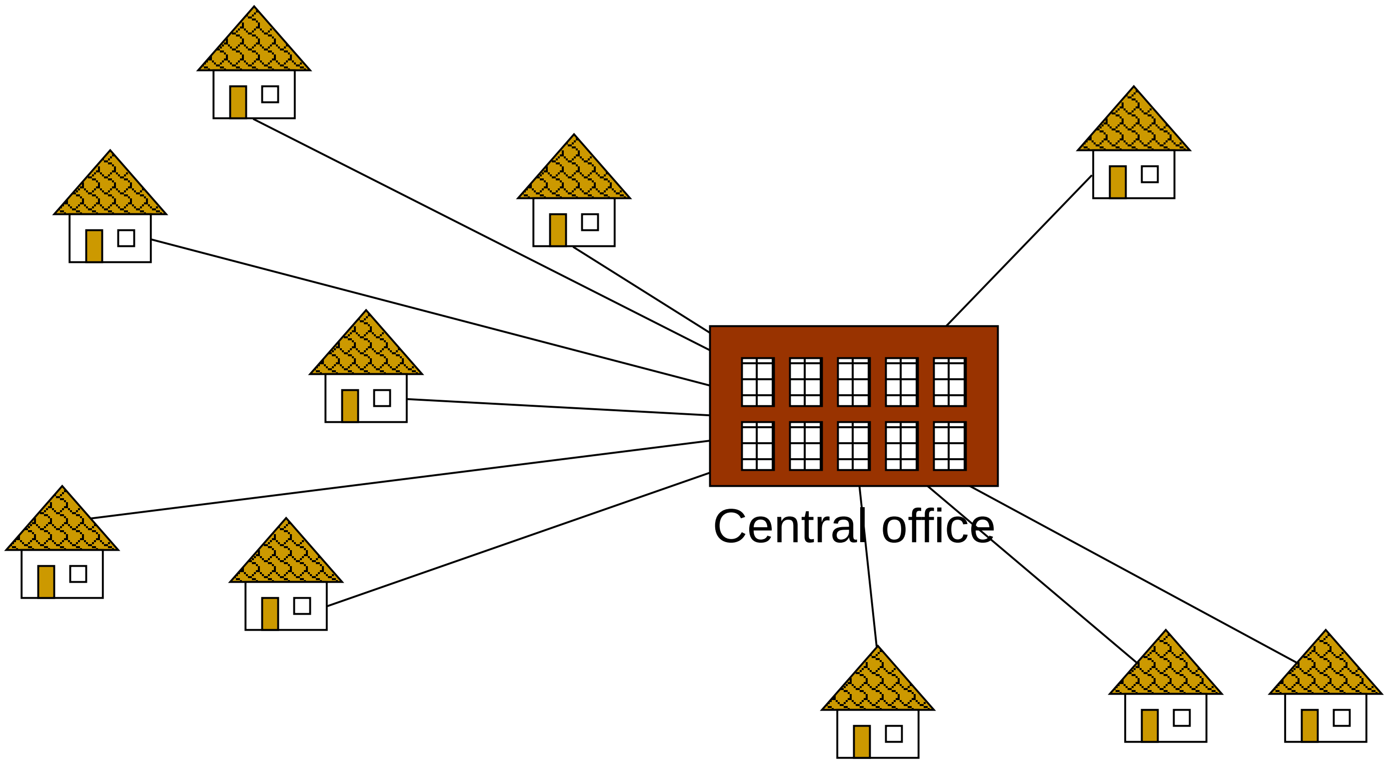
Context

- This is one of several “greedy algorithms” we will examine:
  - Minimum Spanning Tree of a graph
    - Prim’s algorithm
    - Kruskal’s algorithm
  - Shortest Paths from a Single Source in a graph
    - Dijkstra’s algorithm
  - Graph coloring

Problem: Build a (physical) network

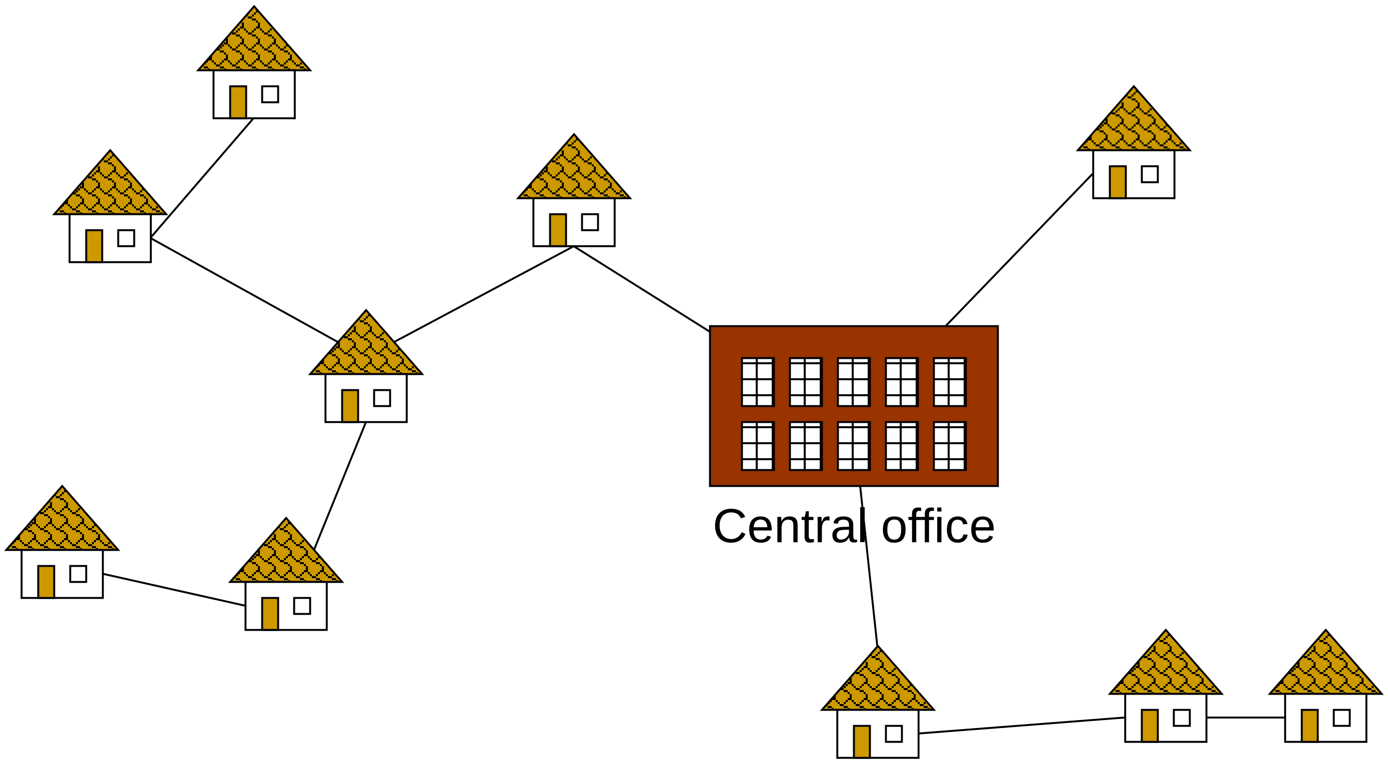


Wiring: Naïve approach



Expensive!

Wiring: Better approach



Minimize the total length of wire connecting the customers

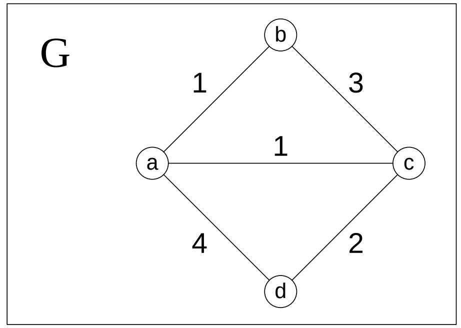
Minimum Spanning Trees

- A **minimum spanning tree** (MST) is a subgraph of an undirected weighted graph ***G***, such that
  - it is acyclic
  - it includes all the vertices
  - the total cost associated with the edges is the minimum among all possible spanning trees

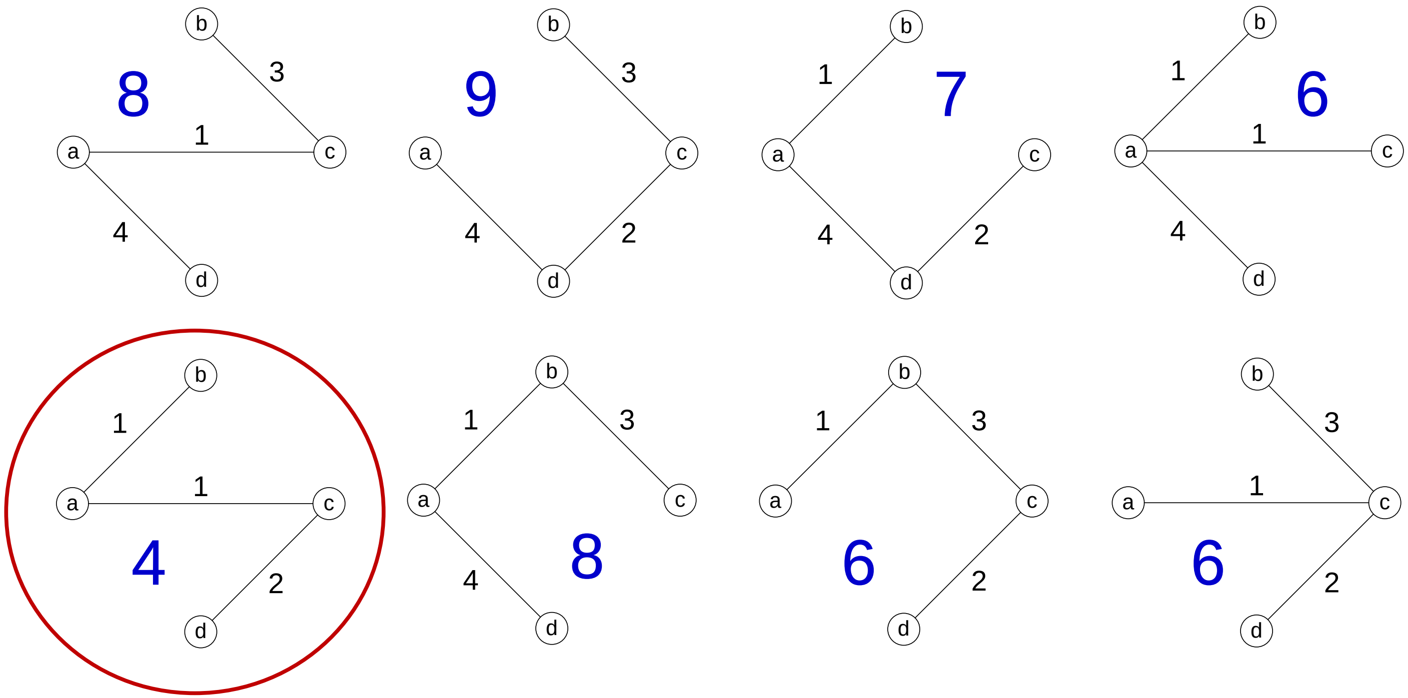
- MST may not be unique

MSTs (cont'd)

Consider all the spanning trees of G:



The weight of each spanning tree is given by the sum of its edges …

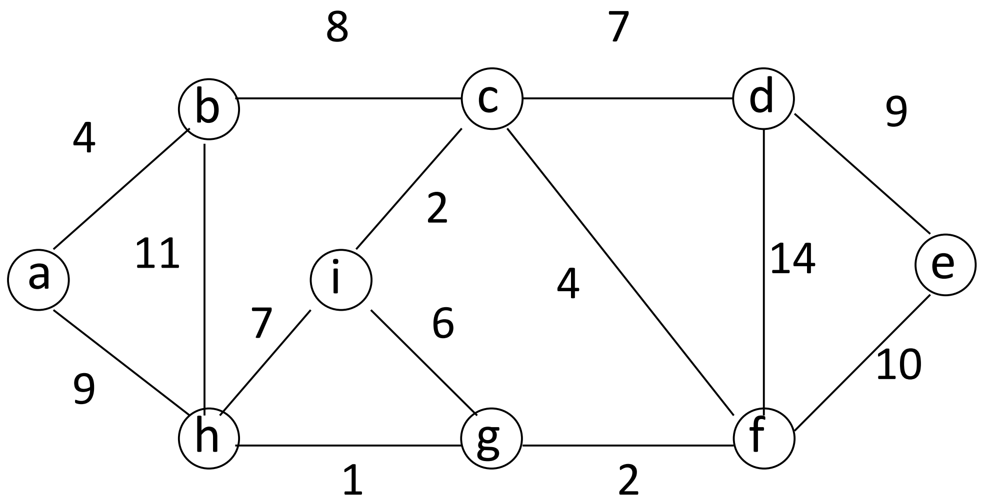


Minimum Spanning Tree of G is this graph, and it has a weight of 4.

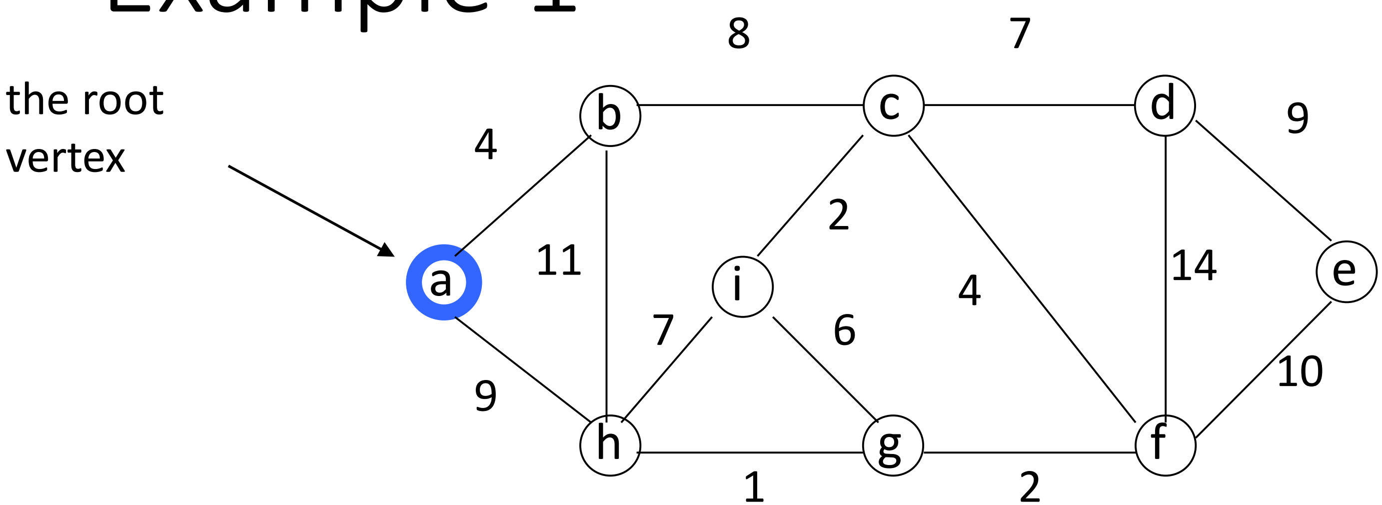
Prim's algorithm

```
Algorithm Prim(G)
    V_T ← {v_0}                          // init tree with one (arbitrary) vertex
    E_T ← ∅                              // init tree with no edges
    for i ← 0 to |V|-1 do                // loop until all vertices added to tree
        find a min-weight edge e=(u,v) from E
            where u is in V_T (in the tree)
            and v is in V-V_T (not yet in the tree)
        V_T ← V_T ∪ {v}                   // add the vertex v to the tree
        E_T ← E_T ∪ {e}                   // add the edge (u,v) to the tree
    return T = (V_T, E_T)
```

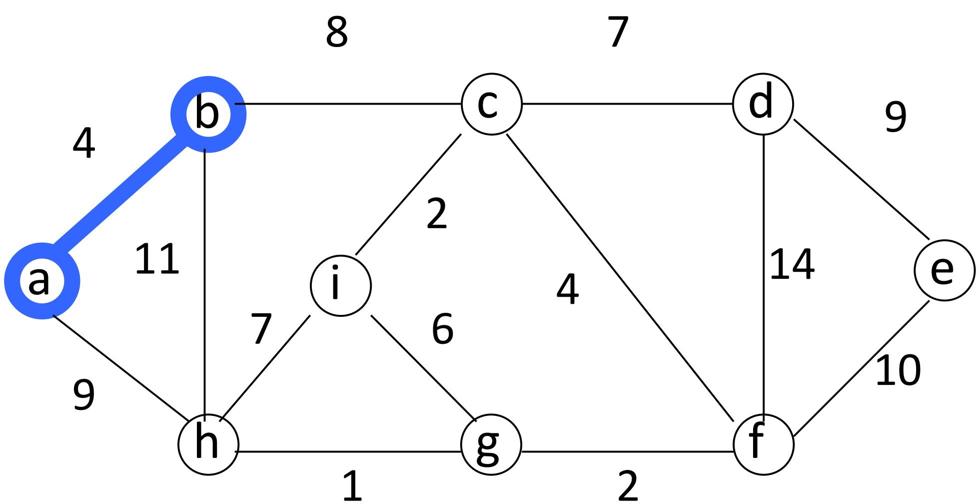
Example 1



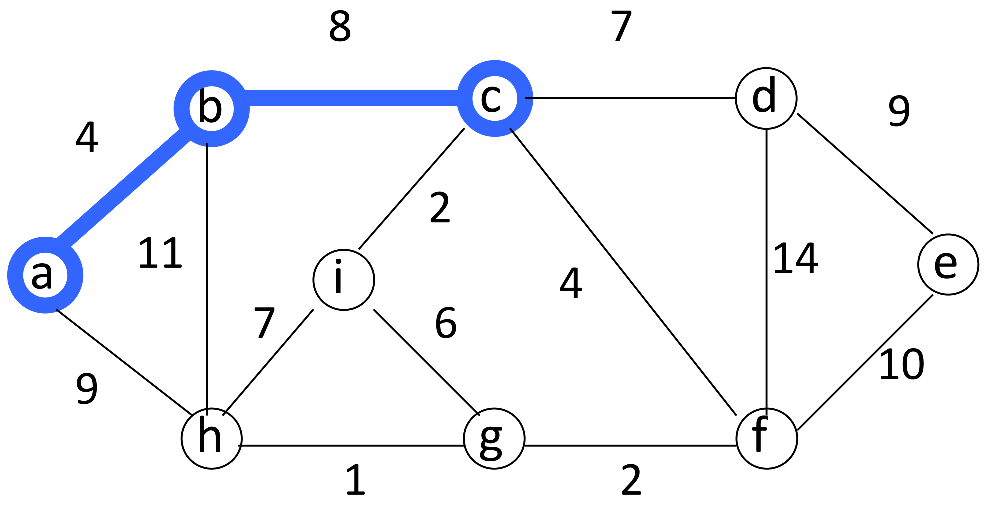
Example 1

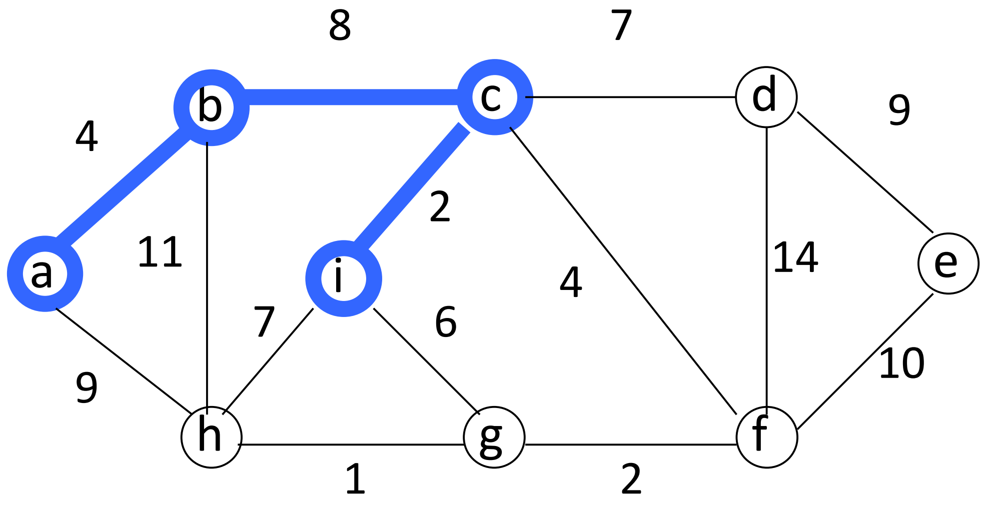


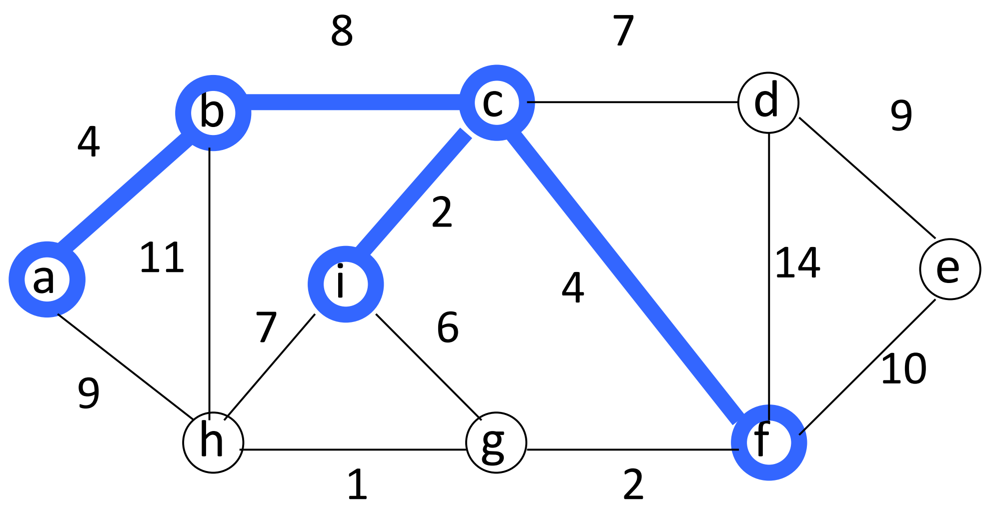
the root vertex
a
b
c
d
e
f
g
h
i
4
8
7
9
11
2
14
4
7
6
10
9
1
2



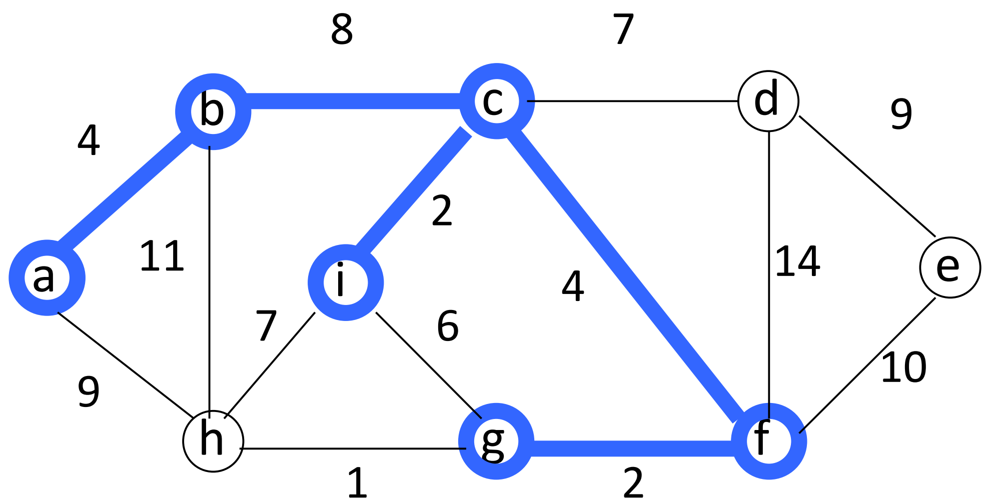
a
b
c
d
e
f
g
h
i
4
8
7
9
11
2
14
4
7
6
10
9
1
2


8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
10
9
h
g
f
1
2

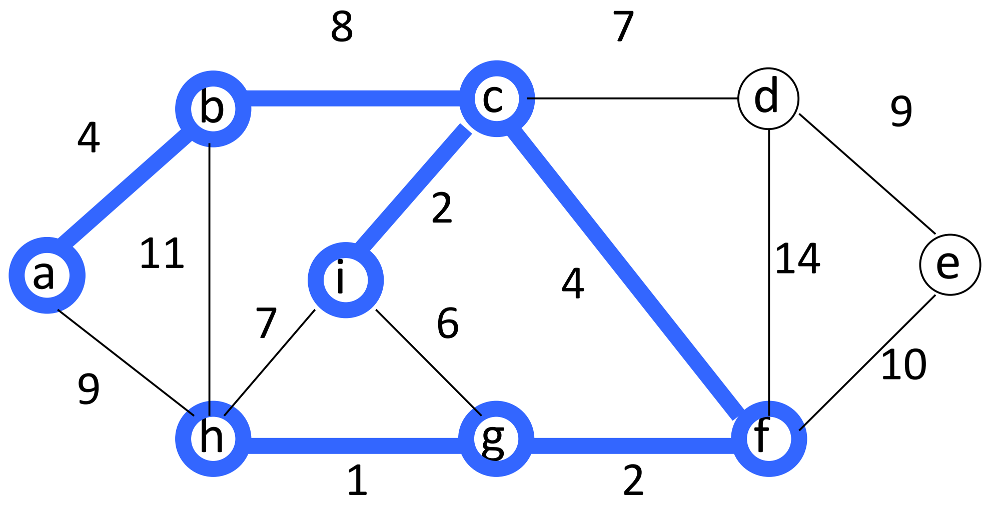

8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
10
9
h
g
f
1
2

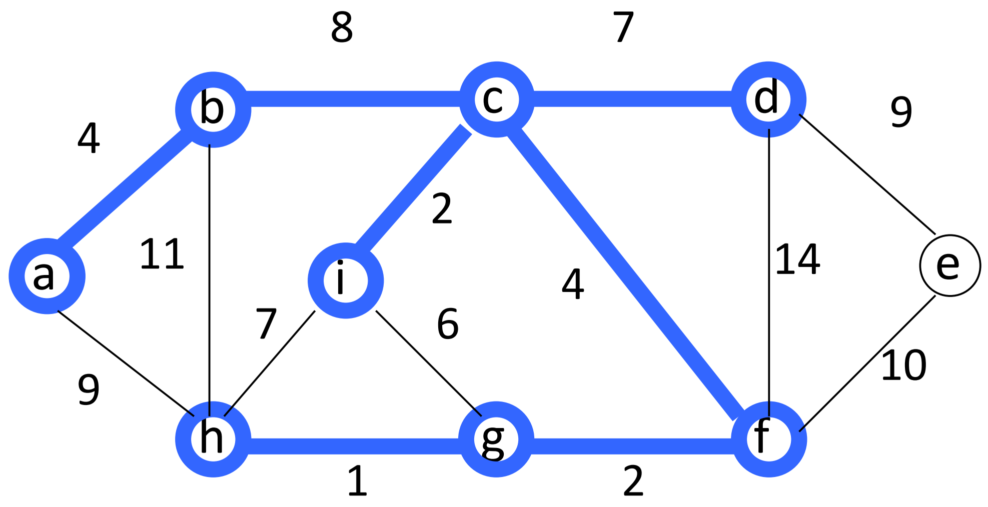


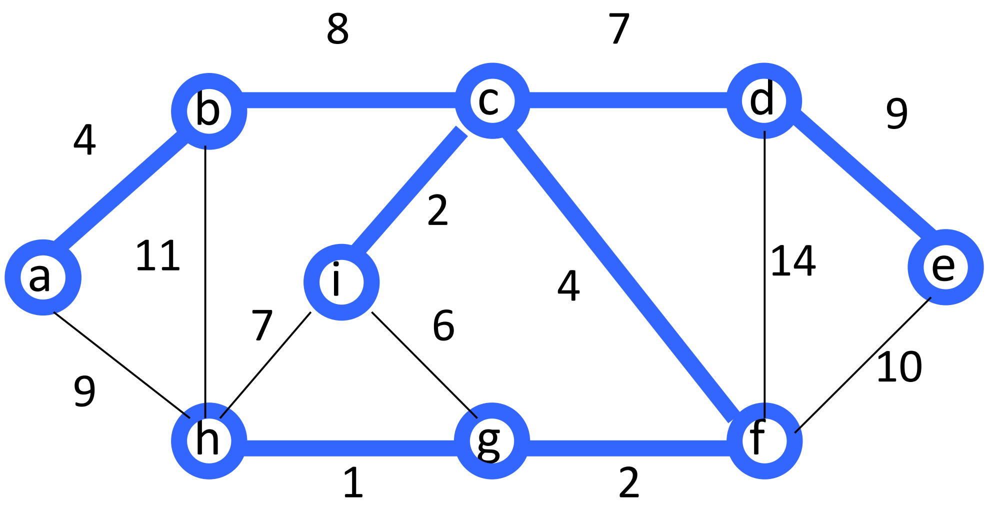
8
7
a
b
c
d
e
f
g
h
i
4
11
2
4
14
9
7
6
10
9
1
2

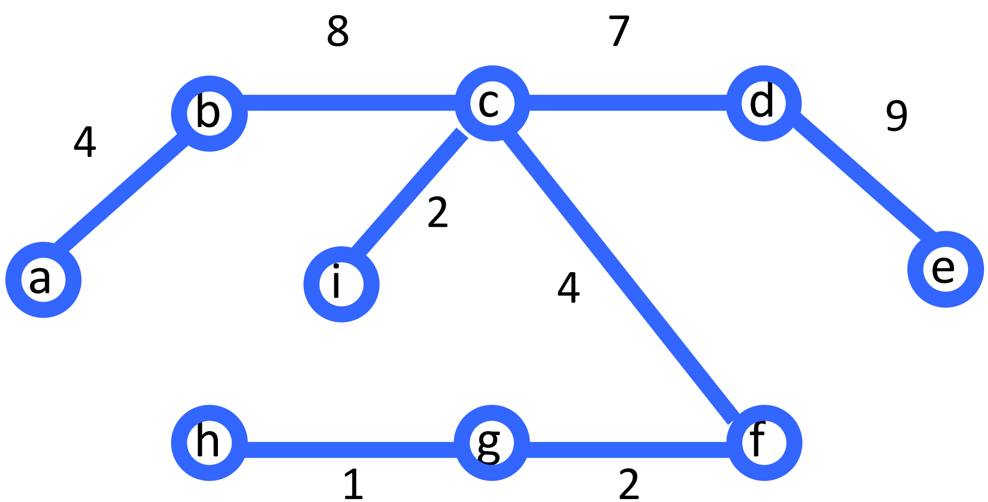


8
7
a
b
c
d
e
f
g
h
i
4
11
2
4
14
9
7
6
10
9
1
2

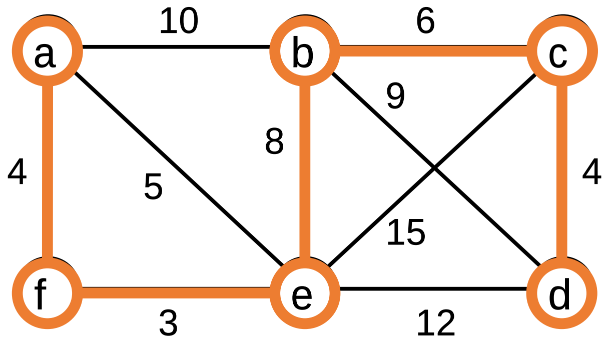

8
7
b
c
d
4
9
2
11
14
a
i
e
4
7
6
9
10
h
g
f
1
2


8
7
b
c
d
4
9
2
11
14
a
i
e
4
7
6
9
10
h
g
f
1
2

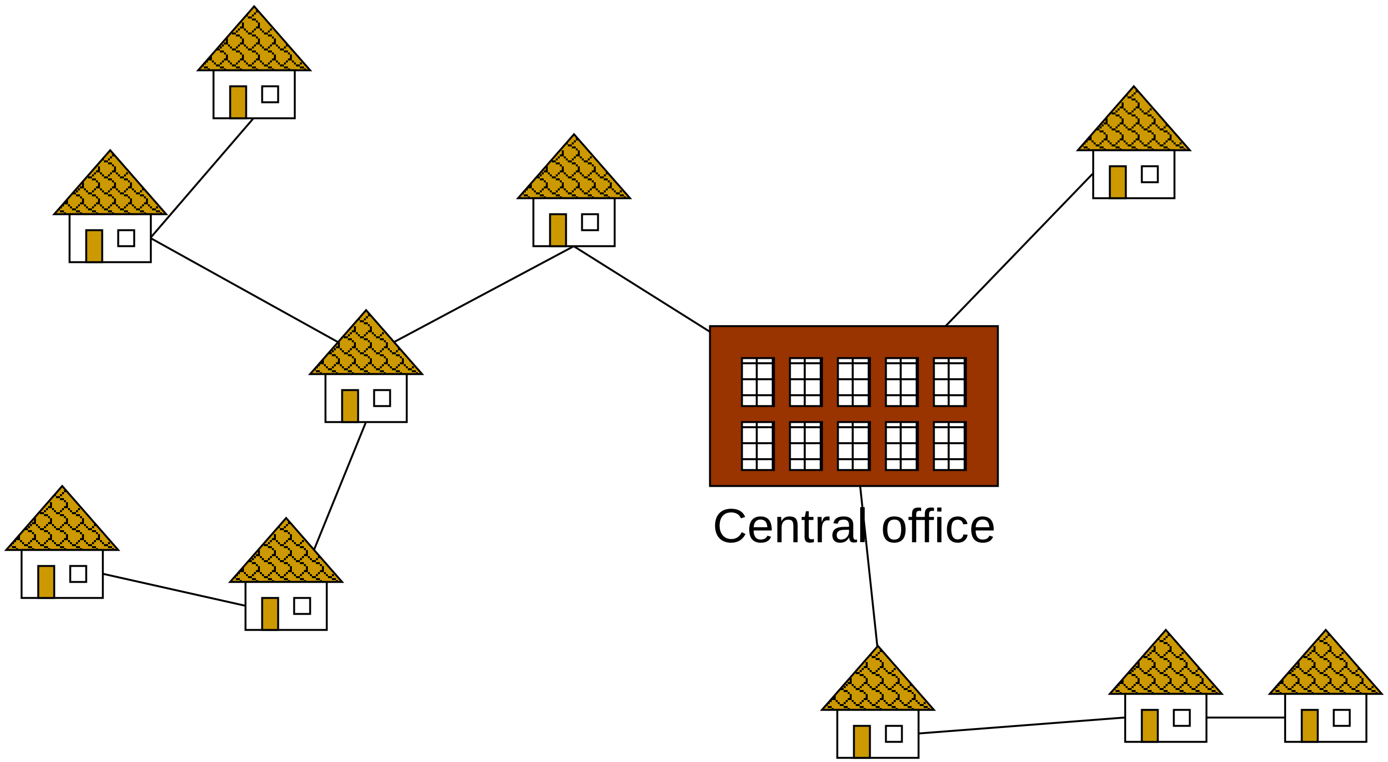

8
7
4
b
c
d
9
2
a
11
i
4
14
e
7
6
10
9
h
g
f
1
2


8
7
4
b
c
d
9
2
a
i
4
e
h
g
f
1
2

Example 2



How can we achieve this with MST?



Graph representation

- Vertices are all the nodes to be connected
- One edge for *every possible* connection
  - I.e. the complete graph of N vertices
- Each edge has a “weight” associated with it
  - Cost of running a wire from node A to node B

- Now find the MST
  - How does this solve the problem?
    - Spanning tree → all nodes are connected
    - Lowest cost tree → cheapest possible network