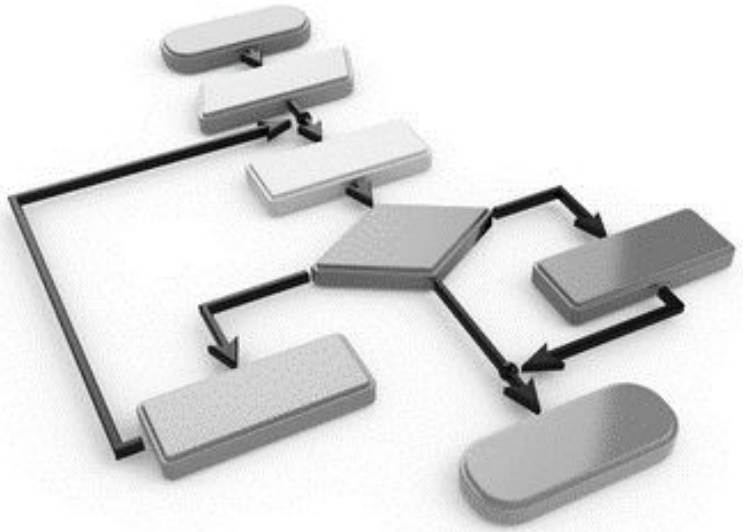


Graph-Based Algorithms

CSE 301: Combinatorial Optimization

Definition of MST

- Let $G = (V, E)$ be a connected, undirected graph.
- For each edge (u, v) in E, we have a weight $w(u, v)$ specifying the cost (length of edge) to connect u and v.
- We wish to find a (acyclic) subset T of E that connects all of the vertices in V and whose total weight is minimized.
- Since the total weight is minimized, the subset T must be acyclic (no circuit).
- Thus, T is a tree. We call it a spanning tree.
- The problem of determining the tree T is called the minimum-spanning-tree problem.

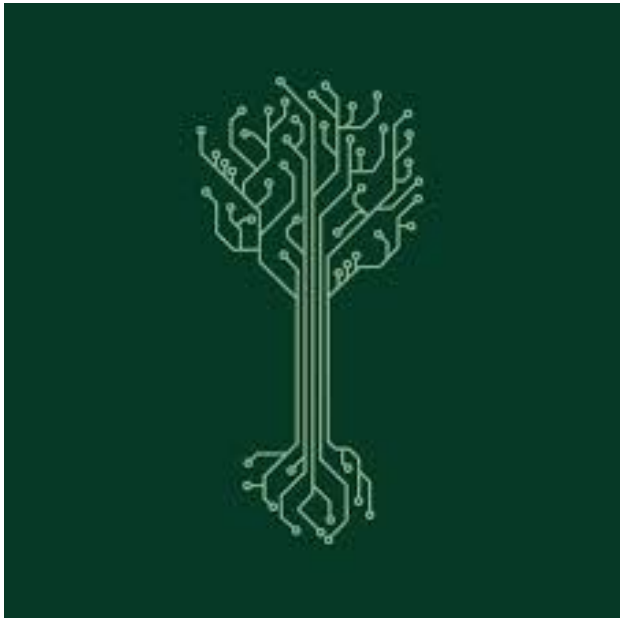
Application of MST

In the design of electronic circuitry, it is often necessary to make a set of pins electrically equivalent by wiring them together.

To interconnect n pins, we can use n-1 wires, each connecting two pins.

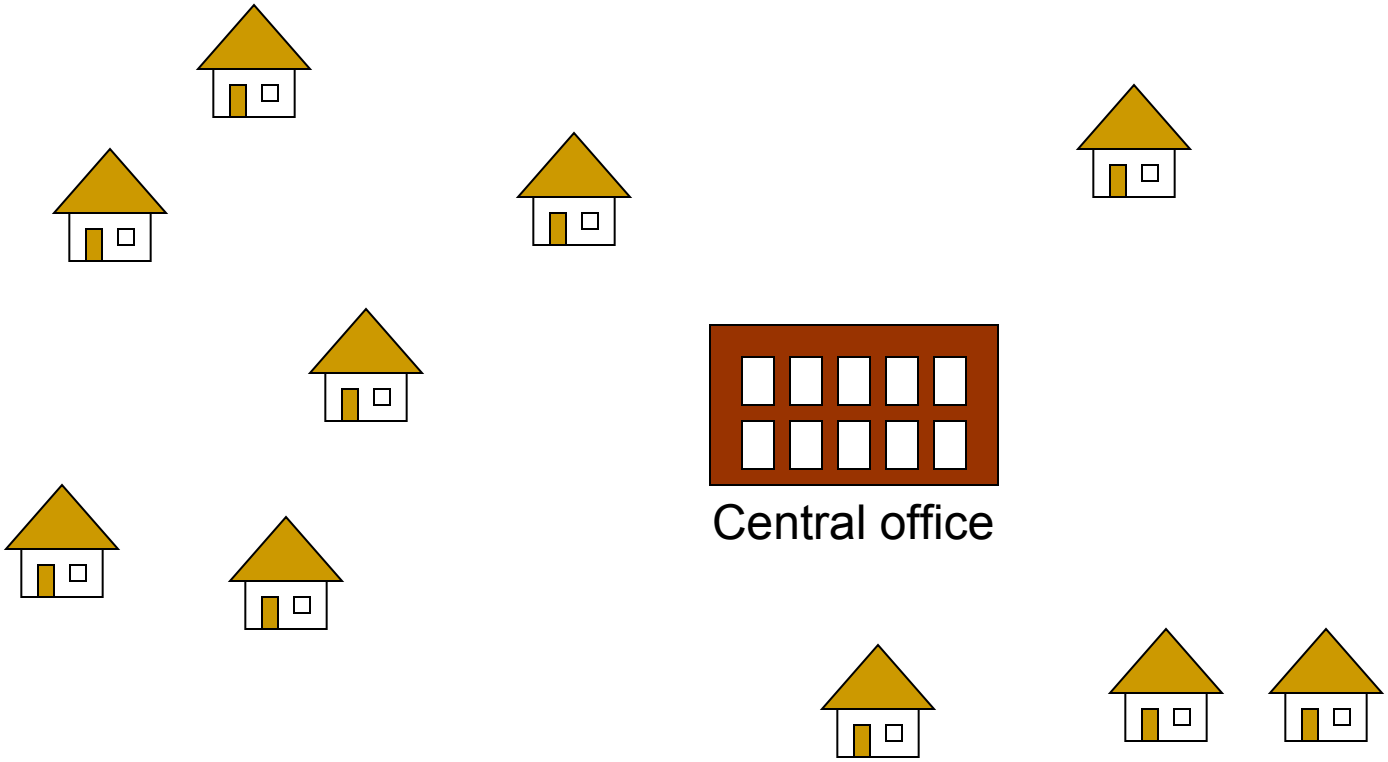
We want to minimize the total length of the wires.

Minimum Spanning Trees can be used to model this problem.



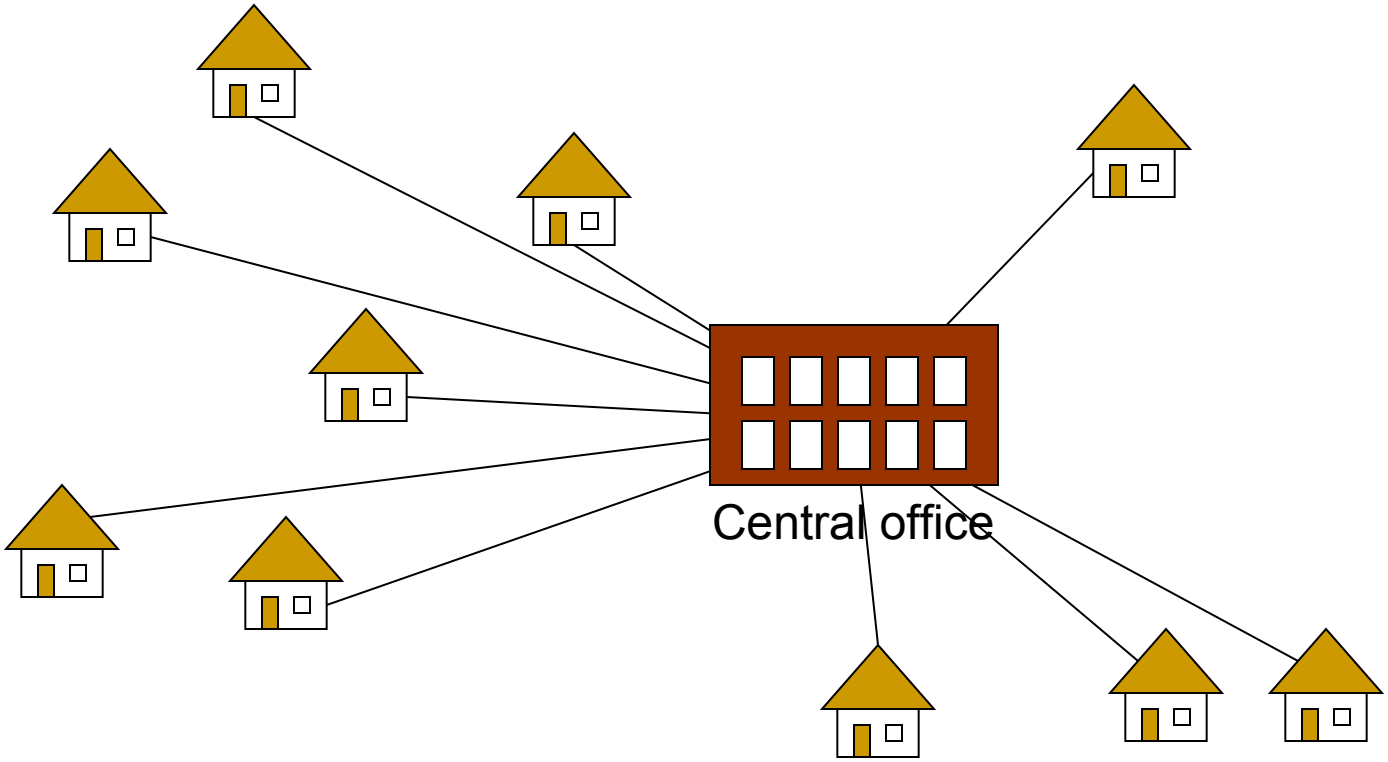
Application of MST

Problem: Laying Telephone Wire



Application of MST

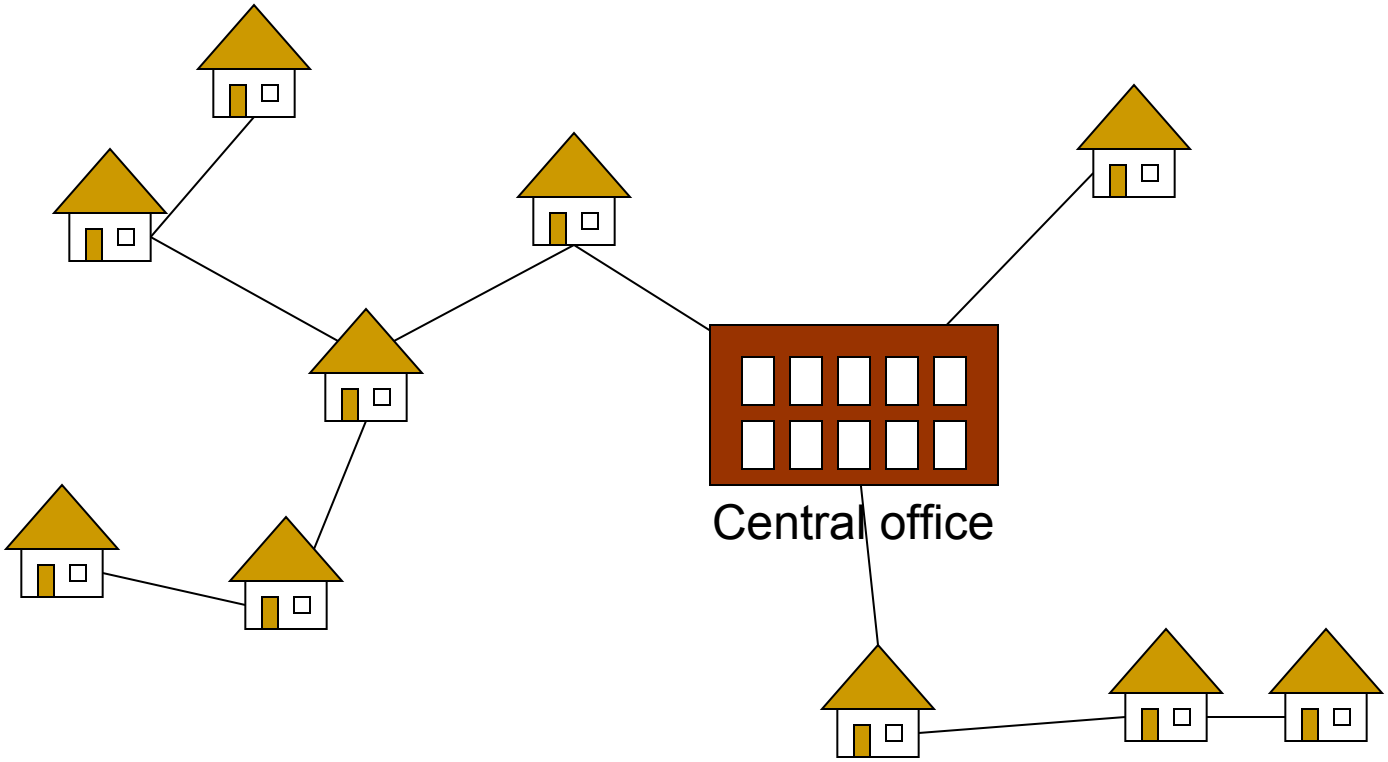
Wiring: Naïve Approach



Expensive!

Application of MST

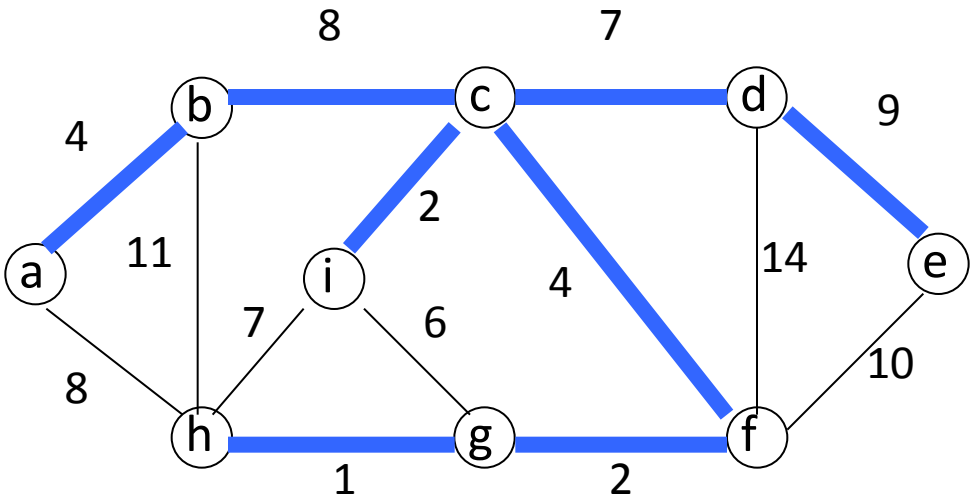
Wiring: Better Approach



Minimize the total length of wire connecting the customers

MST Example

Here is an example of a connected graph and its minimum spanning tree:



Notice that the tree is not unique:
replacing (b,c) with (a,h) yields another spanning tree with the same minimum weight.
ST: a collection of edges (here blue edges) which ensures that there is a path between every pair of nodes/vertices in the input graph via those edges.
MST: a ST whose total edge weight is minimum possible, i.e., there is no other ST whose total edge weight is less.

Growing a MST(Generic Algorithm)

Set A is always a subset of some minimum spanning tree. This property is called the invariant Property.

An edge (u, v) is a safe edge for A if adding the edge to A does not destroy the invariant.

A safe edge is just the CORRECT edge to choose to add to T.

```
GENERIC_MST(G,w)
1       A = ∅
2       while A does not form a spanning tree
3               find an edge (u,v) that is safe for A
4               A = A∪{(u,v)}
5       return A
```

The Algorithms of Kruskal and Prim

- The two algorithms are elaborations of the generic algorithm.
- They each use a specific rule to determine a safe edge in line 3 of GENERIC_MST.
- In Kruskal's algorithm,
  - The set A is a forest.
  - The safe edge added to A is always a least-weight edge in the graph that connects two distinct components.
- In Prim's algorithm,
  - The set A forms a single tree.
  - The safe edge added to A is always a least-weight edge connecting the tree to a vertex not in the tree.

Related Topics

Disjoint-Set (Chapter 21, Page 571)

- Keep a collection of disjoint sets: S_1, S_2, .., S_k,
  - Each S_i is a set, e,g, $S_1=\{v_1, v_2, v_8\}$.
- Three operations
  - **Make-Set(x)** - creates a new set whose only member is **x**.
  - **Union(x, y)** – unites the sets that contain **x** and **y**, say, $\mathbf{S_x}$ and $\mathbf{S_y}$, into a new set that is the union of the two sets.
  - **Find-Set(x)** - returns a pointer to the representative of the set containing **x**.

Kruskal's Algorithm

```
MST_KRUSKAL(G,w)
1    A = ∅
2    for each vertex v ∈ G.V
3        MAKE_SET(v)
4    sort the edges of G.E into nondecreasing order by weight w
5    for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6        if FIND_SET(u) ≠ FIND_SET(v)
7                A = A ∪ {(u, v)}
8                UNION(u, v)
9    return A
```

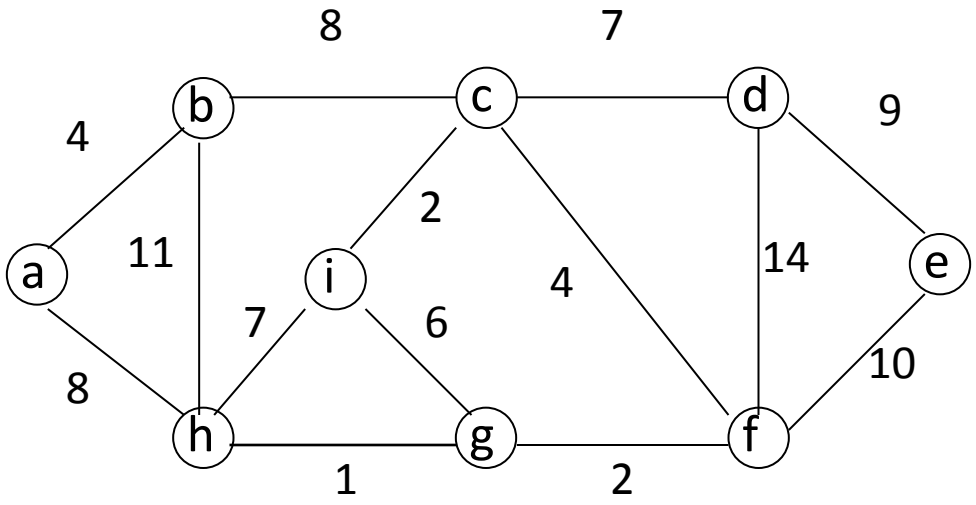
Time Complexity of Kruskal's Algorithm

- Line 4 sorts E edges => it takes O(E lgE) time, if we use MergeSort
  - But lg E = O(lg V), since $E < V^2$ => lg E < 2 lg V => lg E is O(lg V)
  - Hence O(E lg E) = O(E lg V)
- If we make total m calls of Make_set, Union, and Find_set in any algorithm, then these calls will take total O(m lg n) time, where n is the number of Make-Set calls.
- The 1st for loop makes O(V) MAKE_SET() calls and the 2nd for loop makes O(E) Find_SET and UNION calls
  - Total O(V+E) calls of MAKE_SET, FIND_SET, and UNION, out of which O(V) calls were MAKE_SET calls => O((V+E) lgV) time
  - $E \geq V-1$, since the graph is connected

    => V is O(E)=> O((V+E) lgV) = O(E lgV)
- Hence total time = O(E lg V)

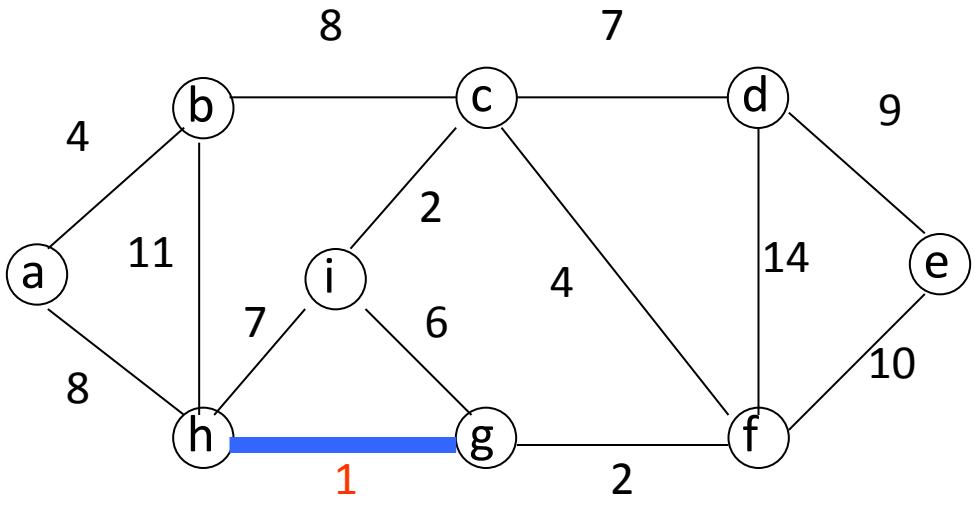
Kruskal's Algorithm: Example

- The edges considered by the algorithm are sorted by weight.
- The edge under consideration at each step is shown with a red weight number.



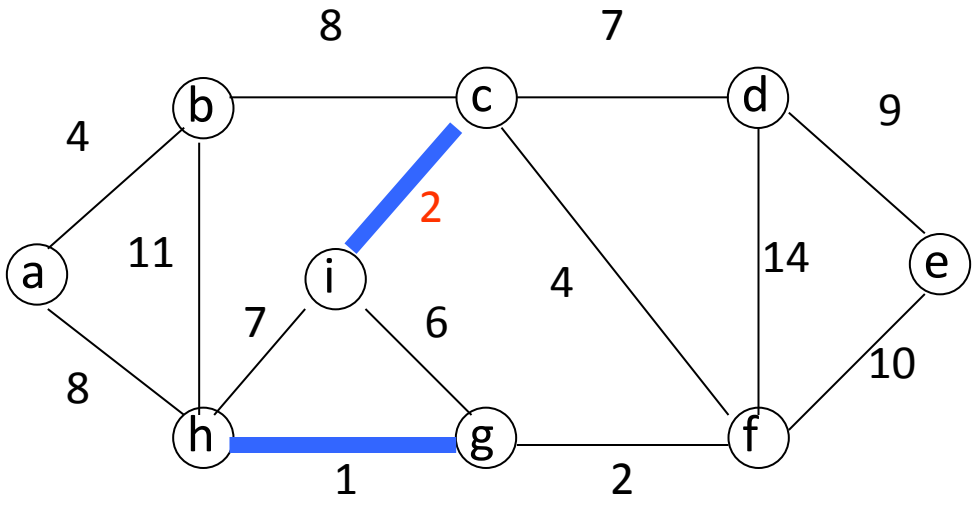
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



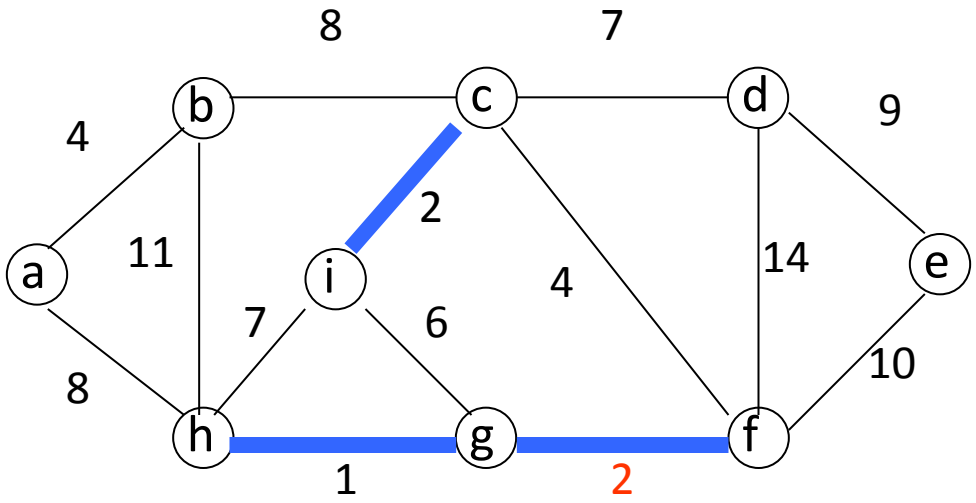
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



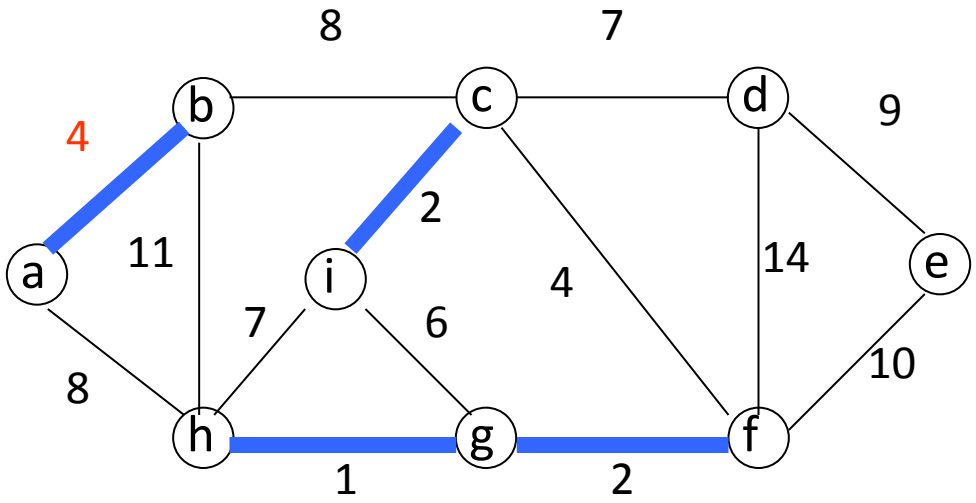
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



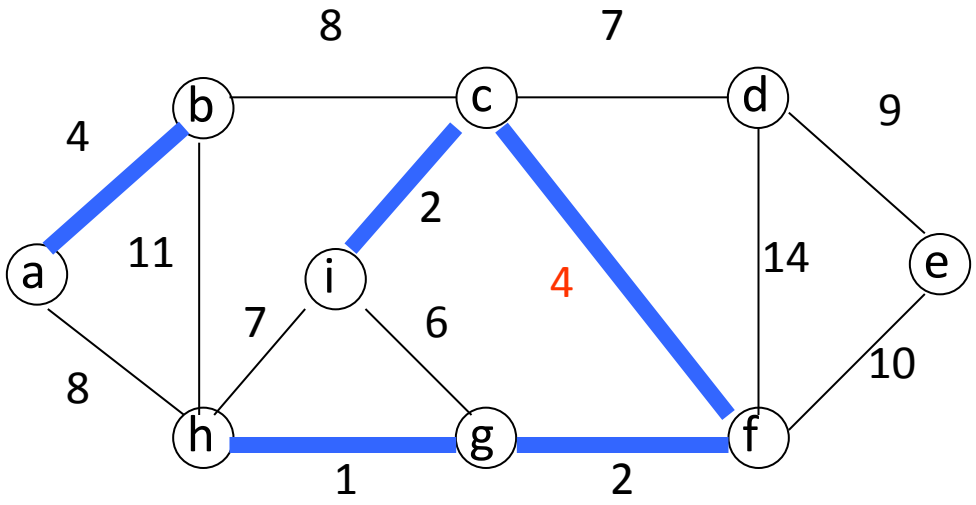
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



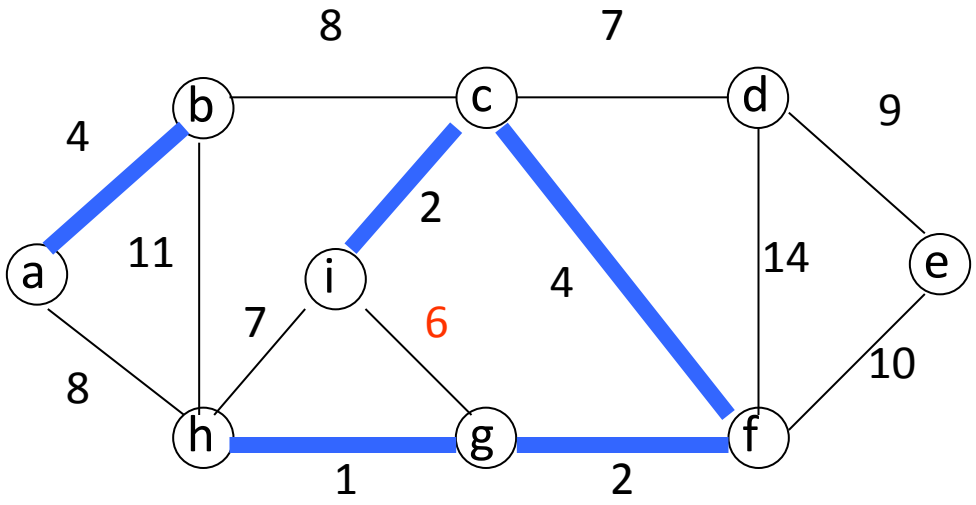
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



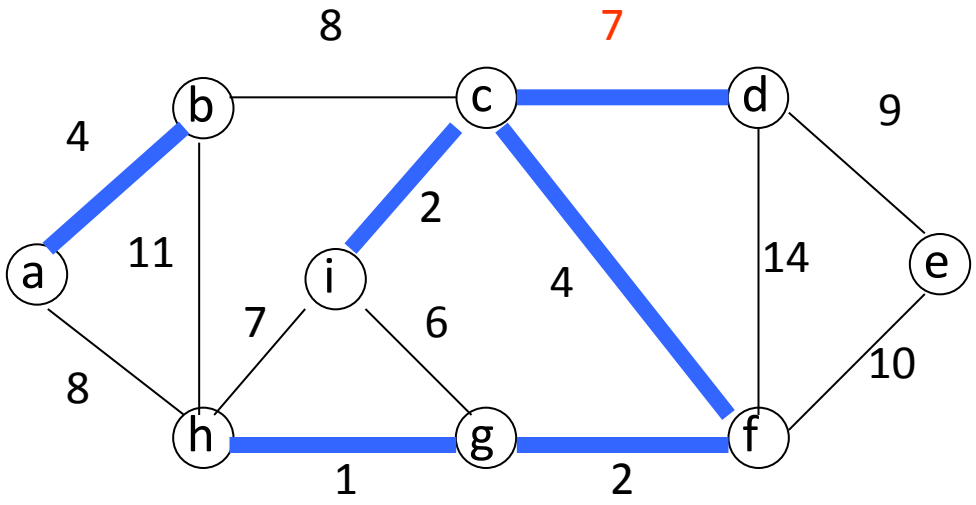
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



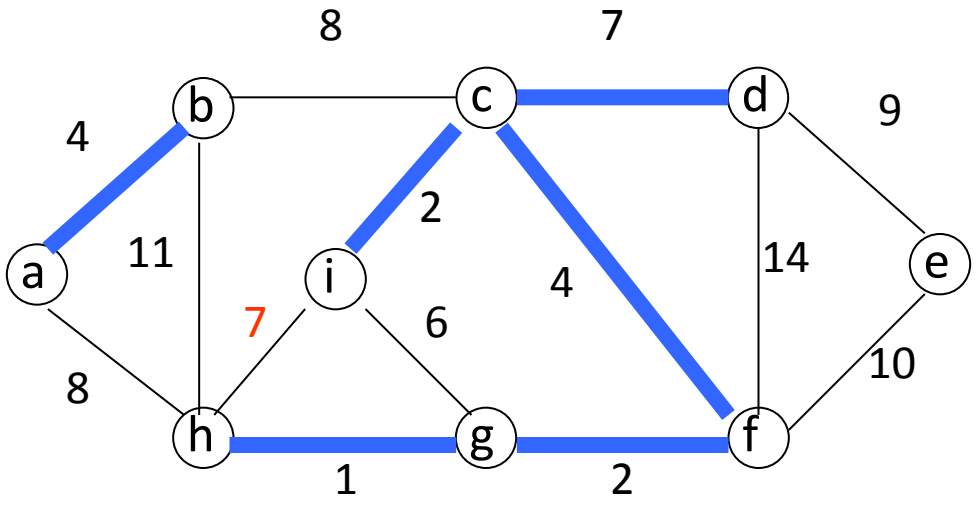
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



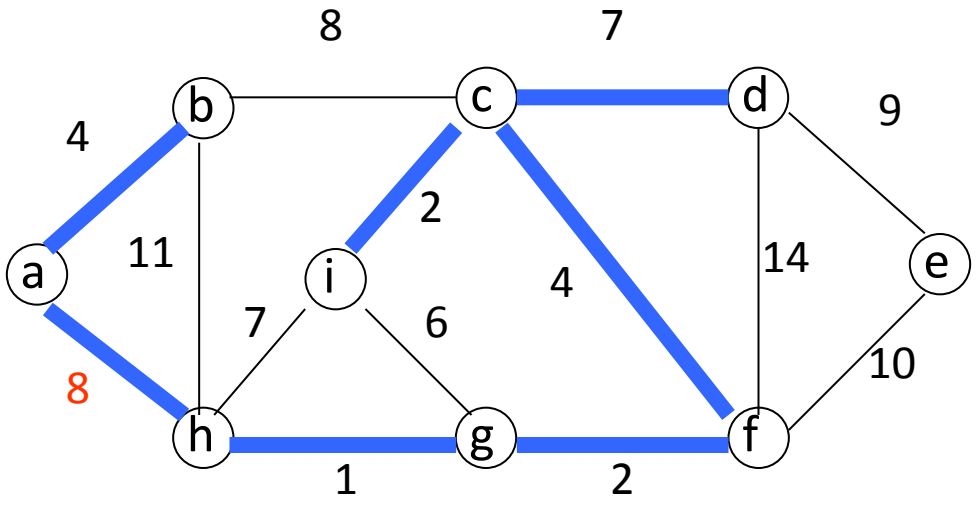
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



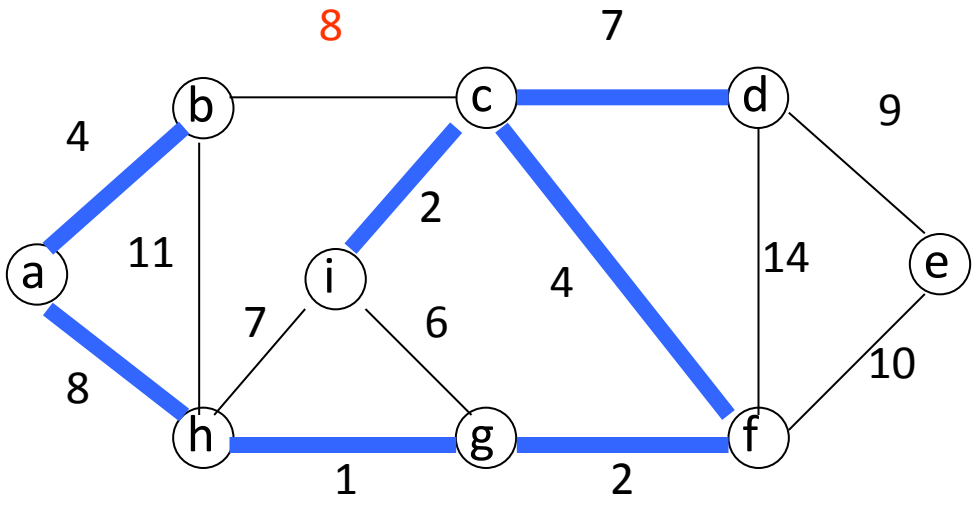
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



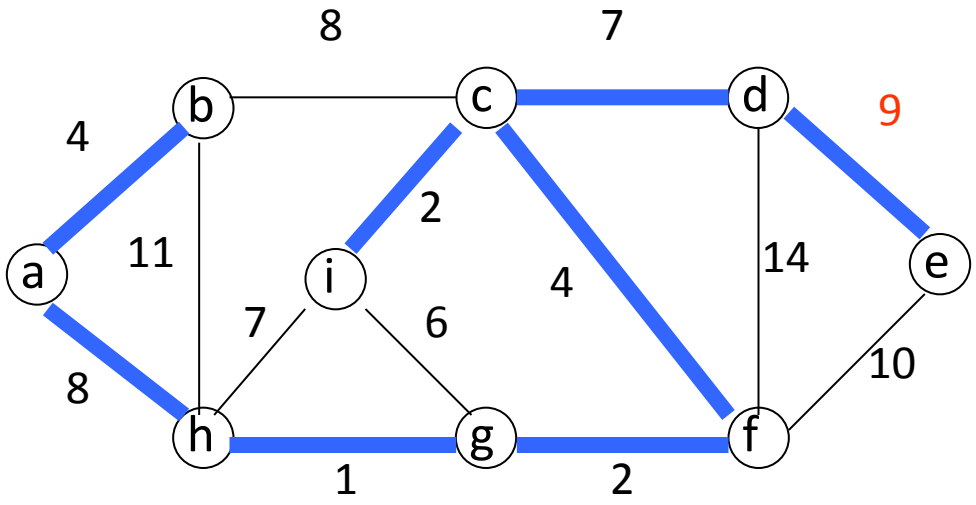
Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



Kruskal's Algorithm: Example

- The edges are considered by the algorithm in sorted order by weight.
- The edge under consideration at each step is shown with a red weight number.



Prim's Algorithm

```
MST_PRIM(G,w,r)
1     for each u in G.V
2         u.key = ∞
3         u.π =NIL
4      r.key = 0
5     Q = G.V    //Q is a min-priority queue
6     while Q ≠ ∅
7         u =EXTRACT_MIN(Q)
8         for each v ∈ G.Adj[u]
9                 if v ∈ Q and w(u,v) < v.key
10                    v.π = u
11                    v.key = w(u,v)
```

Prim's Algorithm

Grow the minimum spanning tree from the root vertex r.

Q is a priority queue, holding all vertices that are not in the tree now.

key[v] is the minimum weight of any edge connecting v to a vertex in the tree.

parent[v] names the parent of v in the tree.

When the algorithm terminates, Q is empty; the minimum spanning tree A for G is thus $A=\{(v,parent[v]):v \in V-\{r\}\}$.

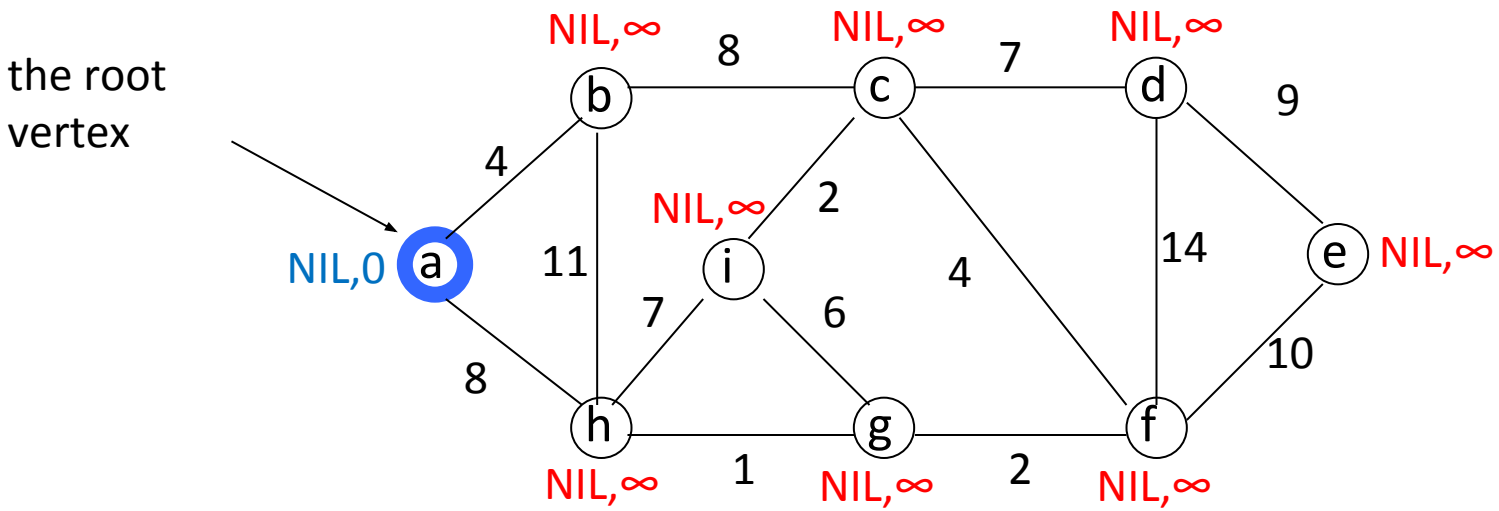
Running time: O(Elg V).

Running time of Prim's Algorithm

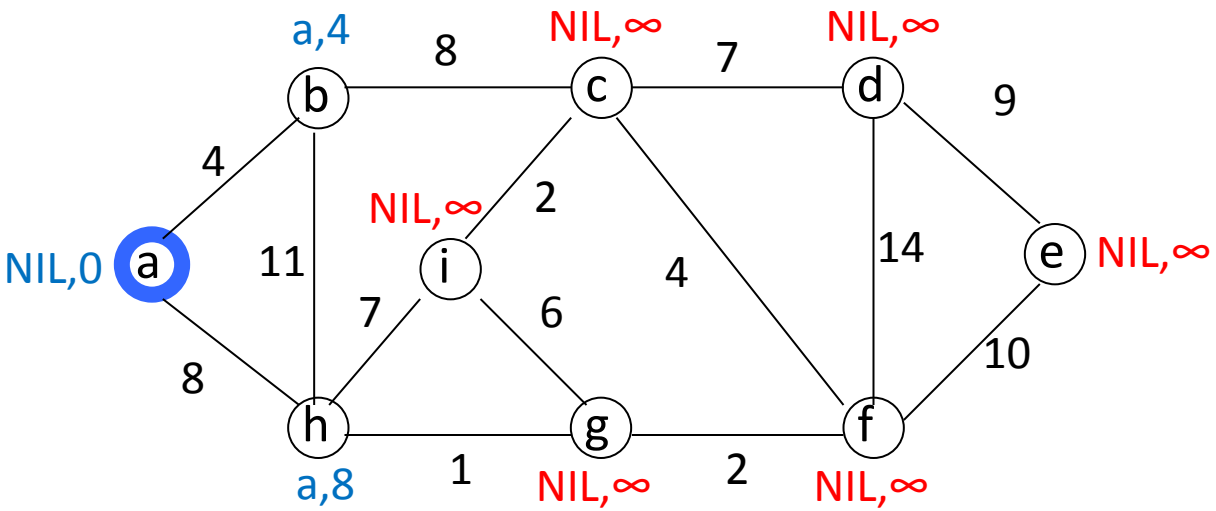
```
MST_PRIM(G,w,r)
1     for each u in G.V                    // O(V)
2         u.key = ∞
3         u.π = NIL
4     r.key = 0
5     Q = G.V    //Q is a min-priority queue   //Θ(V) time to build the heap
6     while Q ≠ ∅
7         u = EXTRACT_MIN(Q)   //V Heap-Extract-Min operations: O(V lg V)
8         for each v ∈ G.Adj[u]
9                 if v ∈ Q and w(u,v) < v.key
10                v.π = u
11                v.key = w(u,v)   //E Heap-Decrease-Key operations: O(E lg V)
```

Total time = O((V+E) lg V) = O(E lg V) since E ≥ V-1 => V is O(E)

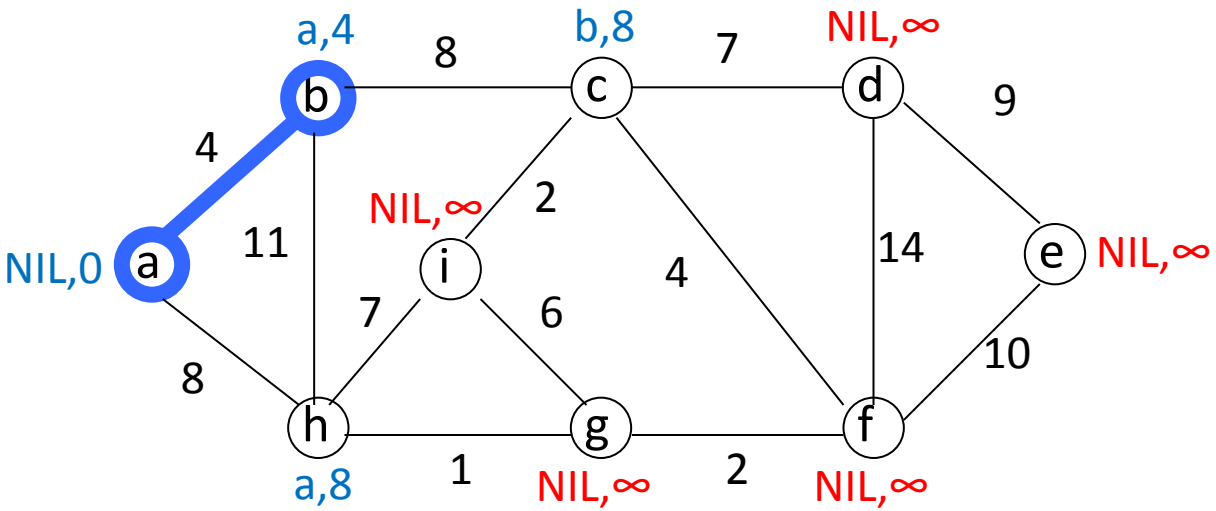
Prim's Algorithm: Example



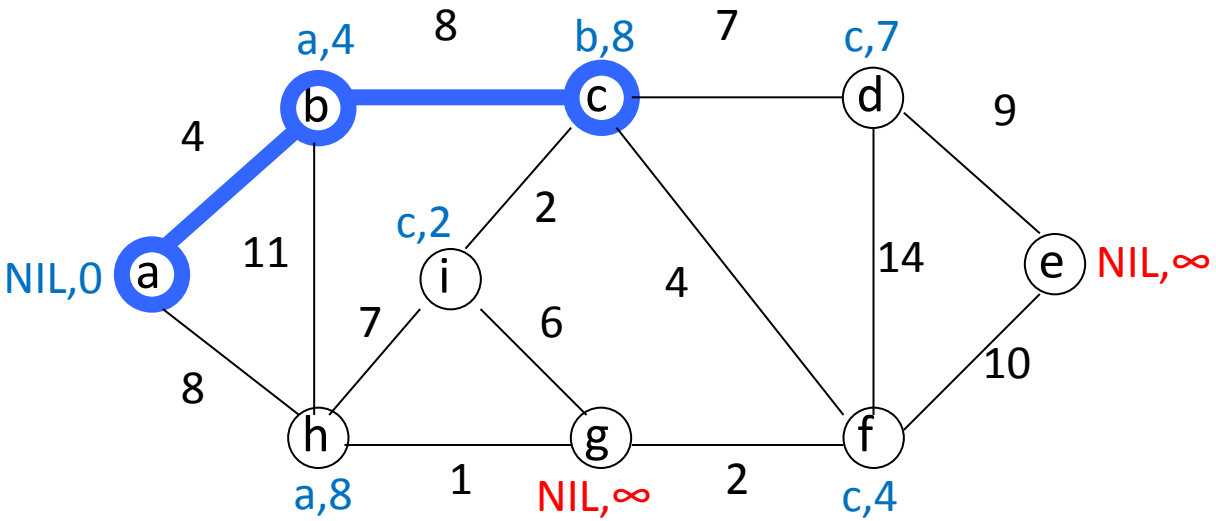
Prim's Algorithm: Example



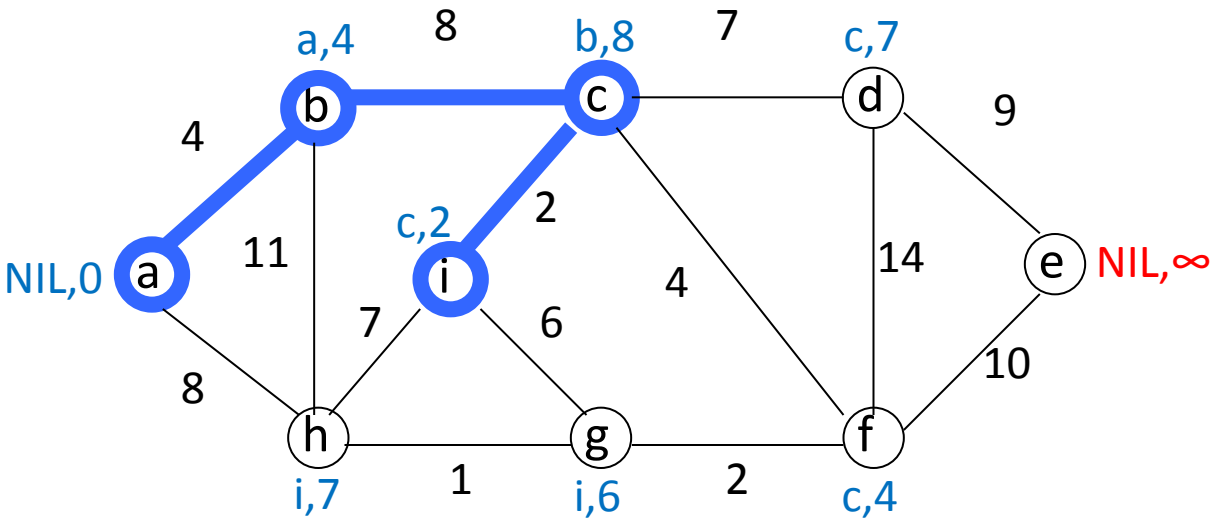
Prim's Algorithm: Example



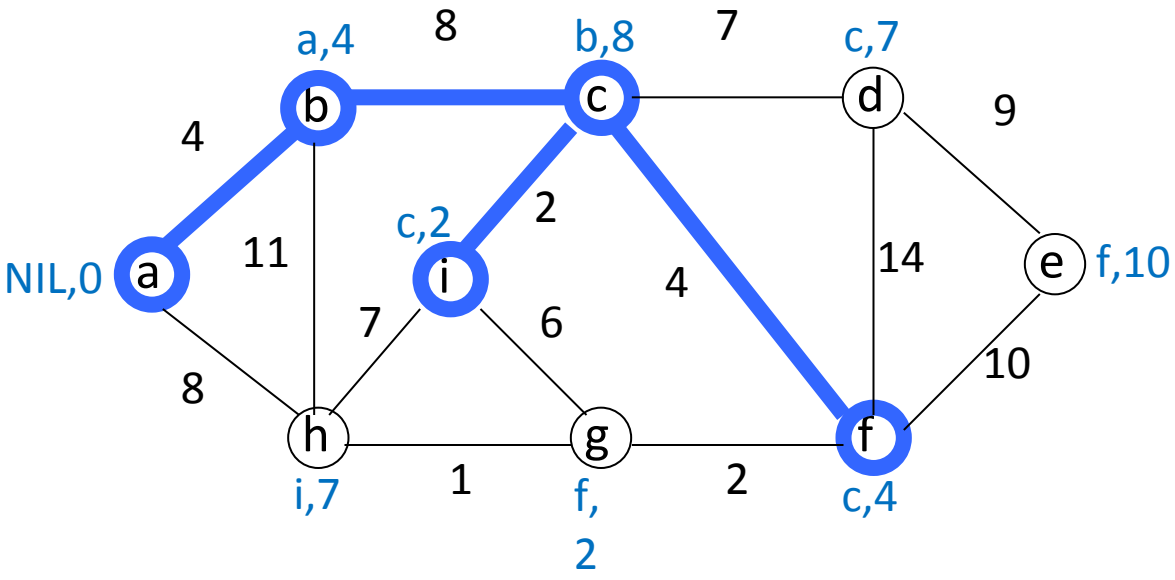
Prim's Algorithm: Example



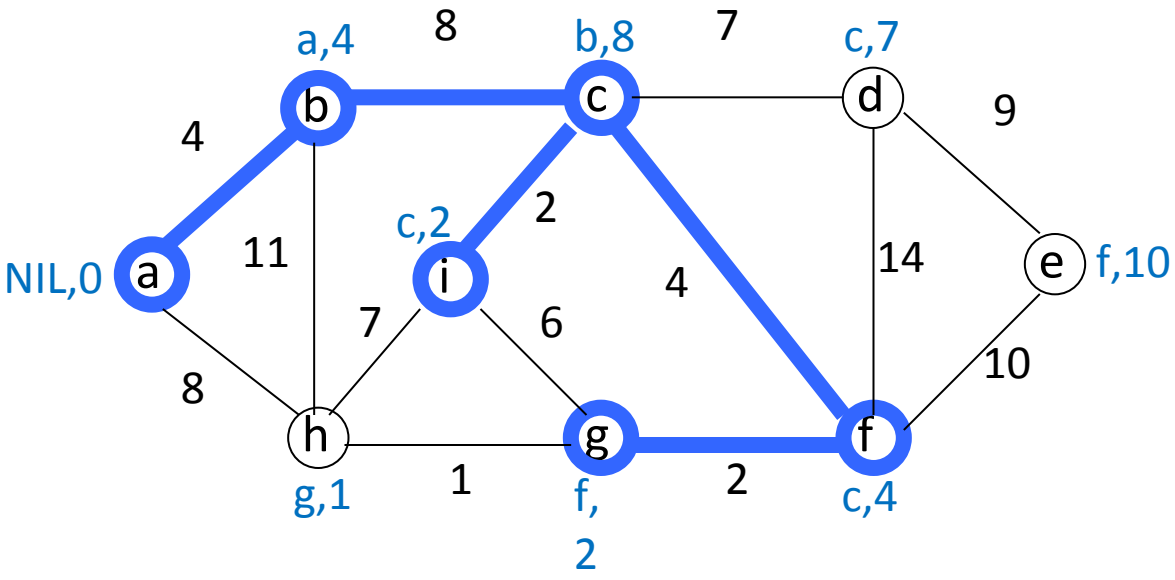
Prim's Algorithm: Example



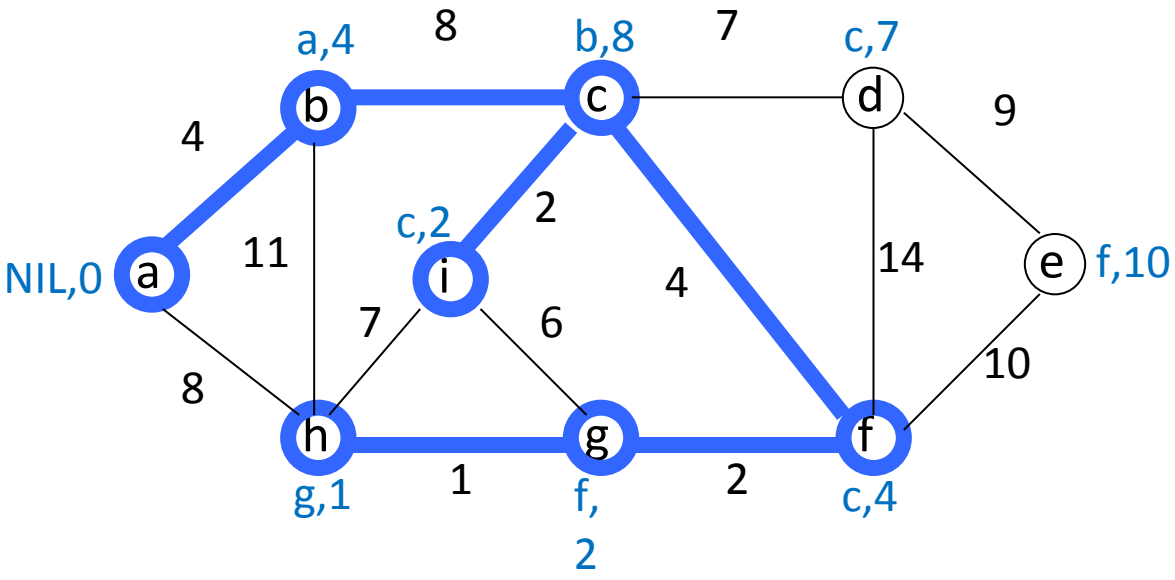
Prim's Algorithm: Example



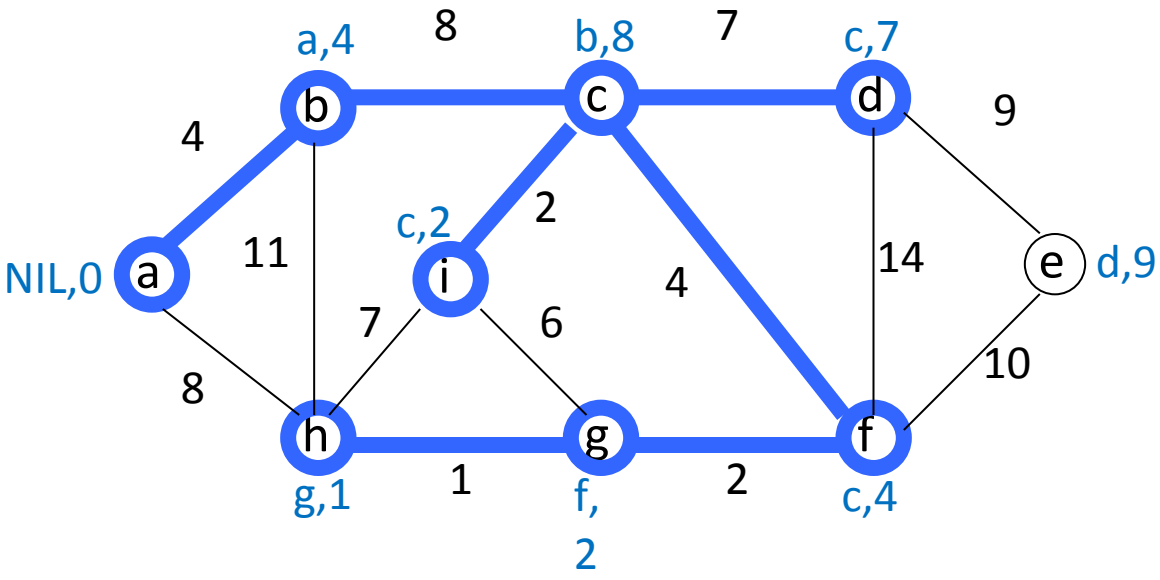
Prim's Algorithm: Example



Prim's Algorithm: Example



Prim's Algorithm: Example



Prim's Algorithm: Example