Minimal Spanning Trees

Chapter 23

Cormen Leiserson Rivest&Stein:
Introduction to Algorithms

Minimal Spanning Trees

Weighted Graphs $G(V, E, w)$

$W: E \rightarrow R$

If $w=1$ for all edges BFS is the solution.

The MST is the way of connecting n vertices at minimal cost of connections.

Two greedy algorithms : Kruskal

Prim

Minimal Spanning Trees

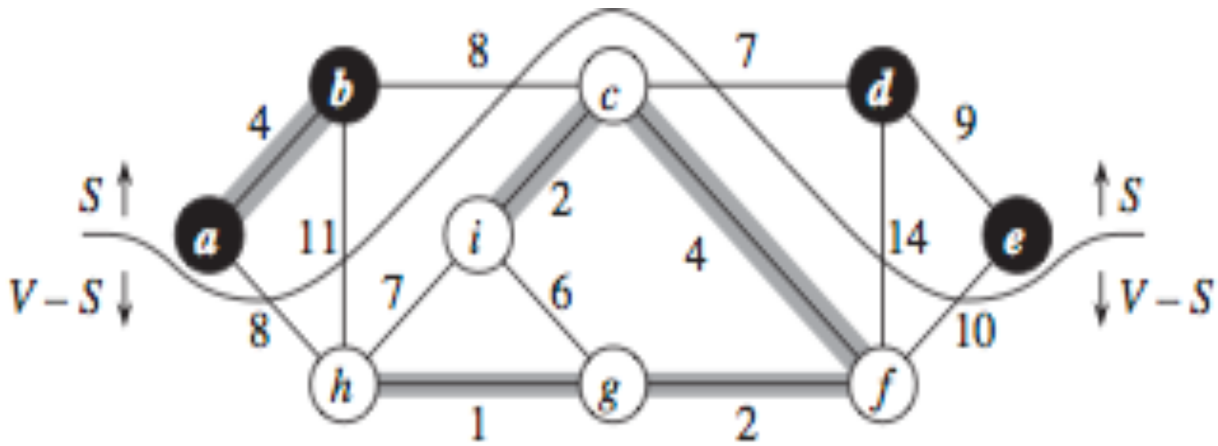
First a generic method utilized by the two algorithms:

```
GENERIC-MST(G, w)
A = ∅
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    A = A ∪ {(u, v)}
return A
```

Prior of each iteration, *A* is a subset of a MST

Determine a safe edge (u,v): *A* U (u,v) is still a subset of a MST.

Minimal Spanning Trees



(a)

S vertices in MST (black). V-S vertices to be selected. The line is the cut. Light vertices crossing the cut can be selected for the MST.

They are safe!

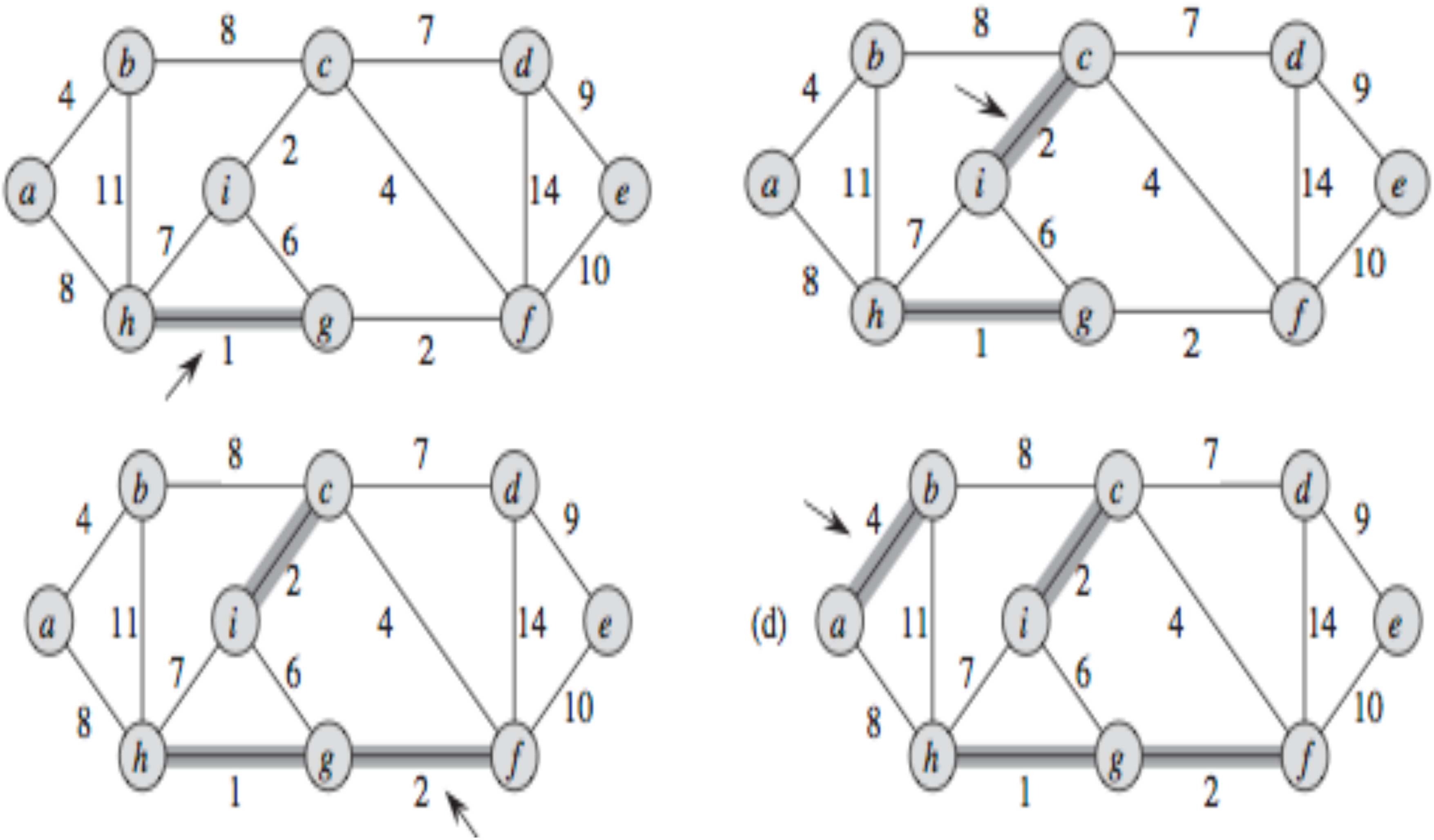
Kruskal Algorithm for MST

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

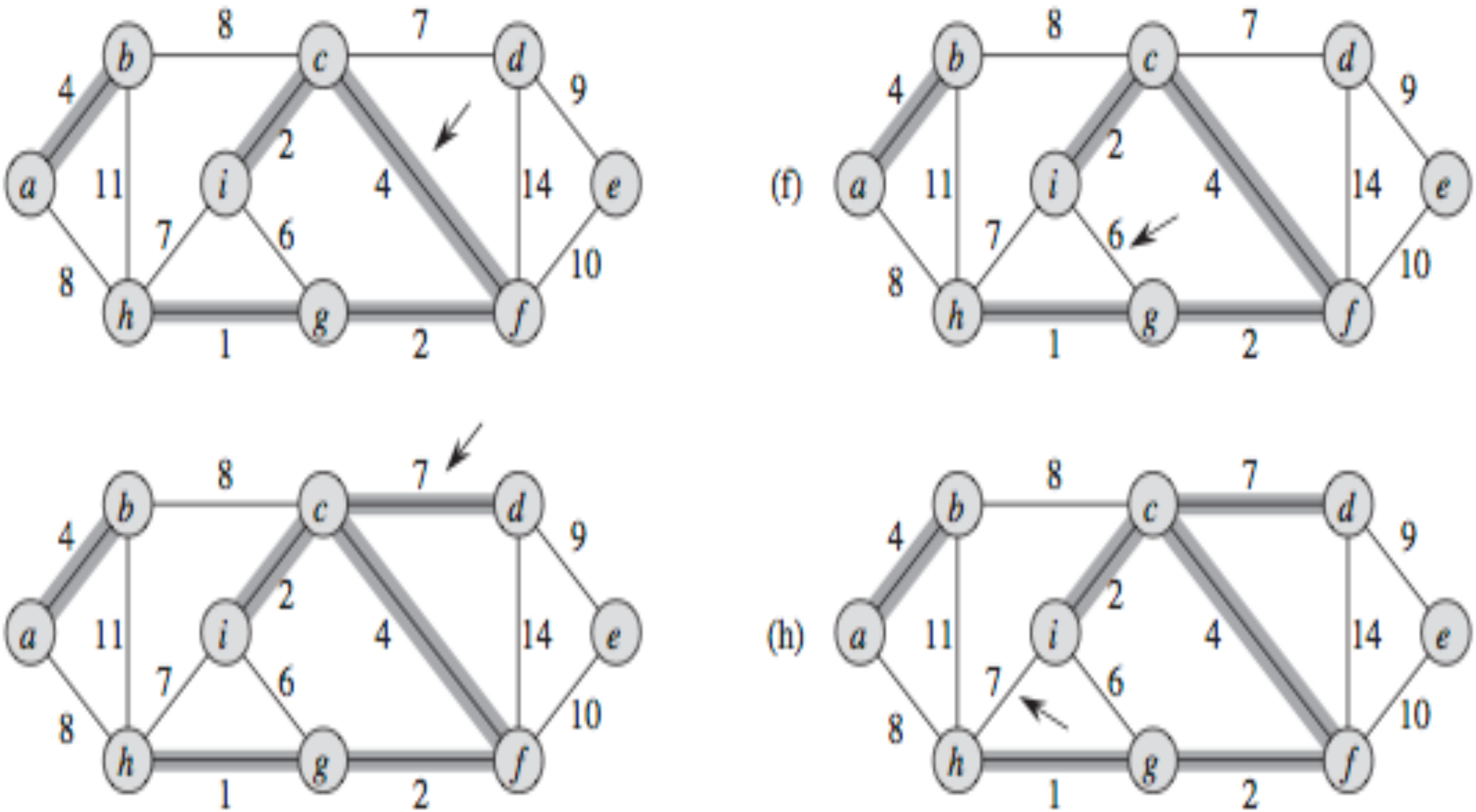
Uses a disjoint-set data structure to maintain several disjoint sets of elements. FIND-SET(u) returns a representative element of the set containing u.

(Disjoint set forest chapter. 21)

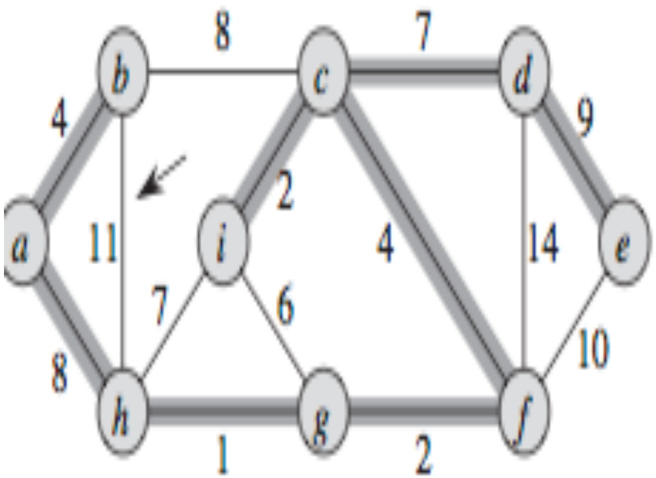
Kruskal Algorithm for MST



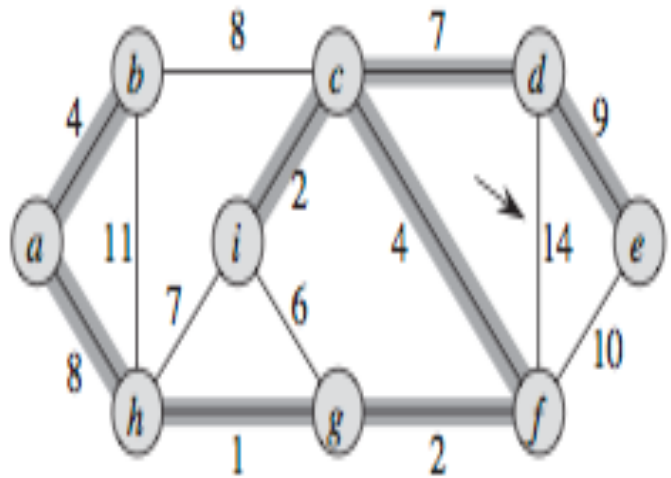
Kruskal Algorithm for MST



Kruskal Algorithm for MST



(n)



Kruskal Algorithm complexity

Complexity depends on how we implement the disjoint-set data structure.

(Disjoint set forest chapter. 21.3)

Sorting of the edges $O(E \log E)$

Loop 5-8 $O(E)$ FIND-SET and UNION operations on the disjoint set forest.

$|V|$ operations of MAKE-SET

In total the loop takes $O((V+E)\,\alpha(V))$ time α is very slowly growing function. Since $E \geq V-1$ and $\alpha(V)=O(\log V) = O(\log E)$ the loop takes $O(E \log E)$.

In total the Kruskal alg. takes $O(E \log E) = O(E \log V)$

PRIM algorithm for MST

Starts from an arbitrary vertex r the root.

The set A forms a single tree at any step.

All vertices not in the spanning tree form a min Heap Q based on the minimum weight of any edge connecting v and the tree. Value v.key.

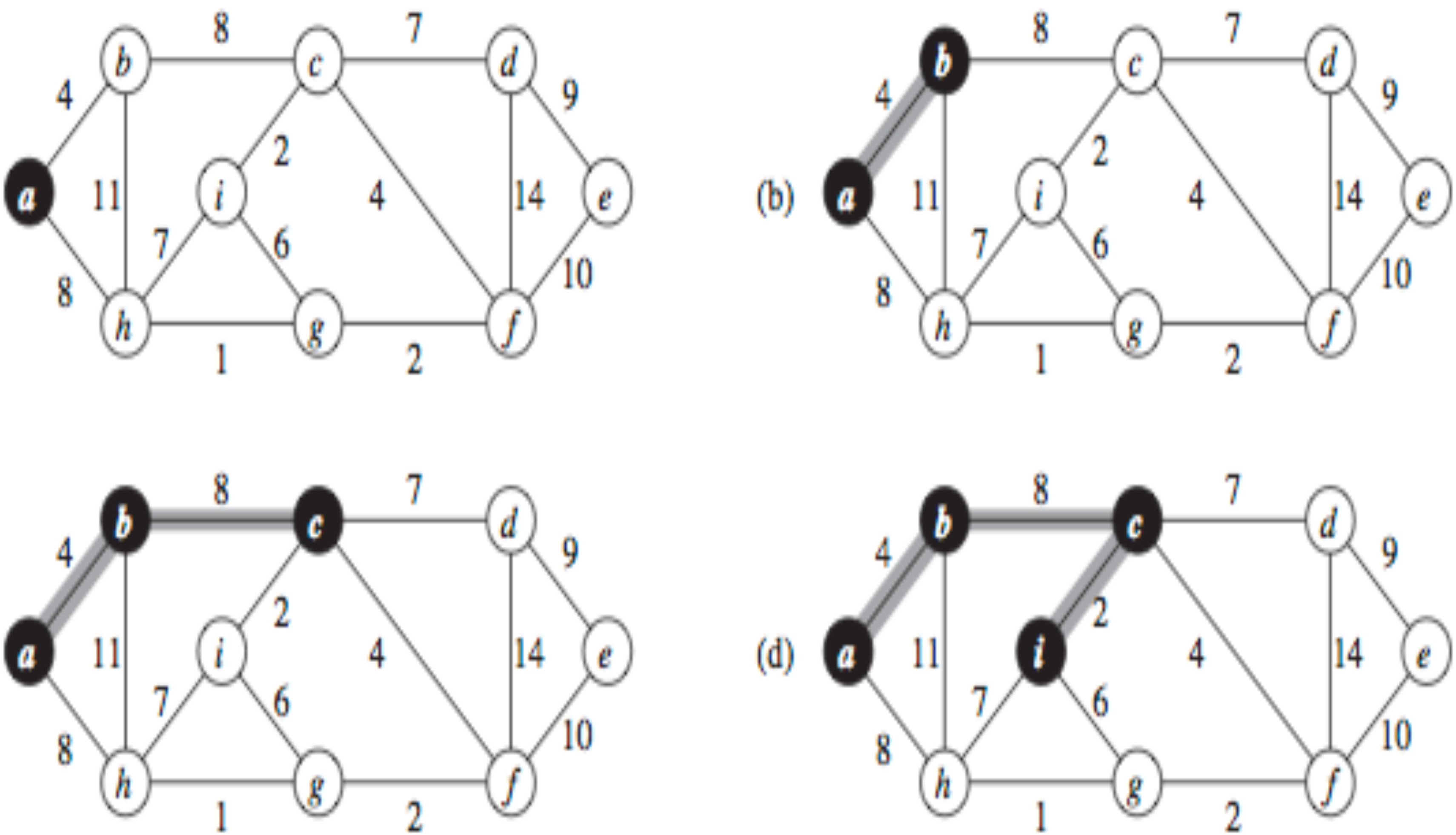
The termination condition is that the heap Q is empty, that means that all vertices have been connecte to the MST.

PRIM algorithm for MST

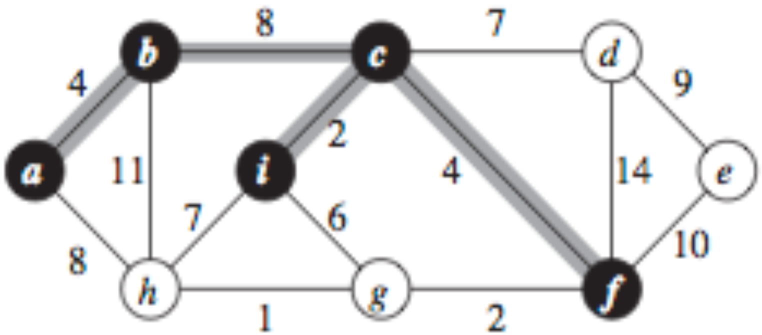
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

The parent in the MST

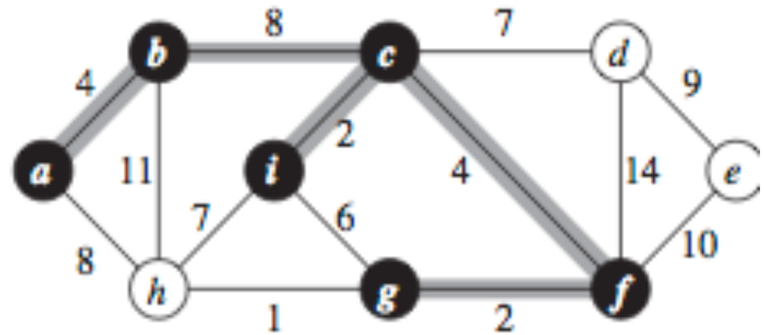
PRIM algorithm for MST

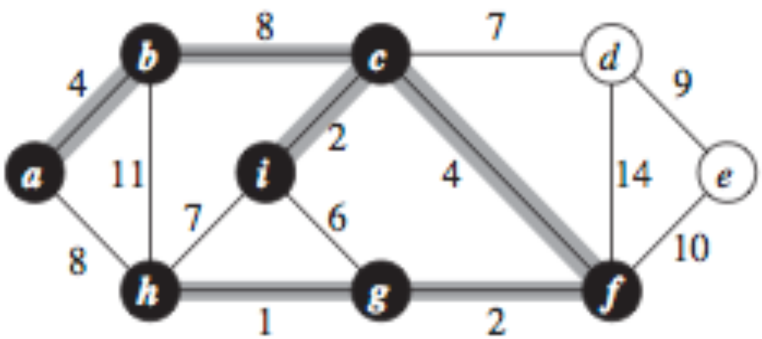


PRIM algorithm for MST

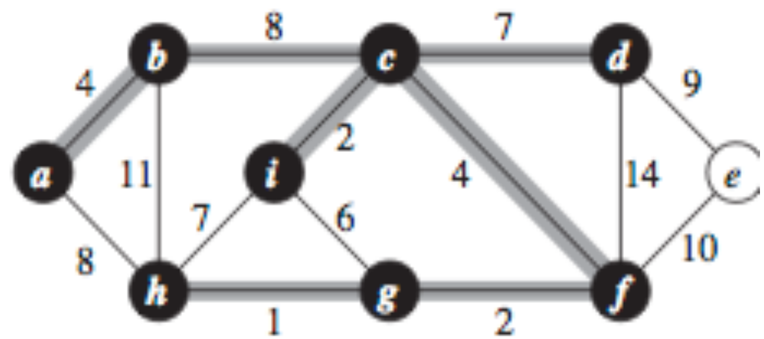


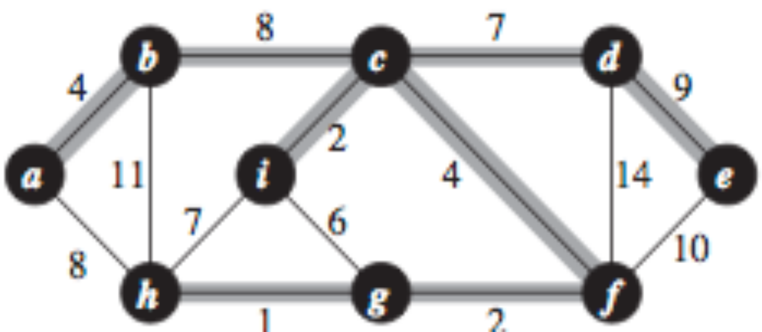
(f)





(h)





Loop 6-11 invariant

1. $A = \{(v, v.\pi) : v \in V - \{r\} - Q\}$.
2. The vertices already placed into the minimum spanning tree are those in $V - Q$.
3. For all vertices $v \in Q$, if $v.\pi \neq$ NIL, then $v.key < \infty$ and $v.key$ is the weight of a light edge $(v, v.\pi)$ connecting v to some vertex already placed into the minimum spanning tree.

Complexity PRIM algorithm

Row 1-5 : Build min heap takes $O(V)$

The while loop is executed $O(V)$ times and EXTRACT-MIN take $O(\log V)$ time hence in totla $O(V \log V)$

The for loop is executed $O(E)$ in total and the last operation is a DECREASE-KEY operation $O(\log V)$.

Prims algorithm takes $O(V \log V + E \log V) = O(E \log V)$ the same as Kruskal algortihm! It can be improved to:

$O(E + V \log V)$

using for Q the data structure Fibonacci Heap