Algorithms II, CS 502

Minimum Spanning Trees

Ugur Dogrusoz

Computer Eng Dept, Bilkent Univ

Minimum spanning trees

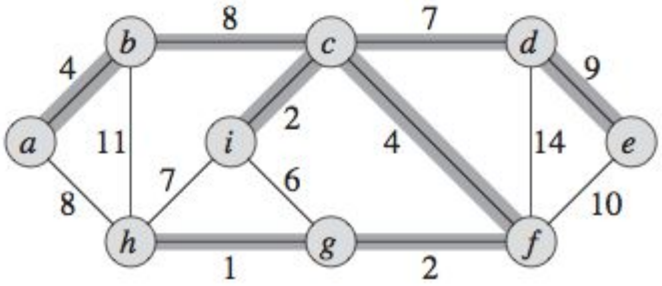
- Given a connected undirected graph G=(V,E), find an acyclic subset T of E that connects all the vertices and its total weight is minimized.

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

- The problem of determining the tree T is called the minimum-spanning-tree problem

Minimum spanning trees

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



Minimum spanning trees

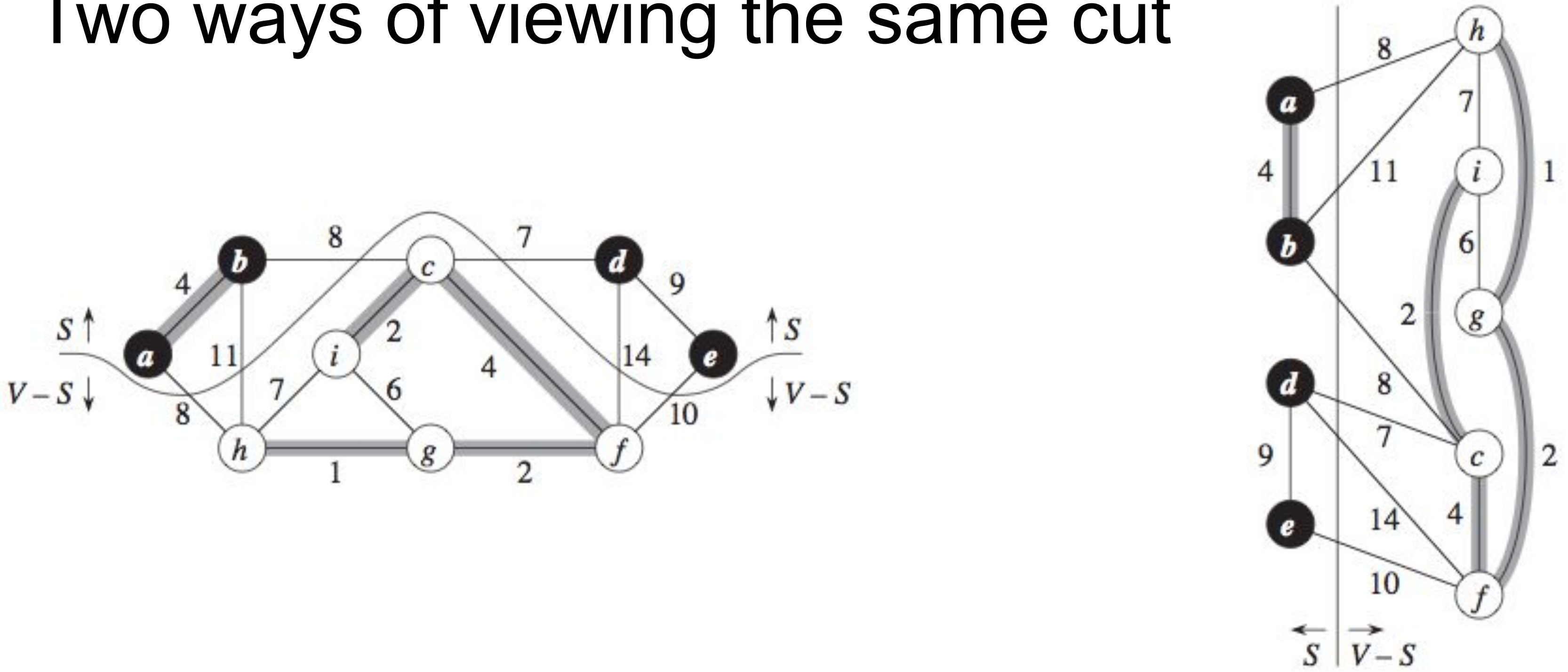
- The invariant here is that A is a subset of some minimum-spanning-tree using a greedy approach.
- An edge is called a safe edge for A, since we can add it safely to A while maintaining the invariant.

Minimum spanning trees

- A cut (S,V-S) of an undirected graph G=(V,E) is a partition of V.
- An edge (u,v) in E crosses the cut (S,V-S) if one of its endpoints is in S and the other is in V-S.
- A cut respects a set A of edges if no edge in A crosses the cut.
- An edge is a light edge crossing a cut if its weight is the minimum of any edge crossing the cut.

Minimum spanning trees

Two ways of viewing the same cut



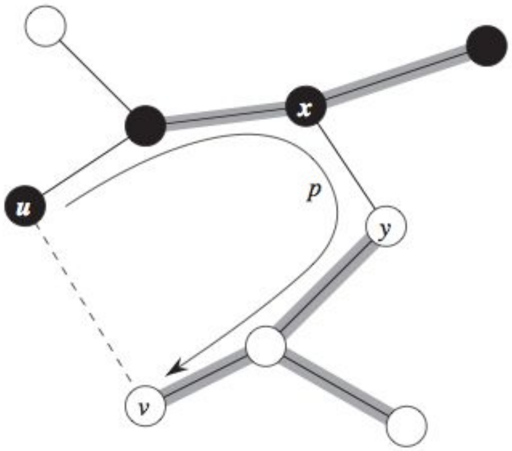
Minimum spanning trees

- **Theorem 23.1** Let G=(V,E) be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let (S,V-S) be any cut of G that respects A, and let (u,v) be a light edge crossing (S,V-S).Then, edge (u,v) is safe for A.

Minimum spanning trees

- **Proof** Use a cut-and-paste argument to construct another minimum spanning tree that includes A U {(u,v)}

$$w(T') = w(T) - w(x,y) + w(u,v)$$
$$\leq W(T)$$



Minimum spanning trees

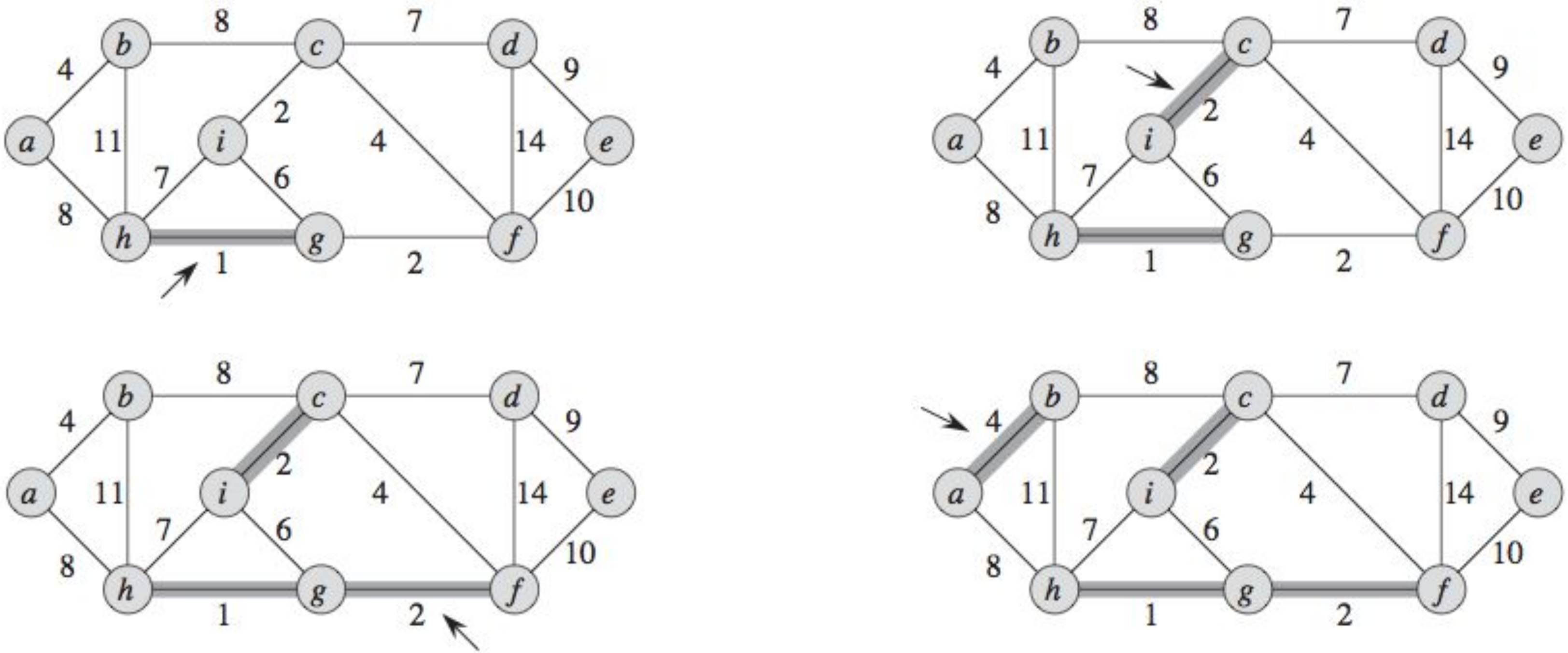
- **Corollary 23.2** Let $G=(V,E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, and let $C=(V_C,E_C)$ be a connected component (tree) in the forest $G_A=(V,A)$. If (u,v) is a light edge connecting C to some other component in G_A, then (u,v) is safe for A.
- **Proof** The cut $(V_C,V-V_C)$ respects A, and (u,v) is a light edge for this cut. Thus, (u,v) is safe for A.

MST, Kruskal's algorithm

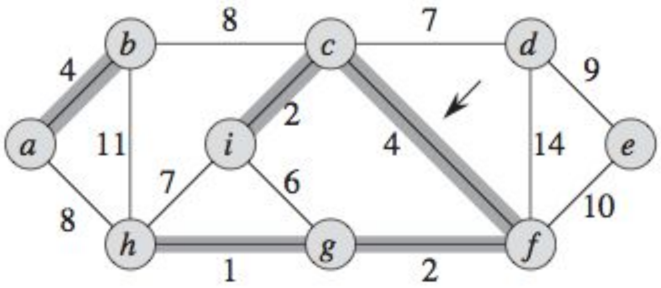
- Use a disjoint set data structure

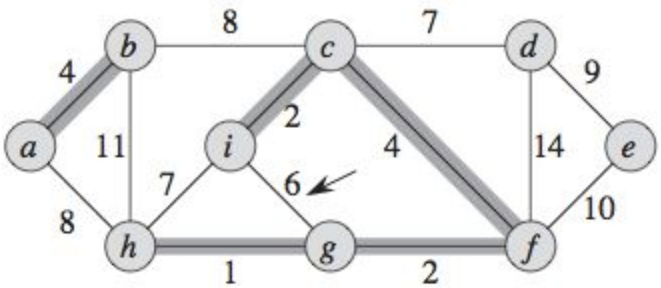
```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

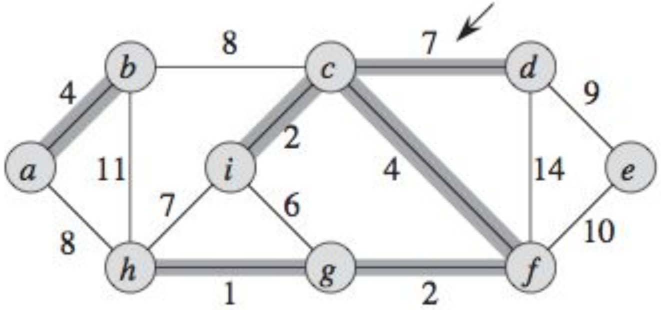
MST, Kruskal's algorithm

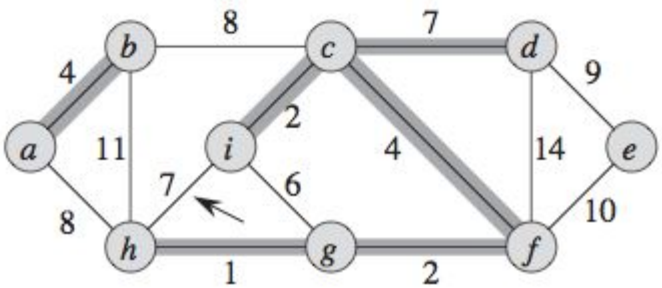


MST, Kruskal's algorithm

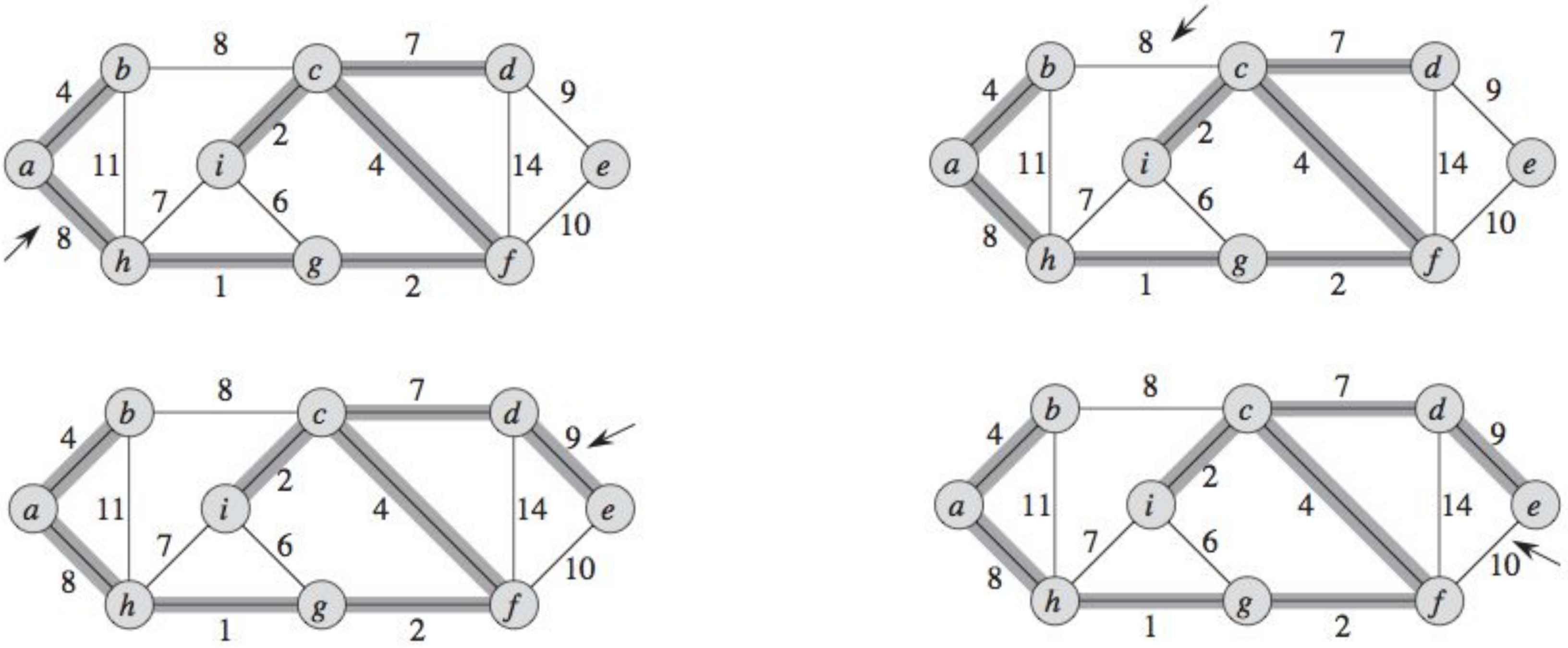




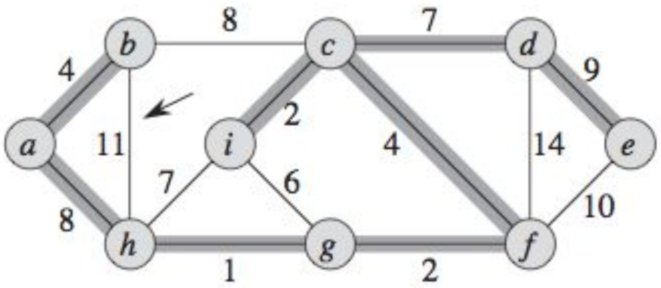


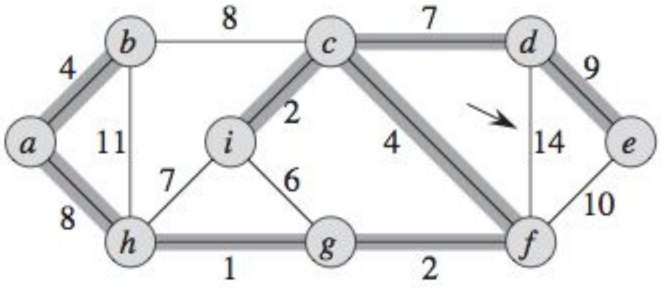


MST, Kruskal's algorithm



MST, Kruskal's algorithm





MST, Kruskal's algorithm

- Running time is

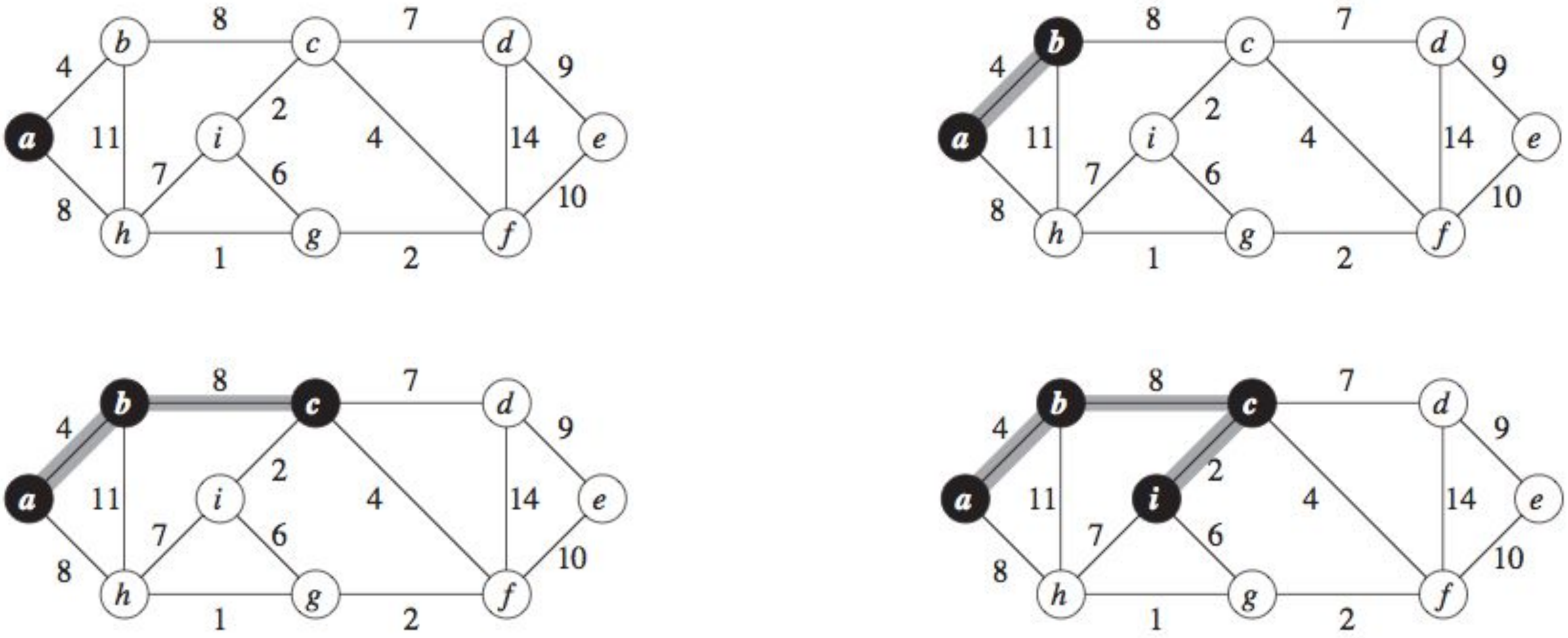
$$
\begin{aligned}
&O(E \lg E) + O((V + E)\alpha(V)) \\
&= O(E \lg E) + O(E\alpha(V)) \\
&= O(E \lg E) \\
&= O(E \lg V)
\end{aligned}
$$

MST, Prim's algorithm

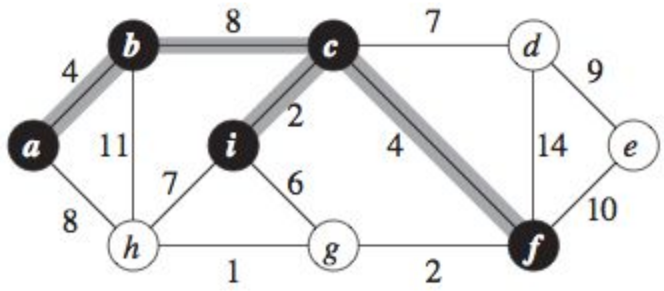
- Use a min-priority queue

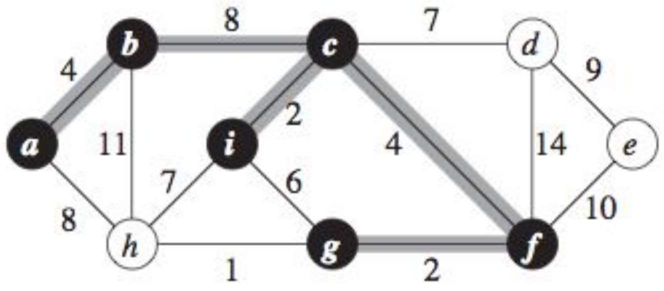
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

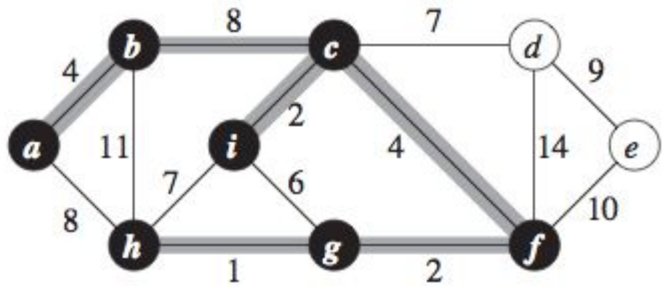
MST, Prim's algorithm

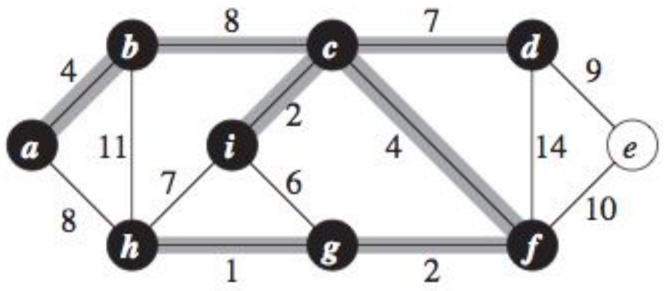


MST, Prim’s algorithm

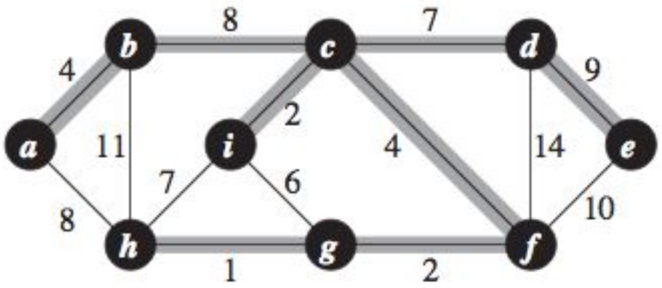








MST, Prim's algorithm



MST, Prim's algorithm

- When using a binary min-heap, running time is:
  - *V* Extract-Min operations of *O(lg V)*
  - *E* Decrease-Key operations of *O(lg V)*

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

MST, Prim's algorithm

- When using a Fibonacci heap, (amortized) running time is
  - V Extract-Min operations of $O(\log V)$ amortized
  - E Decrease-Key operations of $O(1)$ amortized

$$O(E + V \lg V)$$