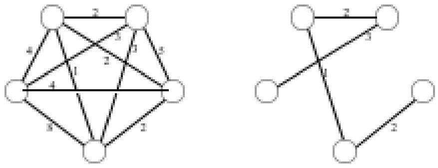
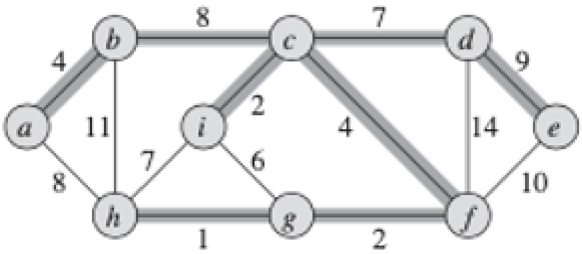
Minimum Spanning Trees

- Given an undirected **weighted** graph $G = (V, E)$
- spanning tree $G_s = (V, E_s)$ where E_s is a subset of E that connects all the nodes in G
- minimum spanning tree : spanning tree with the minimum total weight
  $w(T) = \sum_{(w,v)\in T} w(u, v)$



- An undirected weighed graph and its minimum spanning tree



MST Algorithm

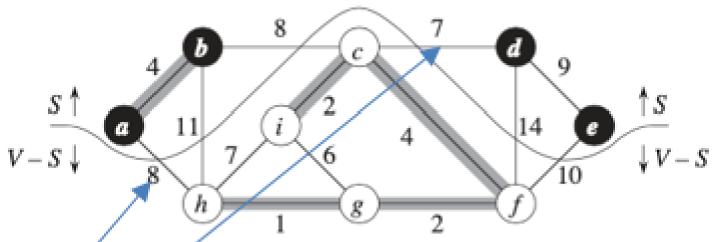
- safe edge : For an edge set A which is a subset of some MST, if $A \cup e$ is still a subset of a MST, then e is a **safe** edge
- loop invariant in GENERIC-MST algorithm:
  - prior to each iteration, A is a subset of some MST

```
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
```

  - spanning tree가 될 때까지 safe edge만 계속 더하기

Theorem 23.1

- Connected undirected weighted graph G에 대해서, edge set A는 G의 한 MST의 부분집합이라 하자. A를 존중(**respect**)하는 G의 cut $(S, V-S)$가 있고, (u, v)가 $(S, V-S)$를 cross 하는 light edge 라면 (u, v)는 A에 대한 **safe edge**이다



cut(S,V-S)=({a,b,d,e},{c,h,i,g,f}) respects
edge set A={(a,b),(c,i),(c,f),(g,f),(g,h)}
(a,h) crosses (S,V-S) and is not a light edge.
(c,d) crosses (S,V-S) and is a light edge. → safe edge

  - light edge : cross edge 중에서 weight가 가장 작은 edge
- Proof
  - A를 포함하는 MST를 T라 하자
    - $A \cup (u, v)$가 T에 포함되면, (u, v)는 safe edge : trivial
    - $A \cup (u, v)$가 T에 포함되지 않으면, T가 spanning tree 이므로 T안에 u ▶ v $path$ p가 있고 그 path 에는 cross edge가 있다
    - 이 cross edge를 (x, y)라 하고 이것을 제거하면 T는 더 이상 connected가 아니고 다시 (u, v)를 추가하면 $T' = T - \{(x, y)\} \cup \{(u, v)\}$는 spanning tree가 되는데 (u, v)가 light edge $w(u, v) \leq w(x, y)$이므로 이 $w(T') = w(T) - w(x, y) + w(u, v) < w(T)$ 이다
    - T가 MST이므로 $w(T') = w(T)$ 즉, T'도 MST

MST-KRUSKAL

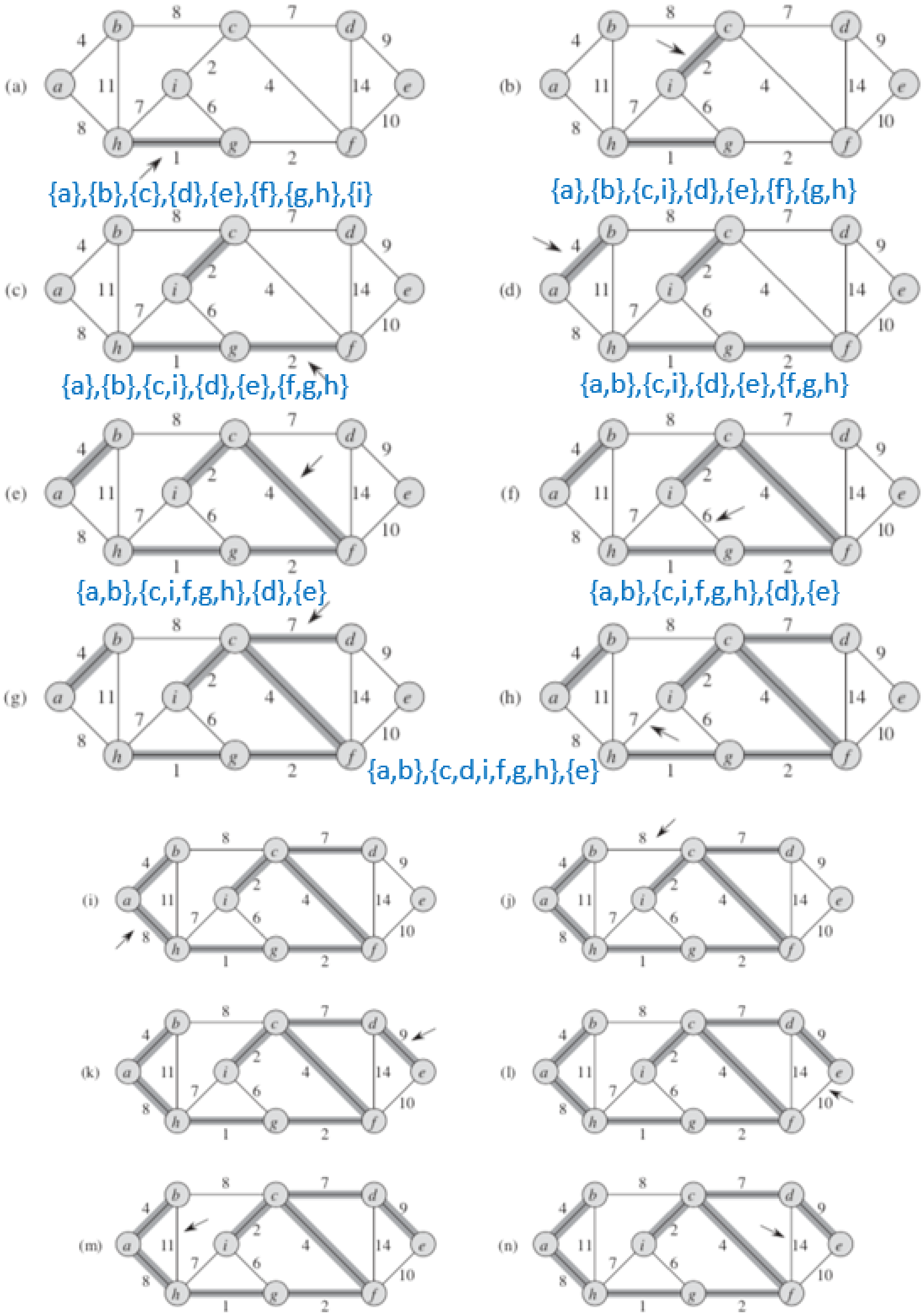
```
GENERIC-MST(G, w)
A = ∅
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    A = A ∪ {(u, v)}
return A
```

```
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   ←O(E lg E)
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
```

in chapter 21,
MAKE-SET(v) : v를 원소로 하는 집합을 만든다.
UNION(u,v) : u가 속한 집합과 v가 속한 집합의 합집합을 만든다.
FIND-SET(v) : v 가 속한 집합

$E < V^2 \dashrightarrow O(\lg V) = O(\lg E)$

$= O(E \lg V)$



Prim's Algorithm

- Main idea
  - Maintain a set S that starts out with a single node s
  - Find the smallest weighted edge $e^* = (u, v)$ that connects $u \in S$ and $v \in S$
  - Add e^* to the MST, add v to S
  - Repeat until $S = V$
- Differs from Kruskal's in that we grow a single supernode S instead of growing multiple ones at the same time

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

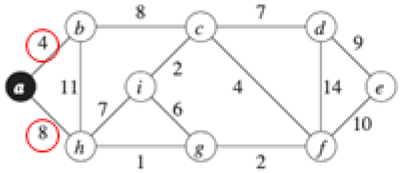
$$= O((E + V)\lg V) = O(E \lg V)$$

$V - 1 \leq E < V^2$ in connected graph

```
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)
```

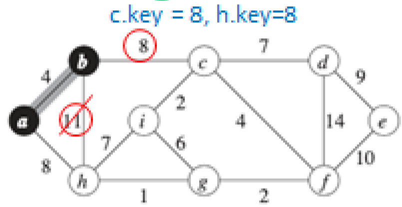
Q ={b,c,d,e,f,g,h,i}
b.key = 4, h.key=8

(a)

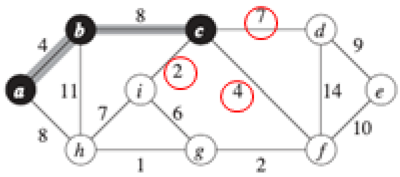


Q ={c,d,e,f,g,h,i}
c.key = 8, h.key=8

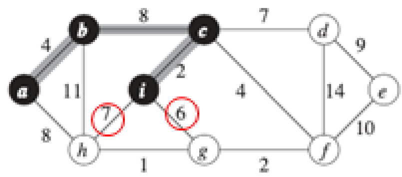
(b)



(c)



(d)

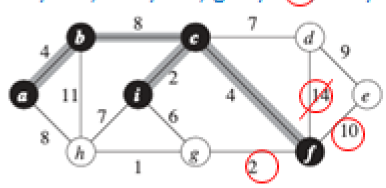


Q ={d,e,f,g,h,i}
d.key = 7, i.key=2, f.key = 4, h.key = 8

Q ={d,e,f,g,h}
d.key = 7, f.key = 4, h.key = 7 g.key = 6

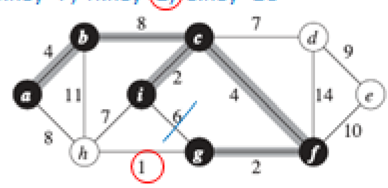
Q ={d,e,g,h}
d.key = 7, h.key = 7, g.key = 2 e.key=10

(e)

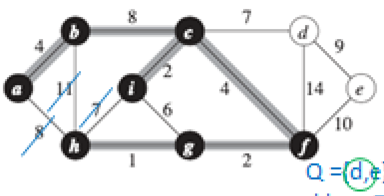


Q ={d,e,h}
d.key=7, h.key=1, e.key=10

(f)

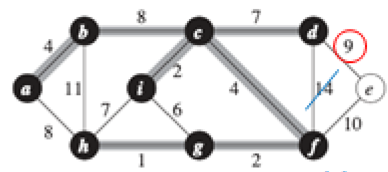


(g)



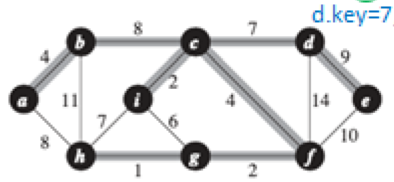
Q ={d,e}
d.key=7, e.key=10

(h)



Q ={e}
e.key=9

(i)




```
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)
```