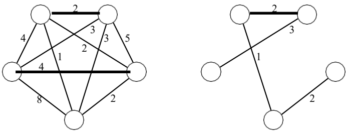
23. Minimum Spanning Trees

Minimum Spanning Tree (MST)

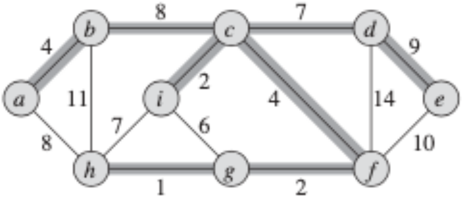
- Given an undirected weighted graph $G = (V, E)$
- spanning tree $G_s = (V, E_s)$ where E_s is a subset of E that connects all the nodes in G
- minimum spanning tree : spanning tree with the minimum total weight $w(T) = \sum_{(u,v)\in T} w(u, v)$



An undirected weighted graph and its minimum spanning tree.

MST Algorithms

- two greedy algorithms:
  - Kruskal's algorithm
  - Prim's algorithm



  - Proof of greedy choice property

Kruskal’s Algorithm

```
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)    cycle 이 생기지 않음
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

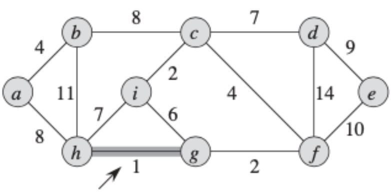
in chapter 21,

MAKE-SET(v) : v를 원소로 하는 집합을 만든다.

UNION(u,v) : u가 속한 집합과 v가 속한 집합의 합집합을 만든다.

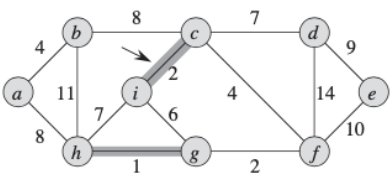
FIND-SET(v) : v 가 속한 집합

(a)



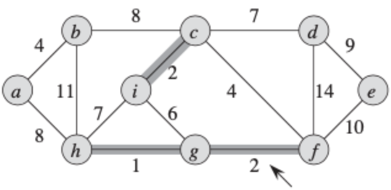
{a},{b},{c},{d},{e},{f},{g,h},{i}

(b)



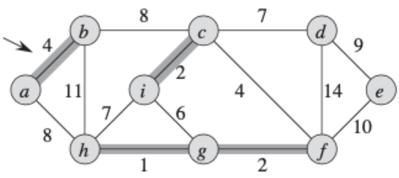
{a},{b},{c,i},{d},{e},{f},{g,h}

(c)

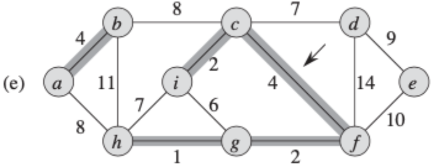


{a},{b},{c,i},{d},{e},{f,g,h}

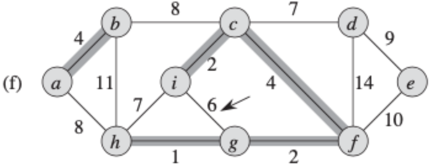
(d)



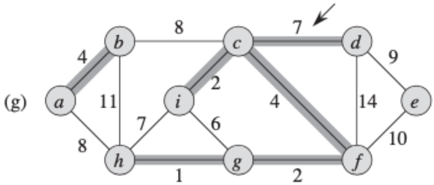
{a,b},{c,i},{d},{e},{f,g,h}



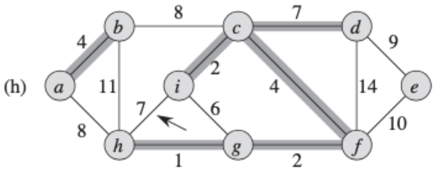
{a,b},{c,i,f,g,h},{d},{e}



{a,b},{c,i,f,g,h},{d},{e}

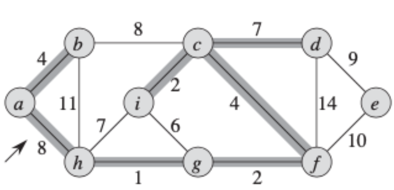


{a,b},{c,d,i,f,g,h},{e}



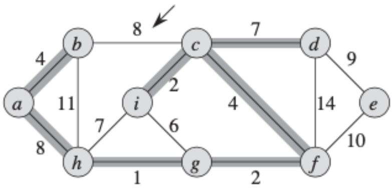
{a,b},{c,d,i,f,g,h},{e}

(i)



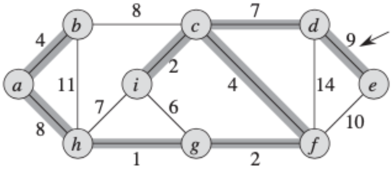
{a,b,c,d,i,f,g,h},{e}

(j)



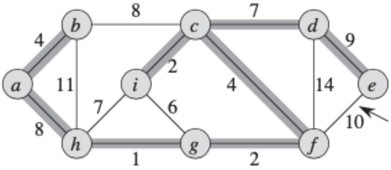
$\boxed{\text{\{a,b,c,d,i,f,g,h\},}}$\{e\}

(k)

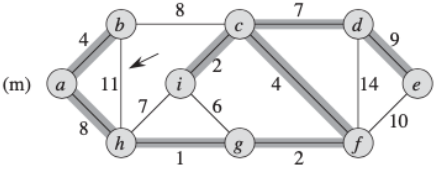


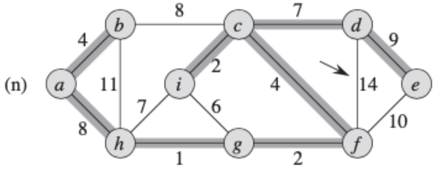
{a,b,c,d,i,f,g,h,e}

(l)



$\boxed{\text{\{a,b,c,d,i,f,g,h,e\}}}$


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


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

$\{a,b,c,d,i,f,g,h,e\}$

Running Time of MST-Kruskal

```
MST-KRUSKAL(G, w)
A = Ø
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   ←O(E)
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

$= O(\mathrm{E} \lg E) = O(\mathrm{E} \lg V)$

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

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

Prim's Algorithm

- Main idea:
  - Maintain a set S that starts out with a single node s
  - Find the smallest weighted edge $e^\star = (u, v)$ that connects $u \in S$ and $v \notin S$
  - Add $e^\star$ 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

Prim's MST Algorithm

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ Ø
    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)
```

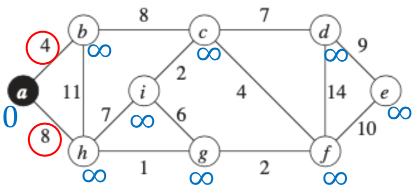
MST-Prim(G,w,a)

```
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)
```

Q ={a,b,c,d,e,f,g,h,i}

b.key = 4, h.key=8

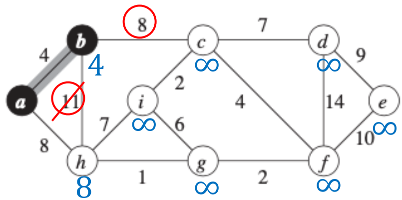
(a)



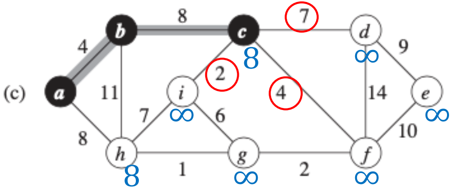
Q ={b,c,d,e,f,g,h,i}

c.key = 8, h.key=8

(b)

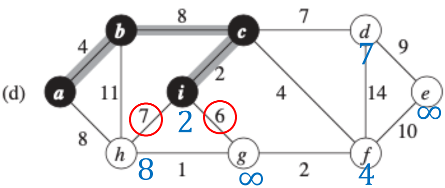


```
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```



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

d.key = 7, i.key=2, f.key = 4, h.key = 8

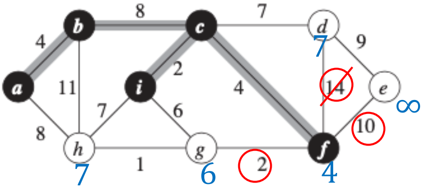


Q ={d,e,f,g,h,i}

d.key = 7, f.key = 4, h.key = 7, g.key = 6

```
while Q ≠ Ø
    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)
```

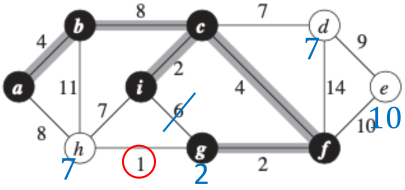
(e)



Q ={d,e,f,g,h}

d.key = 7, h.key = 7, g.key = 2, e.key=10

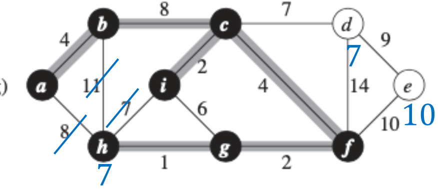
(f)



Q ={d,e,g,h}

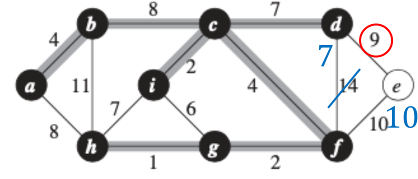
d.key=7, h.key=1, e.key=10

(g)



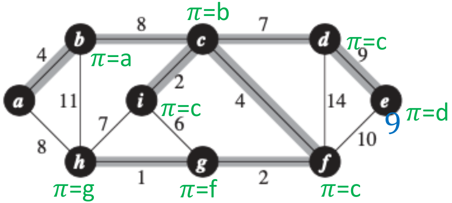
Q ={d,e,h}
d.key=7, e.key=10

(h)



Q ={d,e}
e.key=9

(i)



Q ={e}

```
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)
```

Running Time of Prim's MST Algorithm implemented using a MIN_HEAP

```
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 implying MIN_HEAPIFY
                            : E x O(lg V)
```

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

$since\ V-1 \le E < V^2$ in connected graphs,
$O(E)=O(V)$ in a sparse graph,
$O(E)=O(V^2)$ in a dense graph

generic MST Algorithm

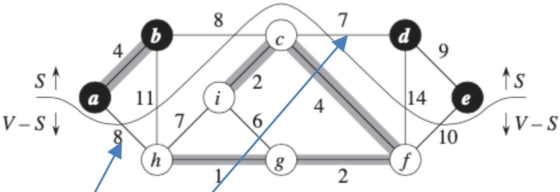
safe edge : For an edge set A which is a subset of some MST, if $A \cup e$ *is still a subset of a MST, then e is a* *safe* *edge.*

loop invariant in GENERIC-MST algorithm :
Prior to each iteration, A is a subset of some MST.

GENERIC-MST(G, w)

$A = \emptyset$
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    $A = A \cup \{(u, v)\}$
return A

cut, respect, cross, light edge



cut(S,V-S)=({a,b,d,e},{c,h,i,g,f})

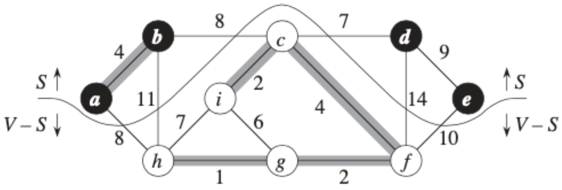
respects edge set A={(a,b),(c,i),(c,f),(g,f),(g,h)}

(a,h) crosses (S,V-S) and is not a light edge.

(c,d) crosses (S,V-S) and is a light edge (특정 성질을 만족하는 에지 중 최소 가중치를 가지는 에지).

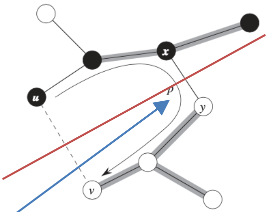
Proof of GENERIC-MST

Thm 23.1 connected undirected weighted graph G 에 대해서,
edge set A 는 G 의 한 MST 의 부분 집합이라 하자.
A를 존중하는 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.



proof of Thm 23.1

Thm 23.1 connected undirected weighted graph G 에 대해서,
edge set A 는 G 의 한 MST 의 부분 집합이라 하자.
A를 존중하는 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.



A를 포함하는 MST를 T라 하자.

1) $A \cup (u,v)$ 가 T에 포함되면, *(u,v)* 는 safe edge : trivial

2) $A \cup (u,v)$ 가 T에 포함되지 않으면

T가 spanning tree 이므로 T 안에 *u->v path p*가 있고 그 path 에는 cross edge 가 있다.
이 cross edge 를 *(x,y)* 라 하고 이것을 제거하면 T는 더 이상 connected 가 아니고 다시 *(u,v)* 를 추가하면 $T' = T - \{(x,y)\} \cup \{(u,v)\}$ 는 spanning tree 가 되는데 *(u,v)* 가 light edge $w(u,v) \leq w(x,y)$이므로 이 $w(T') = w(T) - w(x,y) + w(u,v) \leq w(T)$이다.

T가 MST 이므로 $w(T') = w(T)$ 즉, T'도 MST

```
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
```

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ Ø
    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-KRUSKAL(G, w)
A = Ø
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
```

generic MST Algorithm

```
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
```

Thm 23.1 connected undirected weighted graph G 에 대해서,

edge set A 는 G 의 한 MST 의 부분 집합이라 하자.

A를 존중하는 G의 cut (S, V-S) 가 있고, (u,v) 가 (S,V-S)를 cross

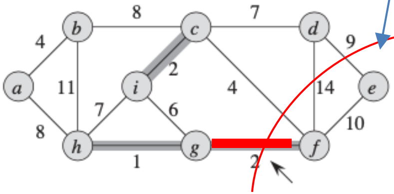
하는 light edge 라면 (u,v) 는 A에 대한 safe edge 이다.

Kruskal’s algorithm 에서의 cut :
(u,v)를 cross 하면서
 같은 집합의 vertices 을 같은 쪽에 포함하는 any cut

MST-KRUSKAL(G, w)

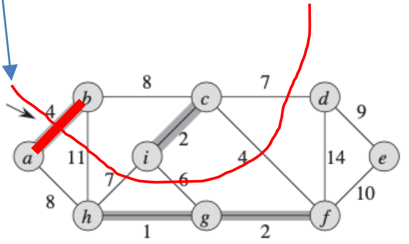
```
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

(c)



{a},{b},{c,i},{d},{e},{f},{g,h}

(d)

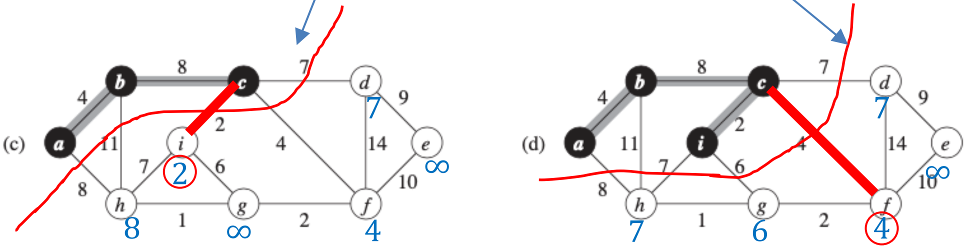


{a},{b},{c,i},{d},{e},{f,g,h}

Prim's algorithm 에서의 cut :

{Q에서 제거된 vertices (검정색)},{Q에 남은 vertices (흰색)}

```
while Q ≠ Ø
    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 ={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