Fundamental of Minimal Spanning Trees

Kuan-Yu Chen (陳冠宇)

2019/05/29 @ TR-310-1, NTUST

Review

- A spanning tree of a connected, undirected graph G is a sub-graph of G which is a tree that connects all the vertices together
  - A graph G can have many different spanning trees

- A **minimum spanning tree** (MST) is defined as a spanning tree with weight less than or equal to the weight of every other spanning tree
  - We can assign **weights** to each edge, and use it to assign a weight to a spanning tree by calculating the sum of the weights of the edges in that spanning
    - Prim's Algorithm
    - Kruskal's Algorithm

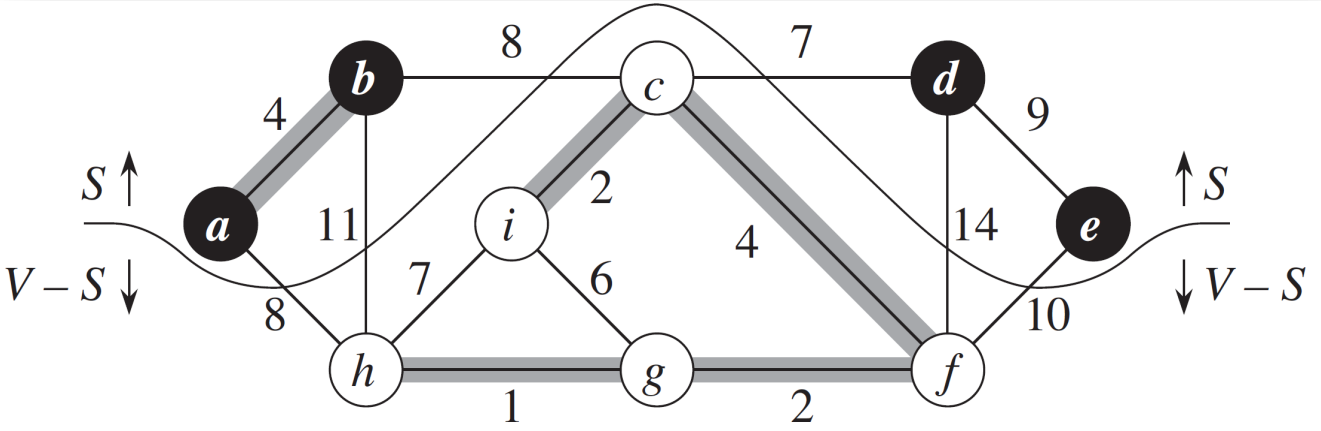
Basic Idea

- Assume that we have a connected, undirected graph $G = (V, E)$ with a weight function $w: E \to \mathbb{R}$, and we wish to find a minimum spanning tree for G
  - The basic idea is to maintain a subset A of some MST
  - At each step, we determine an edge (u, v) and add to A
  - We should make sure that $A \cup \{(u, v)\}$ is still a subset of a MST
  - Such an edge is called a ***safe edge*** for A
    - The tricky part is, of course, finding a safe edge in line 3

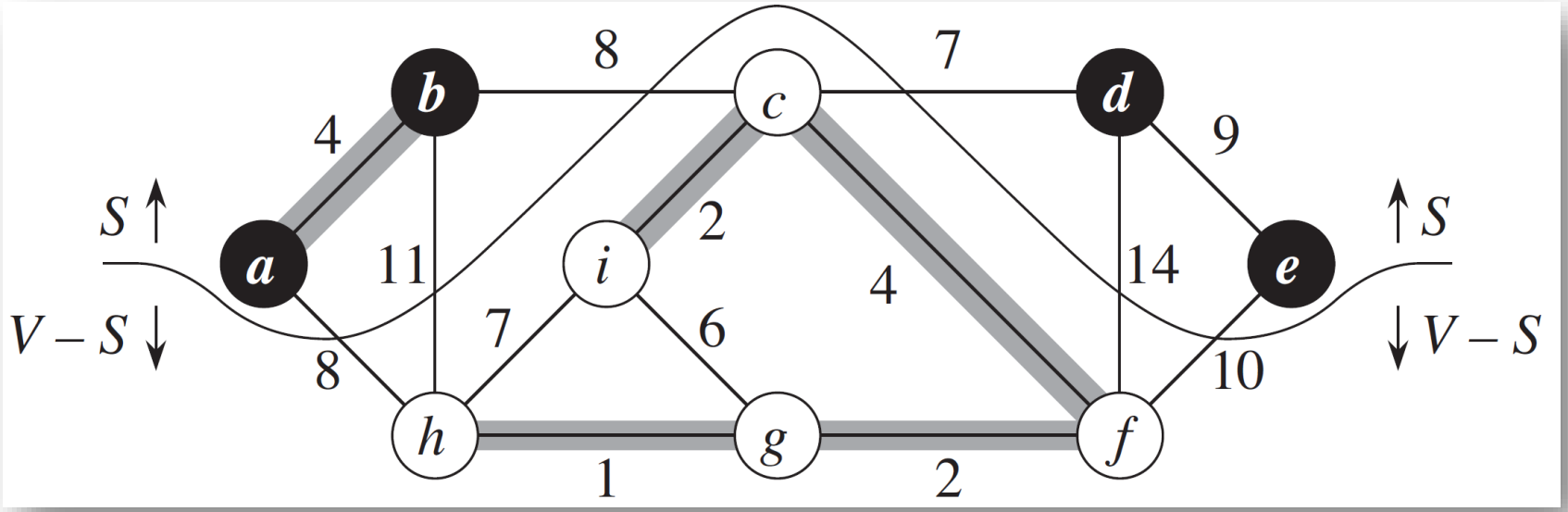
```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3      find an edge (u, v) that is safe for A
4      A = A ∪ {(u, v)}
5  return A
```

Definitions.

- We first define some definitions
  - A ***cut*** $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V
  - We say that 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 $V - S$
  - We say that 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



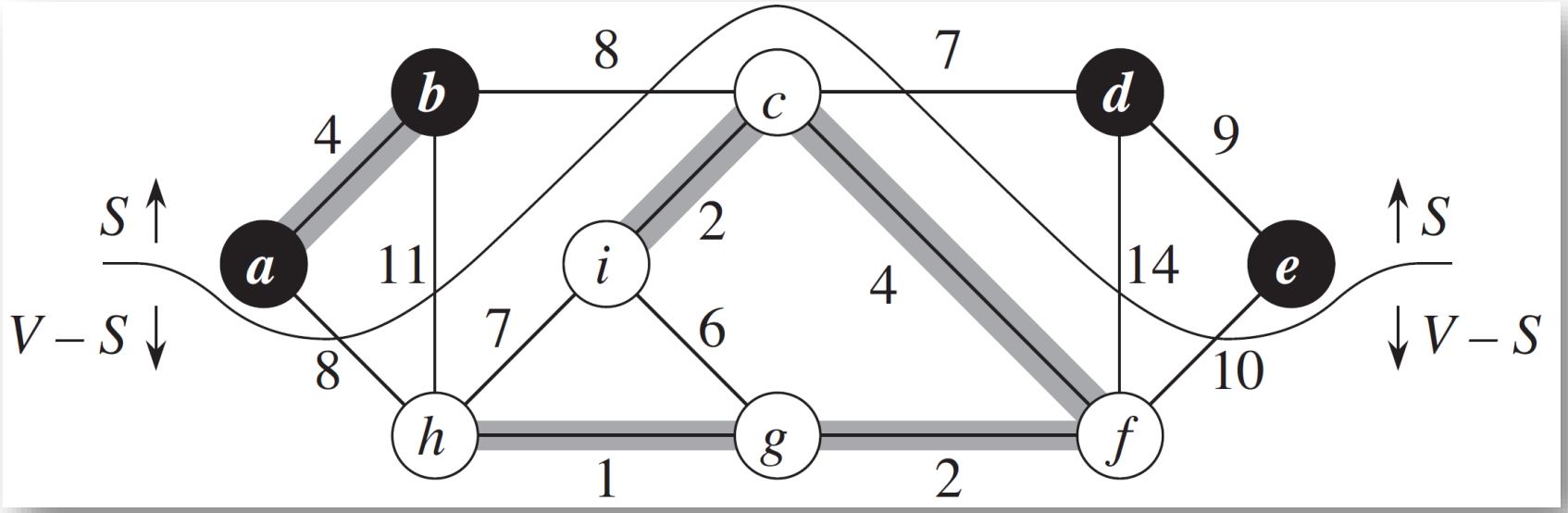
Definitions..



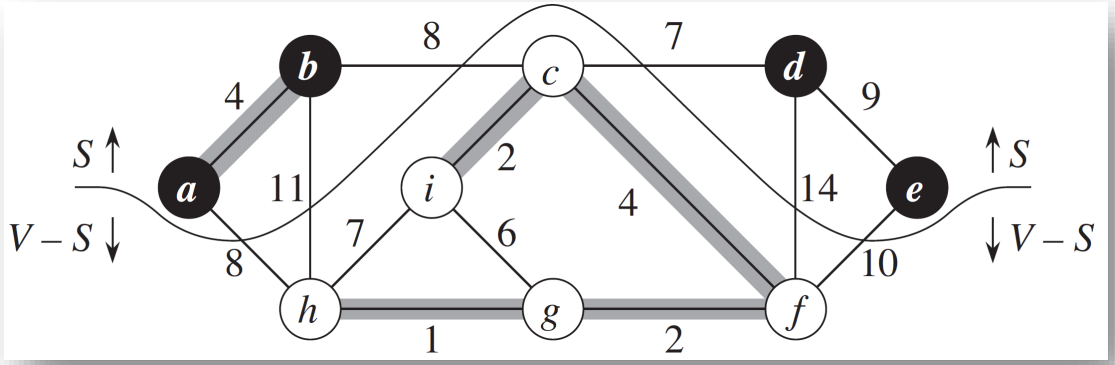
- The edge (d, c) is the unique **light edge** crossing the cut
- A subset A of the edges is shaded
  - The cut $(S, V - S)$ **respects** A, since no edge of A crosses the cut

Theorem.

- 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
  - Let (u, v) be a light edge crossing $(S, V - S)$
  - Edge (u, v) is safe for A

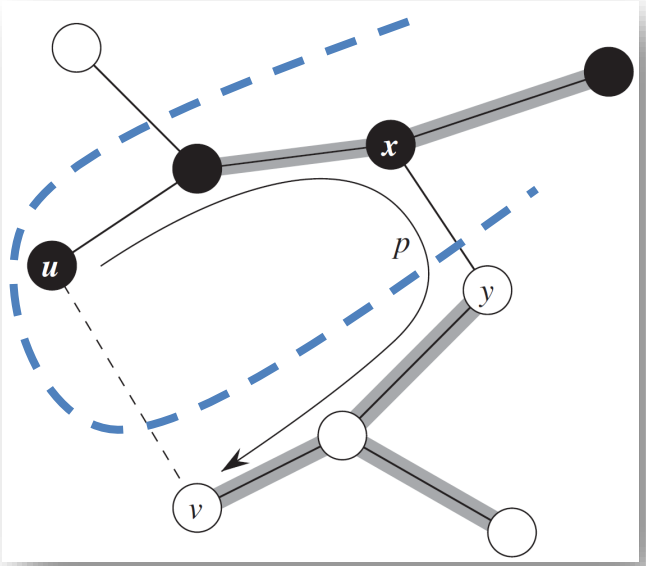


Theorem..



- Say:
  - Let T is a minimum spanning tree that include A, and T' is a tree that include A
  - We assume that $A \cup (x, y) \subseteq T$ and $A \cup (c, d) \subseteq T'$
    - x and y are on opposite sides of the cut $(S, V - S)$, and $(x, y) \neq (c, d)$
  - $T' = T - (x, y) + (c, d)$
    - Although (x, y) crosses the cut, $w(c, d) < w(x, y)$
    - $w(T') = w(T) - w(x, y) + w(c, d) < w(T)$
    - Thus, T' is actually a MST!
    - The result implies that (c, d), the light edge, is safe for A

Theorem...



- Proof:
  - Let T is a minimum spanning tree that include A, and T' is a tree that include A
    - $A \cup (x, y) \subseteq T$ and $A \cup (u, v) \subseteq T'$
  - $T' = T - (x, y) + (c, d)$
    - Although (x, y) crosses the cut, $w(u, v) \leq w(x, y)$
    - $w(T') = w(T) - w(x, y) + w(u, v) \leq w(T)$
    - Thus, T' is actually (also) a MST!
    - The result implies that (u, v), the light edge, is safe for A

Prim and Kruskal Algorithms

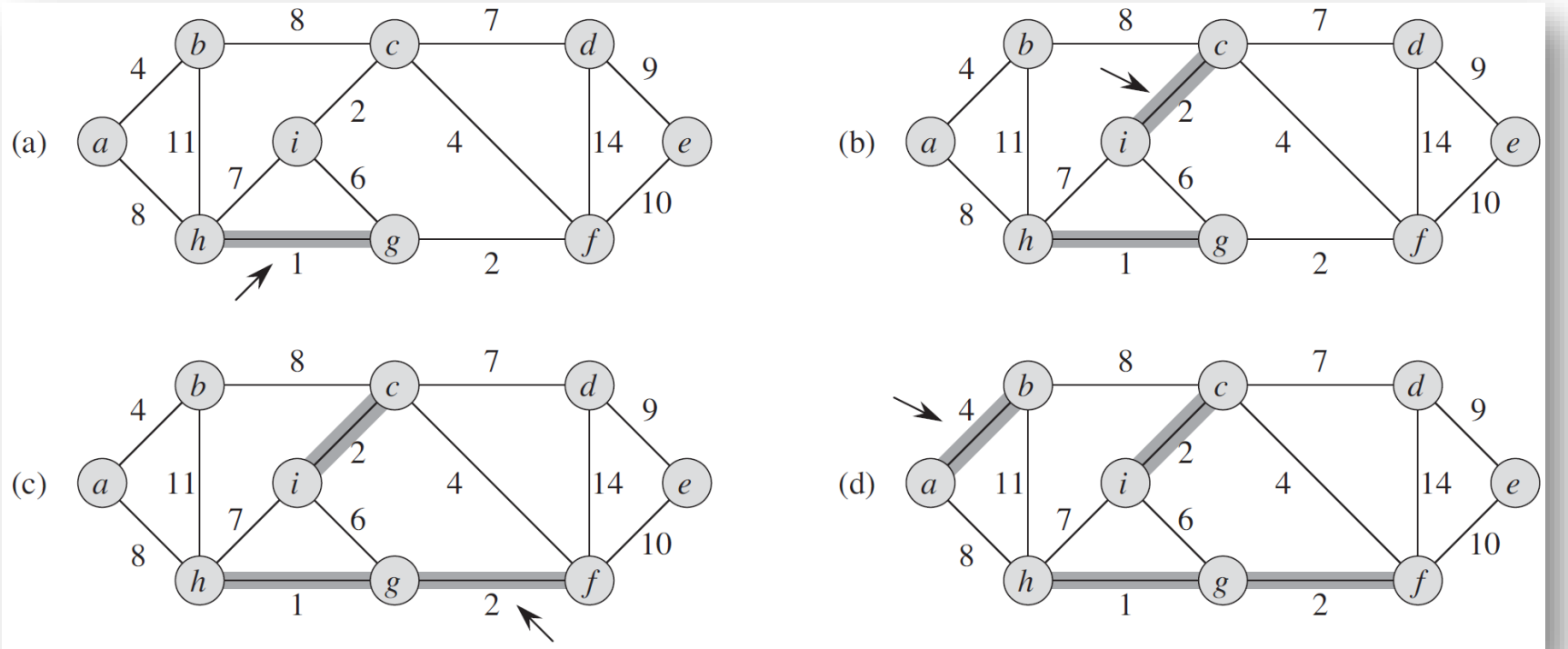
- They each use a specific rule to determine a safe edge in line 3 of GENERIC-MST

```
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
```

  - In Kruskal's algorithm
    - The set A is a forest whose vertices are all those of the given graph
    - 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**

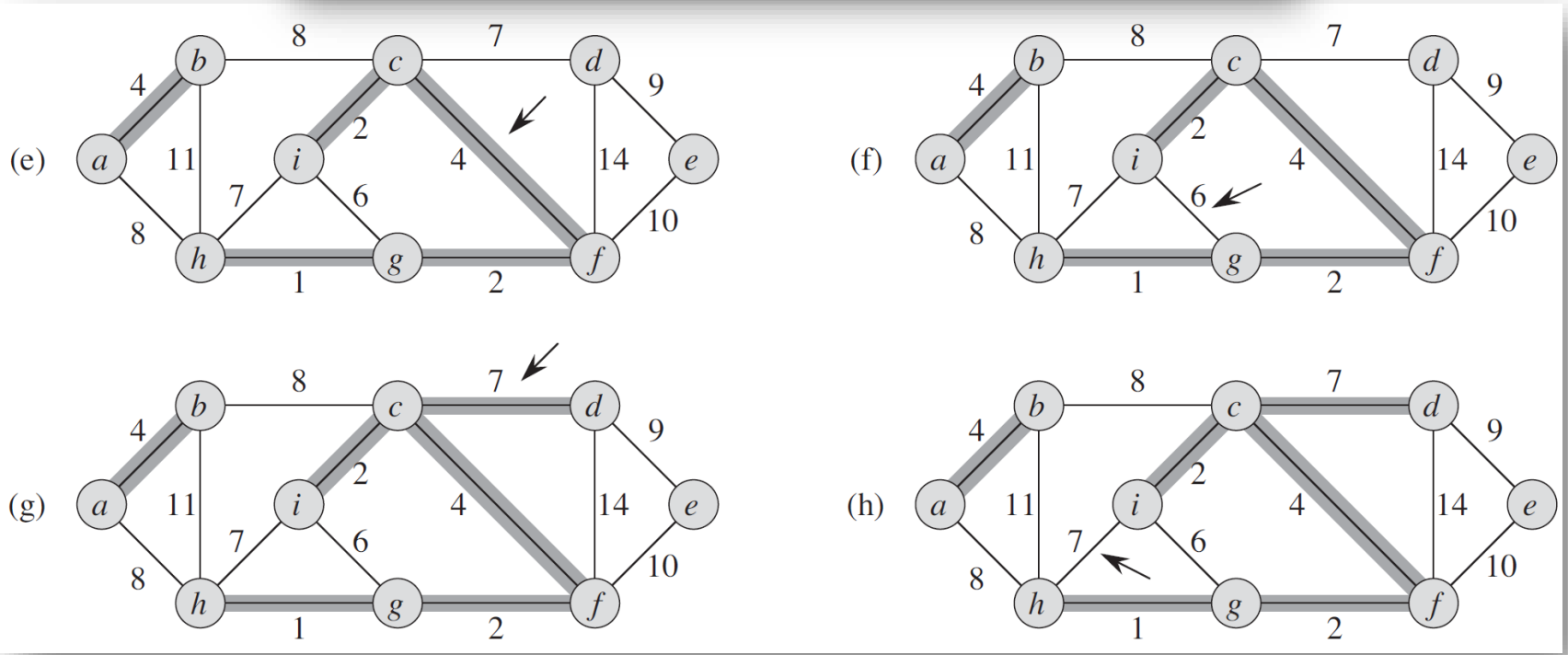
Kruskal's Algorithm.

- In Kruskal's algorithm
  - The set A is a forest whose vertices are all those of the given graph
  - The safe edge added to A is always a least-weight edge in the graph that **connects two distinct components**

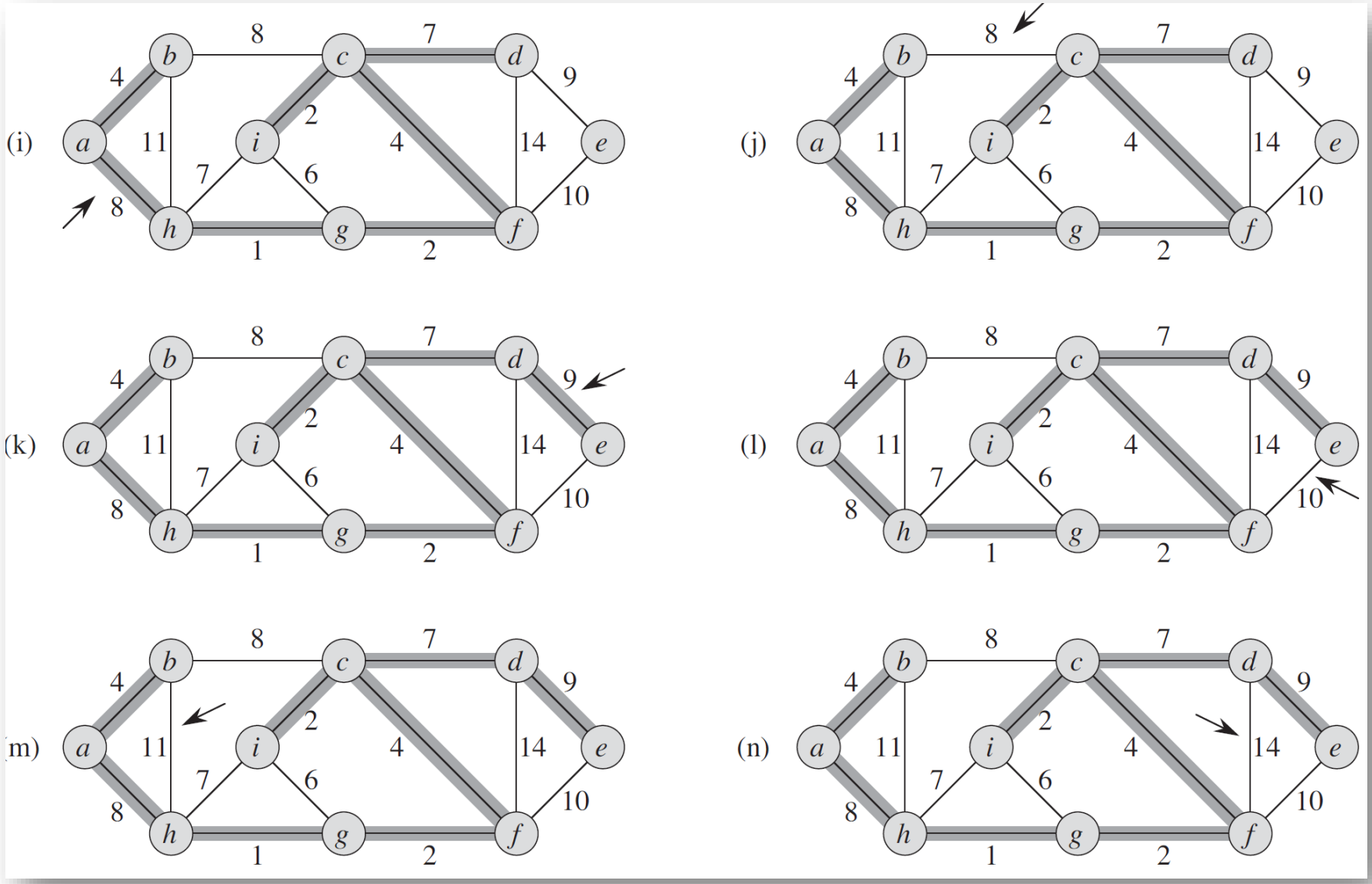


Kruskal's Algorithm..

```
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
```

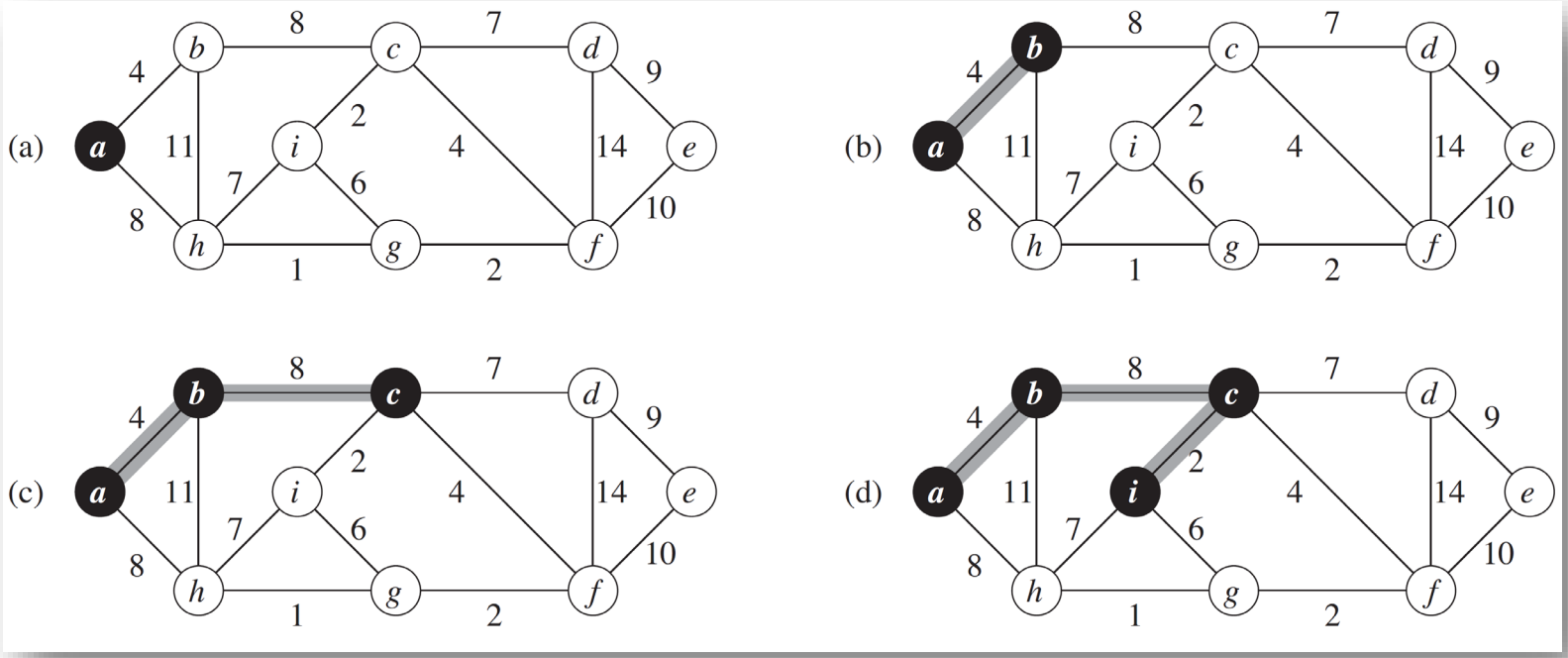


Kruskal's Algorithm…

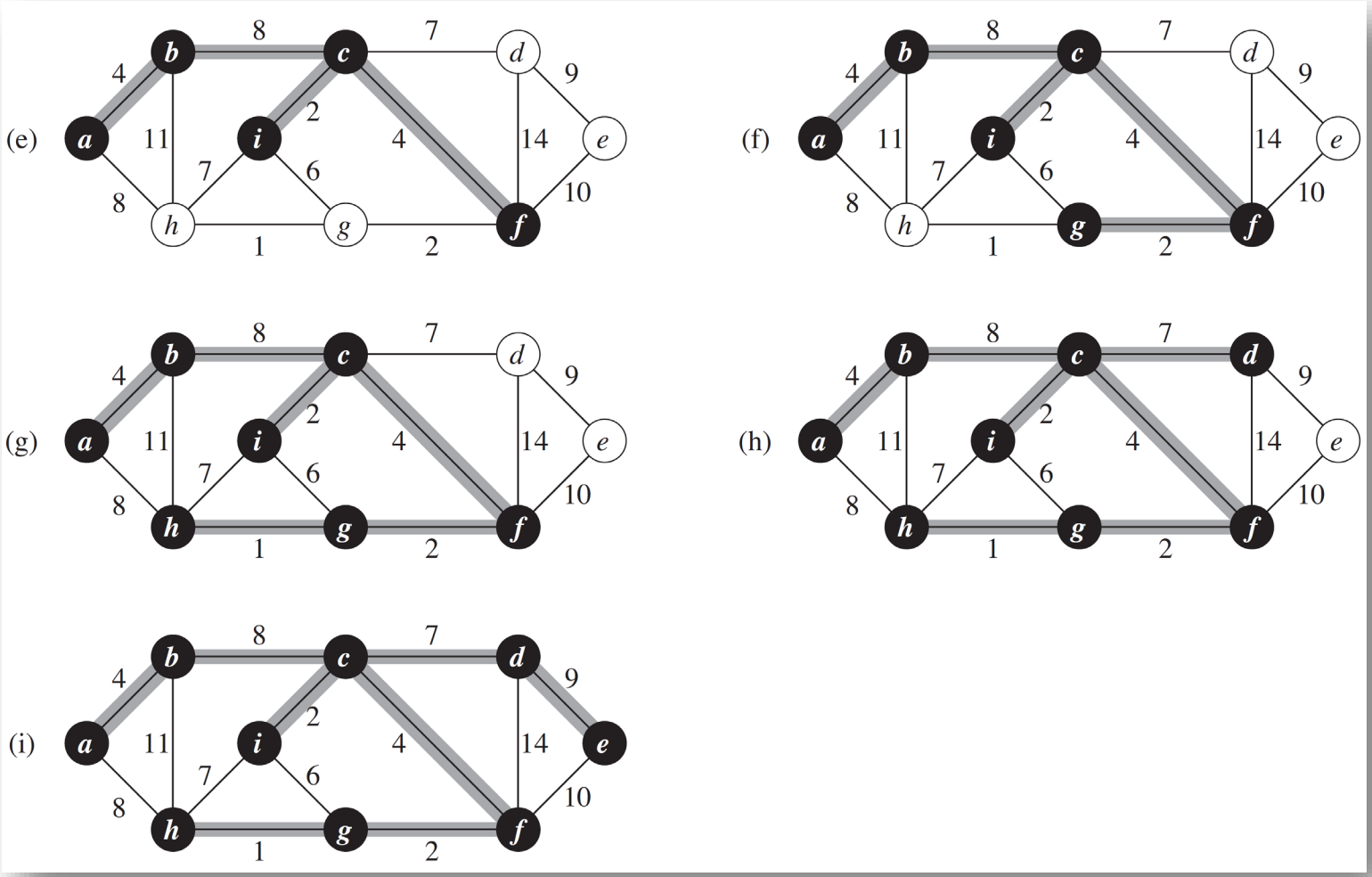


Prim's Algorithm.

- 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



Prim's Algorithm..



Prim's Algorithm...

- Step1

Q	a	b	c	d	e	f	g	h	i
Key	0	∞	∞	∞	∞	∞	∞	∞	∞
π	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

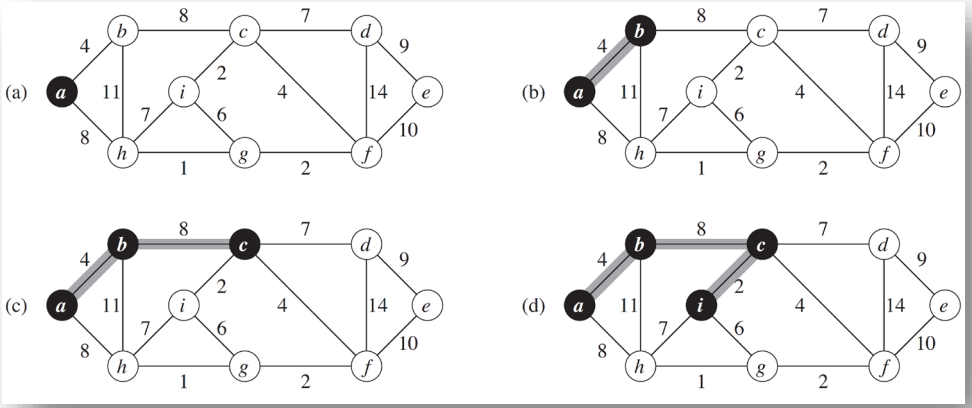
- Step2
  - $u = a$

Q	b	c	d	e	f	g	h	i
Key	4	∞	∞	∞	∞	∞	8	∞
π	a	NIL	NIL	NIL	NIL	NIL	a	NIL

- Step3
  - $u = b$

Q	c	d	e	f	g	h	i
Key	8	∞	∞	∞	∞	8	∞
π	b	NIL	NIL	NIL	NIL	a	NIL

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 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….

- Step4
  - $u = c$

Q	d	e	f	g	h	i
Key	7	∞	4	∞	8	2
π	c	NIL	c	NIL	a	c

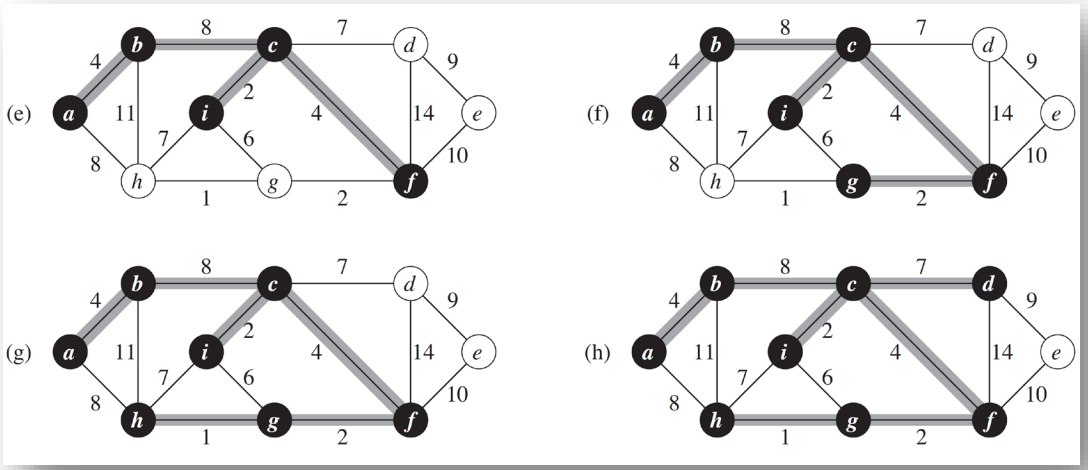
- Step5
  - $u = i$

Q	d	e	f	g	h
Key	7	∞	4	6	7
π	c	NIL	c	i	i

- Step6
  - $u = f$

Q	d	e	g	h
Key	7	10	2	7
π	c	f	f	i

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 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….

- Step7
  - $u = g$

Q	d	e	h
Key	7	10	1
π	c	f	g

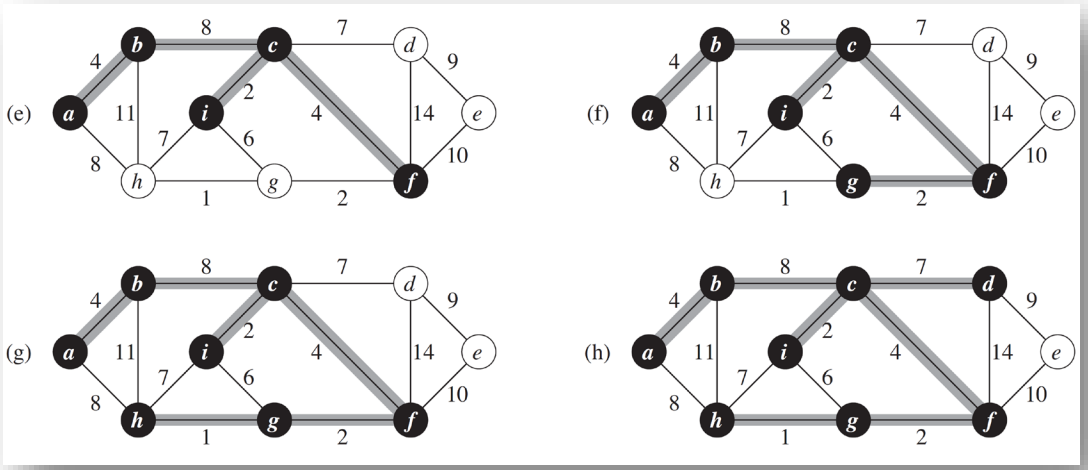
- Step8
  - $u = h$

Q	d	e
Key	7	10
π	c	f

- Step9
  - $u = d$

Q	e
Key	10
π	f

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 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)
```

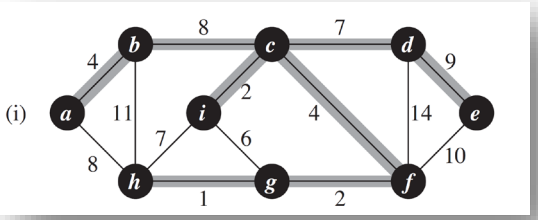


Prim's Algorithm….

- Step10
  - $u = e$

Q
Key
π

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 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)
```



Questions?

kychen@mail.ntust.edu.tw