Minimum Spanning Trees

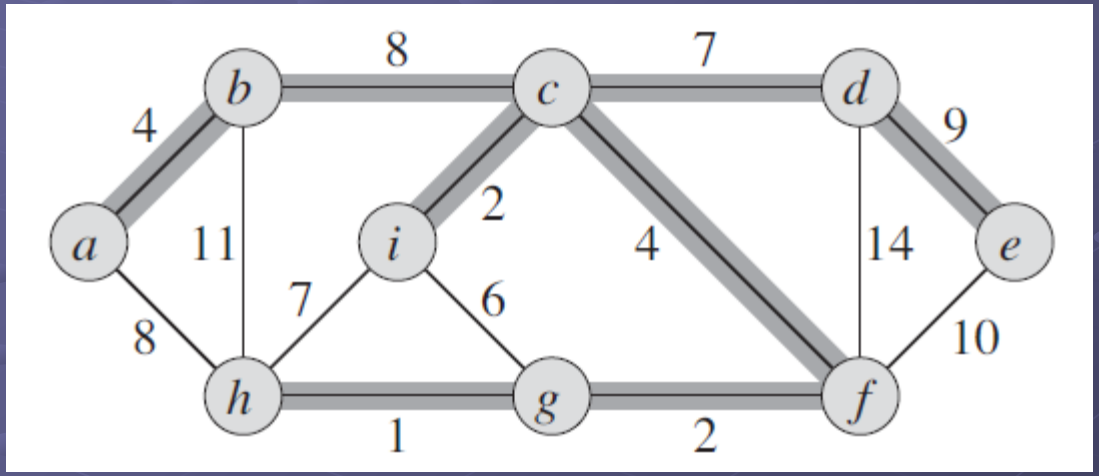
Introduction[1]

- **Electronic circuit designs**
  - **Interconnect a set of n pins**
  - **Model this wiring problem with a connected, undirected graph $G = (V, E)$.**
  - **Minimize**

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

    - T is a spanning tree of G
    - Minimum spanning tree (minimum-weight spanning tree)

Introduction^2



- Two greedy algorithms $O(E \lg V)$
  - Kruskal's algorithm
  - Prim's algorithm
    - Using Fibonacci heaps, $O(E + V \lg V)$.

Growing a Minimum Spanning Tree[1]

- A generic method
  - Grows the minimum spanning tree one edge at a time.
  - Maintaining the following loop invariant:
    - Prior to each iteration, A is a subset of some minimum spanning tree.
  - *Safe edge*
    - An edge (u, v), such that $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree.

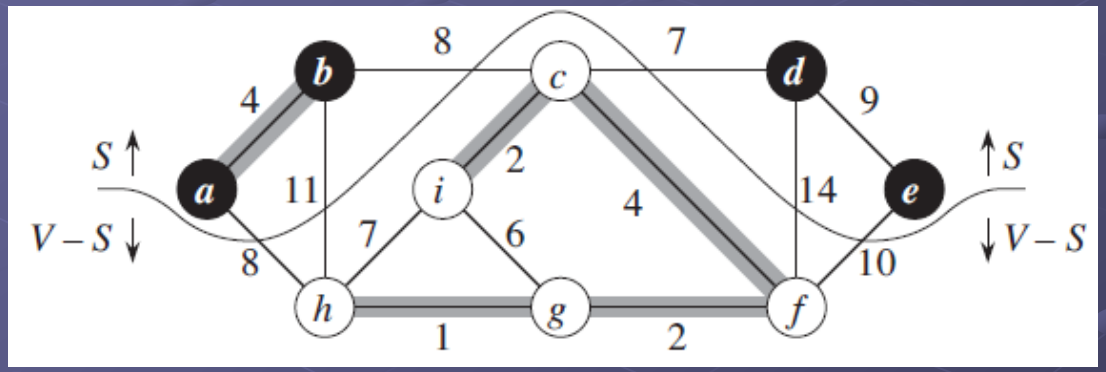
```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3      find an edge (u, v) that is safe for A
4      A = A ∪ {(u, v)}
5  return A
```

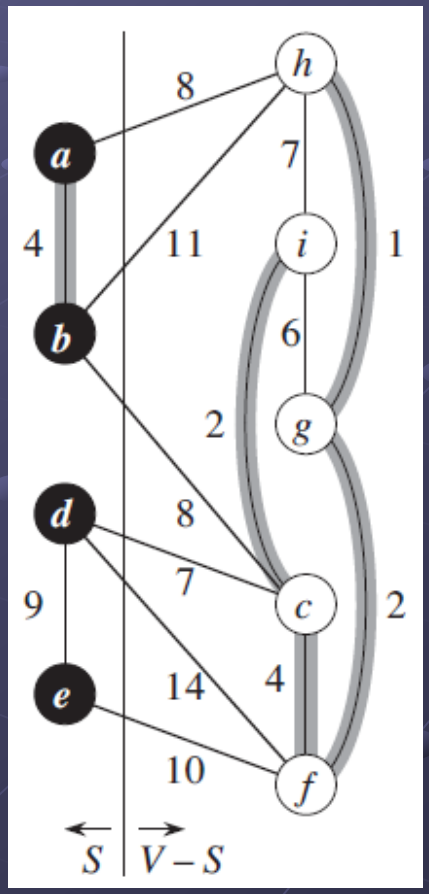
Growing a Minimum Spanning Tree[2]

- 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 $(S, V - S)$ *iff*
  - 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.

Growing a Minimum Spanning Tree[3]

- Two ways to view a cut.



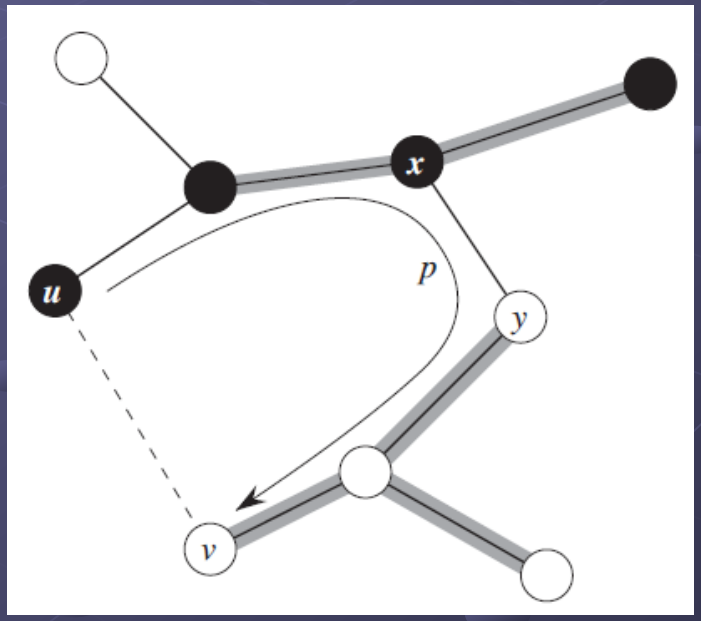


Growing a Minimum Spanning Tree[4]

☞ ***Theorem* 23.1 (A rule for recognizing safe edges)**

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.

- **Proof:**



Growing a Minimum Spanning Tree[5]

- This theorem gives a description of the working of the GENERIC-MST
  - As the method proceeds, the set A is always acyclic.
  - The graph $G_A = (V, A)$ is a forest.
  - When the method begins: A is empty and the forest contains $|V|$ trees, one for each vertex.
- Corollary 23.2

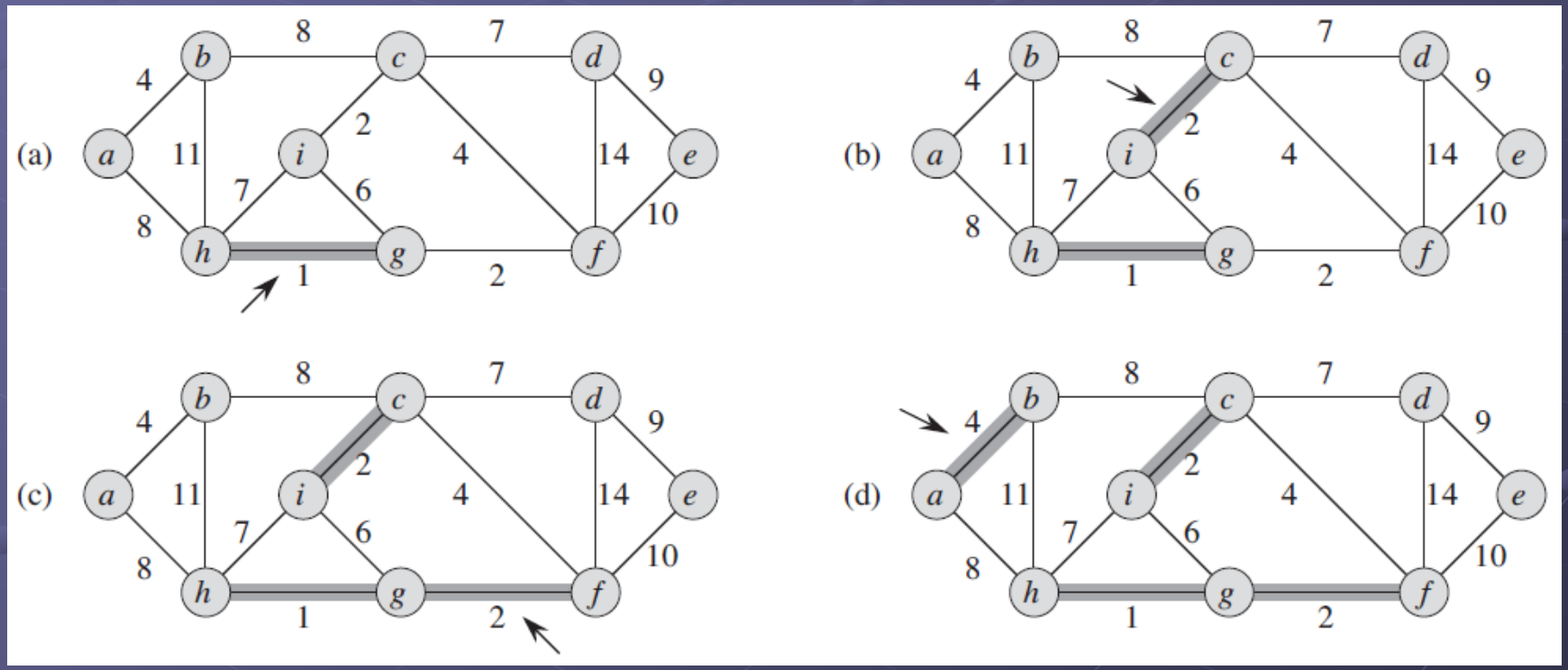
Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, and let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting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

Kruskal's Algorithm[1]

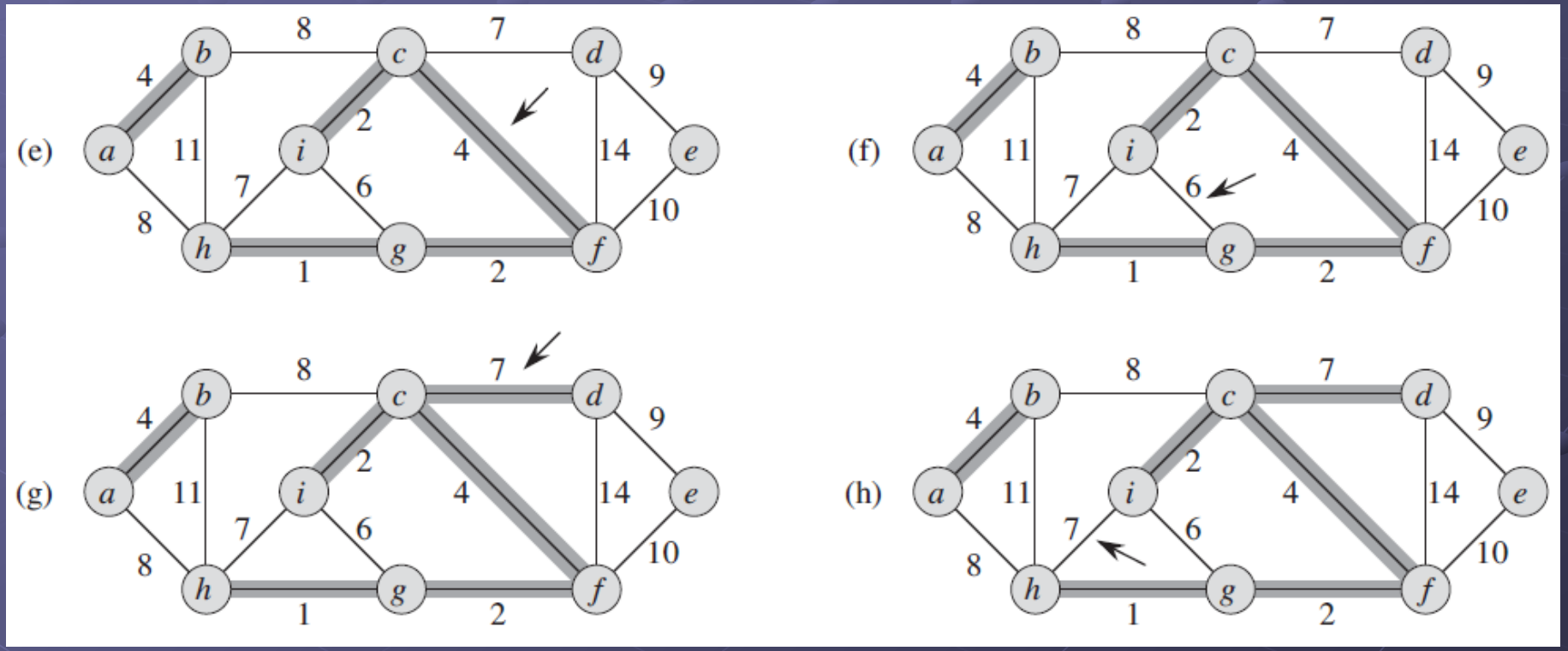
- **Find a safe edge to add to the growing forest by finding an edge (u, v) of least weight.**

```
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
```

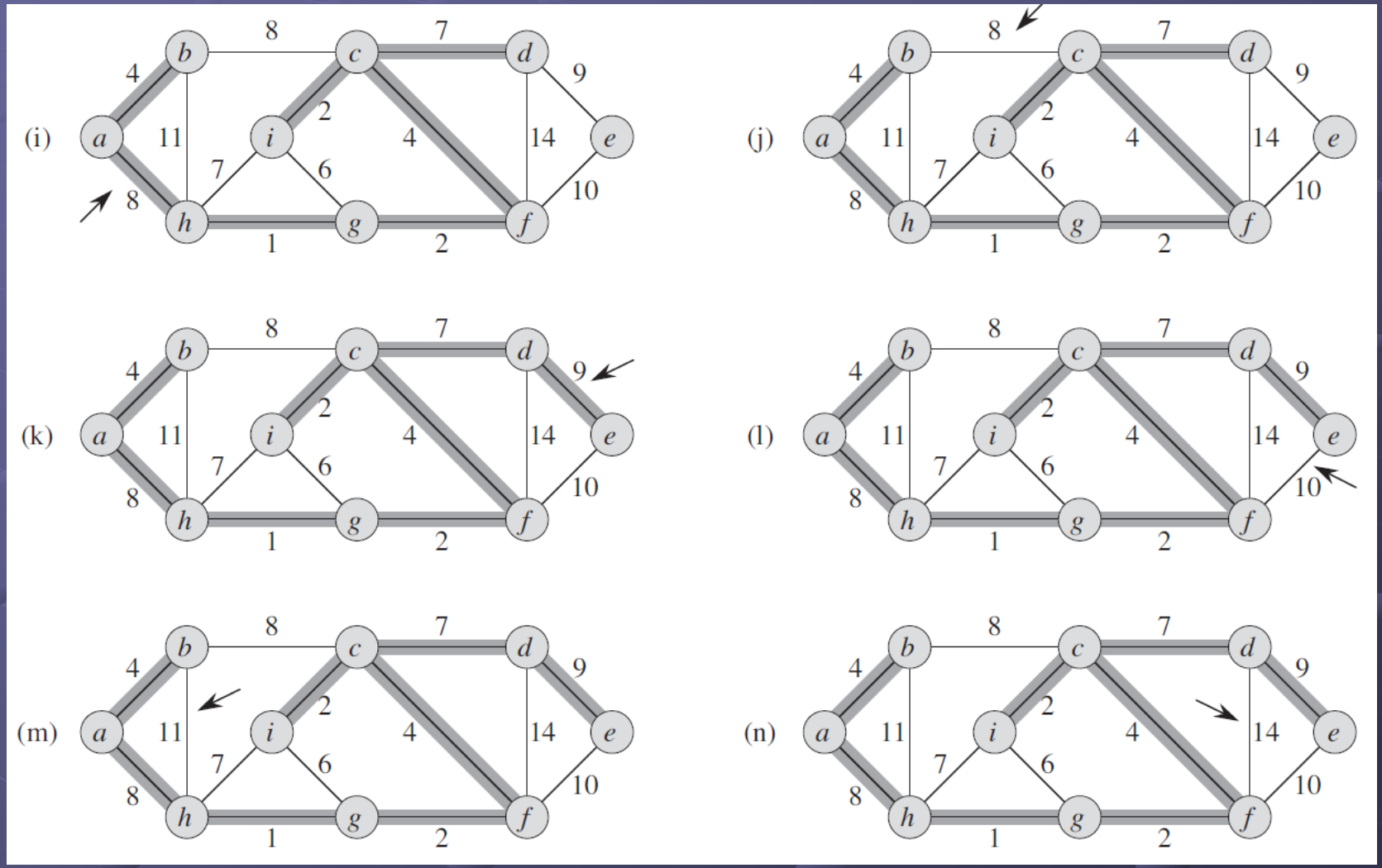
Kruskal's Algorithm[2]



Kruskal's Algorithm[3]



Kruskal's Algorithm[4]

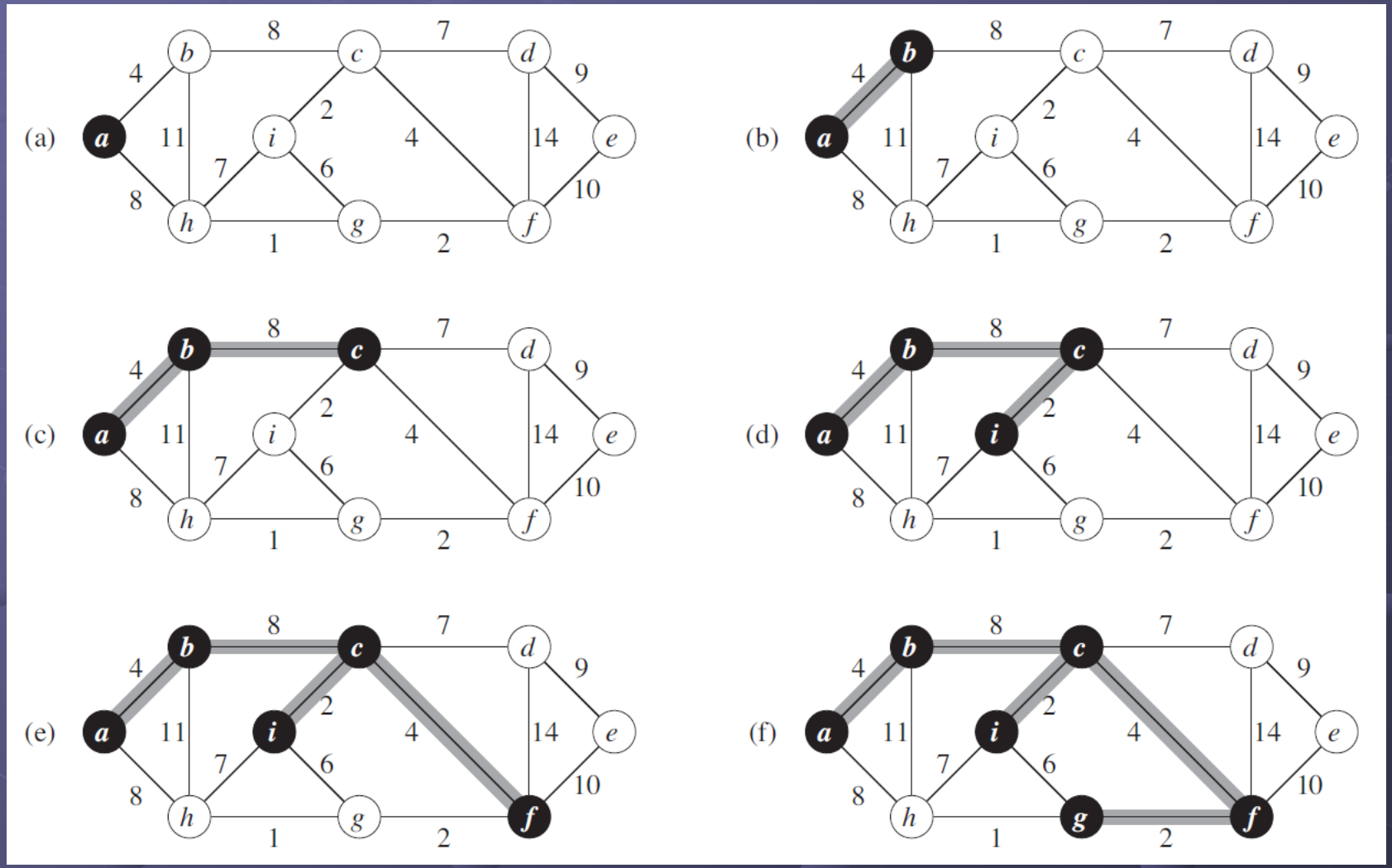


Prim's Algorithm[1]

- **Prim's algorithm has a property**
  - **The edges in the set A always form a single tree.**

```
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[2]



Prim's Algorithm[3]