COT 6405: Analysis of Algorithms

Giri NARASIMHAN

www.cs.fiu.edu/~giri/teach/6405F19.html

Graph Traversal

- Visit every vertex and every edge.
- Traversal has to be systematic so that no vertex or edge is missed.
- Just as tree traversals can be modified to solve several tree-related problems, graph traversals can be modified to solve several problems.

```
DFS(G)
1.  For each vertex u ∈ V[G] do
2.      color[u] ← WHITE
3.      π[u] ← NIL
4.  Time ← 0
5.  For each vertex u ∈ V[G] do
6.      if color[u] = WHITE then
7.          DFS-VISIT(u)
```

Depth First Search

Time: O(m+n) Why?

```
DFS-VISIT(u)
1.  VisitVertex(u)
2.  Color[u] ← GRAY
3.  Time ← Time + 1
4.  d[u] ← Time
5.  for each v ∈ Adj[u] do
6.      VisitEdge(u,v)
7.      if (v ≠ π[u]) then
8.          if (color[v] = WHITE) then
9.              π[v] ← u
10.                 DFS-VISIT(v)
11. color[u] ← BLACK
12. F[u] ← Time ← Time + 1
```

Breadth First Search

```
BFS(G,s)
1.   For each vertex u ∈V[G] – {s} do
2.       color[u] ← WHITE
3.        d[u] ← ∞
4.        π[u] ← NIL
5.   Color[u] ← GRAY
6.   D[s] ← 0
7.   π[s] ← NIL
8.   Q ← Φ
9.   ENQUEUE(Q,s)
10.  While Q ≠ Φ do
11.      u ← DEQUEUE(Q)
12.      VisitVertex(u)
13.      for each v ∈ Adj[u] do
14.           VisitEdge(u,v)
15.           if (color[v] = WHITE) then
16.               color[v] ← GRAY
17.               d[v] ← d[u] + 1
18.              π[v] ← u
19.               ENQUEUE(Q,v)
20.       color[u] ← BLACK
```

Minimum Spanning Tree

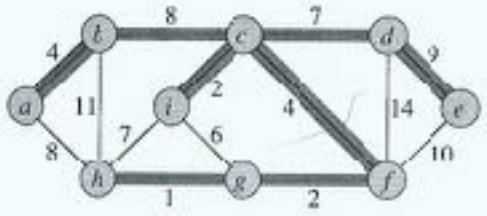


Figure 23.1 A minimum spanning tree for a connected graph. The weights on edges are shown, and the edges in a minimum spanning tree are shaded. The total weight of the tree shown is 37. This minimum spanning tree is not unique: removing the edge (b, c) and replacing it with the edge (a, h) yields another spanning tree with weight 37.

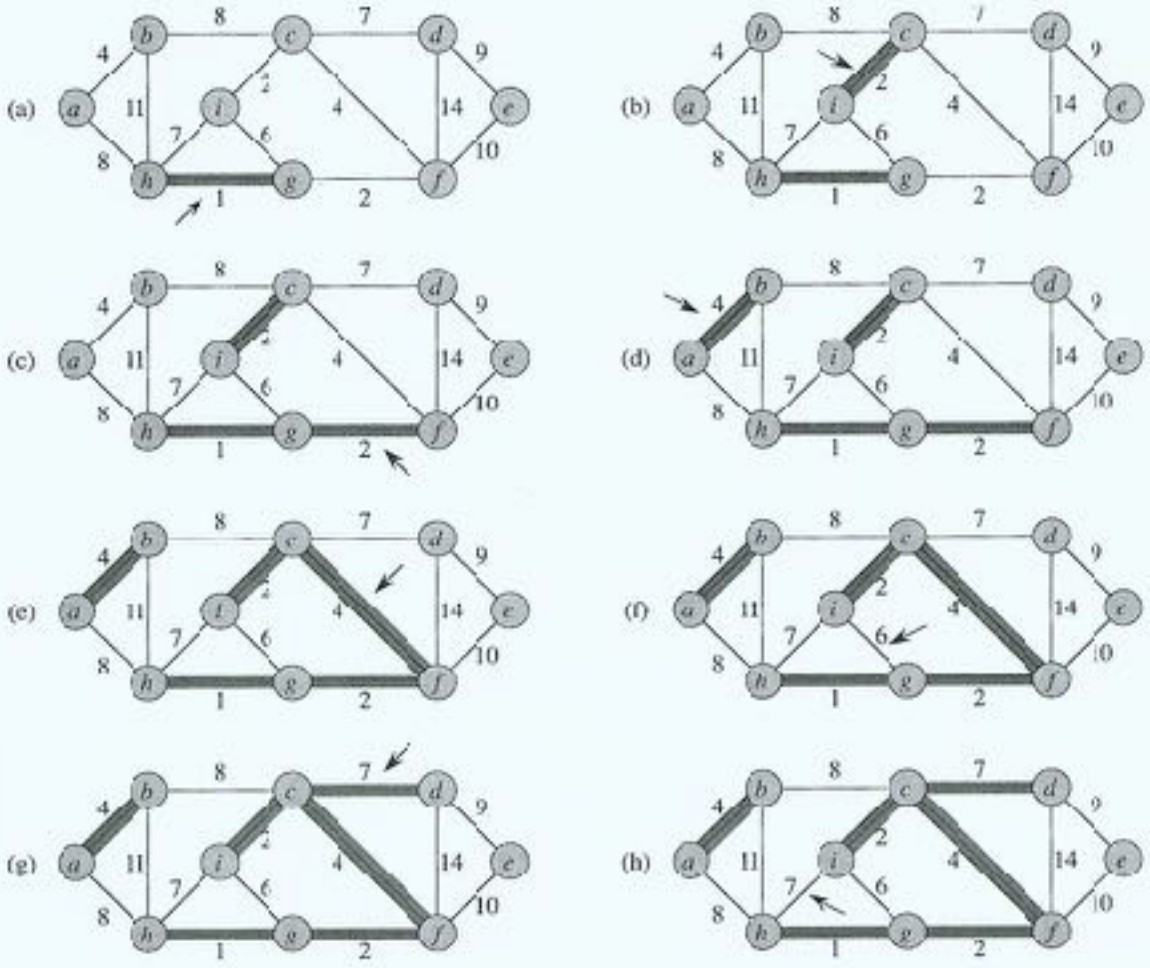
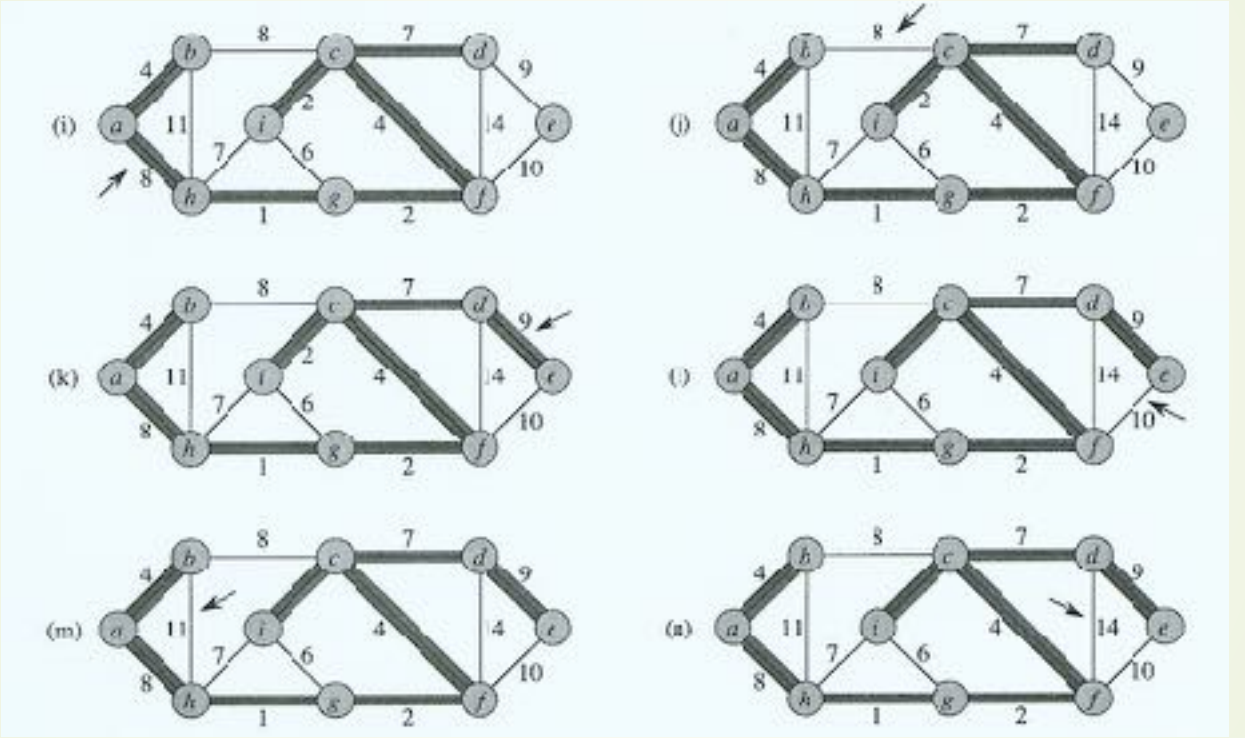


Figure 23.4 The execution of Kruscal's algorithm on the graph from Figure 23.1. Shaded edges belong to the forest A being grown. The edges are considered by the algorithm in sorted order by weight. An arrow points to the edge under consideration at each step of the algorithm. If the edge joins two distinct trees in the forest, it is added to the forest, thereby merging the two trees.


(i)
(j)
(k)
(l)
(m)
(n)

Minimum Spanning Tree

```
MST-KRUSKAL(G, w)
1.  A ← ∅
2.  for each vertex v ∈ V G|
3.      do MAKE-SET(v)
4.  sort the edges of E by nondecreasing weight w
5.  for each edge (u, v) ∈ E, in order by nondecreasing weigh
6.      do if FIND-SET(u) ≠ FIND-SET(v)
7.            then A ← A ∪ {(u, v)}
8.                 UNION(u, v)
9.  return A
```

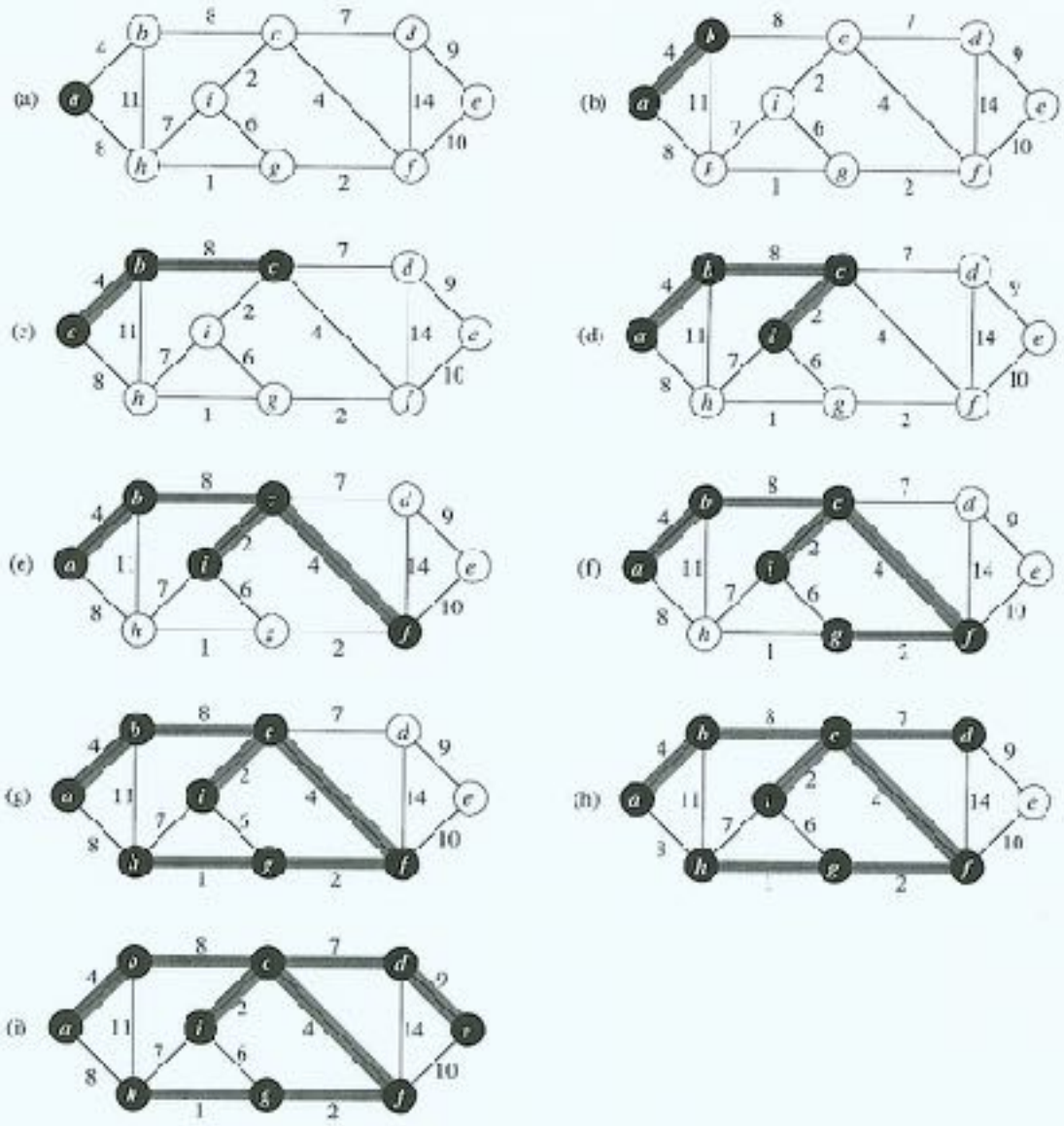


Figure 23.5 The execution of Prim's algorithm on the graph from Figure 23.1. The root vertex is a. Shaded edges are in the tree being grown, and the vertices in the tree are shown in black. At each step of the algorithm the vertices in the tree determine a cut of the graph, and a light edge crossing the cut is added to the tree. In the second step, for example, the algorithm has a choice of adding either edge (b, c) or edge (a, h) to the tree since both are light edges crossing the cut.

```
MST-KRUSKAL(G, w)
1.  A ← ∅
2.  for each vertex v ∈ V[G]
3.      do MAKE-SET(v)
4.  sort the edges of E by nondecreasing weight w
5.  for each edge (u, v) ∈ E, in order by nondecreasing weigh
6.      do if FIND-SET(u) ≠ FIND-SET(v)
7.            then A ← A ∪ {(u, v)}
8.                 UNION(u, v)
9.  return A
```

```
MST-PRIM(G, w, r)
1.  Q ← V[G]
2.  for each u ∈ Q
3.      do key[u] ← ∞
4.  key[r] ← 0
5.  π[r] ← NIL
6.  while Q ≠ ∅
7.        do u ← EXTRACT-MIN(Q)
8.           for each v ∈ Adj[u]
9.               do if v ∈ Q and w(u, v) < key[v]
10.                 then π[v] ← u
11.                      key[v] ← w(u, v)
```

Proof of Correctness: MST Algorithms

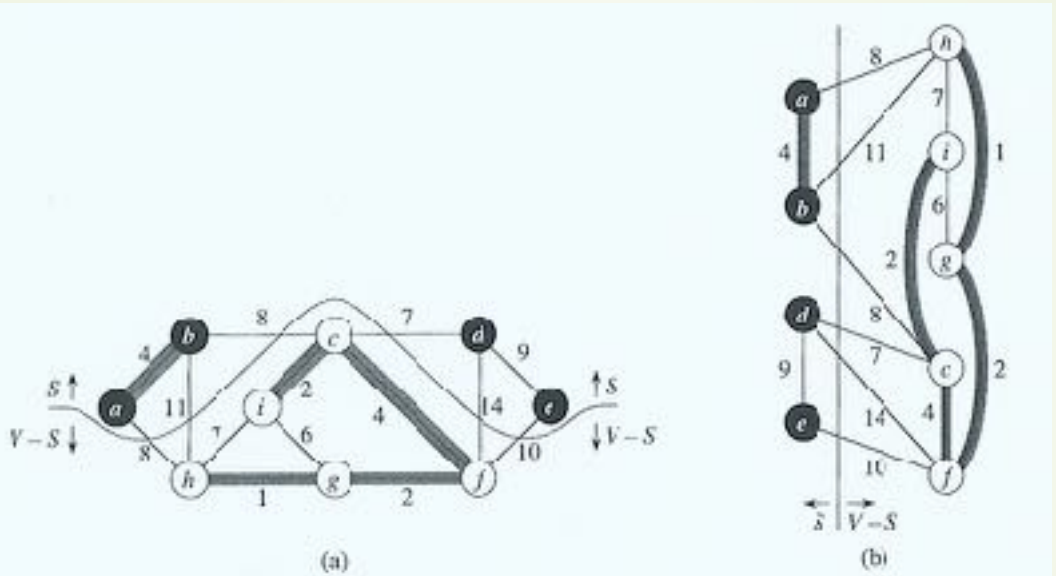


Figure 23.2 Two ways of viewing a cut $(S, V - S)$ of the graph from Figure 23.1. **(a)** The vertices in the set S are shown in black, and those in $V - S$ are shown in white. The edges crossing the cut are those connecting white vertices with black vertices. The edge (d, c) is the unique light edge crossing the cut. A subset A of the edges is shaded; note that the cut $(S, V - S)$ respects A, since no edge of A crosses the cut. **(b)** The same graph with the vertices in the set S on the left and the vertices in the set $V - S$ on the right. An edge crosses the cut if it connects a vertex on the left with a vertex on the right.