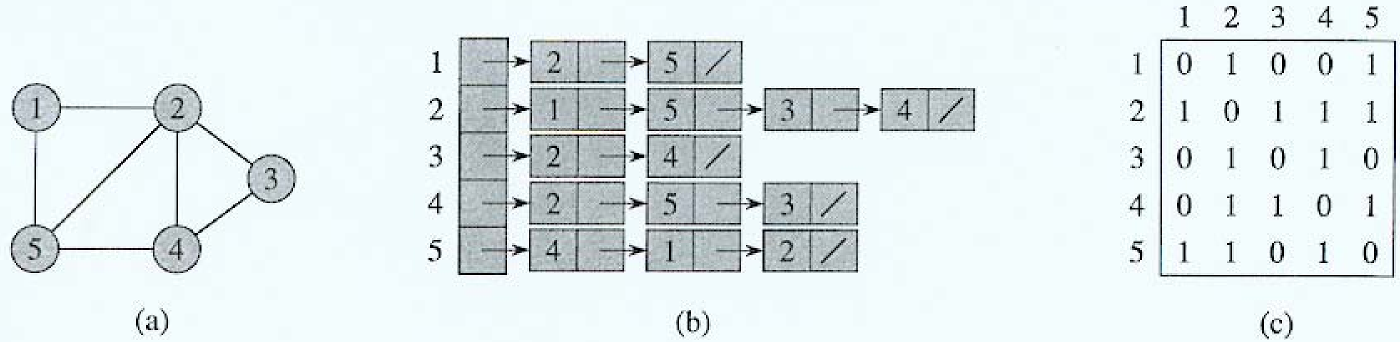


Figure 22.1 Two representations of an undirected graph. **(a)** An undirected graph G having five vertices and seven edges. **(b)** An adjacency-list representation of G. **(c)** The adjacency-matrix representation of G.

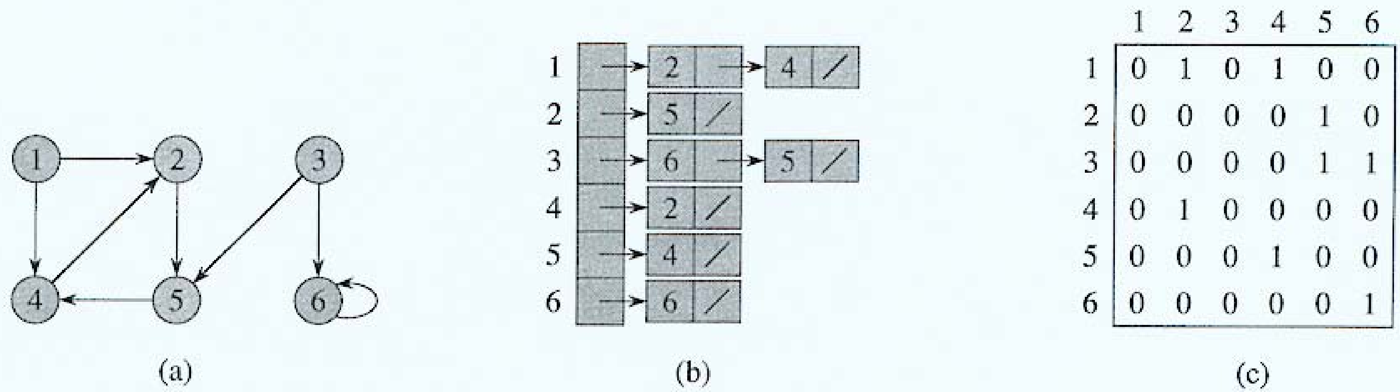


Figure 22.2 Two representations of a directed graph. **(a)** A directed graph G having six vertices and eight edges. **(b)** An adjacency-list representation of G. **(c)** The adjacency-matrix representation of G.

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.

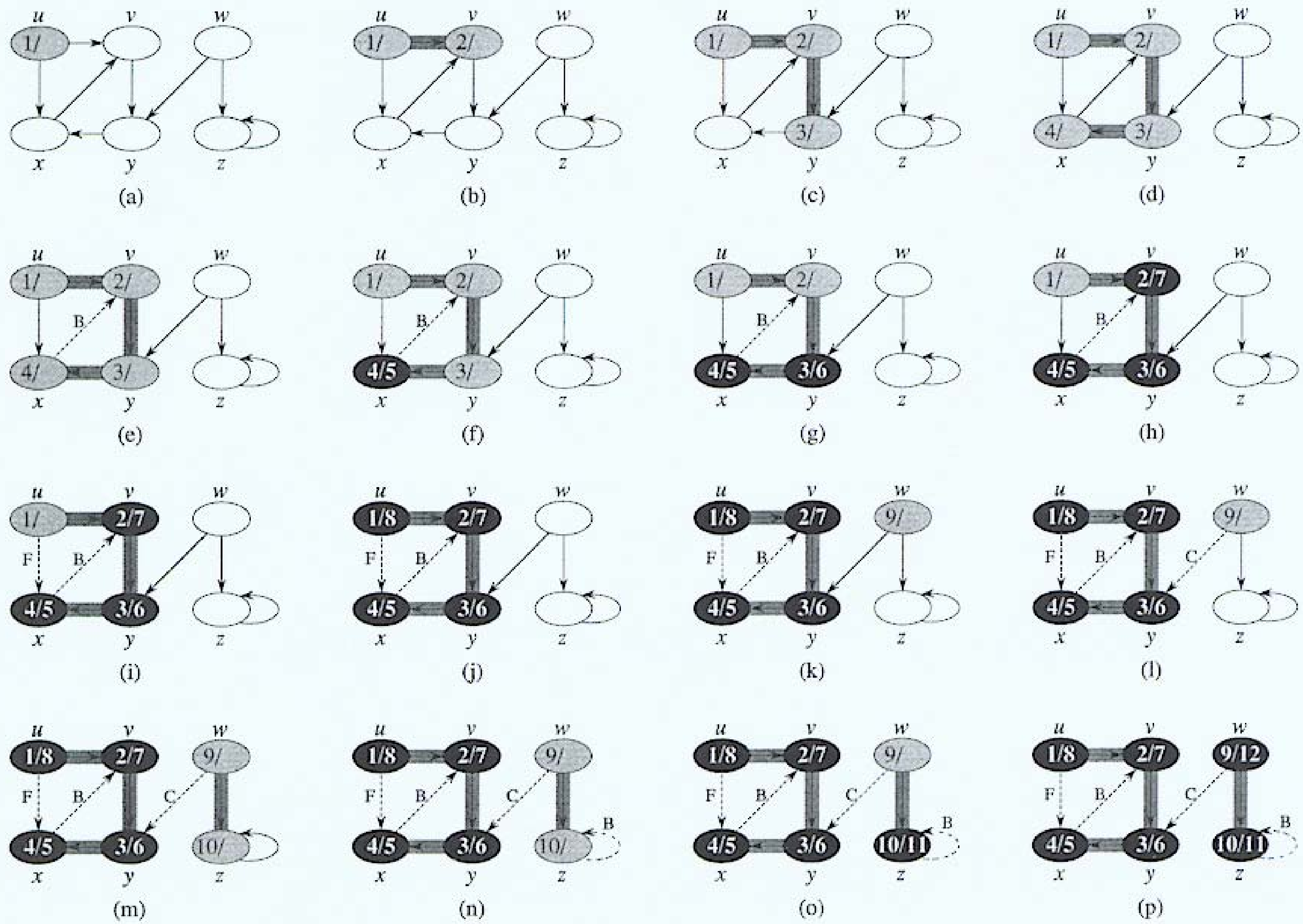


Figure 22.4 The progress of the depth-first-search algorithm DFS on a directed graph. As edges are explored by the algorithm, they are shown as either shaded (if they are tree edges) or dashed (otherwise). Nontree edges are labeled B, C, or F according to whether they are back, cross, or forward edges. Vertices are timestamped by discovery time/finishing time.

```
DFS(G)
For each vertex u ∈ V[G] do
      color[u] ← WHITE
      π[u] ← NIL
Time ← 0
For each vertex u ∈ V[G] do
      if color[u] = WHITE then
            DFS-VISIT(u)
```

Depth First Search

```
DFS-VISIT(u)
VisitVertex(u)
Color[u] ← GRAY
Time ← Time + 1
d[u] ← Time
for each v ∈ Adj[u] do
    VisitEdge(u,v)
    if (v ≠ π[u]) then
        if (color[v] = WHITE) then
            π[v] ← u
            DFS-VISIT(v)
color[u] ← BLACK
F[u] ← Time ← Time + 1
```

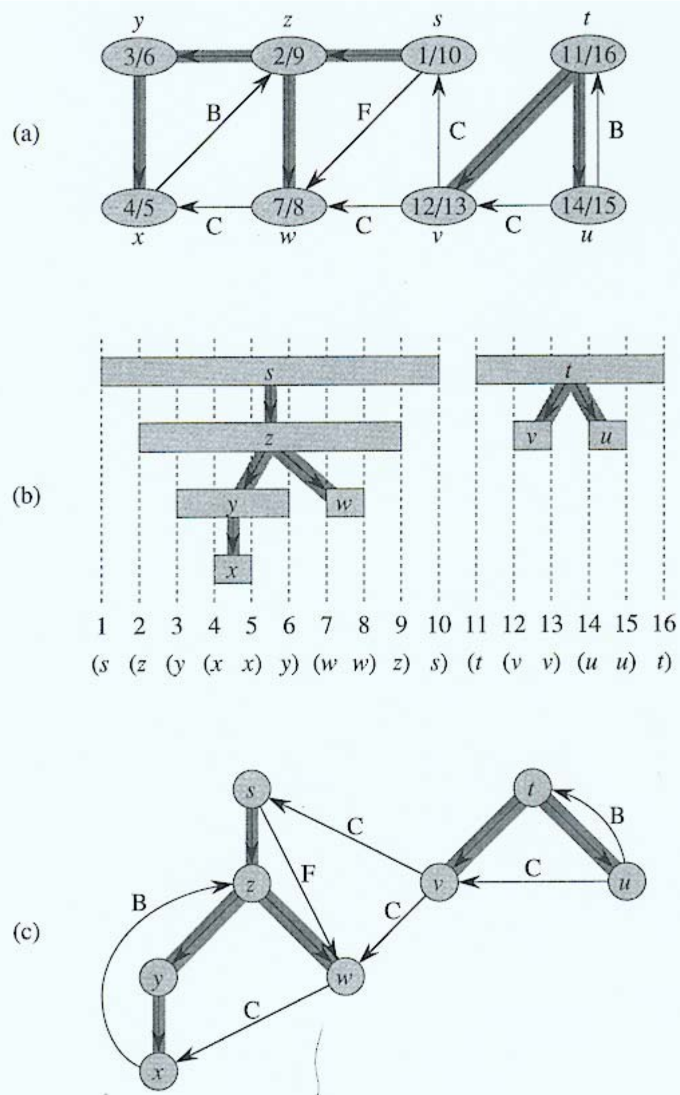


Figure 22.5 Properties of depth-first search. **(a)** The result of a depth-first search of a directed graph. Vertices are timestamped and edge types are indicated as in Figure 22.4. **(b)** Intervals for the discovery time and finishing time of each vertex correspond to the parenthesization shown. Each rectangle spans the interval given by the discovery and finishing times of the corresponding vertex. Tree edges are shown. If two intervals overlap, then one is nested within the other, and the vertex corresponding to the smaller interval is a descendant of the vertex corresponding to the larger. **(c)** The graph of part (a) redrawn with all tree and forward edges going down within a depth-first tree and all back edges going up from a descendant to an ancestor.

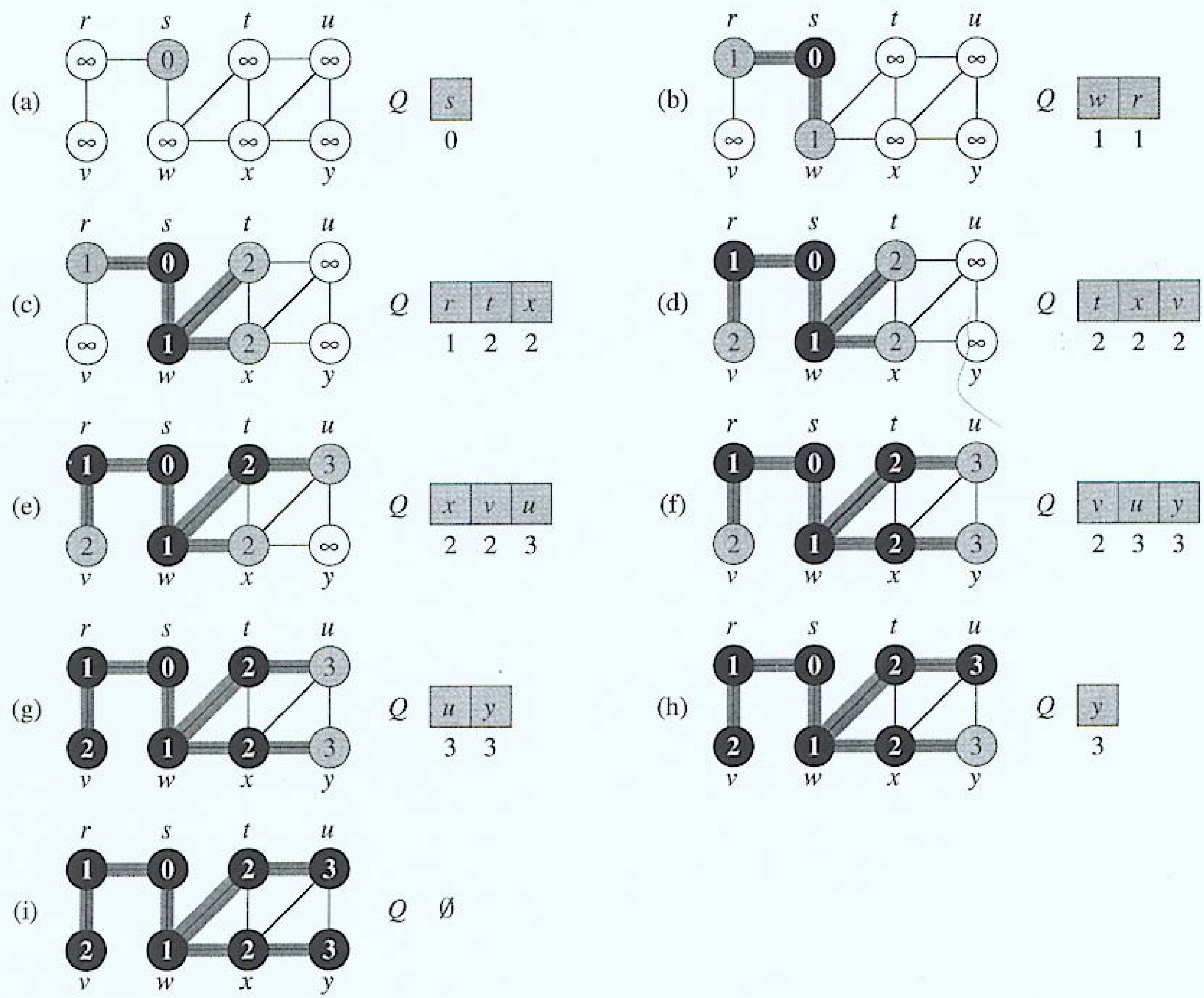


Figure 22.3 The operation of BFS on an undirected graph. Tree edges are shown shaded as they are produced by BFS. Within each vertex u is shown $d[u]$. The queue Q is shown at the beginning of each iteration of the **while** loop of lines 10–18. Vertex distances are shown next to vertices in the queue.

Breadth First Search

```
BFS(G,s)
For each vertex u ∈V[G] - {s} do
    color[u] ← WHITE
     d[u] ← ∞
     π[u] ← NIL
Color[u] ← GRAY
D[s] ← 0
π[s] ← NIL
Q ← Φ
ENQUEUE(Q,s)
While Q ≠ Φ do
    u ← DEQUEUE(Q)
    VisitVertex(u)
    for each v ∈ Adj[u] do
        VisitEdge(u,v)
        if (color[v] = WHITE) then
            color[v] ← GRAY
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q,v)
    color[u] ← BLACK
```

Figure 14.33

An activity-node graph

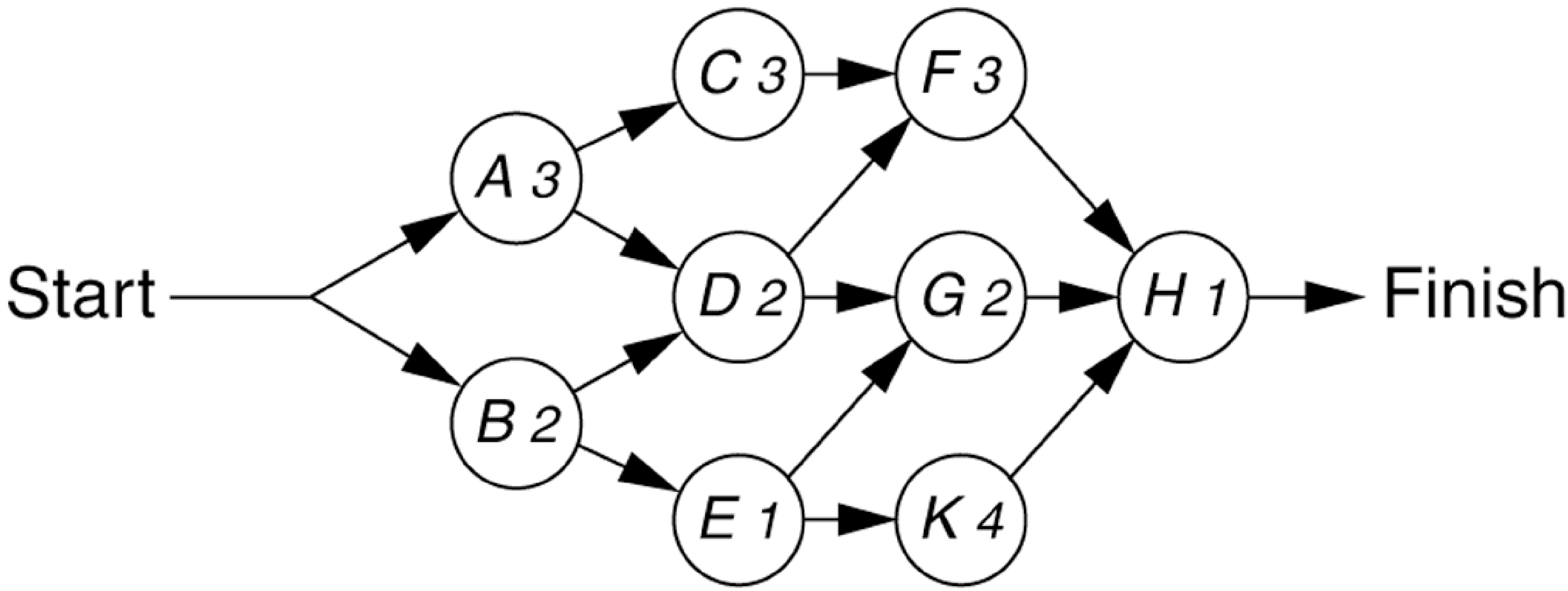


Figure 14.30A

A topological sort. The conventions are the same as those in Figure 14.21 (continued).

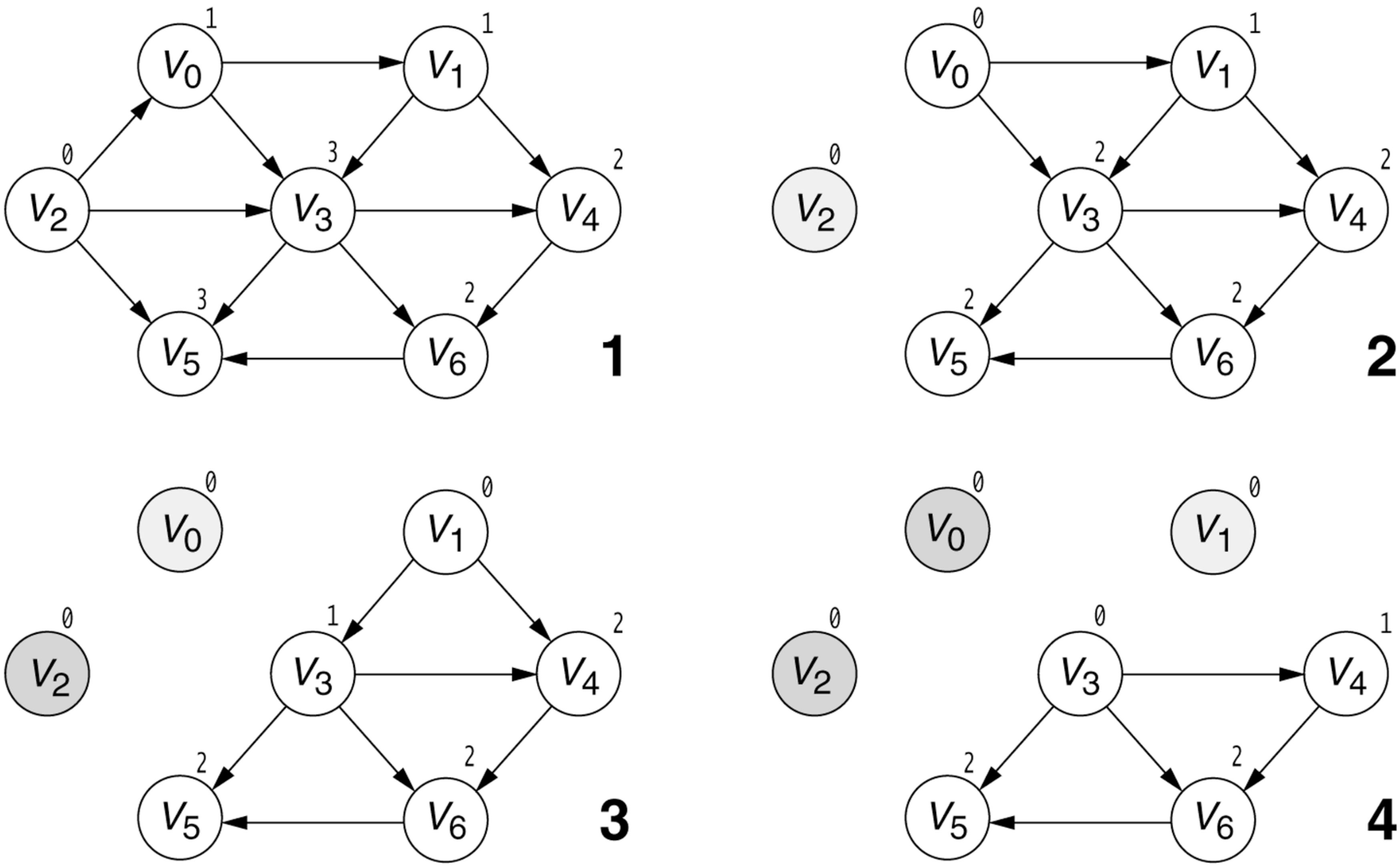


Figure 14.30B

A topological sort. The conventions are the same as those in Figure 14.21.

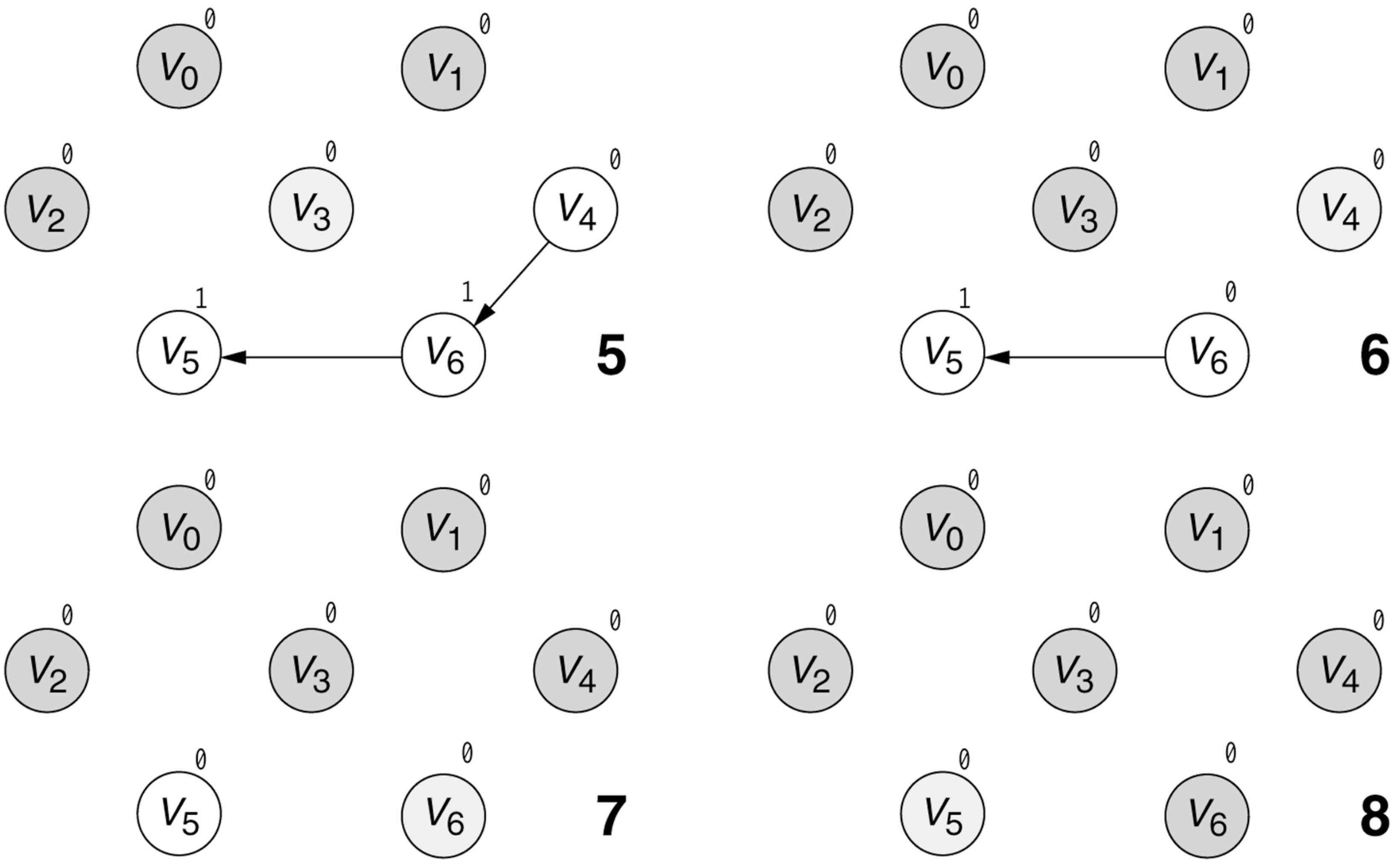


Figure 14.31A

The stages of acyclic graph algorithm. The conventions are the same as those in Figure 14.21 (*continued*).

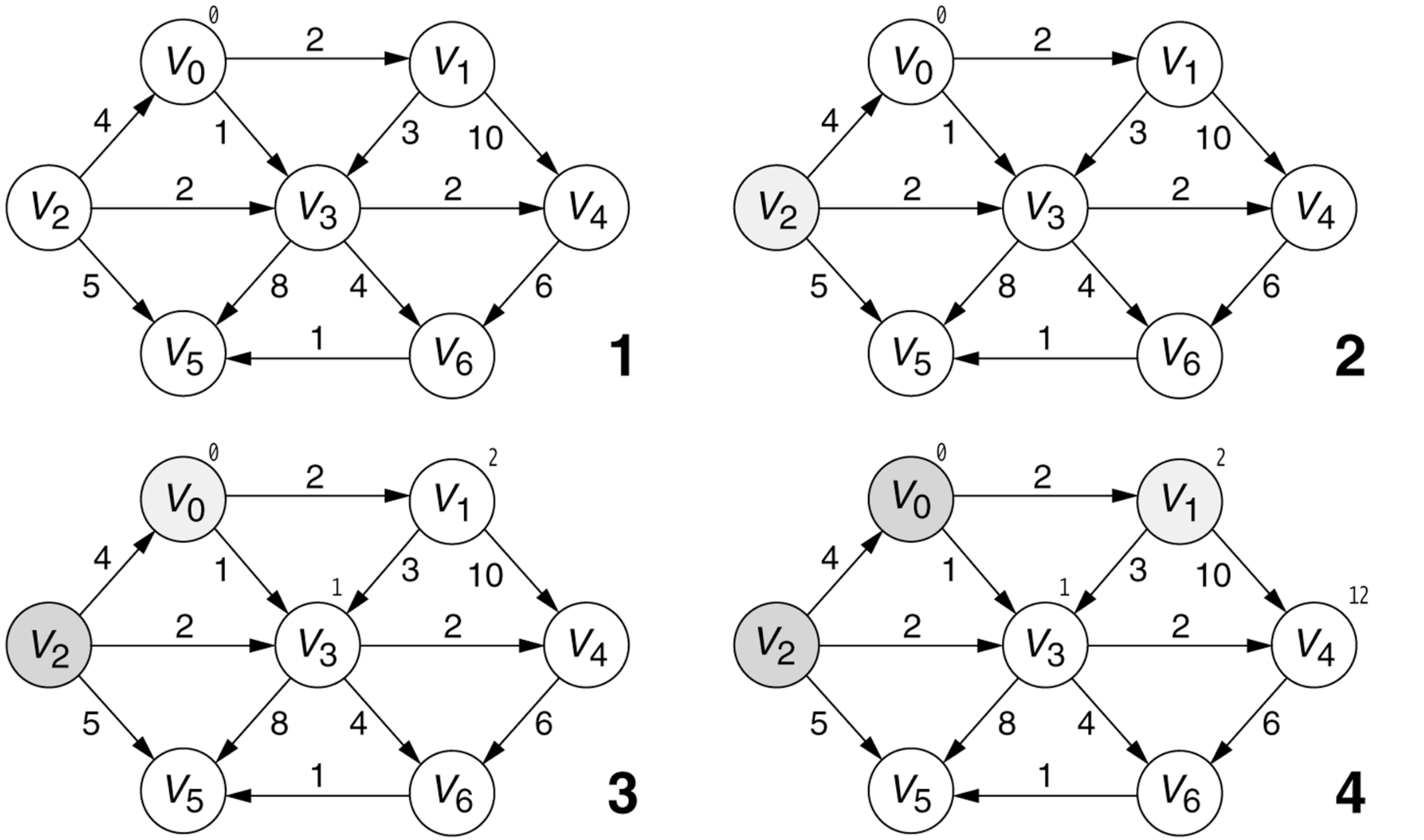


Figure 14.31B

The stages of acyclic graph algorithm. The conventions are the same as those in Figure 14.21.

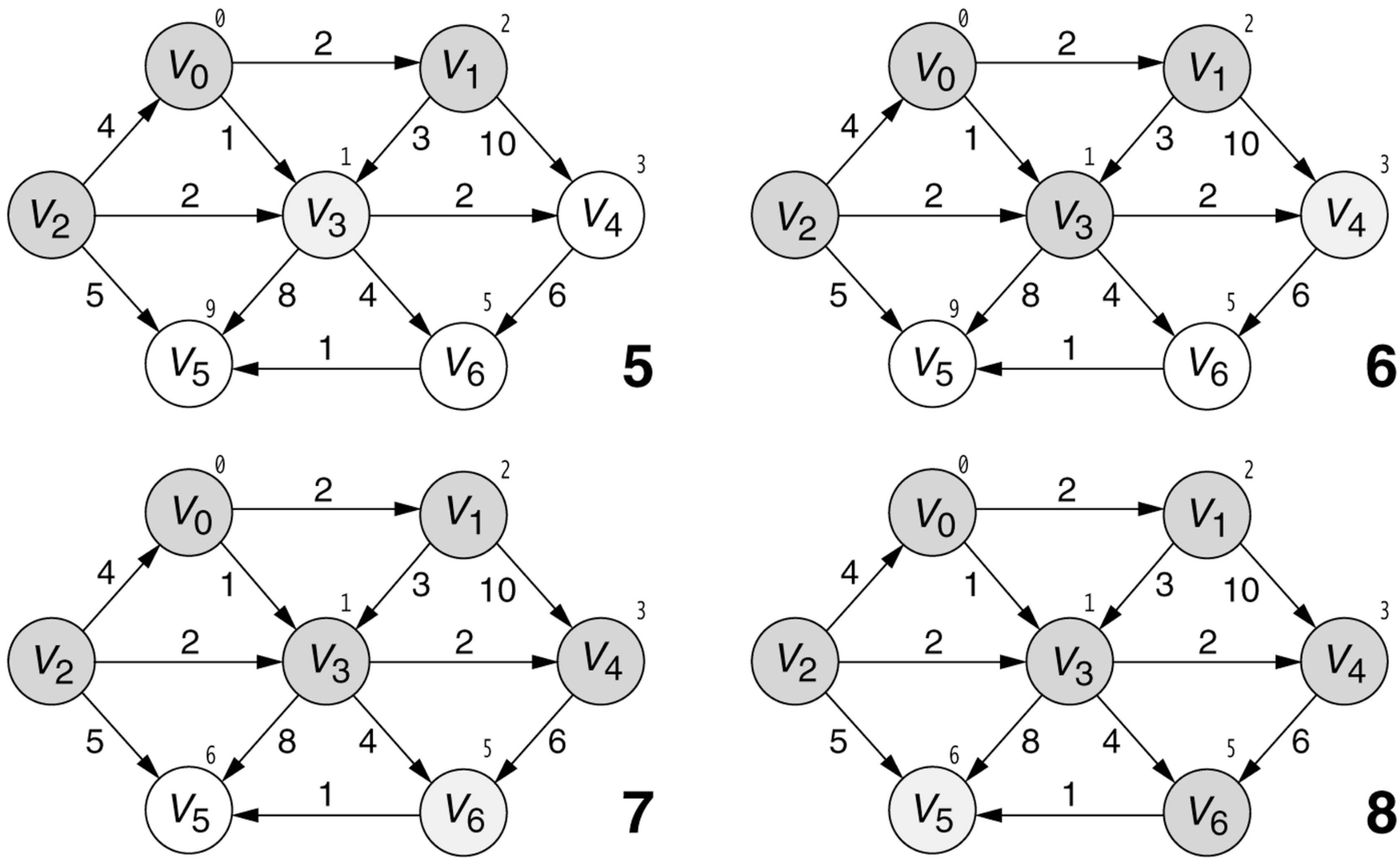


Figure 14.34
An event-node graph

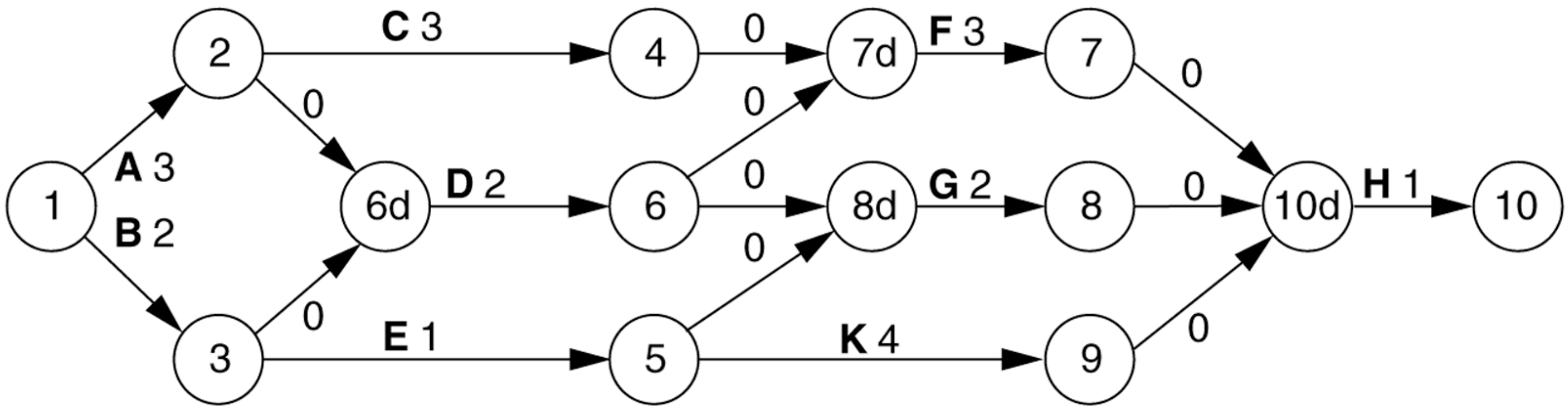


Figure 14.35

Earliest completion times

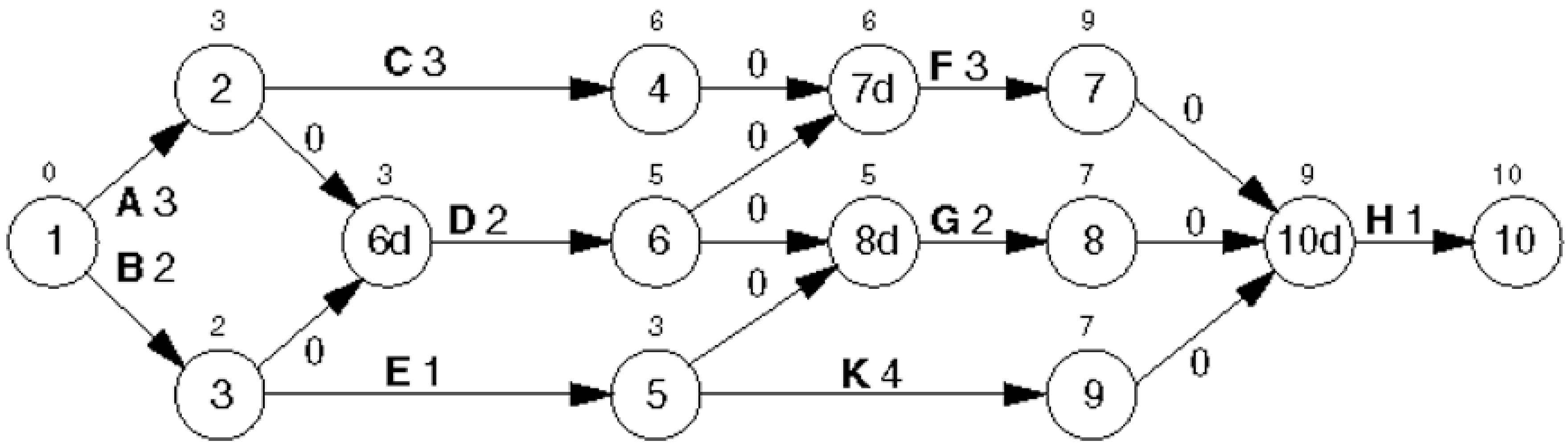


Figure 14.36
Latest completion times

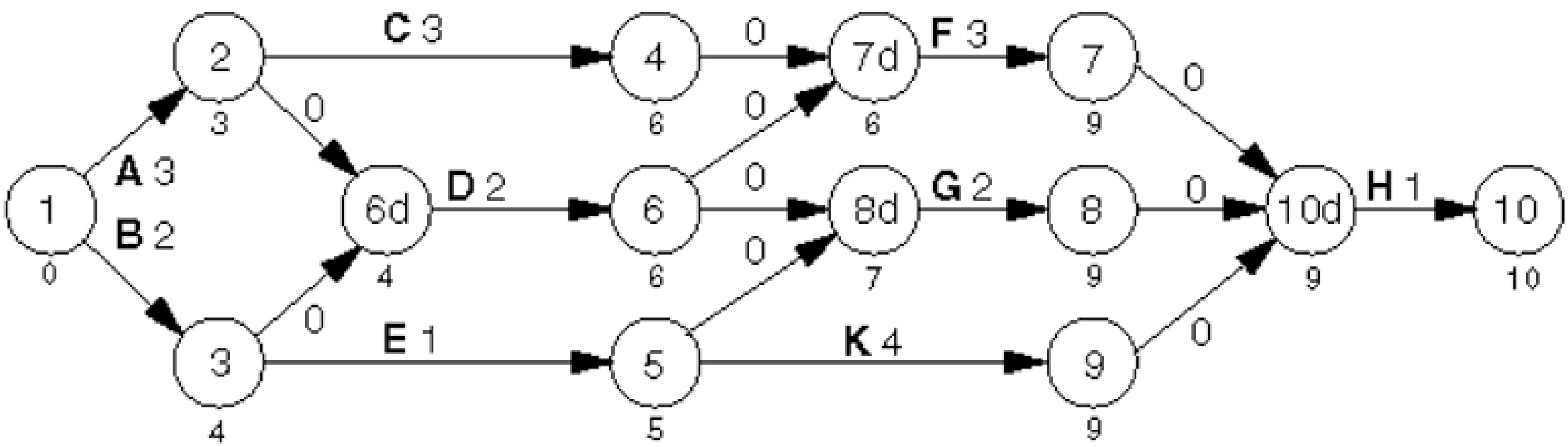
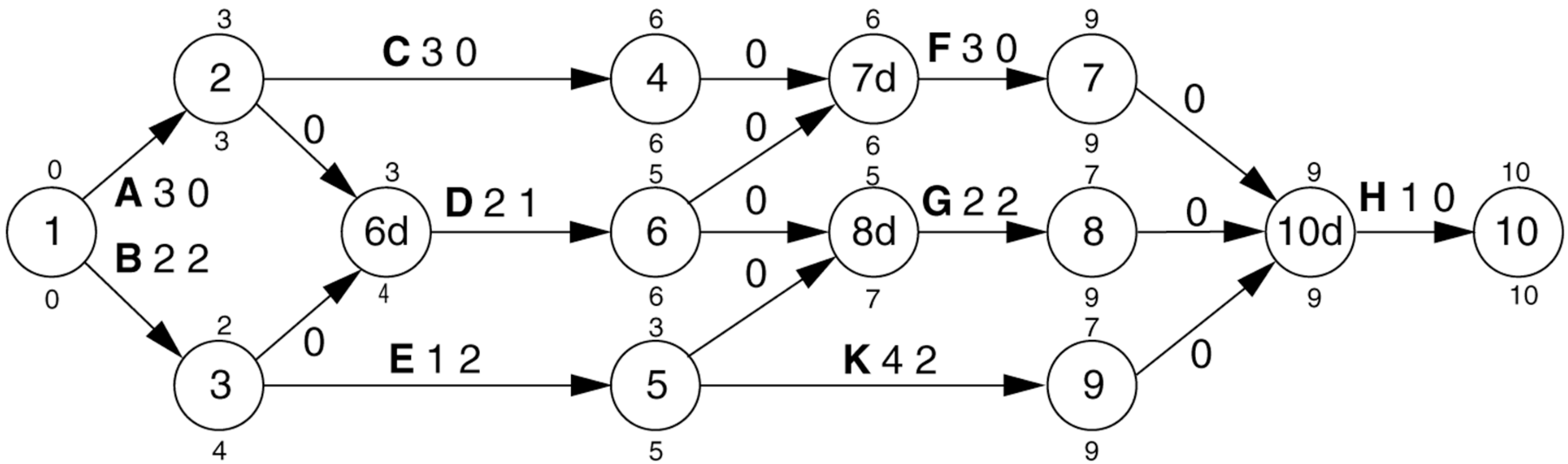


Figure 14.37

Earliest completion time, latest completion time, and slack (additional edge item)



Connectivity

- A (simple) undirected graph is connected if there exists a path between every pair of vertices.
- If a graph is not connected, then G'(V',E') is a connected component of the graph G(V,E) if V' is a maximal subset of vertices from V that induces a connected subgraph. (What is the meaning of maximal?)
- The connected components of a graph correspond to a partition of the set of the vertices. (What is the meaning of partition?)
- How to compute all the connected components?
  - Use DFS or BFS.

Minimum Spanning Tree

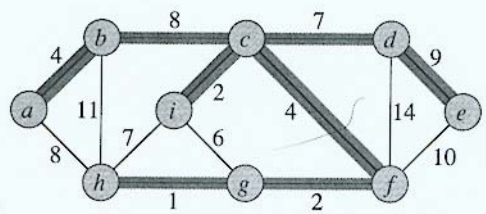


Figure 23.1 A minimum spanning tree for a connected graph. The weights on edges are shown, and the edges in a minimum spanning tree are shaded. The total weight of the tree shown is 37. This minimum spanning tree is not unique: removing the edge (b, c) and replacing it with the edge (a, h) yields another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

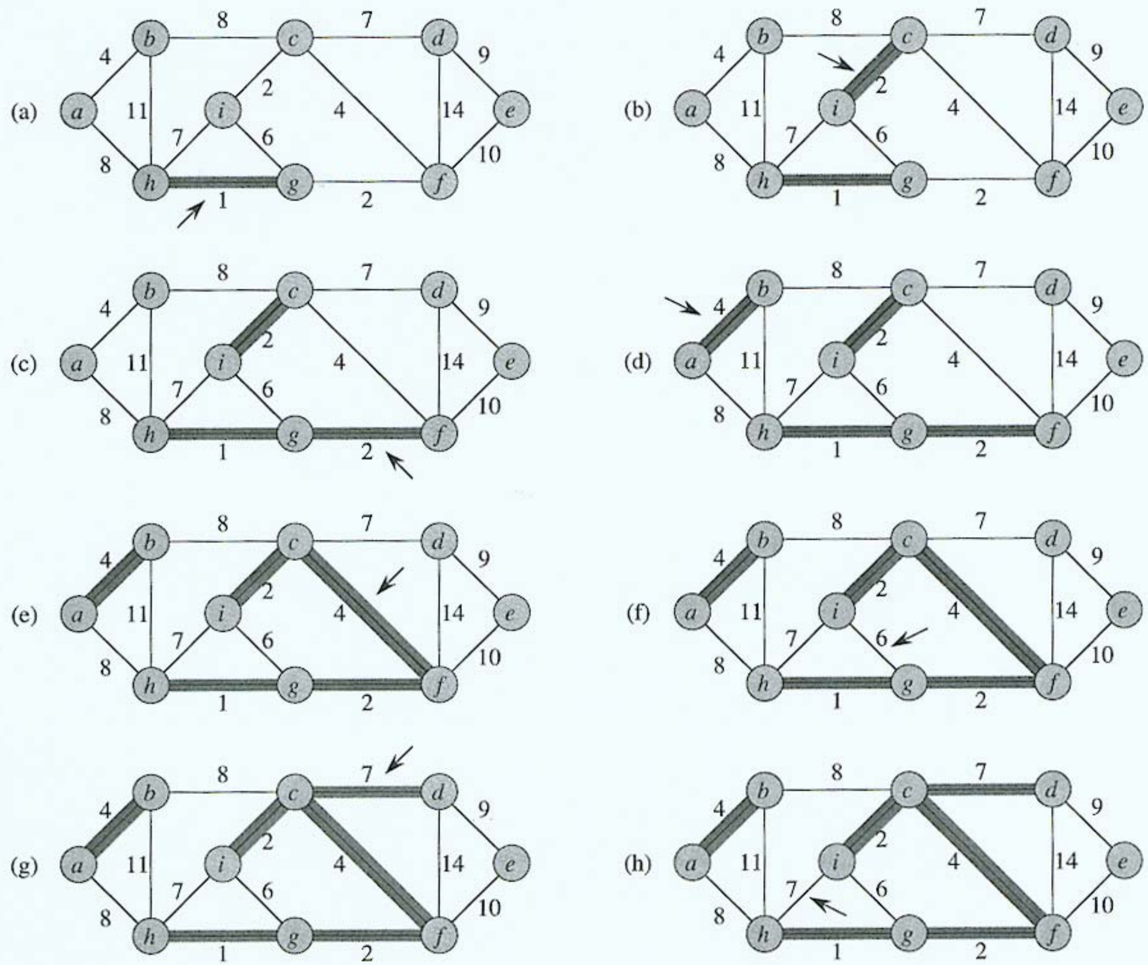
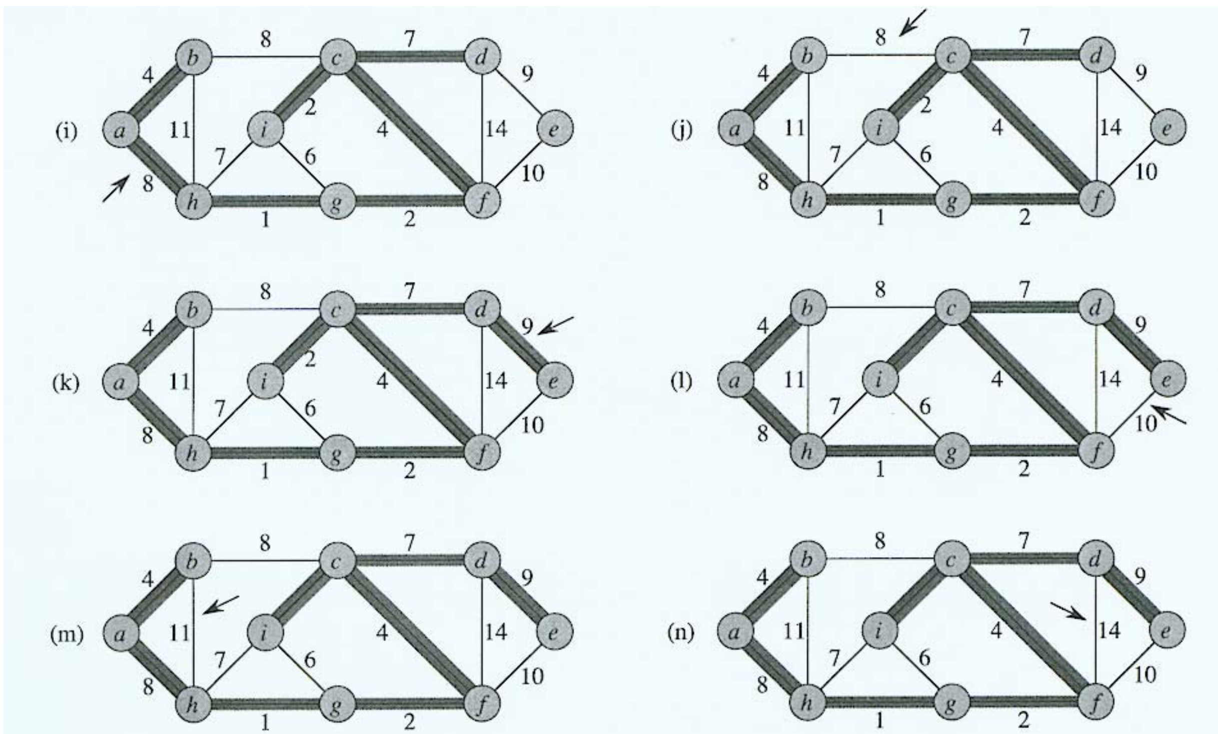


Figure 23.4 The execution of Kruskal's algorithm on the graph from Figure 23.1. Shaded edges belong to the forest A being grown. The edges are considered by the algorithm in sorted order by weight. An arrow points to the edge under consideration at each step of the algorithm. If the edge joins two distinct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


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