```
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)
```

```
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
```

Depth First Search

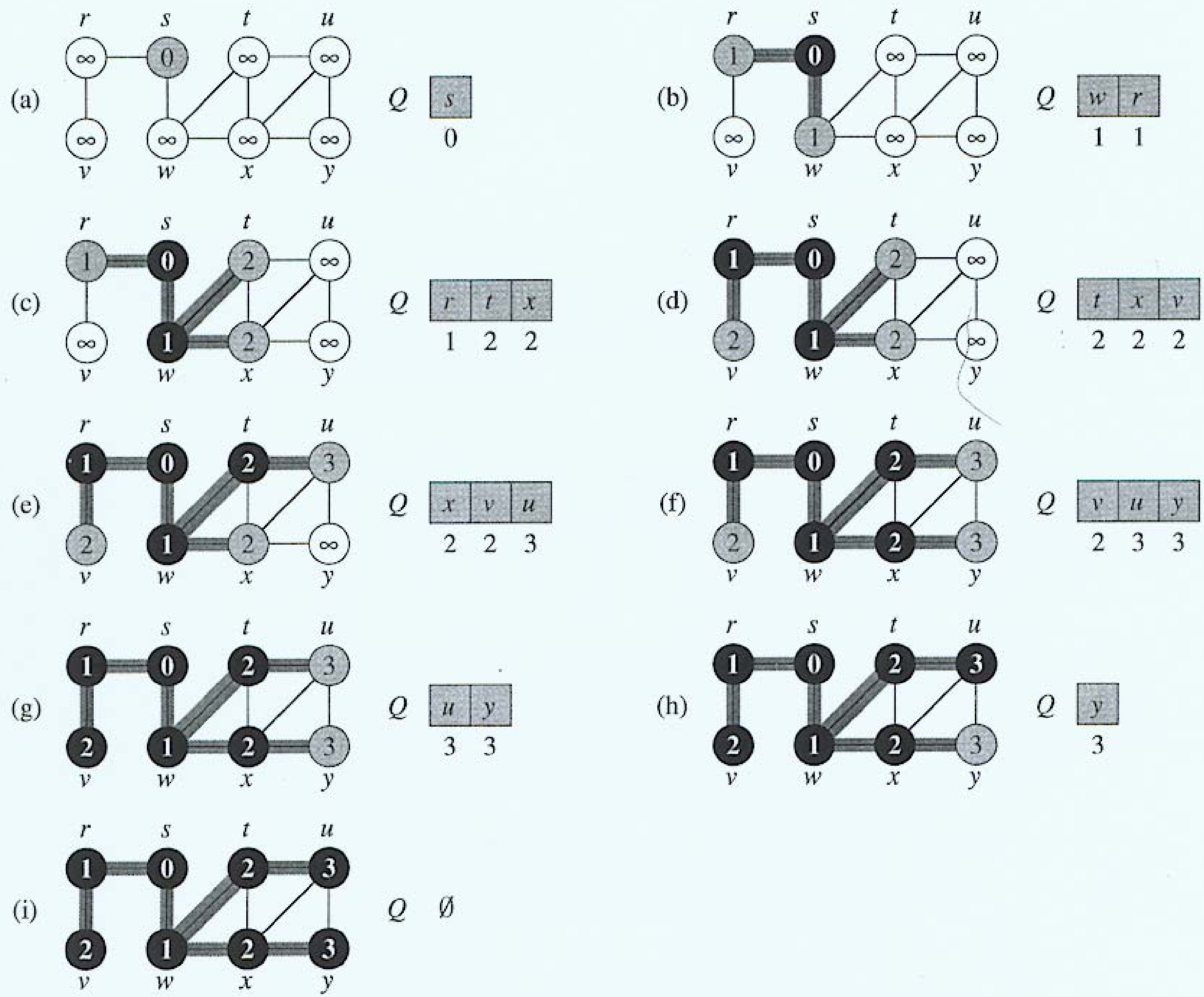


Figure 22.3 The operation of BFS on an undirected graph. Tree edges are shown shaded as they are produced by BFS. Within each vertex u is shown $d[u]$. The queue Q is shown at the beginning of each iteration of the **while** loop of lines 10–18. Vertex distances are shown next to vertices in the queue.

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 14.30A

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

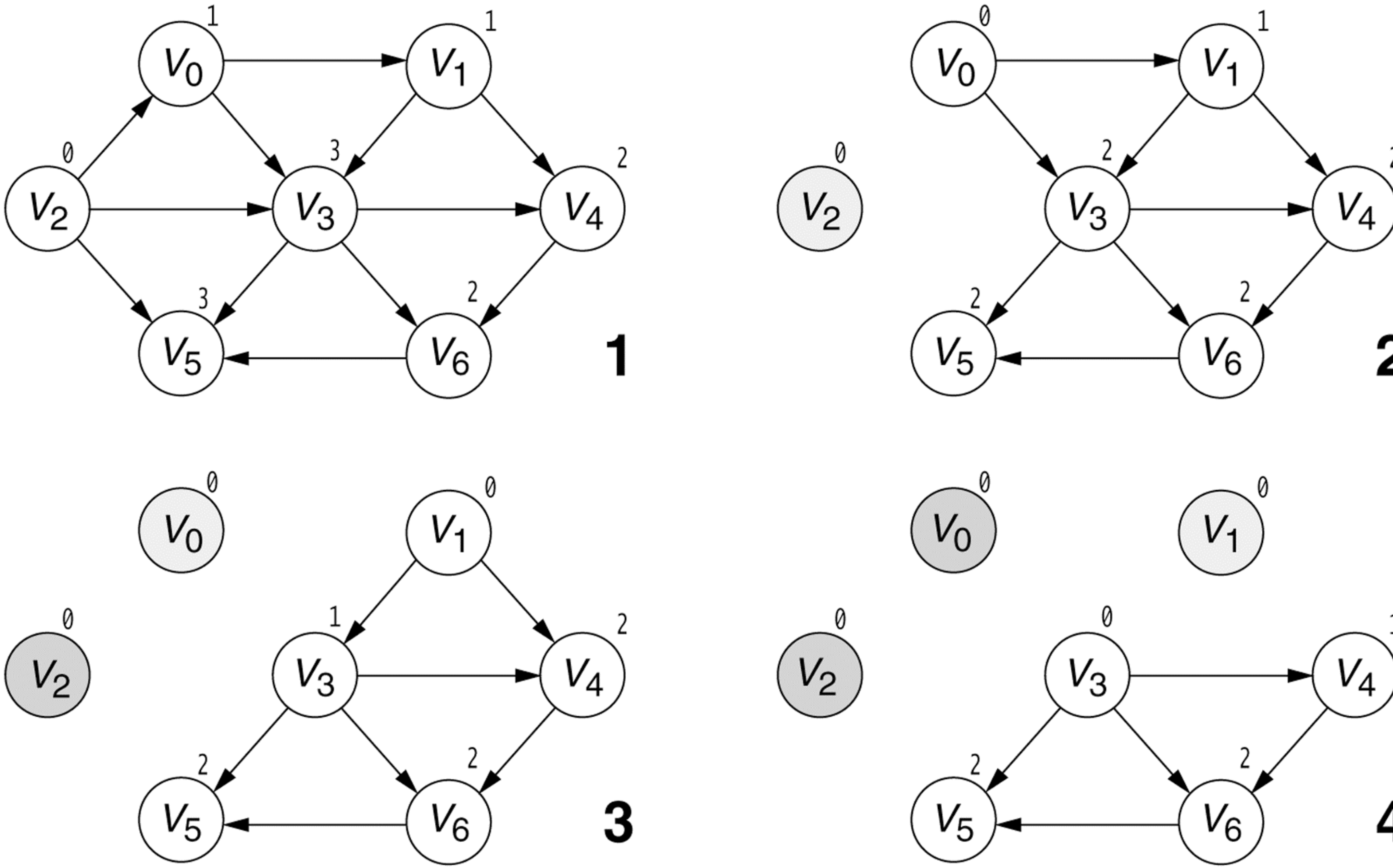


Figure 14.30B

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

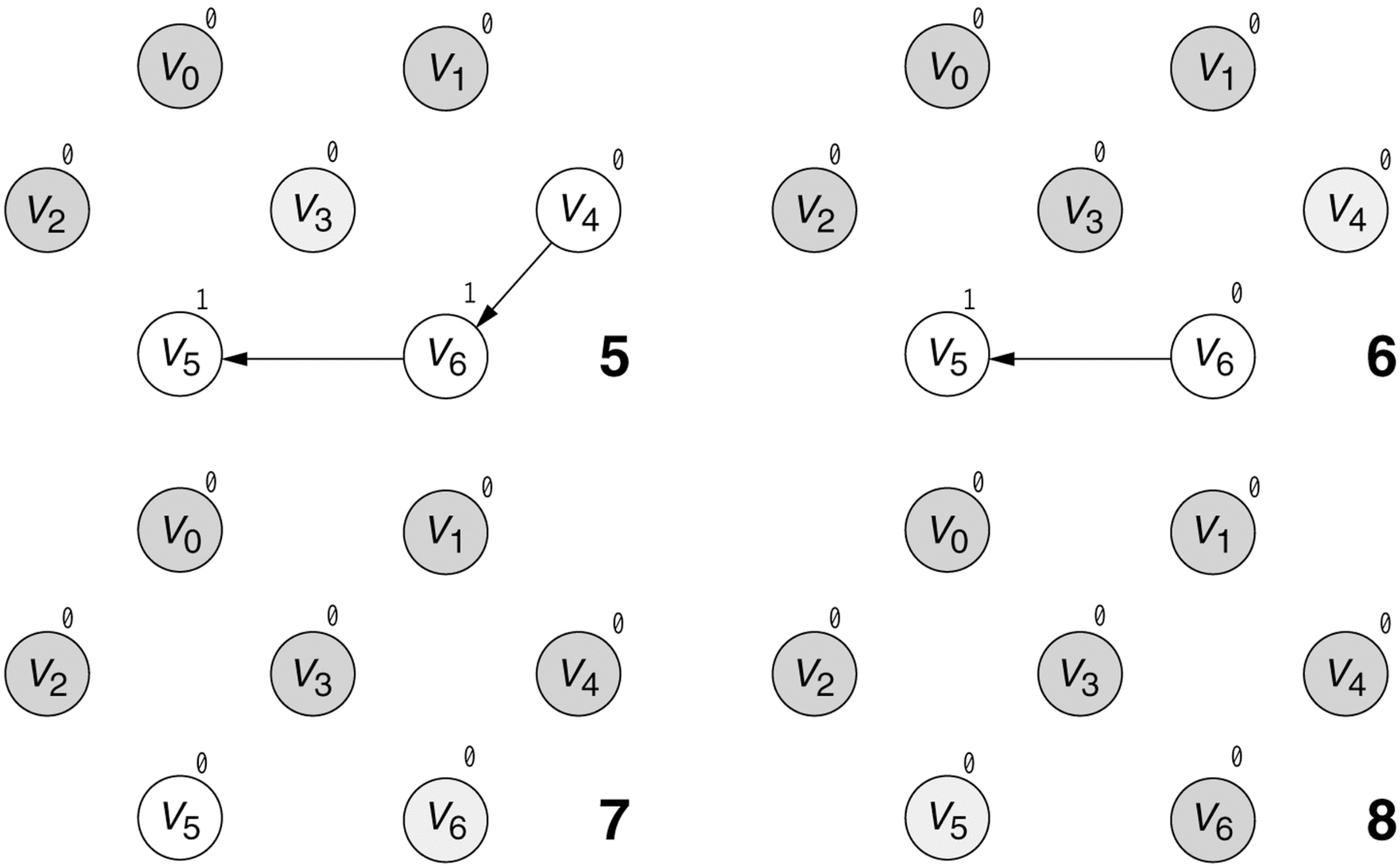


Figure 14.31A

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

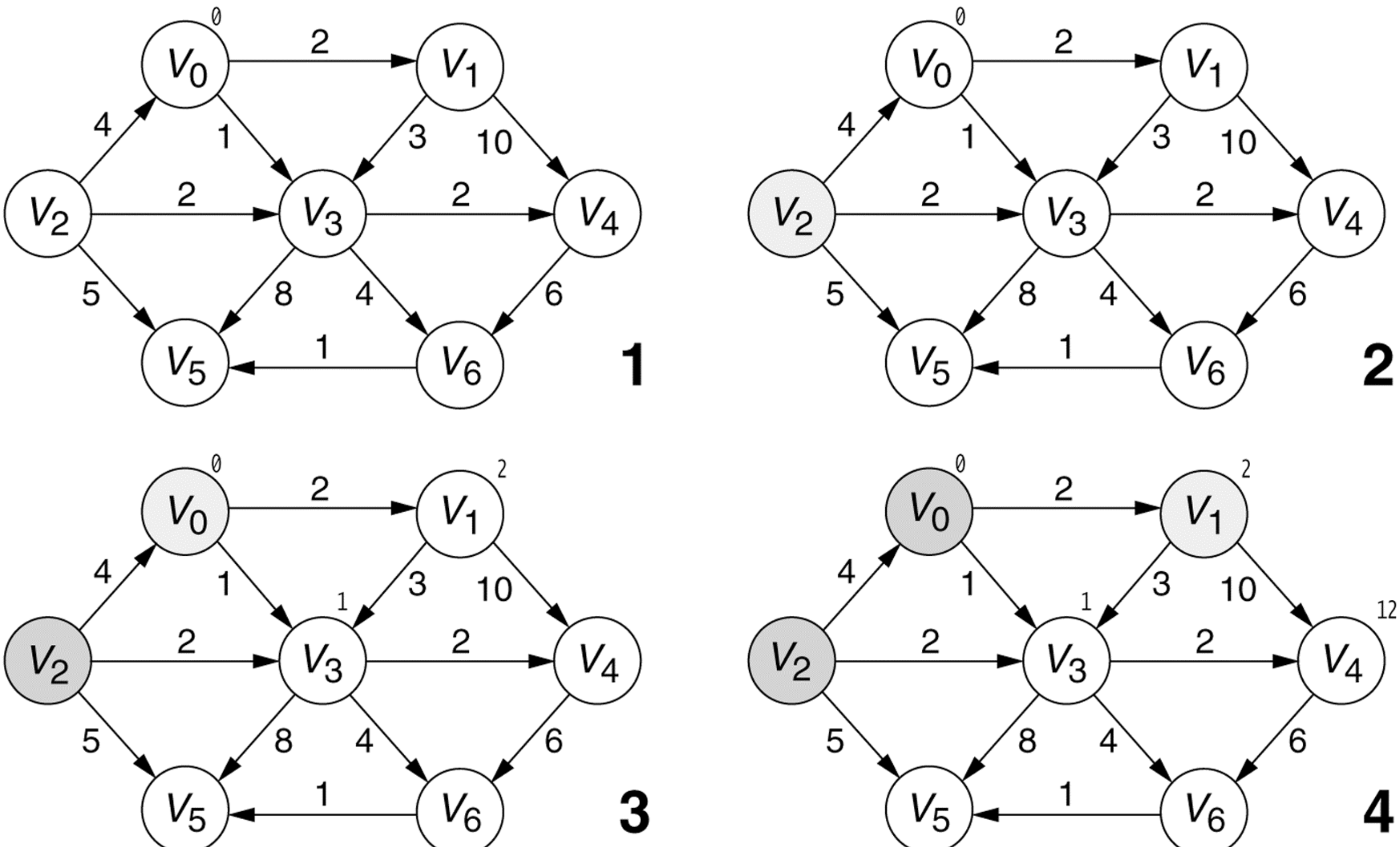
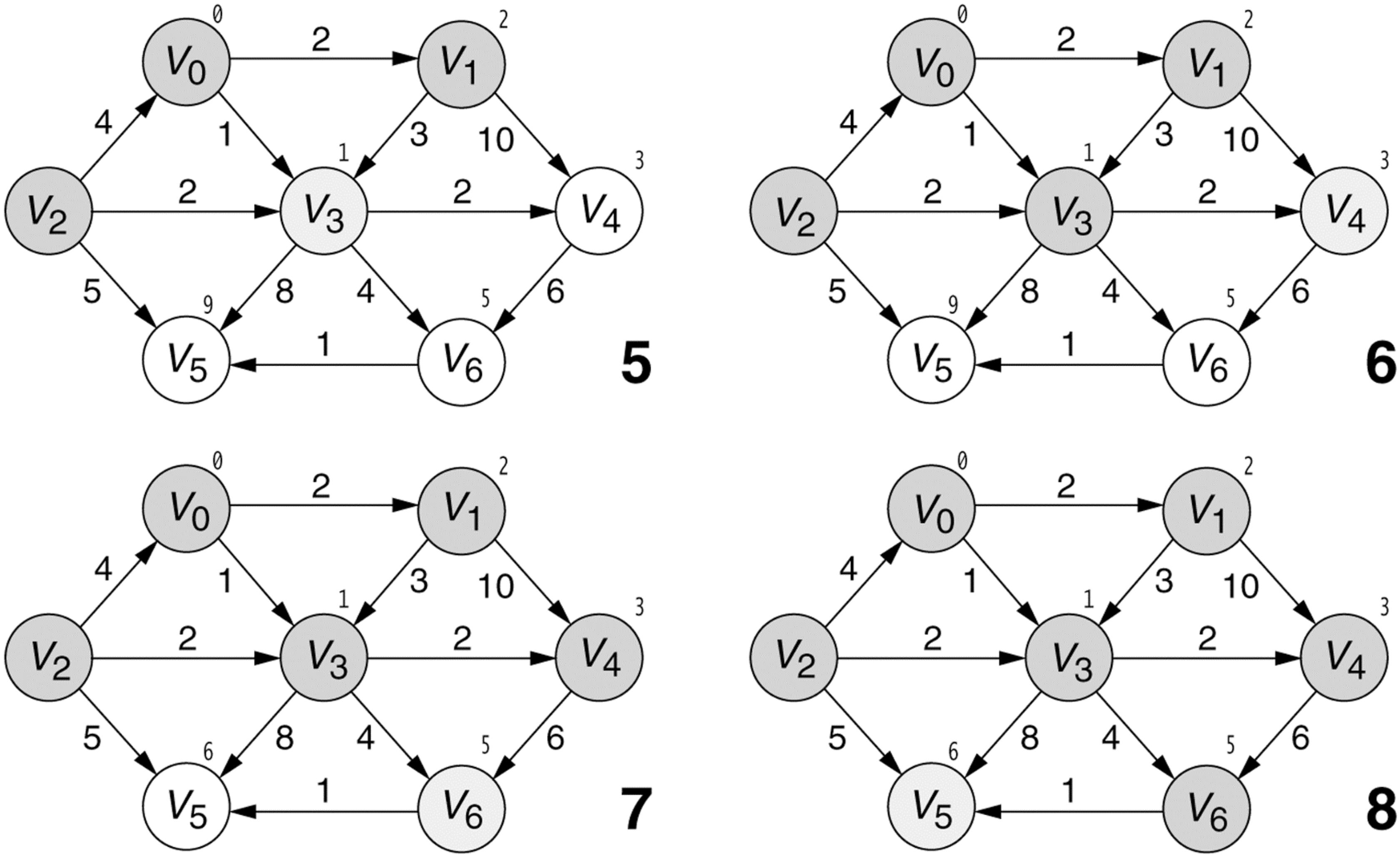


Figure 14.31B

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



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.

Biconnectivity: Generalizing Connectivity

- A tree is a minimally connected graph.
- Removing a vertex from a connected graph may make it disconnected.
- A graph is biconnected if removing a single vertex does not disconnect the graph.
- Alternatively, a graph is biconnected if for every pair of vertices there exists at least 2 disjoint paths between them.
- A graph is k-connected if for every pair of vertices there exists at least k disjoint paths between them. Alternatively, removal of any k-1 vertices does not disconnect the graph.

Biconnected Components

- If a graph is not biconnected, it can be decomposed into biconnected components.
- An articulation point is a vertex whose removal disconnects the graph.
- **Claim:** If a graph is not biconnected, it must have an articulation point. Proof?
- A biconnected component of a simple undirected graph G(V,E) is a maximal set of edges from E that induces a biconnected subgraph.

Biconnected Components

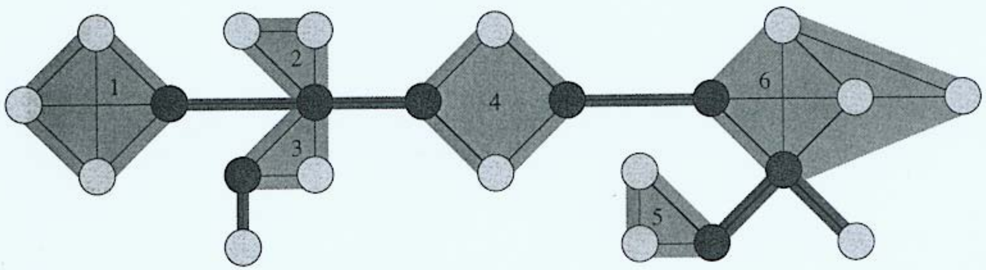


Figure 22.10 The articulation points, bridges, and biconnected components of a connected, undirected graph for use in Problem 22-2. The articulation points are the heavily shaded vertices, the bridges are the heavily shaded edges, and the biconnected components are the edges in the shaded regions, with a *bcc* numbering shown.