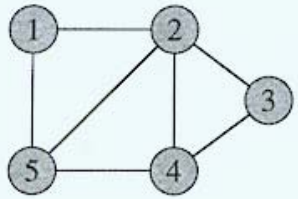
Priority Queue

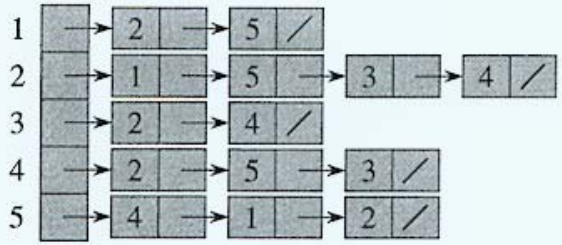
- **Operations**
  - MAXIMUM(S) ,
  - INSERT(S, x)
  - EXTRACT-MAX(S)
  - INCREASE-KEY(S, x, k)
- **Implementation**
  - Use a HEAP.
  - MAXIMUM:
    - Return value stored at root.
  - INSERT:
    - Insert at last leaf and percolate up tree.
  - EXTRACT-MAX:
    - Delete root of heap and call HEAPIFY.
  - INCREASE-KEY:
    - Change value and percolate up tree.

Graphs

- Graph $G(V,E)$
- V Vertices or Nodes
- E Edges or Links: pairs of vertices
- D Directed vs. Undirected edges
- Weighted vs Unweighted
- Graphs can be augmented to store extra info (e.g., city population, oil flow capacity, etc.)
- Paths and Cycles
- Subgraphs $G'(V',E')$, where V' is a subset of V and E' is a subset of E
- Trees and Spanning trees



(a)

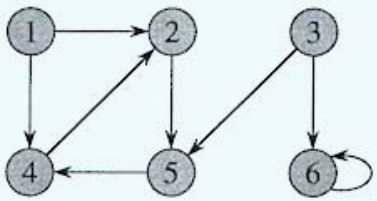


(b)

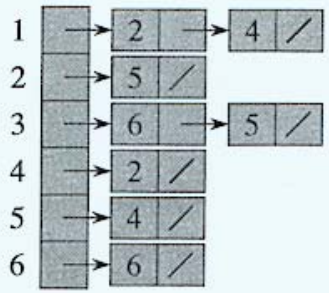
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

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.



(a)



(b)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

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.

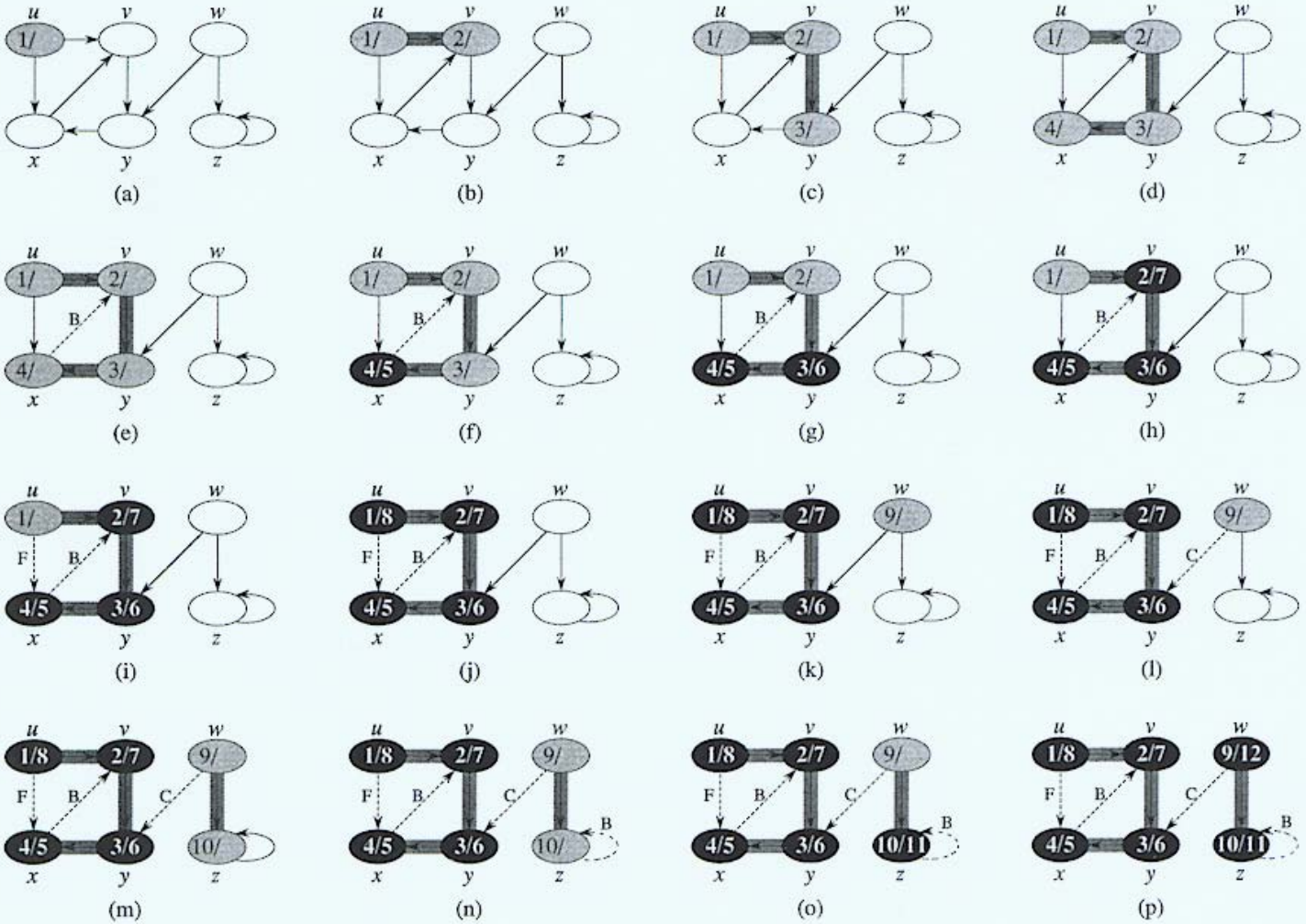


Figure 22.4 The progress of the depth-first-search algorithm DFS on a directed graph. As edges are explored by the algorithm, they are shown as either shaded (if they are tree edges) or dashed (otherwise). Nontree edges are labeled B, C, or F according to whether they are back, cross, or forward edges.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

```
DFS(G)
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

```
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 (color[v] = WHITE) then
8.              π[v] ← u
9.              DFS-VISIT(v)
10. color[u] ← BLACK
11. F[u] ← Time ← Time + 1
```

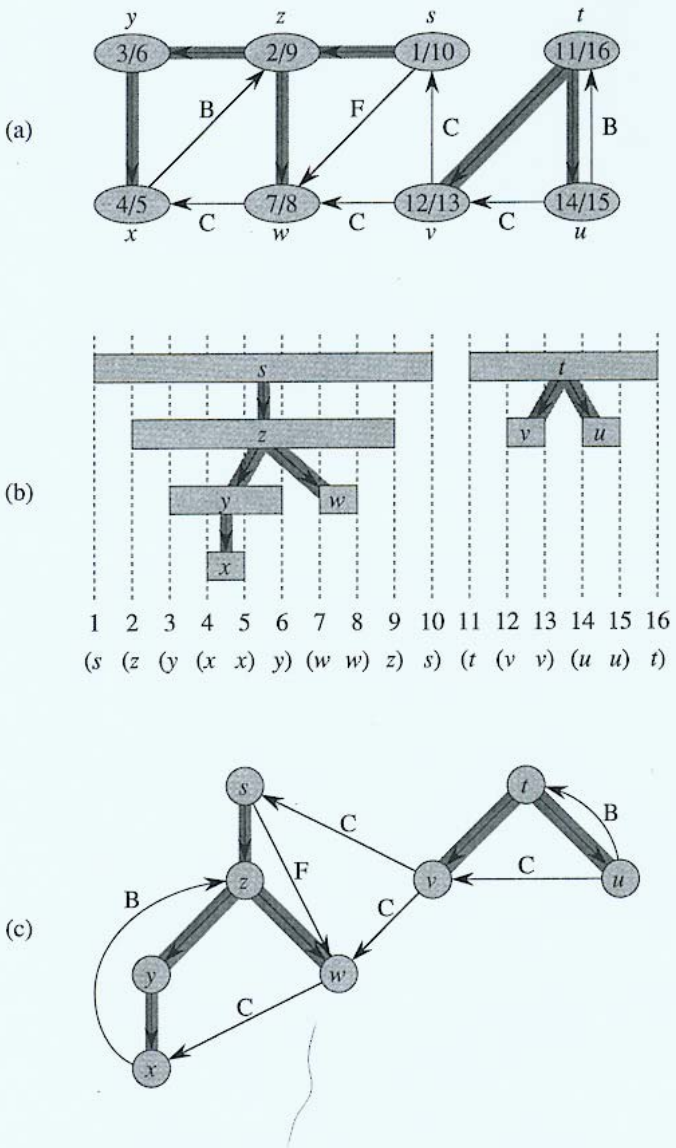


Figure 22.5 Properties of depth-first search. **(a)** The result of a depth-first search of a directed graph. Vertices are timestamped and edge types are indicated as in Figure 22.4. **(b)** Intervals for the discovery time and finishing time of each vertex correspond to the parenthesization shown. Each 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.

Mid Term Exam 1

MidTem 1 Statistics		
Range	Number	Approx. Grade
50 to 75	1	A- to A
35 to 49	3	B+ to A-
20 to 34	4	B- to B+
AVERAGE = 24		
15 to 19	6	C- to C+
10 to 14	5	D to C-
0 to 9	1	F to D