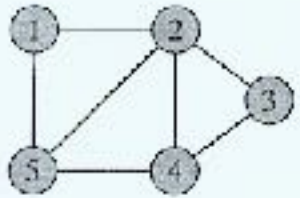
COT 6405: Analysis of Algorithms

Giri NARASIMHAN

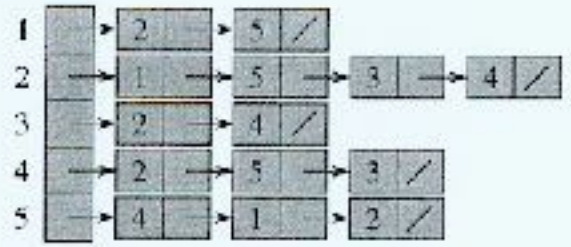
www.cs.fiu.edu/~giri/teach/6405F19.html

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)

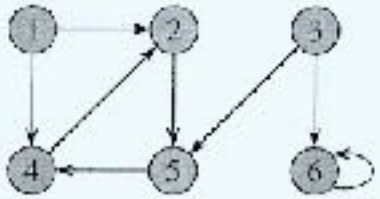


(b)

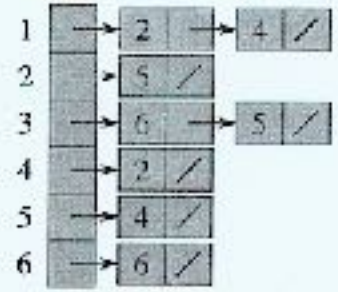
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(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.

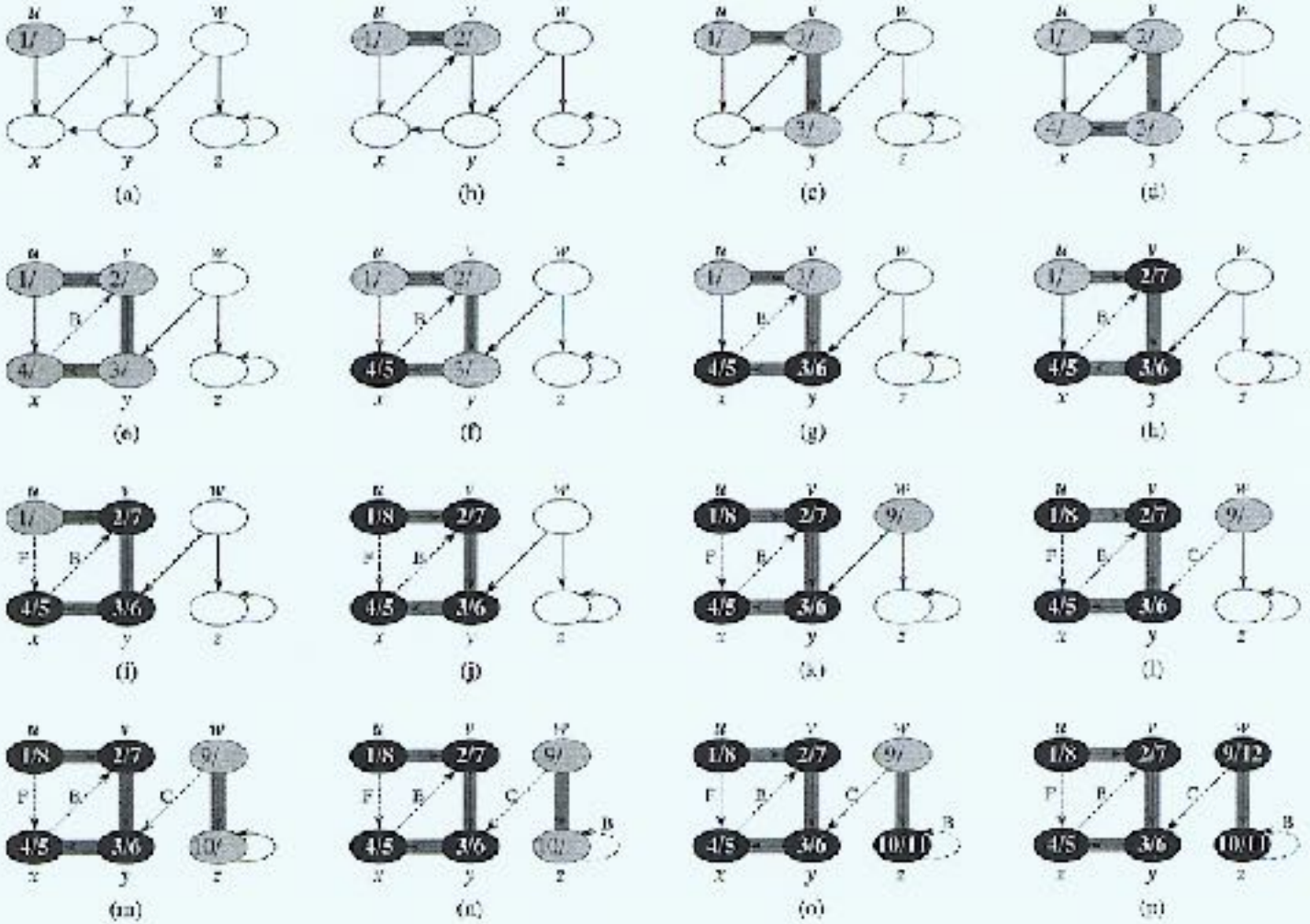


Figure 22.4 The progress of the depth-first-search algorithm DFS on a directed graph. As edges are explored by the algorithm, they are shown as either shaded (if they are tree edges) or dashed (otherwise). Nontree edges are labeled B, C, or F according to whether they are back, cross, or forward edges.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

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
```

DFS Tree

- Consider all the edges (π[v], v), for all v in V.
- For a **connected** undirected graph, these edges form a tree called the DFS tree.
  - If the graph is not connected, these edges form a DFS forest
- For a **strongly connected** directed graph, these edges form a directed tree called the DFS tree.

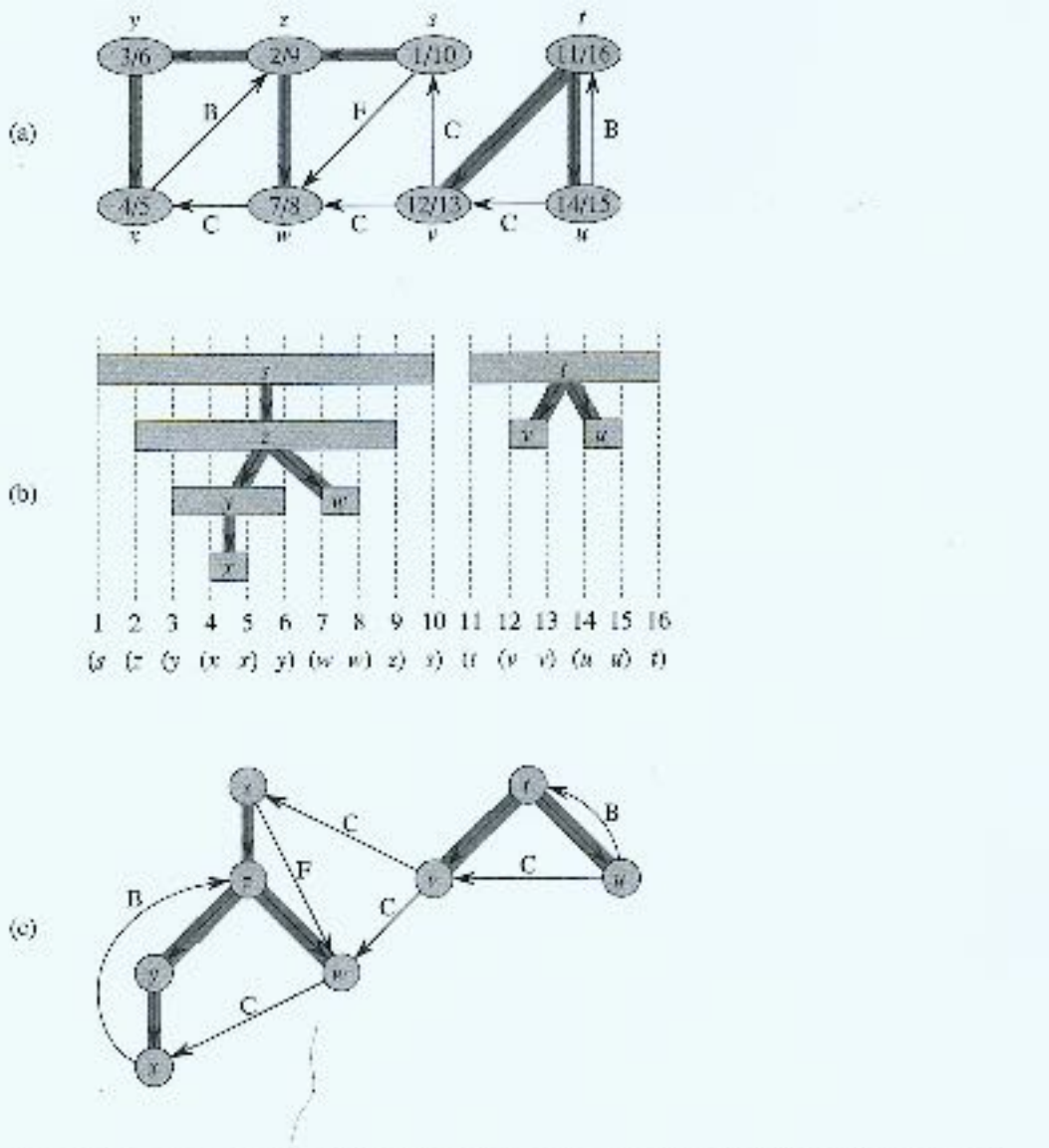


Figure 22.5 Properties of depth-first search. (a) The result of a depth-first search of a directed graph. Vertices are timestamped and edge types are indicated as in Figure 22.4. (b) Intervals for the discovery time and finishing time of each vertex correspond to the parenthesization shown. Each rectangle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

Connectivity for Undirected Graphs

- A (simple) undirected graph is connected if there exists a path between every pair of vertices.
- G'(V',E') is a connected component of the graph G(V,E) if V' induces a maximal 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.

Applications of Graph Traversal

- Checking for connectivity in undirected graphs
  - If we return to statement 7 in DFS(G), then we are starting a new connected component, and therefore G is not connected.
  - The number of time we return to statement 7 in DFS(G) is the number of connected components in G.
  - The vertices and edges visited between returning to statement 7 in DFS(G) constitute the connected components of G.
- Checking for cycles
  - If the condition in statement 8 in DFS-Visit(u) fails, then a cycle is detected.
  - Any edge that passes the condition in statement 8 does not form a cycle and is part of the DFS tree.
  - Warning: The number of times the the condition in statement 8 fails is not equal to the number of cycles in the graph

Connected Components: Undirected

- An undirected graph is connected if every vertex is reachable from every other vertex
- A connected component is a maximal connected subgraph of a graph
- Every graph can be partitioned (i.e., broken down) into a set of connected components such that every vertex and every edge belongs to exactly one connected component

Computing Connected Components

- Undirected Case
  - If graph is disconnected, then DFS/BFS will not visit all vertices and all edges when initiated from a vertex v.
  - It will only visit the component to which v belongs
  - Restart DFS/BFS from an unvisited vertex if traversal is needed
  - Number of restarts =
    - Number of connected components
- Directed case
  - Even if graph is not disconnected, DFS/BFS may not visit all vertices and all edges
  - This is because you can have a case where
    - A is reachable from B, but B is not reachable from A
    - Need to redefine connected components
  - Restart DFS/BFS from an unvisited vertex if traversal is needed

Weak/Strong Connected Components

- A directed graph is strongly connected if every vertex is reachable from every other vertex.
  - Note that this means that for a pair of vertices (A,B), we must have that A is reachable from B and B is reachable from A
- A directed graph is weakly connected if for every pair of vertices (A,B), either A is reachable from B or B is reachable from A.
- We will not study this further for this class …

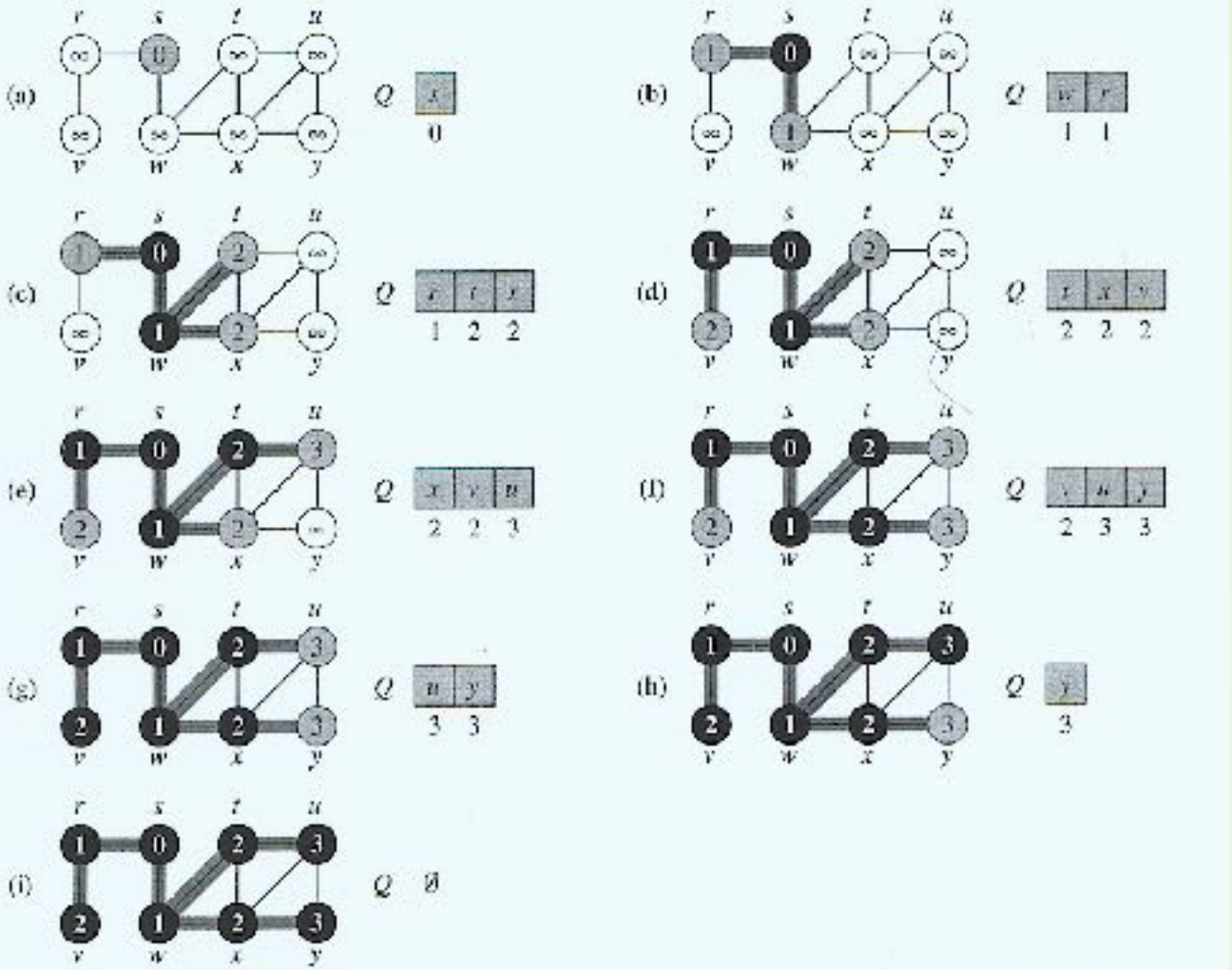


Figure 22.3 The operation of BFS on an undirected graph. Tree edges are shown shaded as they are produced by BFS. Within each vertex u is shown $d[u]$. The queue Q is shown at the beginning of each iteration of the **while** loop of lines 10–18. Vertex distances are shown next to vertices in the queue.

Breadth First Search

```
BFS(G,s)
1.    For each vertex u ∈V[G] – {s} do
2.        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
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
```