CMPN302: Algorithms Design and Analysis

Lecture 07: Graph Algorithms

Ahmed Hamdy

Computer Engineering Department

Cairo University

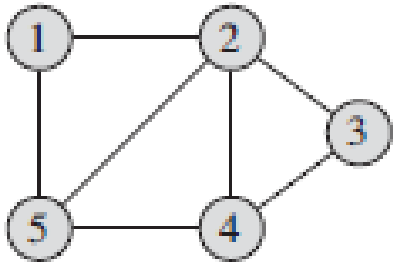
Fall 2017

Graph representations

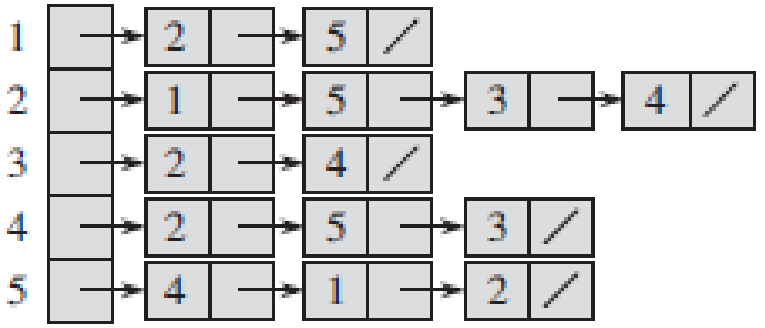
- Graph notation
  - $G(V, E)$ denotes a graph with a set of vertices V and a set of edges E
  - $|V|$ number of vertices
  - $|E|$ number of edges
  - (u, v) denotes edge going from node u to node v

Graph representations

- Undirected graphs
- Adjacency list ($\Theta(V+E)$ space) vs adjacency matrix ($\Theta(V^2)$ space)



(a)



(b)

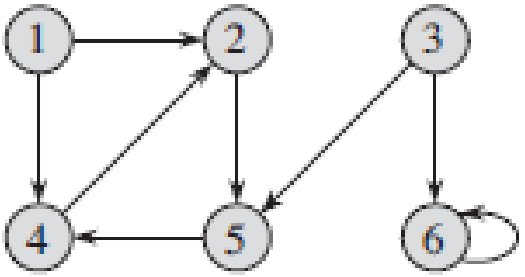
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

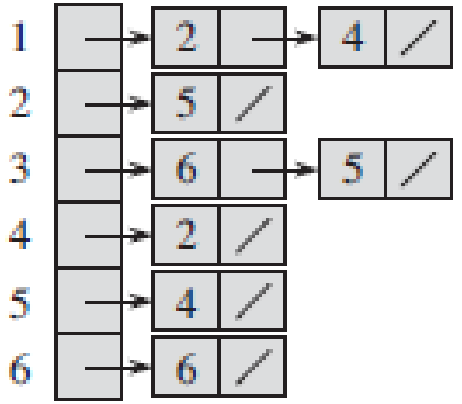
Figure 22.1 Two representations of an undirected graph. (a) An undirected graph G with 5 vertices and 7 edges. (b) An adjacency-list representation of G. (c) The adjacency-matrix representation of G.

Graph representations

- Directed graphs



(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 with 6 vertices and 8 edges. (b) An adjacency-list representation of G. (c) The adjacency-matrix representation of G.

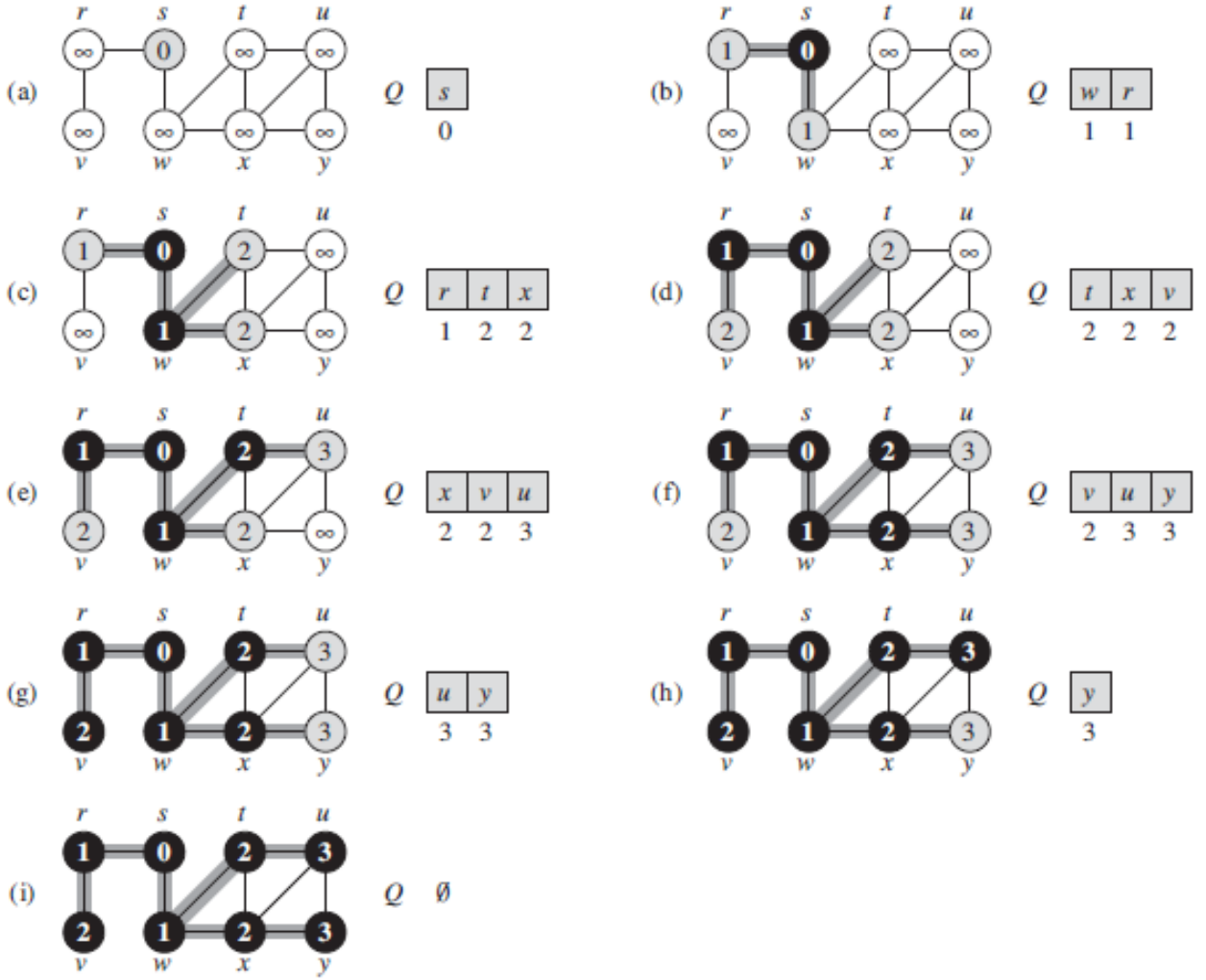
- How to adapt representation for weighted graphs when each edge has a weight?

Breadth First Search (BFS)

- Has to start from a given *source* node s.
- Goals:
  - "Discovers" every vertex that is reachable from s.
  - It computes the distance (smallest number of edges) from s to each reachable vertex.
  - It also produces a "breadth-first tree" with root s that contains all reachable vertices.
  - For any vertex v reachable from s, the simple path in the breadth-first tree from s to corresponds to a "shortest path" from s to v in G, that is, a path containing the smallest number of edges.
  - Works on both directed and undirected graphs.

Breadth First Search (BFS)

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```



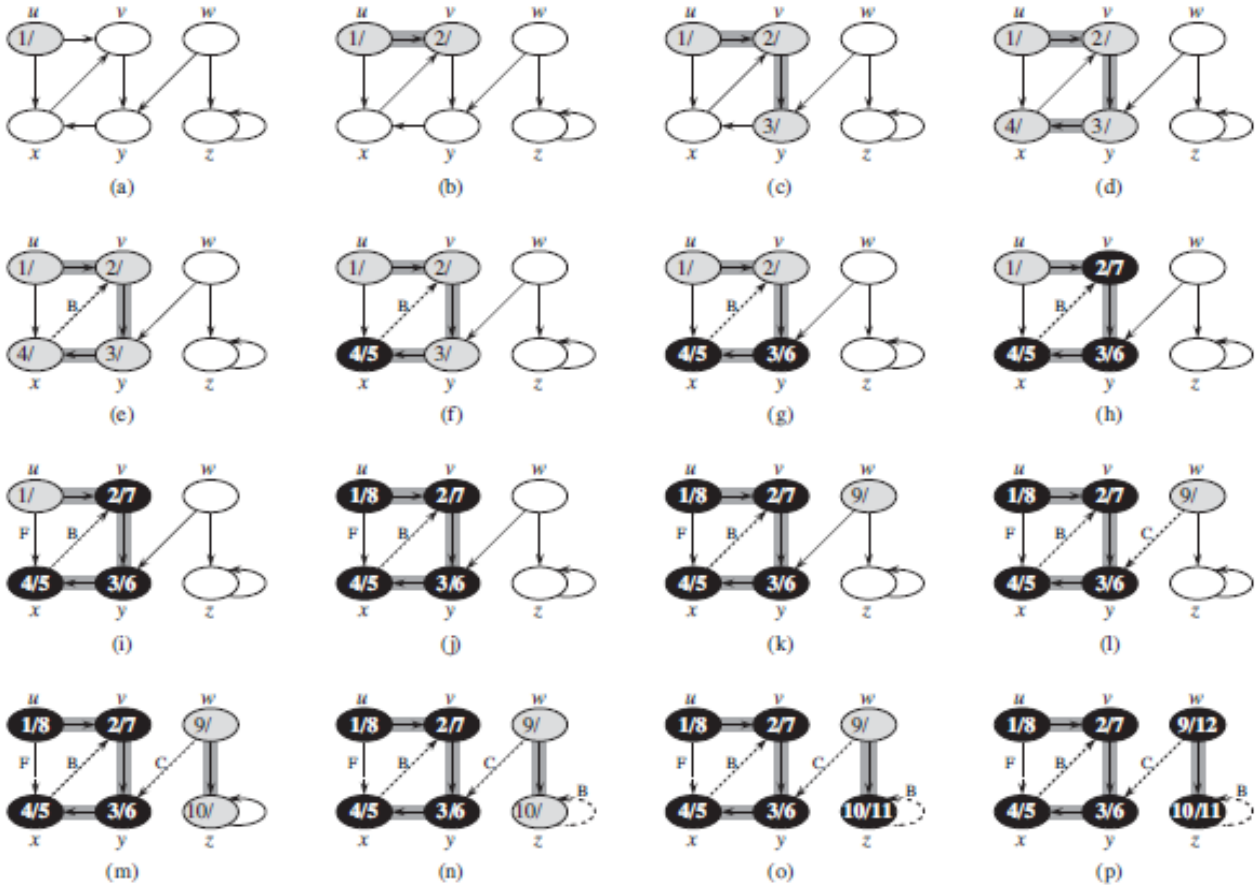
Complexity: $O(V + E)$

Notice the resulting breadth-first tree in black

Depth First Search (DFS)

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
 1  time = time + 1
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK
 9  time = time + 1
10  u.f = time
```



Complexity: $O(V + E)$

Notice the forest of trees

Depth First Search (DFS)

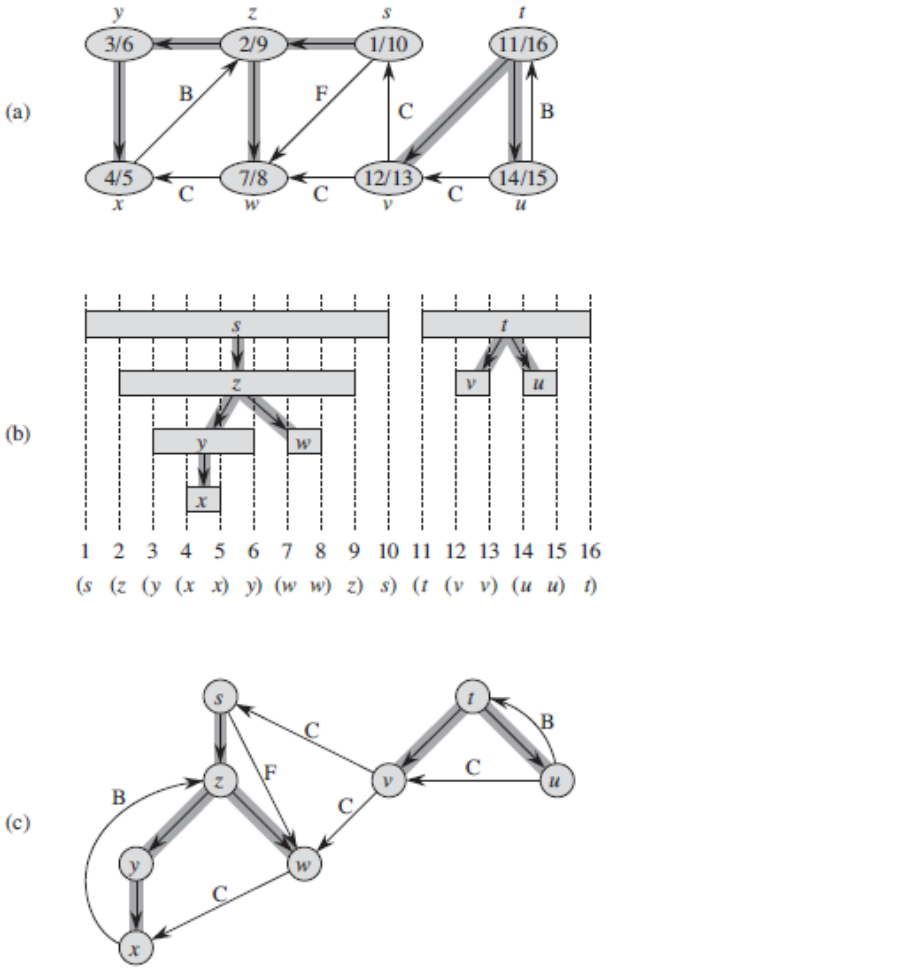


Figure 22.5 Properties of depth-first search. **(a)** The result of a depth-first search of a directed graph. Vertices are timestamped and edge types are indicated as in Figure 22.4. **(b)** Intervals for the discovery time and finishing time of each vertex correspond to the parenthesization shown. Each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

Depth First Search (DFS)

1. ***Tree edges*** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
2. ***Back edges*** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. We consider self-loops, which may occur in directed graphs, to be back edges.
3. ***Forward edges*** are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.
4. ***Cross edges*** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.

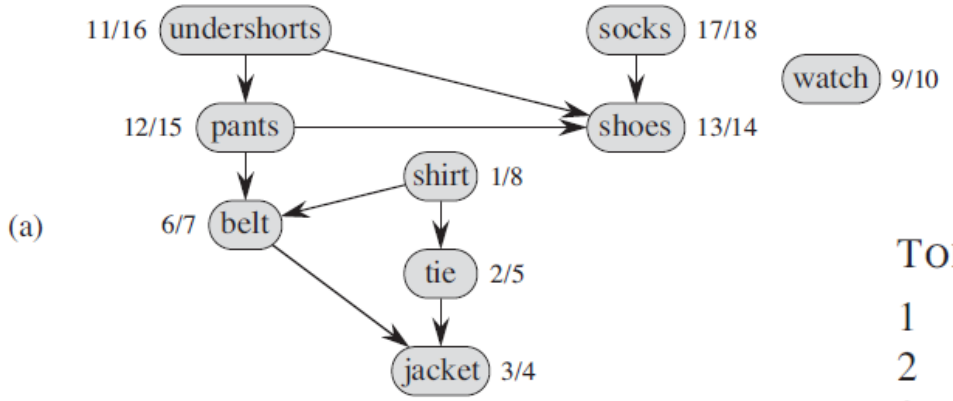
Depth First Search (DFS)

- A graph is acyclic if DFS results in *no* back edges
- Meaning of edge color in DFS:
  1. WHITE indicates a tree edge,
  2. GRAY indicates a back edge, and
  3. BLACK indicates a forward or cross edge.

Topological sort

- Performed on a directed acyclic graph (DAG).
- It is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering.
- If the graph contains a cycle, then no linear ordering is possible.
- Can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right.

Topological sort



TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

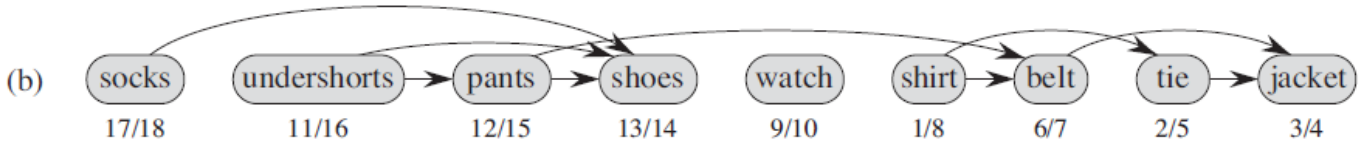
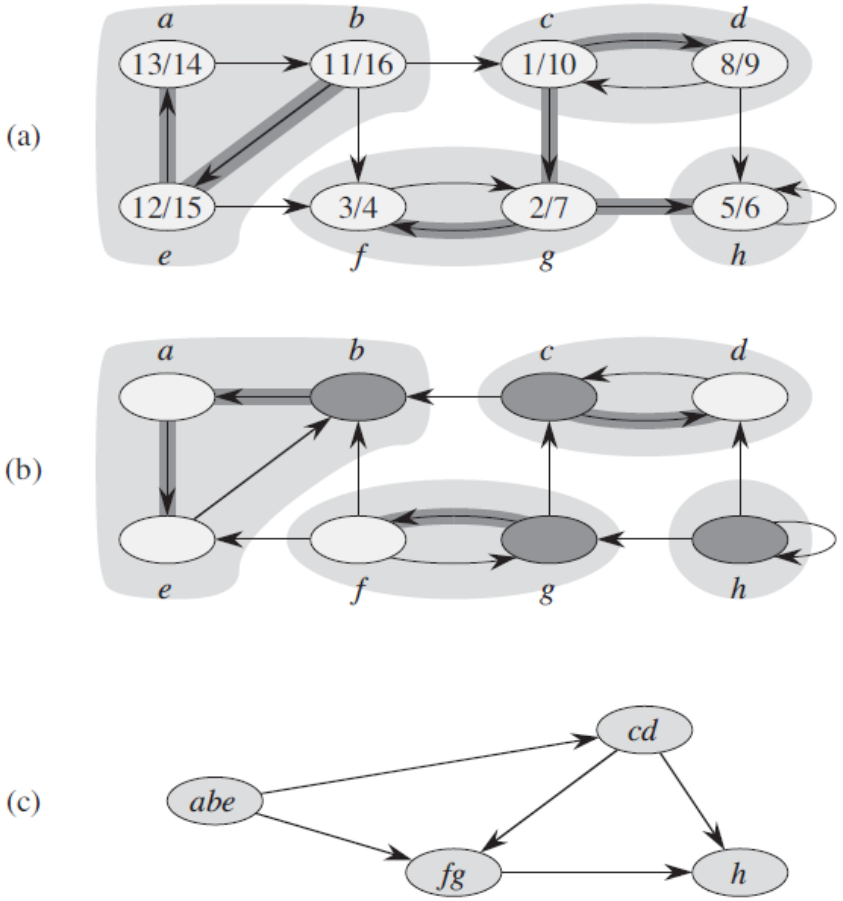


Figure 22.7 **(a)** Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finishing times from a depth-first search are shown next to each vertex. **(b)** The same graph shown topologically sorted, with its vertices arranged from left to right in order of decreasing finishing time. All directed edges go from left to right.

Strongly connected components

- It is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v in C, we have both $u \to v$ and $\mathrm{v} \to u$; that is, vertices u and v are reachable from each other.

Strongly connected components



STRONGLY-CONNECTED-COMPONENTS (G)

1 call DFS(G) to compute finishing times $u.f$ for each vertex u
2 compute G^{T}
3 call DFS(G^{T}), but in the main loop of DFS, consider the vertices in order of decreasing $u.f$ (as computed in line 1)
4 output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component

Figure 22.9 **(a)** A directed graph G. Each shaded region is a strongly connected component of G. Each vertex is labeled with its discovery and finishing times in a depth-first search, and tree edges are shaded. **(b)** The graph G^{T}, the transpose of G, with the depth-first forest computed in line 3 of STRONGLY-CONNECTED-COMPONENTS shown and tree edges shaded. Each strongly connected component corresponds to one depth-first tree. Vertices b, c, g, and h, which are heavily shaded, are the roots of the depth-first trees produced by the depth-first search of G^{T}. **(c)** The acyclic component graph G^{SCC} obtained by contracting all edges within each strongly connected component of G so that only a single vertex remains in each component.