Chapter 14 excerpts

Graphs (breadth-first-search)

Terminology

- G = {V, E}
- A graph G consists of two sets
  - A set V of vertices, or nodes
  - A set E of edges
- A subgraph
  - Consists of a subset of a graph's vertices and a subset of its edges
- Adjacent vertices
  - Two vertices that are joined by an edge

Terminology

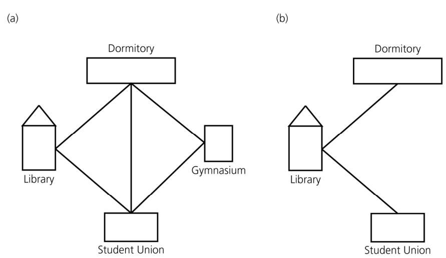


Figure 14-2

a) A campus map as a graph; b) a subgraph

Terminology

- A path between two vertices
  - A sequence of edges that begins at one vertex and ends at another vertex
  - May pass through the same vertex more than once
- A simple path
  - A path that passes through a vertex only once
- A cycle
  - A path that begins and ends at the same vertex
- A simple cycle
  - A cycle that does not pass through a vertex more than once

Terminology

- A connected graph
  - A graph that has a path between each pair of distinct vertices
- A disconnected graph
  - A graph that has at least one pair of vertices without a path between them
- A complete graph
  - A graph that has an edge between each pair of distinct vertices

Terminology

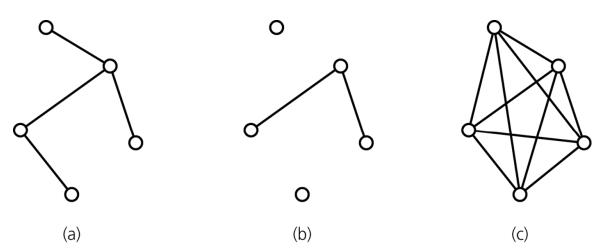


Figure 14-3

Graphs that are a) connected; b) disconnected; and c) complete

Graphs As ADTs

- Graphs can be used as abstract data types
- Two options for defining graphs
  - Vertices contain values
  - Vertices do not contain values
- Operations of the ADT graph
  - Create an empty graph
  - Determine whether a graph is empty
  - Determine the number of vertices in a graph
  - Determine the number of edges in a graph

Graphs As ADTs

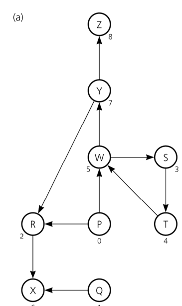
- Operations of the ADT graph (Continued)
  - Determine whether an edge exists between two given vertices; for weighted graphs, return weight value
  - Insert a vertex in a graph whose vertices have distinct search keys that differ from the new vertex's search key
  - Insert an edge between two given vertices in a graph
  - Delete a particular vertex from a graph and any edges between the vertex and other vertices
  - Delete the edge between two given vertices in a graph
  - Retrieve from a graph the vertex that contains a given search key

Implementing Graphs

- Most common implementations of a graph
  - Adjacency matrix
  - Adjacency list
- Adjacency matrix
  - Adjacency matrix for a graph with n vertices numbered 0, 1, …, n – 1
    - An n by n array matrix such that matrix[i][j] is
      - 1 (or true) if there is an edge from vertex i to vertex j
      - 0 (or false) if there is no edge from vertex i to vertex j

Implementing Graphs

(a)



(b)

		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0

Figure 14-6

a) A directed graph and b) its adjacency matrix

Implementing Graphs

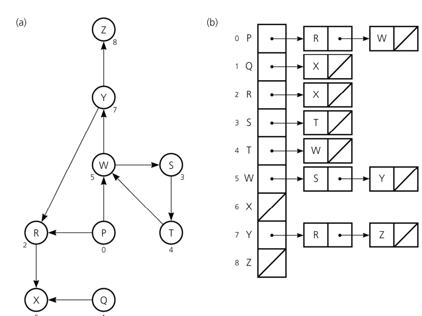
- Adjacency list
  - An adjacency list for a graph with n vertices numbered 0, 1, …, n – 1
    - Consists of n linked lists
    - The i[th] linked list has a node for vertex j if and only if the graph contains an edge from vertex i to vertex j
      - This node can contain either
        - » Vertex j's value, if any
        - » An indication of vertex j's identity

Implementing Graphs

Figure 14-8

a) A directed graph and

b) its adjacency list



Graph Traversals

- A graph-traversal algorithm
  - Visits all the vertices that it can reach
  - Visits all vertices of the graph if and only if the graph is connected
    - A connected component
      - The subset of vertices visited during a traversal that begins at a given vertex
  - Can loop indefinitely if a graph contains a loop
    - To prevent this, the algorithm must
      - Mark each vertex during a visit, and
      - Never visit a vertex more than once

Graph Traversals

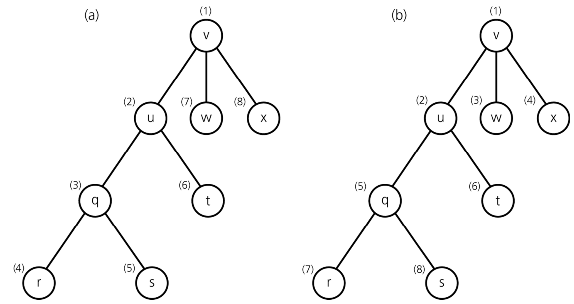


Figure 14-10

Visitation order for a) a depth-first search; b) a breadth-first search

Depth-First Search

- Depth-first search (DFS) traversal
  - Proceeds along a path from *v* as deeply into the graph as possible before backing up
  - Does not completely specify the order in which it should visit the vertices adjacent to *v*
  - A last visited, first explored strategy

Breadth-First Search

- Breadth-first search (BFS) traversal
  - Visits every vertex adjacent to a vertex v that it can before visiting any other vertex
  - A first visited, first explored strategy
  - An iterative form uses a queue
  - A recursive form is possible, but not simple