Announcements

• Assignment 6 questions?
• Quiz 10 (hash tables) is available in SPARK.

• Plan for the remaining lectures:
  – Apr 27: Final review 1
  – Apr 29: Final review 2
  – May 4: more on graphs
CS 187: Programming with Data Structures (Spring 2010)

Lecture 24: Graphs

Rui Wang
Graphs

• Graphs are one of the most versatile structures.

• Some real-world examples that inspire the graph data structure:
  – Social networks
  – Peer-to-peer (P2P) network
  – Roadmaps
Graphs

• A graph consists of many **vertices (nodes)** and **edges** between vertices.
  – An example.
  – Edges define how the vertices are connected.

• A graph reflects topological relationship, thus the edge lengths do not reflect physical distances.

• How is a graph different from a tree?
Terminology

• **Adjacency**
  – Two vertices are adjacent if they are connected by an edge.

• **Neighbors**
  – The vertices adjacent to a given vertex are called its neighbors.

• **Paths**
  – A sequence of edges that connect two vertices.
  – There may be more than one path that connect two given vertices!
Terminology

• **Connected Graphs**
  – A graph is said to be connected if there exists at least one path between every pair of vertices.

• **Directed Graphs**
  – Each edge has a direction (one-way).
  – Example: course prerequisite graph.

• **Weighted Graphs**
  – Each edge has an associated weight (e.g. distance between two cities, travel cost etc.)
Representing a Graph

• How do we store graphs in computer?

• Vertices
  – We can give every vertex an index, from 0 to N-1.
  – N is the number of vertices.
  – Store all vertices in an array.

• Edges
  – In a tree, edges are represented by the children links. Here you can have an edge between any two vertices. So what do we do?
Representing a Graph

• The Adjacency Matrix
  – A two-dimensional array in which elements represent whether an edge exists between two vertices. It’s an NxN matrix.
  – Example:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Representing a Graph

• The Adjacency Matrix
  – What about the diagonal?
  – It’s called the identity diagonal.
  – Let’s look at another example.

  – For an undirected graph, the adjacency matrix is symmetric.
Representing a Graph

• The Adjacency List
  – Another way to represent a graph.
  – For each vertex, store its neighbors in a linked list. Why use a linked list?

<table>
<thead>
<tr>
<th>Vertex</th>
<th>List Containing Adjacent Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B→C→D</td>
</tr>
<tr>
<td>B</td>
<td>A→D</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>A→B</td>
</tr>
</tbody>
</table>
Representing a Graph

• The Adjacency List
  – In terms of data structure, this is essentially an array of linked lists.
Create a Graph

- **Adding vertices:**

```java
class Vertex {
    public char label;
    ...
    ...                           // other data
    Vertex(char l) { label = l; }
}
```

```java
ArrayList<Vertex> vArray = new ArrayList;
vArray.add(new Vertex('A'));
vArray.add(new Vertex('B'));
```
Create a Graph

• **Adding edges:**

```java
adjMatrix = new int[MAX_VERTS][MAX_VERTS];

// fill adjMatrix with all 0s

adjMatrix[0][1] = 1;
adjMatrix[1][0] = 1;
```
Create a Graph

• In summary, you store a graph using two structures: a vertex array, and an adjacency matrix.
More Topics on Graphs

- Searches
- Minimum spanning tree
- Topological sorting
- Shortest paths