Graph Operations And Representation

Sample Graph Problems
- Path problems.
- Connectedness problems.
- Spanning tree problems.

Path Finding
Path between 1 and 8.

Path length is 20.

Another Path Between 1 and 8

Path length is 28.

Example Of No Path

No path between 2 and 9.

Connected Graph
- Undirected graph.
- There is a path between every pair of vertices.
Example Of Not Connected

Connected Graph Example

Connected Components

Connected Component

• A maximal subgraph that is connected.
  • Cannot add vertices and edges from original graph and retain connectedness.
  • A connected graph has exactly 1 component.

Not A Component

Communication Network

Each edge is a link that can be constructed (i.e., a feasible link).
Communication Network Problems

- Is the network connected?
  - Can we communicate between every pair of cities?
- Find the components.
- Want to construct smallest number of feasible links so that resulting network is connected.

Cycles And Connectedness

- Removal of an edge that is on a cycle does not affect connectedness.

Cycles And Connectedness

Connected subgraph with all vertices and minimum number of edges has no cycles.

Tree

- Connected graph that has no cycles.
- n vertex connected graph with n-1 edges.

Spanning Tree

- Subgraph that includes all vertices of the original graph.
- Subgraph is a tree.
  - If original graph has n vertices, the spanning tree has n vertices and n-1 edges.

Minimum Cost Spanning Tree

- Tree cost is sum of edge weights/costs.
A Spanning Tree

Spanning tree cost = 51.

Minimum Cost Spanning Tree

Spanning tree cost = 41.

A Wireless Broadcast Tree

Source = 1, weights = needed power. Cost = 4 + 8 + 5 + 6 + 7 + 8 + 3 = 41.

Graph Representation

- Adjacency Matrix
- Adjacency Lists
  - Linked Adjacency Lists
  - Array Adjacency Lists

Adjacency Matrix

- 0/1 n x n matrix, where n = # of vertices
- A(i,j) = 1 iff (i,j) is an edge

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Adjacency Matrix Properties

- Diagonal entries are zero.
- Adjacency matrix of an undirected graph is symmetric.
- A(i,j) = A(j,i) for all i and j.
### Adjacency Matrix (Digraph)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Diagonal entries are zero.
- Adjacency matrix of a digraph need not be symmetric.

### Adjacency Matrix

- $n^2$ bits of space
- For an undirected graph, may store only lower or upper triangle (exclude diagonal).
  - $(n-1)n/2$ bits
- $O(n)$ time to find vertex degree and/or vertices adjacent to a given vertex.

### Adjacency Lists

- Adjacency list for vertex $i$ is a linear list of vertices adjacent from vertex $i$.
- An array of $n$ adjacency lists.
  
  - $aList[1] = (2,4)$
  - $aList[2] = (1,5)$
  - $aList[3] = (5)$
  - $aList[4] = (5,1)$
  - $aList[5] = (2,4,3)$

### Linked Adjacency Lists

- Each adjacency list is a chain.

### Array Adjacency Lists

- Each adjacency list is an array list.

### Weighted Graphs

- Cost adjacency matrix.
  - $C(i,j) =$ cost of edge $(i,j)$
- Adjacency lists $\Rightarrow$ each list element is a pair (adjacent vertex, edge weight)

Array Length = $n$

- # of list elements = $2e$ (undirected graph)
- # of list elements = $e$ (digraph)

Array Length = $n$

- # of chain nodes = $2e$ (undirected graph)
- # of chain nodes = $e$ (digraph)
Number Of Java Classes Needed

- Graph representations
  - Adjacency Matrix
  - Adjacency Lists
    ➢ Linked Adjacency Lists
    ➢ Array Adjacency Lists
  - 3 representations
- Graph types
  - Directed and undirected.
  - Weighted and unweighted.
  - $2 \times 2 = 4$ graph types
  - $3 \times 4 = 12$ Java classes

Abstract Class Graph

```java
package dataStructures;
import java.util.*;
public abstract class Graph
{
    // ADT methods come here
    public abstract Iterator iterator(int i);
    // create an iterator for vertex i
    public abstract Iterator iterator(int i);
    // implementation independent methods come here
}
```

Abstract Methods Of Graph

```java
// ADT methods
public abstract int vertices();
public abstract int edges();
public abstract boolean existsEdge(int i, int j);
public abstract void putEdge(Object theEdge);
public abstract void removeEdge(int i, int j);
public abstract int degree(int i);
public abstract int inDegree(int i);
public abstract int outDegree(int i);
```