**Graphs 1**

**Adjacency Matrix: Warshall's Algorithm**

Here is a little more interesting directed graph:

**6** Princeton

**3** Phoenix

**1** Pensacola

**4** Pierre

**0** Pendleton

**7** Pueblo

**5** Pittsburgh

**2** Peoria

Here are the names, the rows and columns, and the adjacency matrix for the graph above:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0.** | **1.** | **2.** | **3.** | **4.** | **5.** | **6.** | **7.** |
| **0.** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **1.** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **2.** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| **3.** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| **4.** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5.** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **6.** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **7.** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

|  |
| --- |
| 0-Pendleton 1-Pensacola 2-Peoria 3-Phoenix 4-Pierre 5-Pittsburgh 6-Princeton 7-Pueblo |

If you recall from Graphs0, the numbering of the cities happened when you read the file of cities. The cities in the file happened to be in alphabetical order.

One of the fundamental questions about graphs is the question of *reachability*. For example:

1. If you start from Phoenix, is Peoria reachable? \_\_\_
2. Is it possible to go from Peoria to Phoenix? \_\_\_\_
3. If you start from Peoria, is Princeton reachable? \_\_\_
4. Is it possible to go from Pensacola to Pendleton? \_\_\_
5. Is it possible to go from Pendleton to Pensacola? \_\_\_

Warshall's Algorithm will answer these questions by turning the adjacency matrix into a reachability matrix. Amazingly, Warshall's Algorithm solves the *all-pairs reachability* problem.

You should already have upgraded your AdjMat class to handle names.

public void readNames(String fileName);  
 public Map<String, Integer> getNamesAndNumbers();  
 public String toStringNamesAndNumbers();   
 public boolean isNeighbor(String from, String to);

Now you need to implement the Warshall interface.

interface Warshall  
{

public int countReachables();  
public boolean isReachable(String from, String to);   
public List<String> getReachables(String from);  
public String toStringReachability(); //displays the reachability matrix   
public void allPairsReachability();

}

It's good programming practice separate the adjacency matrix from the reachability matrix. You will need to add a new field. The private fields for our graph are shown below.

public class AdjMat implements AdjacencyMatrix, Warshall, Floyd  
{  
 private int[][] grid = null; //adjacency matrix representation  
 private int[][] reachability = null; //reachability matrix for   
 //Warshall, cost matrix for Floyd  
 private Map<String, Integer> namesAndNumbers = null;// maps name   
 // to number  
 private ArrayList<String> nameList = null; //reverses the map,   
 index --> name

The two-arg constructor needs to instantiate all the fields.

public AdjMat(String fileMatrix, String fileNames)   
 {  
 grid = readGrid( fileMatrix );  
 reachability = readGrid( fileMatrix );  
 namesAndNumbers = new TreeMap<String, Integer>();  
 nameList = new ArrayList<String>(); //Warshall Extension  
 readNames( fileNames );  
 }

**Warshall's Algorithm**

Warshall's Algorithm starts with an adjacency matrix, examines the edges between three vertices, and adds edge information to the matrix. The completed matrix is called a *reachability* matrix. Here is Warshall's algorithm: for any given vertex v, if the paths [r, v] and [v, c] exist, then a path to [r, c] exists. Repeat for every vertex v.

Try it out for v=7, r=0, and c=4.

Now match those numbers to their cities.

What is the Big-O for Warshall’s Algorithm? \_\_**\_**\_\_ The disadvantage of Warshall’s Algorithm is that it may take too long to run in practice.

**Assignment**

In your AdjMat implement the Warshall interface. Notice that AdjMat uses two grids, one for the adjacency data and the other for the reachability results. The first step in our allPairsReachability is to copy the adjacency data into the reachability matrix. The second step is to perform Warshall's algorithm to produce the reachability matrix. Implement toStringReachability, isReachable, getReachables. Turn in AdjMat.

The WarshallDriver.java driver program, which is given to you, prompts the user for the necessary files (citymatrix.txt and cities.txt). Here is the output:

**Sample Run**

|  |  |
| --- | --- |
| Warshall's Algorithm!  Enter file with matrix: citymatrix Enter file of cities: cities  Adjacency Matrix 0 0 0 0 0 0 0 1  0 0 0 1 0 0 0 0  0 0 0 0 0 1 0 1  0 0 0 0 0 1 0 1  1 0 0 0 0 0 0 0  0 1 0 1 0 0 0 0  0 0 0 0 0 1 1 0  1 0 0 0 1 0 0 0   0-Pendleton 1-Pensacola 2-Peoria 3-Phoenix 4-Pierre 5-Pittsburgh 6-Princeton 7-Pueblo  Reachability Matrix 1 0 0 0 1 0 0 1  1 1 0 1 1 1 0 1  1 1 0 1 1 1 0 1  1 1 0 1 1 1 0 1  1 0 0 0 1 0 0 1  1 1 0 1 1 1 0 1  1 1 0 1 1 1 1 1  1 0 0 0 1 0 0 1 | Is it reachable? Enter name of start city (-1 to exit): Peoria  Enter name of end city: Phoenix true  Is it reachable? Enter start city (-1 to exit): -1 List of every city's reachable cities:  Pendleton--> [Pendleton, Pierre, Pueblo] Pensacola--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo] Peoria--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo] Phoenix--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo] Pierre--> [Pendleton, Pierre, Pueblo] Pittsburgh--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo] Princeton--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Princeton, Pueblo] Pueblo--> [Pendleton, Pierre, Pueblo]  List the reachable cities from: Pittsburgh [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]  List the reachable cities from: -1 |

**Sample Run**

Warshall's Algorithm!  
 Enter file with the matrix: citymatrix  
 Enter file of cities: cities  
 Adjacency Matrix  
 0 0 0 0 0 0 0 1  
 0 0 0 1 0 0 0 0  
 0 0 0 0 0 1 0 1  
 0 0 0 0 0 1 0 1  
 1 0 0 0 0 0 0 0  
 0 1 0 1 0 0 0 0  
 0 0 0 0 0 1 1 0  
 1 0 0 0 1 0 0 0  
   
 Number of edges: 13

Map: {Pendleton=0, Pensacola=1, Peoria=2, Phoenix=3, Pierre=4, Pittsburgh=5, Princeton=6, Pueblo=7}  
   
 0-Pendleton  
 1-Pensacola  
 2-Peoria  
 3-Phoenix  
 4-Pierre  
 5-Pittsburgh  
 6-Princeton  
 7-Pueblo  
   
 Reachability Matrix  
 1 0 0 0 1 0 0 1  
 1 1 0 1 1 1 0 1  
 1 1 0 1 1 1 0 1  
 1 1 0 1 1 1 0 1  
 1 0 0 0 1 0 0 1  
 1 1 0 1 1 1 0 1  
 1 1 0 1 1 1 1 1  
 1 0 0 0 1 0 0 1  
   
 Number of reachables: 40  
   
 Is it reachable? Enter name of start city (-1 to exit): Pittsburgh  
 Enter name of end city: Pendleton  
 true  
   
 Is it reachable? Enter name of start city (-1 to exit): Pendleton  
 Enter name of end city: Pittsburgh  
 false  
   
 Is it reachable? Enter name of start city (-1 to exit): -1  
 List of every city's reachable cities:   
 Pendleton--> [Pendleton, Pierre, Pueblo]  
 Pensacola--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]  
 Peoria--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]  
 Phoenix--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]  
 Pierre--> [Pendleton, Pierre, Pueblo]  
 Pittsburgh--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]  
 Princeton--> [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Princeton, Pueblo]  
 Pueblo--> [Pendleton, Pierre, Pueblo]  
   
 List the reachable cities from: Pittsburgh  
 [Pendleton, Pensacola, Phoenix, Pierre, Pittsburgh, Pueblo]