**Graphs 0: AdjMat**

If you want to answer questions about reachability (Warshall) or lowest cost (Floyd), and the number of nodes is "small," then the graph can conveniently be represented by an *adjacency matrix*. For reachability, we use a 0 to indicate no edge and a 1 to indicate an edge. The picture on the left and the adjacency matrix (indicating the neighbors of each vertex) on the right contain the same information, but the computer can process the adjacency matrix.

**target**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | 0 | 0 | 1 | 0 |
| **1** | 1 | 0 | 0 | 0 |
| **2** | 0 | 0 | 1 | 1 |
| **3** | 1 | 0 | 1 | 0 |

**source**

An alternative implementation has the adjacency matrix storing booleans instead of 0’s and 1’s. In our case, using 0 and 1 generalizes nicely when we reach *weighted* graphs (Floyd).

Looking only at the adjacency matrix, answer the same questions as before:

1. If you start at 0, is 1 reachable? Y/N
2. If you start at 0, is 2 reachable? Y/N
3. If you start at 2, is 0 reachable? Y/N
4. If you start at 2, is 2 reachable? Y/N
5. How many vertices are in the graph above?
6. How many edges are in the graph above?
7. List the neighbors of 0.
8. List the neighbors of 3.
9. Is there a path from source 2 to target 0?

Obviously, we need a class with methods that manipulate a 2-D array of ints. We will call it AdjMat. We will add functionality to AdjMat as we go. Here is the header:

public class AdjMat implements AdjacencyMatrix//, WithNames//,   
 Warshall//, Floyd  
{  
 private int[][] grid = null;

1. What constructor do we need? Does it need arguments? Since graphs require lots of data, let's pass the name of the text file to the constructor. Write the header here:

|  |
| --- |
| 4 |
| 0 0 1 0 |
| 1 0 0 0 |
| 0 0 1 1 |
| 1 0 1 0 |

1. A typical text file is structured as shown. The constructor will instantiate the grid of the correct size and fill the grid with the correct data. We can use a helper method,

public int[][] readGrid(String filename)

Use a try-catch in readGrid. Here is an example of a try-catch:

Scanner file = null;  
 try {  
 file = new Scanner(new File(fileName));  
 }  
 catch(FileNotFoundException e)  
 {  
 e.printStackTrace(System.out);  
 }

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |

1. Write the header of the accessor method for AdjMat's private field:

1. For example, vertex 1 is a neighbor of vertex 0, but vertex 0 is not a neighbor of vertex 1. Write the header of the method isNeighbor
2. Write the header of the method that counts the edges (the number of 1's) in the grid.

|  |
| --- |
| 0: [2] |
| 1: [0] |
| 2: [2, 3] |
| 3: [0, 2] |

1. Write the header of the method that returns a list of a vertex's neighbors.

1. Write the header of the method that returns a string listing each vertex with its neighbors.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |

1. We need to display the matrix. Let's return a toString of the grid, showing only the 0's and 1's in the grid, without labeling rows and columns.

**Stop and test everything**

Test your AdjMat using the first part of AdjMat\_0\_Driver, which has been written for you. The text file is named matrix4x4 . Your teacher may direct you to submit this version of AdjMat. Or your teacher may direct you to continue with **interface WithNames**, described below.

The methods so far work directly on the rows and columns of the matrix, but we will want to match each row and column, with, say, the names of cities in Warshall and Floyd. We are going to need Strings and two more private fields. The two fields allow us to go both ways, from name to number and also from index to name:

private Map<String, Integer> namesAndNumbers = null;

private ArrayList<String> nameList = null; //Extension

|  |
| --- |
| 4  Albert Claire Larry Marylin |

1. Since you now have to read two files, it is convenient to write a two-arg constructor which will (of course) instantiate and populate these two new private fields. The constructor will call two helper methods, readGrid and readNames. The file of names "names4" is in this format, which happens to be in alphabetical order, and the first name gets to be number 0, etc.
2. You will need an accessor method for the namesAndNumbers map. If you instantiated a TreeMap, the names will be stored in alphabetical order. Write its header:

|  |
| --- |
| 0-Albert  1-Claire  2-Larry 3-Marylin |

1. Return a String with the names and their numbers, in toStringNamesAndNumbers, in this format:
2. Overload the isNeighbor method so that the user can deal with names, not numbers. Write its header:

**Assignment**: complete the AdjMat class as specified by the AdjacencyMatrix interface. Test your resource class using the second part of the AdjMat\_0\_Driver, which has been written for you. The two text files you need are matrix4x4 and names4. You will submit AdjMat.

**Sample Run**

Adjacency Matrix!

Enter file with the matrix: matrix4x4  
   
 0 0 1 0  
 1 0 0 0  
 0 0 1 1  
 1 0 1 0  
   
 Number of Warshall edges: 6  
   
 The neighbors of each vertex:   
 0: [2]  
 1: [0]  
 2: [2, 3]  
 3: [0, 2]  
   
 Enter by number <source> <target>. Is target a neighbor of source?  
 3 2  
 true  
 Enter by number <source> <target>. Is target a neighbor of source?  
 -1  
 Enter <source>. Its neighbors are:   
 3  
 [0, 2]  
 Enter <source>. Its neighbors are:   
 -1  
 Adjacency Matrix!

Enter file with the matrix: matrix4x4  
 Enter file with the names: names4  
   
 0 0 1 0  
 1 0 0 0  
 0 0 1 1  
 1 0 1 0  
   
 Number of Warshall edges: 6

Map: {Albert=0, Claire=1, Larry=2, Marylin=3}  
 Names and numbers:   
 0-Albert  
 1-Claire  
 2-Larry  
 3-Marylin  
   
  
 Enter by name <source> <target>. Is target a neighbor of source?  
 Albert Claire  
 false  
 Enter by name <source> <target>. Is target a neighbor of source?  
 Claire Albert  
 true  
 Enter by name <source> <target>. Is target a neighbor of source?  
 -1