CSC 340 GRAPH SEARCH ALGORITHMS

PROFESSOR GODFREY MUGANDA

1. The Assignment

The goal of this project is to write a program that finds different types of paths through a directed graph. Your program will allow the user to specify a file containing adjacency list data for a graph, read the file and construct an internal representation of the adjacency list, and then allow the user to find paths through the graph.

2. The Syntax for the Adjacency list Data

The format of data in the file that holds the graph data is as follows. The file begins with number of vertices N, followed by a list of N space-separated strings that are the names of the vertices. This order will determine the integer that is used to represent each vertex.

This will be followed by N sets of data, where each data set represents a vertex X and a list of neighbours directly reachable from X. Each data set consist of a vertex X followed by a space-separated listing of neighbours reachable from X.

The line structure is not significant. Line structure of the file may be chosen to enhance readability, but the parsing of the graph data should not on it.

Here is an example input file

```
9
a b c d e f g h i
a 2 f b
f 2 g e
e 1 f
d 3 e h c
c 1 h
b 3 a i c
i 1 a
h 2 i g
g 2 e i
```

Part of the assignment will involve reading in the graph data, constructing the internal representation of the adjacency list, and printing the adjacency list. The adjacency list from the above data should print like this:

```
a : [f, b]
b : [a, i, c]
c : [h]
d : [e, h, c]
e : [f]
```

```
f : [g, e]
g : [e, i]
h : [i, g]
i : [a]
```

3. Project Requirements

Use a class `Graph` to represent the adjacency list of the graph. The graph should have the member shown in the following outline:

```
public class Graph
{
    public int [][] adj; // Adjacency list
    private final List<String> vertNames; // Ordered list of vertex names
    private final Map<String, Integer> nameToNumberMap;
    private final StringBuilder stringForm;  // used for text output of the graph

    // Reads the graph data form the file
    // and builds the graph. Sets up the adj, vertNames,
    // nameToNumberMap, and stringForm variables.
    public Graph(File file) throws FileNotFoundException
    {

    }

    // Number of vertices in the graph
    public int size()
    {
        return vertNames.size();
    }

    @Override
    public String toString()
    {
        return stringForm.toString();
    }

    // Get the name corresponding to a vertex
    public final String getName(int v)
    {
        return vertNames.get(v);
    }

    public int getVNumber(String name)
    {
        return nameToNumberMap.get(name);
    }
}
```

The constructor for this class is passed a `File` object containing the adjacency list data. Note that the adjacency list data in the file uses strings for the names of the vertices, but the internal representation of the adjacency list used integers for the

names of the vertices. Output displayed to the user must be translated back into string names.

The project needs to use a file chooser dialog to select the file with the data.

Use a main method that looks like this

```
public static void main(String[] args) throws FileNotFoundException
{
    JFileChooser chooser = new JFileChooser();
    int result = chooser.showOpenDialog(null);
    if (result != JFileChooser.APPROVE_OPTION)
    {
       System.out.println("No file selected. Program will terminate.");
    }
    File file = chooser.getSelectedFile();

    Graph g = new Graph(file);
    System.out.println("The adjacency list for the graph is");
    System.out.print(g);
    System.out.println();
    findBFSPaths(g);
    System.out.println();
    findDFSPaths(g);
}
```

Here is a sample run:

```
The adjacency list for the graph is
a : [f, b]
b : [a, i, c]
c : [h]
d : [e, h, c]
e : [f]
f : [g, e]
g : [e, i]
h : [i, g]
i : [a]

How many breadth first paths do you want to find? 2
Enter source and destination vertices: d b
The path is [d,  h,  i, a, b].
Enter source and destination vertices
a f
Queue is [0]
The paths is [a, f].

How many depth first paths do you want to find? 2
Enter source and destination vertices: d b
The path is [d, e, f, g, i, a, b].
Enter source and destination vertices: a f
The path is [a, f].
```