CSC 231 HOMEWORK 5

PROFESSOR GODFREY C. MUGANDA

1. Project Description

Breadth-First search can be used to solve the problem of finding the shortest paths from an initial vertex to every other path. By shortest, we mean the fewest number of edges in the path.

Building on the graph representation project, write a program that finds the shortest paths in digraph from a given start vertex to every other vertex.

Organize your project as follows;

1. Have a method

```
static void BFS(GraphRepresentation g, int v, int[] pred)
```

that takes a graph `g` as parameter, a vertex `v`, and an array `pred` that has an entry for each vertex in the graph. This method uses breadth first search to find the shortest paths from `v`.

The array `pred` is used to record the paths. For each vertex `w`, the value `pred[w]` is the predecessor of `w` along the path from the start vertex `v` to `w`. If `w` has no predecessor, then the corresponding entry `pred[w]` is set to -1.

2. Write a method

```
static void getPath(GraphRepresentationA g, int v, int [] pred,
                    LinkedList<Integer> path)
```

that takes a graph representation as parameter, and an array `pred` that was computed by a previous call to `BFS`. The method also takes as parameter a `LinkedList<Integer>` object `path`. The method uses the data stored in `pred` to build a path to `v` that is then stored in `path`.

2. User Interaction and Output

Your program's user interaction and output should look like what you see below.

```
[a, b, c, d, e, f, g, h, i]
{a=0, b=1, c=2, d=3, e=4, f=5, g=6, h=7, i=8}

0 : [1, 2]
1 : [0, 2, 4]
2 : [5]
3 : [6]
4 : [5, 8]
5 : [3, 0, 1, 7]
6 : [5, 7]
```

```
7 : [8]
8 : [5]

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

Breadth First Paths are:
Path to the vertex v 0 | a
[0]
[a]
Path to the vertex v 1 | b
[0, 1]
[a, b]
Path to the vertex v 2 | c
[0, 2]
[a, c]
Path to the vertex v 3 | d
[0, 2, 5, 3]
[a, c, f, d]
Path to the vertex v 4 | e
[0, 1, 4]
[a, b, e]
Path to the vertex v 5 | f
[0, 2, 5]
[a, c, f]
Path to the vertex v 6 | g
[0, 2, 5, 3, 6]
[a, c, f, d, g]
Path to the vertex v 7 | h
[0, 2, 5, 7]
[a, c, f, h]
Path to the vertex v 8 | i
[0, 1, 4, 8]
[a, b, e, i]
```

The program uses a file chooser to open a file containing adjacency list data, just like in the last project. The program builds an adjacency list, and then performs a breadth-first search on the graph while populating the `pred` array. After that, the program uses the `pred` array and the `getPath` method to recover the paths to all vertices. These paths are printed in both the numeric form and the in the form that shows the original name of the vertices.

Due date: This is due Saturday at the end of Week 8.