Programming Assignment 10

Due 30 NOV @ 11:59pm

Write a program that implements the ADT Graph. The Graph is undirected and should be implemented as an adjacency list. The Bag object should serve as the neighbor list for each vertex. The Graph should hold an array of the Bag objects where the index serves as the vertex identifier. When adding edges to the Graph, you do not need to worry about self-loops, parallel (duplicate) edges, or cycles.

Using the Graph, you must then implement the DFS algorithm using the provided Stack and the BFS algorithm using the provided Queue. Both implementations must keep track of the connectivity (using the marking method discussed in lecture) to a given source and also the path from each vertex to that source. You must properly implement the API's indicated in the javadoc.

The main method in both DFS and BFS read in a file (formatted below) and the source vertex to conduct the search from. It will create the Graph, call your search method, and print the results to standard output.

Grading Notes

You must:

- Use the template provided for you
- Have a style (indentation, good variable names, etc.)
- Comment your code well (no need to over do it, just do it well)

You may not:

- Make your program part of a package.
- Use *code* from anywhere except your own brain.

Submission Instructions:

- Name a folder with your gmu username
- Put your java files in the folder (but not your .class)
- Zip the folder (not just the files) and name the zip “username-pa10.zip”
- Submit to blackboard

Grading Rubric

No Credit:

- Non-submitted assignments
- Late assignments
- Non-compiling assignments
- Non-independent work

1pt	Submission Format
1pt	Style and Comments
1pts	Graph constructor, addEdge and neighbors, V, and E methods
3pts	DFS search
3pts	BFS search
1pt	pathFromSource (DFS and BFS)

Example input file "graph.txt":

First line is number of vertices, subsequent lines are the edges

```
13
0  5
4  3
0  1
9  12
6  4
5  4
0  2
11  12
9  10
0  6
7  8
9  11
5  3
```

Example DFS Run

```
>  java DFS graph.txt 0
Paths to source: 0
  path for 0 : ->0
  path for 1 : ->0->1
  path for 2 : ->0->2
  path for 3 : ->0->5->4->3
  path for 4 : ->0->5->4
  path for 5 : ->0->5
  path for 6 : ->0->5->4->6
  path for 7 :
  path for 8 :
  path for 9 :
  path for 10 :
  path for 11 :
  path for 12 :

>   java DFS graph.txt 12
Paths to source: 12
  path for 0 :
  path for 1 :
  path for 2 :
  path for 3 :
  path for 4 :
  path for 5 :
  path for 6 :
  path for 7 :
  path for 8 :
  path for 9 : ->12->9
  path for 10 : ->12->9->10
  path for 11 : ->12->9->11
  path for 12 : ->12
```

Example BFS Run

```
>  java BFS graph.txt 0
Paths to source: 0
  path for 0 : ->0
  path for 1 : ->0->1
  path for 2 : ->0->2
  path for 3 : ->0->5->3
  path for 4 : ->0->6->4
  path for 5 : ->0->5
  path for 6 : ->0->6
  path for 7 :
  path for 8 :
  path for 9 :
  path for 10 :
  path for 11 :
  path for 12 :

>  java BFS graph.txt 12
Paths to source: 12
  path for 0 :
  path for 1 :
  path for 2 :
  path for 3 :
  path for 4 :
  path for 5 :
  path for 6 :
  path for 7 :
  path for 8 :
  path for 9 : ->12->9
  path for 10 : ->12->9->10
  path for 11 : ->12->11
  path for 12 : ->12
```