Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 9 REPORT

Deniz Can Erdem YILMAZ
151044001

Course Assistant: Ahmet SOYYİĞİT

5. Problem Solutions Approach

There are 5 methods to implement between interface/abstract classes and normal classes. Most of them are similar to each other as algorithm: finding a vertex and vertices bound to them which basically is a search algorithm. So in generally all methods use similar techniques with little difference to solve the problem. Here is the deep explaination for all desired methods:

addRandomEdgesToGraph:
Random number of edges must be inserted in graph between 0 and given number parameter. To accomplish this Random class used to get a random number from the interval. nextInt method throws exception in any number <= 0. First of all an input check has done to prevent errors from that method. After that a for loop checks if an edge with random destination and source value contains in graph. Inserts that edge if it doesn't and increases count. Returns count to indicate number of edges inerted to graph. Calls with invalid parameters gets 0 as return value immediately.

breadthFirstSearch:
Base algorithm taken from the course book. It simply holds a queue to visit after current index. Each vertex inserts its unvisited or not queued neighbors to queue until the queue is empty. Each vertex inserted to queue sets current index as their parent. Finally parent table (array) returned to caller.

getConnectedComponentUndirectedGraph:
A breadth first search finds all vertices connected to given start index which is a connected graph itself. If a breadth first search applied to all not found vertices it would give each connected component as a search table which shows all connections for a starter position. A helper method graphCounter calls search method for all not found indices in order. Each not found index is a connected components smallest index, these indices stored in an ArrayList to return our main method. Size of this ArrayList is also the size of return value, Graph array. Another helper method buildGraphFromTable used to build new graph using parent tabels obtained from breadth first search of smallest elements and placed in return array. After that result array returned to caller. If graph is directed those methods would cause errors. So this is an undirected only method. Call from a directed graph gets null as return value immediately.

isBipartiteUndirectedGraph:
The algorithm is just like the search method. Instead of using a boolean array to indicate a vertex is identified (queued for search) an integer array holds data of vertices' colours. 0 represents an unidentified vertex. 1 or -1 represents blue or red colour (equivalent of identified). An unidentified neighbor vertex queued for search and painted with the opposite of current vertex. An identified vertex colour controlled if it is appropriate (opposite of current colour). Methods returns false the moment there an inappropriate vertex has encountered. Else continues to search until the queue is empty. Just like the previous method this is also undirected graph only method. Call from a directed graph gets false as return value.

writeGraphToFile:
As the name refers, this method writes all contained edge to an output file on the example input format. isEdge method of Graph interface used to obtain all edges of graph by trying each combination one by one. String come from helper format method, written to file after the vertex number of the graph.

6. Test Cases

To test this methods a main class has constructed by the request of homework; test each graph (MatrixGraph & ListGraph) twice, with different input files. 2 methods (one for MatrixGraph, one for ListGraph) takes input file name and direction feature indicator implemented. These methods called from a test method which takes input file name. This could be done in 1 method same with Matrix/ListGraph methods which also takes graph as input with type AbstractGraphExtended to use polymorphism. But I had a different test plan on my mind and returned here from that plan. So this just remained compatible with my old idea, which I admit is bad coding style. Graph types, input numbers and directions are all generated randomly to test different conditions each time. Comparisons made and printed to terminal. Also result graphs written into output files and generated output file names shared with user via terminal too. After the test ends, program asks user to delete the output files or not. Deleting them after checking graphs recommended to prevent pollution.

7. Running and Results

Please see the screen shots (Can be found under the SS folder with higher quality)

```
********************************************************************************************************************
********************************************************************************************************************
****************************************      TEST ONE      ********************************************************
********************************************************************************************************************
********************************************************************************************************************
++++++++++++++++++++++++++++++++++++     TESTING LIST     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing a ListGraph
From file: bipartite.txt
Direction: false
--------------------------------------------------------------------------------------------------------------------
Trying to insert max 1 edges to ListGraph
1 edges inserted to ListGraph
--------------------------------------------------------------------------------------------------------------------
2 graphs produced from graph (with random insertions) given on file: bipartite.txt
Writing sub-Graph1 with 6 vertex to file: test0_subGraph1.txt
Writing sub-Graph2 with 4 vertex to file: test0_subGraph2.txt
Separated total count: 10
Graph's initial count: 10
--------------------------------------------------------------------------------------------------------------------
Parent table:
0  :  1
1  :  0
2  :  1
3  :  1
4  :  2
5  :  0
6  :  -1
7  :  -1
8  :  -1
9  :  -1
--------------------------------------------------------------------------------------------------------------------
Current graph constructed from file bipartite.txt and 1 new edges added is BIPARTITE
--------------------------------------------------------------------------------------------------------------------
Writing result graph to the file: test0_result.txt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++++++++++++++++++     TESTING MATRIX     ++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing a MatrixGraph
From file: bipartite.txt
Direction: true
--------------------------------------------------------------------------------------------------------------
Trying to insert max 1 edges to MatrixGraph
1 edges inserted to MatrixGraph
--------------------------------------------------------------------------------------------------------------
getConnectedComponentUndirectedGraph method failed due to directed graph (result: null)
--------------------------------------------------------------------------------------------------------------
Parent table:
0  :  -1
1  :  0
2  :  1
3  :  1
4  :  2
5  :  0
6  :  -1
7  :  -1
8  :  -1
9  :  -1
--------------------------------------------------------------------------------------------------------------
isBipartiteUndirectedGraph method failed due to directed graph (result: NOT BIPARTITE)
--------------------------------------------------------------------------------------------------------------
Writing result graph to the file: test1_result.txt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



**********************************************************************************************************
**********************************************************************************************************
****************************************      TEST TWO      ******************************************
**********************************************************************************************************
**********************************************************************************************************
++++++++++++++++++++++++++++++++++++     TESTING LIST     ++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing a ListGraph
From file: mixed.txt
Direction: true
--------------------------------------------------------------------------------------------------------------
Trying to insert max 3 edges to ListGraph
2 edges inserted to ListGraph
--------------------------------------------------------------------------------------------------------------
getConnectedComponentUndirectedGraph method failed due to directed graph (result: null)
--------------------------------------------------------------------------------------------------------------
Parent table:
0  :  -1
1  :  0
2  :  1
3  :  0
4  :  1
5  :  1
6  :  3
7  :  4
8  :  6
9  :  -1
10  :  4
11  :  -1
12  :  -1
13  :  10
14  :  10
15  :  4
--------------------------------------------------------------------------------------------------------------
isBipartiteUndirectedGraph method failed due to directed graph (result: NOT BIPARTITE)
--------------------------------------------------------------------------------------------------------------
Writing result graph to the file: test2_result.txt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++++++++++++++++++     TESTING MATRIX     ++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing a MatrixGraph
From file: mixed.txt
Direction: false
------------------------------------------------------------------------------------------------------------
Trying to insert max 1 edges to MatrixGraph
1 edges inserted to MatrixGraph
------------------------------------------------------------------------------------------------------------
4 graphs produced from graph (with random insertions) given on file: mixed.txt
Writing sub-Graph1 with 12 vertex to file: test3_subGraph1.txt
Writing sub-Graph2 with 2 vertex to file: test3_subGraph2.txt
Writing sub-Graph3 with 1 vertex to file: test3_subGraph3.txt
Writing sub-Graph4 with 1 vertex to file: test3_subGraph4.txt
Separated total count: 16
Graph's initial count: 16
------------------------------------------------------------------------------------------------------------
Parent table:
0  :  1
1  :  0
2  :  1
3  :  0
4  :  1
5  :  1
6  :  3
7  :  4
8  :  6
9  :  -1
10  :  4
11  :  -1
12  :  -1
13  :  10
14  :  10
15  :  -1
------------------------------------------------------------------------------------------------------------
Current graph constructed from file mixed.txt and 1 new edges added is NOT BIPARTITE
------------------------------------------------------------------------------------------------------------
Writing result graph to the file: test3_result.txt
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



Would you like to delete (10) output files created after tests?
Press ENTER to DELETE all of them (any other input will allow them to exist)
```

```
Would you like to delete (10) output files created after tests?
Press ENTER to DELETE all of them (any other input will allow them to exist)

Deleting file: test0_subGraph1.txt
Deleting file: test0_subGraph2.txt
Deleting file: test0_result.txt
Deleting file: test1_result.txt
Deleting file: test2_result.txt
Deleting file: test3_subGraph1.txt
Deleting file: test3_subGraph2.txt
Deleting file: test3_subGraph3.txt
Deleting file: test3_subGraph4.txt
Deleting file: test3_result.txt

Process finished with exit code 0
```