Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 9 REPORT

VAKHID BETRAKHMADOV
141044086

Course Assistant: Ahmet SOYYİĞİT

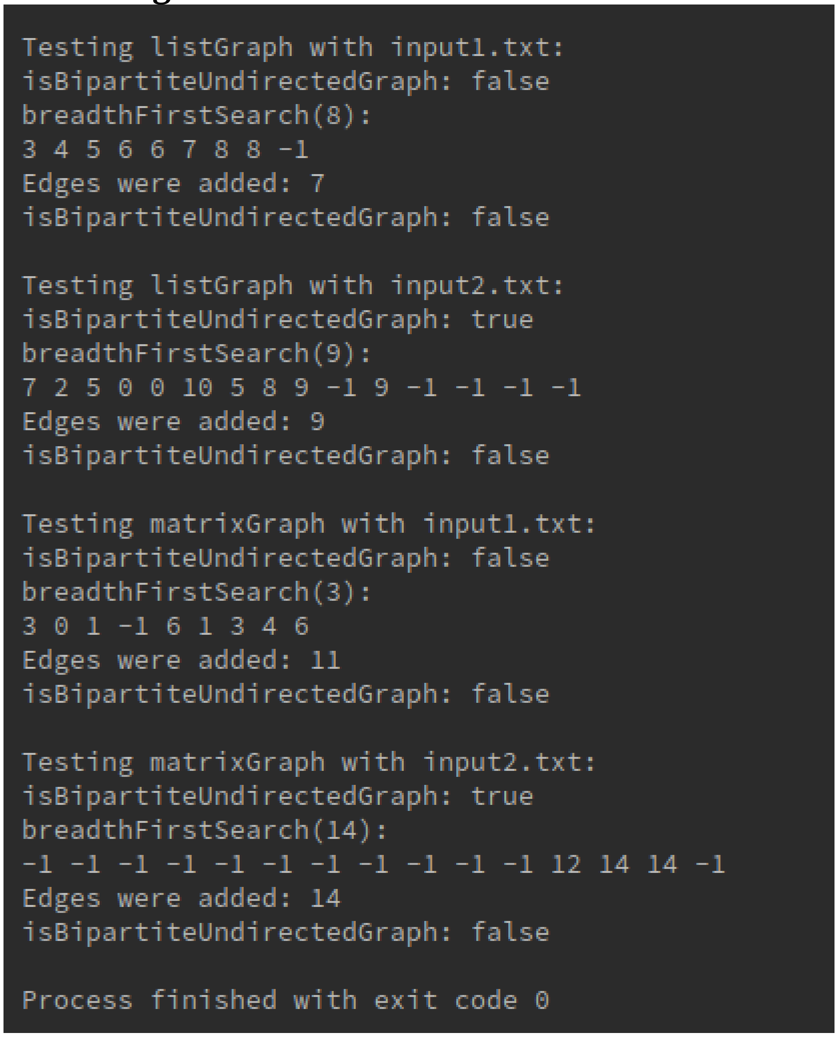
1.Problem solution approach

Method ***addRandomEdgesToGraph*****:** In this method we were supposed to add random number (between 0 and edgeLimit) of random edges to the existing graph. The method is quite explicit, the only detail we had to take care of was repeating edges, while this is not a problem for *MatrixGraph*, it would create unnecessary duplicate nodes in *ListGraph* class, so before adding a new random edge, we first check if such edge already exists in the graph.
Method ***breadthFirstSearch:*** This method is supposed to perform breadth first search of a graph. This method was taken from the book, made concrete and modified just a bit for further use in other methods, besides marking identified vertexes in the local *identified* array, it also marks identified vertexes in the class's data field array.
Method ***getConnectedComponentUndirectedGrap:*** In order to get all connected components of an unconnected graph, we iterate through all vertexes of the graph and call *breadthFirstSearch* on vertexes which are no marked identified in the class's *identified* array. Thus we get arrays representing vertexes of all connected components of the unconnected graph. Then we create appropriate sets of edges for all arrays of vertexes and map them to new values for subclass consistency. From these new sets of edges we create new graphs.
Method ***isBipartiteUndirectedGraph:*** An unconnected graph is bipartite if its all connected components are bipartite. So first we call *getConnectedComponentUndirectedGrap* method to get all connected components and then we call private *isBipartiteUndirectedGraph* helper method on each of them. Private *isBipartiteUndirectedGraph* helper method recursively traverses the graph coloring its vertexes to red and black colors. If a vertex can not be colored to black or red color (already has neighbor of either color) the method returns false, otherwise if all vertexes were successfully colored it returns true.
Method ***writeGraphToFile:*** This method traverses all edges of the graph and adds them in a non repeating way (if [1 2] is in the set [2 1] will not be added) to the set of edges. Then number of vertexes of this graph and contents of the set of edges are written to the output file.

2.Test cases

We have 2 input files (input1.txt and input2.txt), first one contains one connected graph which is not bipartite (contains odd cycle, Konig's Theorem), second one contains unconnected graph consisting of 2 connected graphs which are bipartite.
We call *testAbstractGraphExtended* test method with ListGraph and MatrixGraph instances created from input1.txt and input2.txt in order to test all our methods.
TestAbstractGraphExtended method first checks if the graph is bipartite, then performs breadth first search on a random vertex and prints contents of the array which contains the predecessor of each vertex in the breadth-first search tree, then adds random number of edges between zero and 15, and checks if the graph is bipartite now.
TestAbstractGraphExtended method also finds all connected elements of the graph and prints them into appropriate output files, as well as the new graph obtained by insertion of random number of edges.
Results are shown below.

3.Running command and results



```
Testing listGraph with input1.txt:
isBipartiteUndirectedGraph: false
breadthFirstSearch(8):
3 4 5 6 6 7 8 8 -1
Edges were added: 7
isBipartiteUndirectedGraph: false

Testing listGraph with input2.txt:
isBipartiteUndirectedGraph: true
breadthFirstSearch(9):
7 2 5 0 0 10 5 8 9 -1 9 -1 -1 -1 -1
Edges were added: 9
isBipartiteUndirectedGraph: false

Testing matrixGraph with input1.txt:
isBipartiteUndirectedGraph: false
breadthFirstSearch(3):
3 0 1 -1 6 1 3 4 6
Edges were added: 11
isBipartiteUndirectedGraph: false

Testing matrixGraph with input2.txt:
isBipartiteUndirectedGraph: true
breadthFirstSearch(14):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 14 14 -1
Edges were added: 14
isBipartiteUndirectedGraph: false

Process finished with exit code 0
```