GIT Department of Computer Engineering

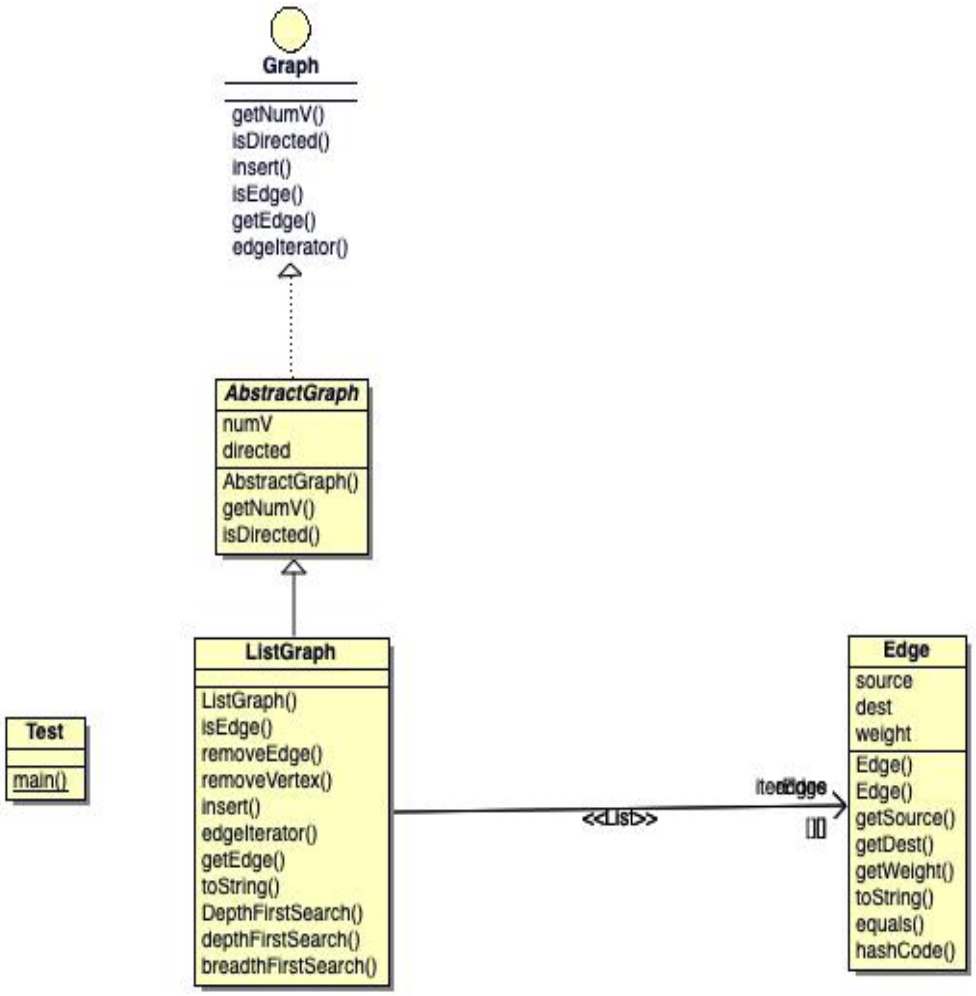
CSE 222/505 - Spring 2020

Homework 8 Report

Dilara KARAKAŞ
171044010

Q2

1. CLASS DIAGRAM



2. PROBLEM SOLUTION

This class is implemented the extended graph ADT using 2-D linked-list structure. Deletion is defined in two different ways. If a vertex is given, all its connections are found and the edge erasing process is called. If two vertexes are given, if there is such an edge in the graph, it performs the deletion. While creating the graph, it is stated that the constructor is either directional or non-directional. If the directional graph is "true", then the non-directional graph is sent "false". BFS and DFS are defined as required and print the path.

3. TEST CASE

TEST CASE NAME	ACTION	TEST DATA	EXPEXTED	TEST RESULTS
Undirected graph	Constructor	false	Undirected graph is created	pass
Add	adding	0,1,8 Ect.	adding	pass
DFS	searching	3	3 0 1 4 2	pass
BFS	searching	3	3 0 4 1 2	pass
DFS	searching	4	4 1 0 3 2	pass
BFS	searching	4	4 1 0 2 3	pass
Remove vertex	removing	3	Remove	pass
Remove edge	removing	4,0	remove	pass

TEST CASE NAME	**ACTION**	**TEST DATA**	**EXPEXTED**	**TEST RESULTS**
Directed graph	Constructor	true	Directed graph is created	pass
Add	adding	0,1,8 Ect.	adding	pass
DFS	searching	3	3 0 1 4 2	pass
BFS	searching	3	3 0 1 4 2	pass
DFS	searching	4	4 0 1 2 3	pass
BFS	searching	4	4 0 2 3 1	pass
Remove vertex	removing	3	Remove	pass
Remove edge	removing	4,0	remove	pass

TEST CASE NAME	**ACTION**	**TEST DATA**	**EXPEXTED**	**TEST RESULTS**
Directed graph	Constructor	true	Directed graph is created	pass
Add	adding	0,1,6 Ect.	adding	pass
DFS	searching	3	3 4 1 0 7 2 6	pass
BFS	searching	3	3 4 1 0 2 7 6	pass
DFS	searching	6	6 7 3 4 1 0 2	pass
BFS	searching	6	6 7 3 0 4 1 2	pass
Remove vertex	removing	3	Remove	pass
Remove edge	removing	6,0	remove	pass

4. TEST RESULT

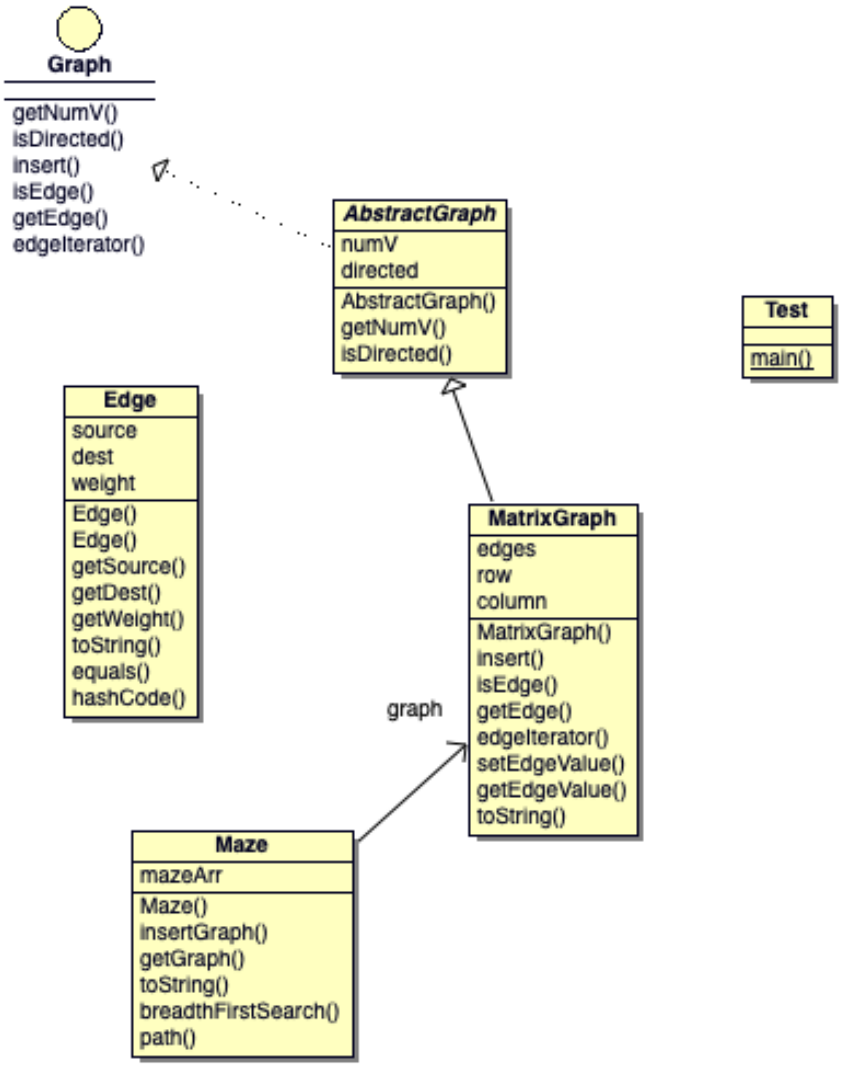
```
****** FIRST GRAPH (UNDIRECTED) ******

0) []          [[(0, 1): 8.0], [(0, 1): 7.0]]          []          [[(0, 3): 4.0]]         [[(0, 4): 1.0]]
1) [[(1, 0): 8.0], [(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          [[(2, 4): 5.0]]
3) [[(3, 0): 4.0]]          []          []          []          [[(3, 4): 6.0]]
4) [[(4, 0): 1.0]]          [[(4, 1): 2.0]]         [[(4, 2): 5.0]]         [[(4, 3): 6.0]]         []


*** Depth First Search 3 ***

3 0 1 4 2
*** Breadth First Search 3 ***

3 0 4 1 2
*** Depth First Search 4 ***

4 1 0 3 2
*** Breadth First Search 4 ***

4 1 0 2 3
*** REMOVE VERTEX 3
***
0) []          [[(0, 1): 8.0], [(0, 1): 7.0]]          []          []          [[(0, 4): 1.0]]
1) [[(1, 0): 8.0], [(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          [[(2, 4): 5.0]]
3) []          []          []          []          []
4) [[(4, 0): 1.0]]          [[(4, 1): 2.0]]         [[(4, 2): 5.0]]         []          []


*** REMOVE EDGE 4->0
***
0) []          [[(0, 1): 8.0], [(0, 1): 7.0]]          []          []          []
1) [[(1, 0): 8.0], [(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          [[(2, 4): 5.0]]
3) []          []          []          []          []
4) []          [[(4, 1): 2.0]]         [[(4, 2): 5.0]]         []          []
```

```
****** SECOND GRAPH (DIRECTED) ******

0) []          [[(0, 1): 8.0]]        []          []          []
1) [[(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          []
3) [[(3, 0): 4.0]]          []          []          []          []
4) [[(4, 0): 1.0]]          []          [[(4, 2): 5.0]]        [[(4, 3): 6.0]]        []


*** Depth First Search 3 ***

3 0 1 4 2
*** Breadth First Search 3 ***

3 0 1 4 2
*** Depth First Search 4 ***

4 0 1 2 3
*** Breadth First Search 4 ***

4 0 2 3 1
*** REMOVE VERTEX 3 ***

0) []          [[(0, 1): 8.0]]        []          []          []
1) [[(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          []
3) []          []          []          []          []
4) [[(4, 0): 1.0]]          []          [[(4, 2): 5.0]]        []          []


*** REMOVE EDGE 4->0 ***

0) []          [[(0, 1): 8.0]]        []          []          []
1) [[(1, 0): 7.0]]          []          []          []          [[(1, 4): 2.0]]
2) []          []          []          []          []
3) []          []          []          []          []
4) []          []          [[(4, 2): 5.0]]        []          []
```

```
****** THIRD GRAPH (DIRECTED) ******

0) []          [[(0, 1): 6.0]]        []         []         []         []         [[(0, 7): 8.0]]
1) []          []         []         [[(1, 3): 6.0]]        []         []         []
2) []          []         []         []         []         []         [[(2, 6): 5.0]]        []
3) [[(3, 0): 4.0]]         [[(3, 1): 4.0]]        [[(3, 2): 4.0]]        [[(3, 3): 4.0]]        [[(3, 4): 2.0]]        []         []         []
4) []          []         []         [[(4, 3): 4.0]]        []         []         []         []
5) []          []         []         [[(5, 3): 7.0]]        []         []         []         []
6) [[(6, 0): 5.0]]         []         []         [[(6, 3): 1.0]]        []         []         []         [[(6, 7): 1.0]]
7) []          []         []         []         []         []         []         []


***Depth First Search 3***
3 4 1 0 7 2 6
***Breadth First Search 3***
3 4 1 0 2 7 6
***Depth First Search 6***
6 7 3 4 1 0 2
***Breadth First Search 6***
6 7 3 0 4 1 2
*** REMOVE VERTEX 3 ***

0) []          [[(0, 1): 6.0]]        []         []         []         []         [[(0, 7): 8.0]]
1) []          []         []         []         []         []         []         []
2) []          []         []         []         []         []         [[(2, 6): 5.0]]        []
3) []          []         []         []         []         []         []         []
4) []          []         []         []         []         []         []         []
5) []          []         []         []         []         []         []         []
6) [[(6, 0): 5.0]]         []         []         []         []         []         []         [[(6, 7): 1.0]]
7) []          []         []         []         []         []         []         []


*** REMOVE EDGE 6->0 ***

0) []          [[(0, 1): 6.0]]        []         []         []         []         [[(0, 7): 8.0]]
1) []          []         []         []         []         []         []         []
2) []          []         []         []         []         []         [[(2, 6): 5.0]]        []
3) []          []         []         []         []         []         []         []
4) []          []         []         []         []         []         []         []
5) []          []         []         []         []         []         []         []
6) []          []         []         []         []         []         []         [[(6, 7): 1.0]]
7) []          []         []         []         []         []         []         []
```

Q3

1. CLASS DIAGRAM

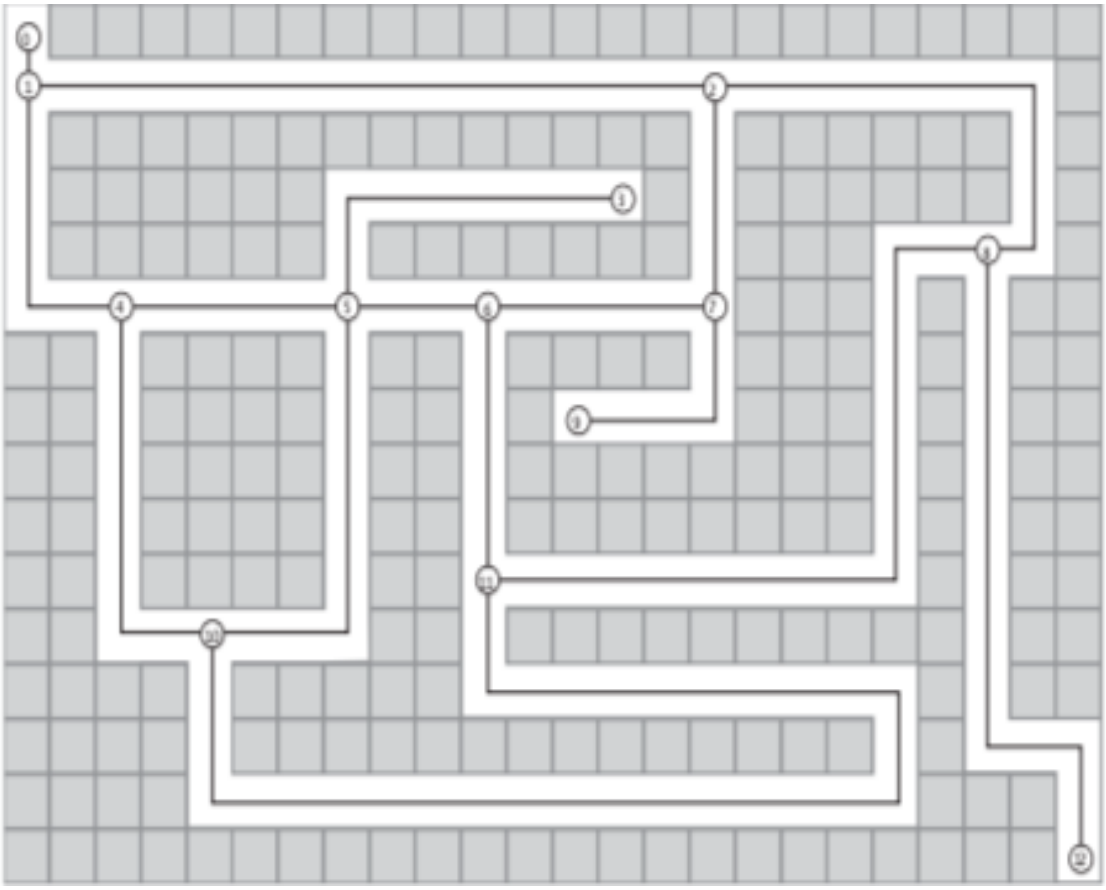


2. PROBLEM SOLUTION

First of all, the program asks you to enter the name of the file that should be read. For example, a file is loaded (test.txt). It is expected to log into the console with the extension of the file. Otherwise it cannot open the file and throws an exception. Assigns the file to double-sided. 0 is the path of our maze. Based on this, a graph is created and the paths now print as 1. Then the shortest route is found with BFS based on the exit gate.

3. TEST CASE

The sample in the book has been tried. The way to go is 0 -> 1 -> 2 -> 8 -> 12. Since this increase is adjusted in direct proportion with the weight, it is expected to be 23.


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

4. TEST RESULT