

Homework #8 Part2 Report

For compile in Virtual machine terminal:

In src file type: `javac -cp . fidan/*.java`

Then for running type: `java -cp . fidan/Main`

I write implementation mentioned in the pdf in TwoDimensionList class and my node class contains Edge,rprev,rnext,cprev,cnext. Either you can create a random graph or you can create a graph with using insert method.

Test screenShot Examples



The screenshot displays an IDE with a project named 'fidan' in the left sidebar. The project contains several classes: AbstractGraph, DepthFirstSearch, Edge, Graph, Main, Node, and TwoDimensionList. The 'Main' class is selected, and its code is shown in the editor. The code defines a TwoDimensionList object 'myGraph' with 5 vertices and directed edges. It then performs a Breadth First Search starting from vertex 1 and a Depth First Search starting from vertex 0. The output window at the bottom shows the results of these searches.

```
public static void main(String[] args) {  
  
    TwoDimensionList myGraph = new TwoDimensionList( numV: 5, directed: true);  
  
    myGraph.insert(new Edge( source: 0, dest: 1));  
    myGraph.insert(new Edge( source: 1, dest: 0));  
    myGraph.insert(new Edge( source: 1, dest: 4));  
    myGraph.insert(new Edge( source: 3, dest: 0));  
    myGraph.insert(new Edge( source: 4, dest: 0));  
    myGraph.insert(new Edge( source: 4, dest: 2));  
    myGraph.insert(new Edge( source: 4, dest: 3));  
  
    System.out.println("Breadth First Search Start from vertex 1");  
    myGraph.breadthFirstSearch( start: 1);  
    System.out.println("Depth First Search Start from vertex 0");  
    DepthFirstSearch depthSearch= new DepthFirstSearch(myGraph);  
}
```

Output:

```
Breadth First Search Start from vertex 1  
1  
0  
4  
2  
3  
Depth First Search Start from vertex 0  
2  
3  
4  
1  
0
```

Here I create mygraph then insert it edges and i build same graph in pdf. Then calling breathFirst algorithm with starting from vertex 1 and calling depth first algorithm with starting from vertex 0.

```

src
└─ fidan
   ├── AbstractGraph
   ├── DepthFirstSearch
   ├── Edge
   ├── Graph
   ├── Main
   ├── Node
   └── TwoDimensionList

Main > main()
Main <
Directed Graph
Vertex number = 5
2,4
0,4
1,3
0,3
2,3
1,0
0,2
Breadth First Search start from vertex 0:
0
2
3
4
Depth First Search start from vertex 0:
3
4
2
0
1

```

Here I use createRandomGraph and its create a random graph. Then I print graph properties and edges. Then I call Breath First Search from starting vertex 0 and I call Depth first search from starting vertex 0.

Homework #8 Part3 Report

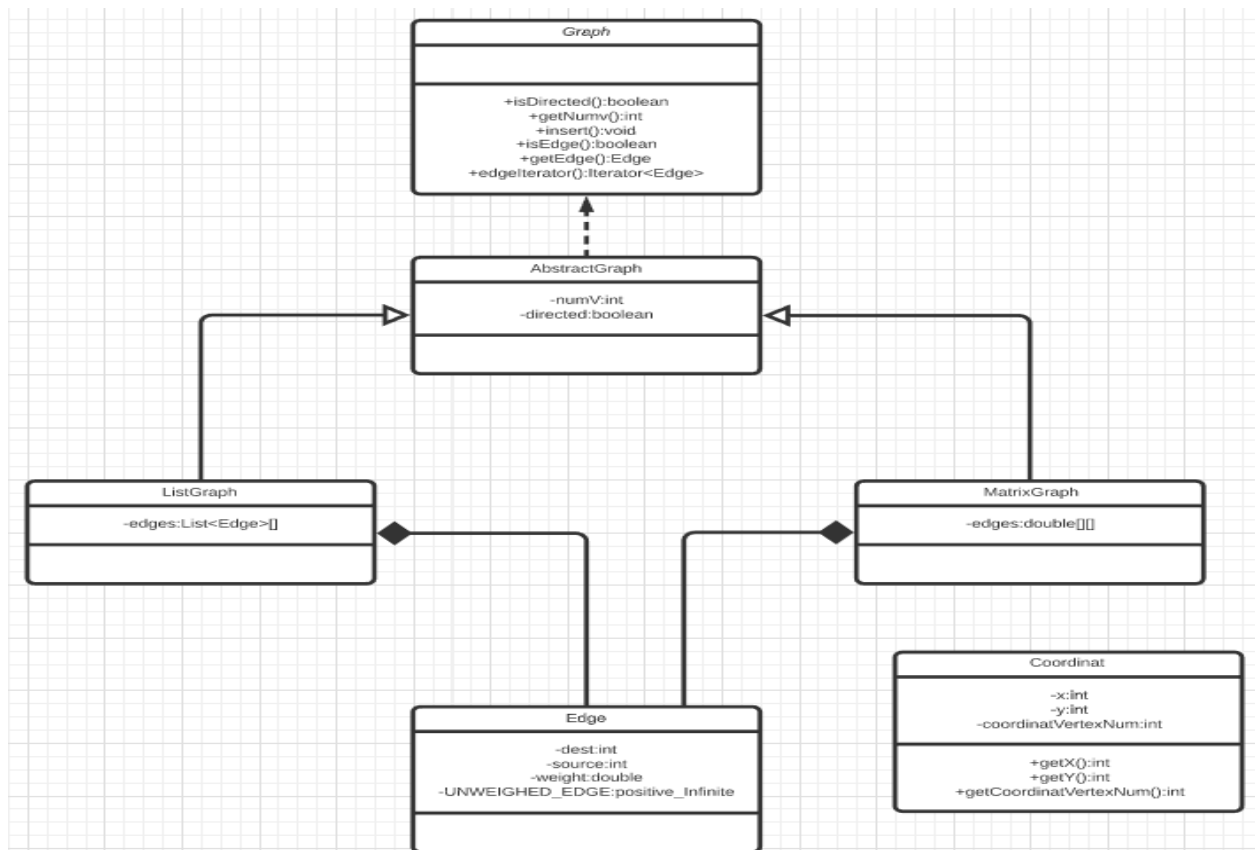
For compile in Virtual machine terminal:

In src file type: `javac -cp . fidan/*.java`

Then for running type: `java -cp . fidan/Main`

There are 5 class and 1 interface in my design. Graph interface , AbstractGraph , ListGraph, MatrixGraph and Edge classes from the book. Also I write a Coordinat class to determine graph vertex points and edge point in the maze. I read graph.txt file maze and first calculate edge and vertex number. Then using this vertex number and edge number I determine my graph is a dense or sparse graph. If graph is a dense graph , I created my graph using matrix implementation, but if graph is a sparse graph I implement my graph as a ListGraph in main. Then insert all edges with its source and destination points into my graph with help of my coordinat class.After my graph is completed I use dijkstra algorithm in the book, so with this algorithm I calculate all shortest paths from the vertex 0(maze start point) to that vertex.

UML DIAGRAM



Test ScreenShots

```
Vertex size = 26
Edge size = 58
Node, Predecessor, and Distance:
0: 0 0.0
1: 0 1.0
2: 1 16.0
3: 2 23.0
4: 10 14.0
5: 6 30.0
6: 7 27.0
7: 3 26.0
8: 1 5.0
9: 8 7.0
10: 9 12.0
11: 10 15.0
12: 11 20.0
13: 12 23.0
14: 11 20.0
15: 14 28.0
16: 9 13.0
17: 16 15.0
18: 10 18.0
19: 14 22.0
20: 19 31.0
21: 6 36.0
22: 21 38.0
23: 17 18.0
24: 23 33.0
25: 22 40.0
```

```
From Upper-left(begin) to lower-right(finish) distant = 40.0
```

This is my program screen shot for maze given in pdf. First it prints Vertex number and Edge number, Then prints all vertices shortest distances. After that finally I print maze shortest path from start point to finish point. So maze in the pdf shortest path from the upper-left to lower-right distance is 40 square.

Muhammed Yasir Fidan

161044056