Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 8 REPORT

ELİF AKGÜN
1801042251

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

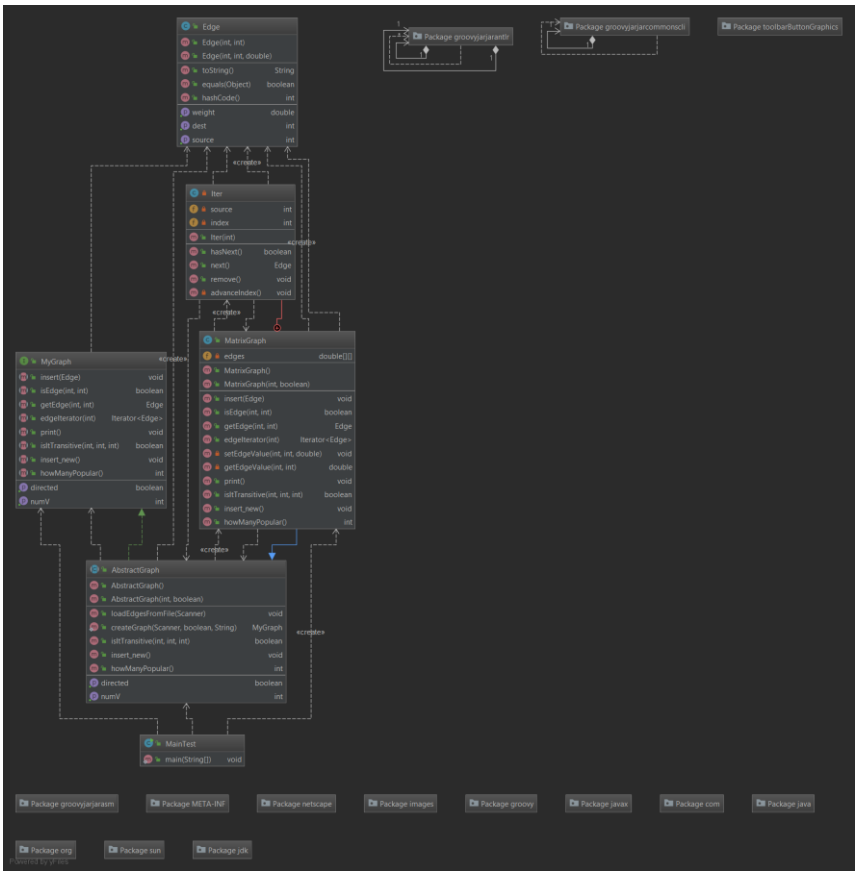
In this homework, there is a group of people with bilateral relations. These people think that the person with whom they have contact is popular. For example, you get a relationship between the X person and the Y person. (X, Y). X thinks Y is popular. At the same time, when X thinks that Y is popular, and Y thinks Z is popular, X also thinks Z is popular. Just like the transitivity in discrete mathematics. More than one person can think of a person as popular. What we're asked for is to find the person most popular.

1.2 System Requirements

My solution also does not require a specific piece of hardware. This program can work anywhere with JVM and can work with 128KB of memory, it does not keep memory so much and it needs the interface to work on the smart phone. It can work if it is provided

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

The only thing that a user needs to do is texts of dataset. User can run this program to click the run button.

2.3 Problem Solution Approach

During the project development phase, the objects that should be used first were considered and classes were created for those objects. Graph structure is used. First of all, graph adt was created. To this MyGraph interface is created. Then to implement method of MyGraph interface, AbstractGraph is created. To created graph structure, adjacent matrix was used. So MatrixGraph class was created. This class extends AbstractGraph. Also Edge class was created to keep the source, destination and weight.

MyGraph

Interface to specify a Graph ADT. A graph is a set of vertices and a set of edges. Vertices are represented by integers from 0 to n - 1. Edges are ordered pairs of vertices. Each implementation of the Graph interface should provide a constructor that specifies the number of vertices and whether or not the graph is directed.

Methods:

int getNumV() : Return the number of vertices
boolean isDirected(): Determine whether this is a directed graph.
void insert(Edge edge): Insert a new edge into the graph.
boolean isEdge(int source, int dest): Determine whether an edge exists.
Edge getEdge(int source, int dest): Get the edge between two vertices.
Iterator < Edge > edgeIterator(int source): Return an iterator to the edges connected to a given vertex.
void print(): Prints the matrix
boolean isItTransitive(int i, int j, int k): Chech whether the matrix is transitive
void insert_new(): After inserting matrix, it can be wrong values. This method checks whether matrix is true. If there is lack, corrects.
int howManyPopular(): Calculate number of populars.

AbstractGraph

Abstract base class for MyGraphs. A MyGraph is a set of vertices and a set of edges. Vertices are represented by integers from 0 to n - 1. Edges are ordered pairs of vertices.

Methods:

AbstractGraph implements the MyGraph interface so methods are the same with MyGraph interface.

MatrixGraph

A MatrixGraph is an implementation of the MyGraph. Abstract class that uses an array to represent the edges. This class also have an inner class, Iter.

***Iter*:** An iterator to the edges. An Edge iterator is similar to an Iterator except that its next method will always return an edge.

Methods:

MatrixGraph extends the AbstractGraph and its methods are the same with AbstractGraph.

Edge

An Edge represents a relationship between two vertices.

Methods:

Edge(int source, int dest): Construct an Edge with a source of from and a destination of to. Set the weight to 1.0.
Edge(int source, int dest, double w): Construct a weighted edge with a source of from and a destination of to. Set the weight to w.
int getSource(): Get the source
int getDest(): Get the destination
double getWeight(): Get the weight
String toString(): Return a String representation of the edge
boolean equals(Object obj): Return true if two edges are equal. Edges are equal if the source and destination are equal. Weight is not conidered.
int hashCode(): Return a hash code for an edge. The hash code is the source shifted left 16 bits exclusive or with the dest.

Complexity

Space complexity is O(V^2). It can be empty locations.

void insert(Edge edge): Its complexity is O(1). With the indexs, new element inserts.
boolean isEdge(int source, int dest): This method includes getEdgeValue method and it includes only if else conditions so its complexity is O(1). Also isEdge's complexity is O(1).
Edge getEdge(int source, int dest): This method includes getEdgeValue method and it includes only if else conditions so its complexity is O(1). Also isEdge's complexity is O(1).
Iterator < Edge > edgeIterator(int source): This method returns an iterator so its complexity is O(1).
void setEdgeValue(int source, int dest, double wt): Its complexity is O(1). It just sets edges.
getEdgeValue(int source, int dest): It includes only if else conditions so its complexity is O(1).
void print(): It has two inner for loop. This loops goes to V so its complexity is O(V^2).

boolean isItTransitive (int i, int j, int k): It includes only if else conditions so its complexity is O(1).
void insert_new(): It has three inner for loop. This loops goes to V so its complexity is O(V^3).
int howManyPopular(): It has two inner for loop. This loops goes to V so its complexity is O(V^2).

3 RESULT

3.1 Test Cases

With different input files, different relations, program was tested.

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class MainTest {

    public static void main( String [] arg) throws IOException {

        Scanner file = new Scanner( new File( pathname: "input.txt")); //reading input
        MyGraph graph;
        graph = new MatrixGraph();
        graph=AbstractGraph.createGraph(file, isDirected: true, type: "Matrix"); //creating graph

        for(int i=0; i<graph.getNumV(); ++i){ //If there is lack, corrects.
            graph.insert_new();
        }

        System.out.println("\nMatrix of people:");
        graph.print();
        System.out.println("Number of people who are considered popular by every other person:\n"+graph.howManyPopula
    }
```

Driver Class

input.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
7 11
1 2
1 3
2 1
2 5
4 5
5 3
5 4
6 1
6 3
7 2
7 5
```

Test File 1

input.txt - Not Defteri

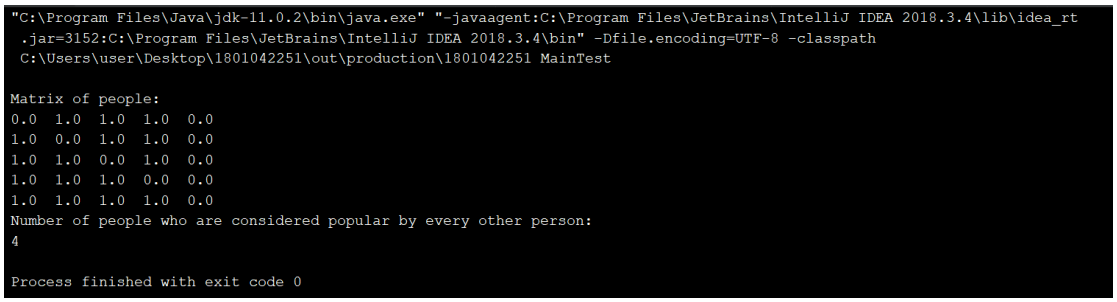
Dosya Düzen Biçim Görünüm Yardım

```
5 6
1 2
1 3
2 1
3 4
4 1
5 2
```

Test File 2

3.2 Running Results

Result for Test File 1



Result for Test File 2