Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 08 REPORT

Muhammed ÖZKAN
151044084

Course Assistant: Ayşe ŞERBETÇİ TURAN

1 INTRODUCTION

1.1 Problem Definition

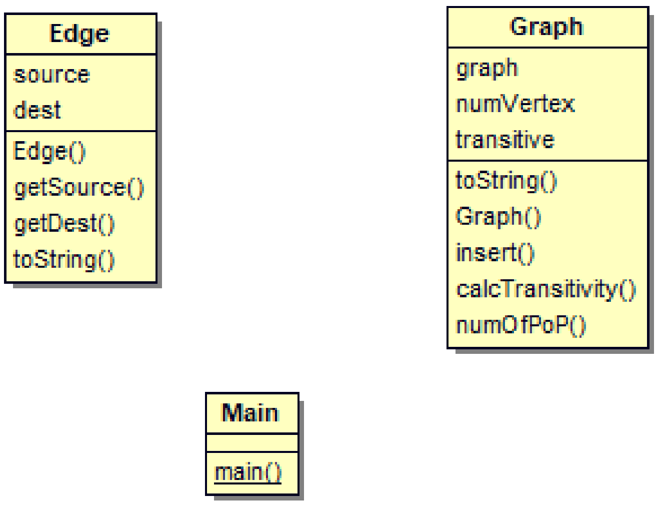
There is a group of people with a regular popularity relationship between the pairs. Our problem is find the number of these people's most popular. Given a directed graph,we must find out if a vertex i is accessible from another vertex j for all vertex pairs (i, j) in the given graph. In this assignment, the meaning of accessible that there is a path from Person1 to Person2.

1.2 System Requirements

We will develop this program for all devices running Java. The classes used in the solution of the problems have been developed considering the minimum possible memory consumption. The size of the graph class on memory is to vary depending on the type of data(vertex) to keep. Memory size shows a quadradic increase. In case of proper use, the prepared programs can be used in any environment, even on a smartphone.

2 METHOD

2.1 Class Diagrams



2.2 Use Case

The software works on the console screen. The user must specify the input file to be used as the parameter to the program before running the programs.

For example:

-java program_name input.txt

2.3 Problem Solution Approach

We use the Graphs to solve this problem. Two different classes are required for this. First we develop Edge class, second we develop Graph class. It holds the location of the source and destination information within the edge class. The graph class holds the relation of the vertices in the adjacency matrix, according to the information of the edges. The relations are transitive. We find this transitivity. We look for whether or not it can establish the connection between the two people who have no relationship.

2.4 Complexity of Functions

The complexity of functions is calculated according to the number and structure of the loops they contain. Since the complexity calculations are considered infinite, the comparison, assignment and similar operations within the functions are not included in the calculations since they do not have any meaning in infinity. V is number of vertices in the given graph.

Function Name	**Complexity**	**Big O Notation**
Graph.Graph()	$T_1(n)=v*v$	$O(V^2)$
Graph.insert()	$T_2(n)=1$	*O(1)*
Graph.calcTransitivity()	$T_3(n)=v*v*v$	$O(V^3)$
Graph.numOfPoP()	$T_4(n)=T_3+(v*v)$	$O(V^3)$

3 RESULT

3.1 Test Cases

I tested the program with input of various vertices and relations.

3.2 Running Results

```
Input:
6 7
1 2
2 3
4 1
4 3
6 4
5 6
3 4

Output:
4
```

```
Input:
6 7
2 4
1 3
3 6
4 2
5 6
3 2
6 1

Output:
2
```

```
Input:
6 6
1 2
2 3
4 1
4 3
6 4
5 6

Output:
1
```

```
Input:
6 8
1 2
2 3
4 1
4 3
6 4
5 6
5 1
3 4

Output:
4
```