Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 8 REPORT

Yusuf Can Kan
161044007

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

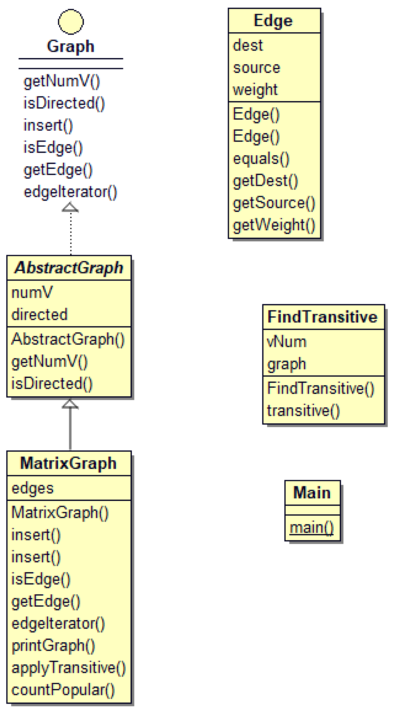
In this homework we have a group of people which has ordered popularity relation between them. (P1,P2) means that P1 thinks P2 is popular. When we calculating the relation we also have to add transitive relation. An example of transitive relation is that if there are exist two relation such as (P1,P2) and (P2,P3), than with respect to transitive relation (P1,P3) also exist. We have to add this situations our relation sets. The main purpose of this homework is finding count of people who everybody thinks popular. For representing the relation and people i used graph structure.

1.2 System Requirements

My solution does not require spesific piece of hardware. It requires just a little memory and CPU but not too much.You can use my solution with every computer unless the computer mush have JVM. It works every machine which has JVM. It does not requires spesific operating system. You can use it on Linux, Windows or Mac operating system. My solution is so simple, you can use it almost everywhere.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

This software can be uses as finding different relations between object with by taking into consideration transitive relation. User can easily seperate main objects which is related from all the other objects. The only thing which expected from user is giving the relation input file. The input file must consist of number of people, number of ordered relation and relations. The first line must consist number of people and number of ordered relation. The other line must consist relations with space-seperated numbers. For example if input ‘1 2’, than it means 1 thinks 2 is popular. Numbers must be between 0 and number of people. After the user gives the input file, the answer is going to be written to the terminal screen.

2.3 Problem Solution Approach

First i decided to use the graph structure and after that i decided which graph implementation should I use? .I decided matrix graph. I choose matrix graph because when i calculating the transitive relations it can be more usefull than list graph. So i implemented the Graph interface and i added the main methods inside it so i can easily manupulate the all code in the future. After that i implement the AbstractGraph class and i put vertex number and directed value variables inside of it. After that i create the MatrixGraph class. I holded the two dimentional integer array. Between the main methods (insert,getEdge,isEdge) i implemented the printGraph() method for make easy to testing. I implement the appyTransitive method which calculates the transitivity and applies it on to the all graph matrix. For that i created new class called FindTransitive. The all transitive algorithm is inside of it. And my final method is countPopular(). This method checks the graph matrix and counts the people which everybody thinks they are popular. My class time and space complexity table is below.

Matrix Graph Class		
Method Name	**Time Complexity**	**Space Complexity**
MatrixGraph(int,boolean) (Constructor)	O(n^2) which n is the vertex number.	O(n^2) which n is the vertex number.
insert(Edge)	O(1). It just sets the matrix.	O(1)
insert(int,int)	O(1). It just sets the matrix.	O(1)
printGraph()	O(n^2)	O(n^2)
applyTransitive()	O(n^3). It comes from FindTransitive method. The transitive algorithm traverses vertex for n^3 times. It has to check every vertex transitive situation and after checks it it has to goes to the beginning and checks all again because making changes to graph creates new transitive relations.	O(n^3).
countPopular()	O(n^2). It just controls the matrix and looks for appropriate values for colums.	O(n^2).

The main code time and space complexity is O(n^3) which n is the number of vertex.

3 RESULT

3.1 Test Cases

My test input values is below;

Input 1:

```
3 3
1 2
2 3
3 1
```

Input2:

```
3 3
1 2
2 1
2 3
```

Input 3:

```
4 8
1 1
1 2
1 4
2 2
2 3
3 3
3 4
4 4
```

Input 4:

```
7 10
1 2
2 1
2 3
4 6
4 3
2 4
3 7
4 7
5 7
6 7
```

Input 5:

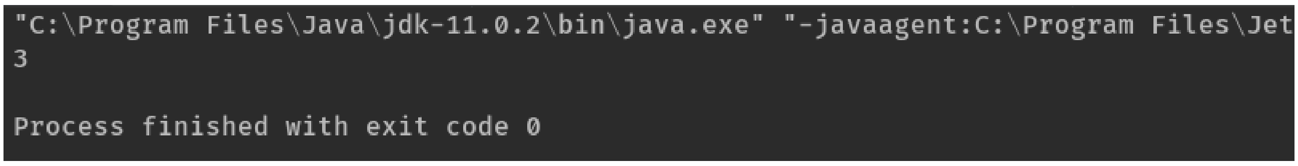
```
7 10
1 2
2 1
2 3
4 6
4 3
2 4
3 7
4 7
5 7
6 7
7 1
```

Input 6:

```
4 4
1 2
2 3
3 4
4 1
```

3.2 Running Results

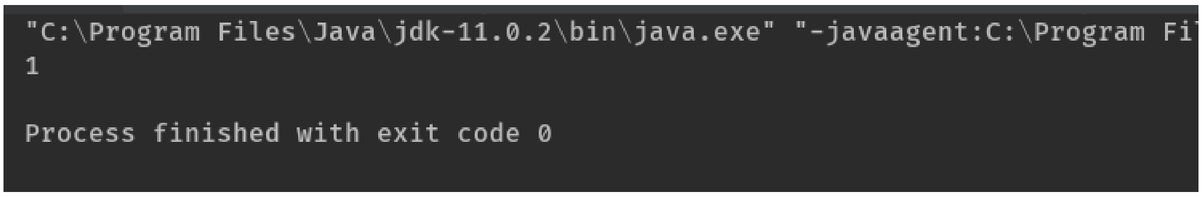
Output 1:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\Jet
3

Process finished with exit code 0
```

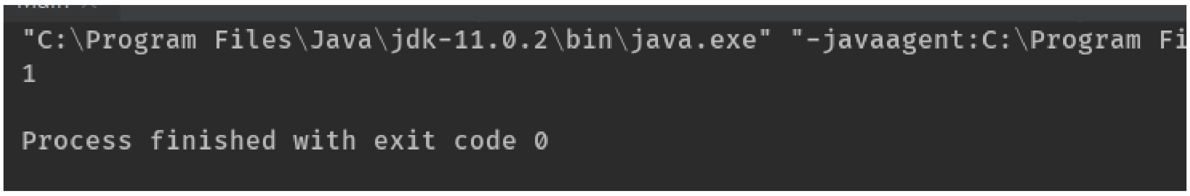
Output 2:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Fi
1

Process finished with exit code 0
```

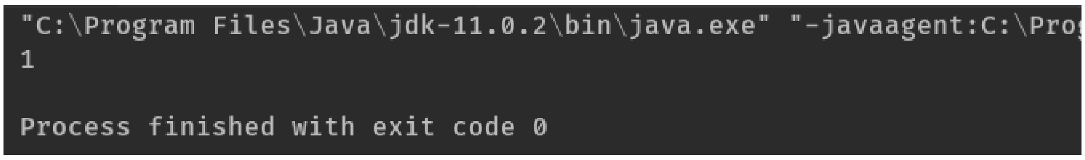
Output 3:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Fi
1

Process finished with exit code 0
```

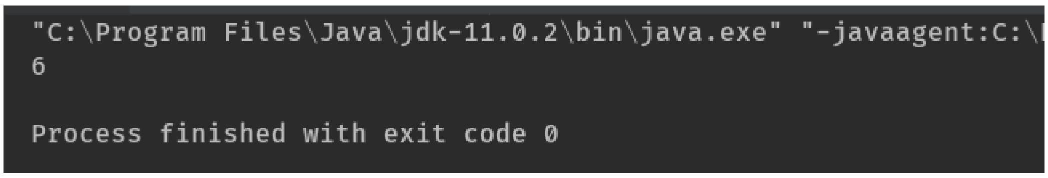
Output 4:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Pro
1

Process finished with exit code 0
```

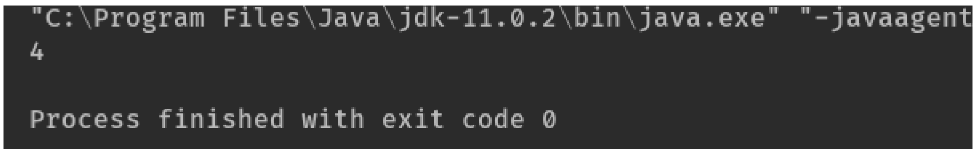
Output 5:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\
6

Process finished with exit code 0
```

Output 6:



```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent
4

Process finished with exit code 0
```