Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 8 REPORT

GÖKTUĞ ALİ AKIN
161044018

Course Assistant:AYŞE ŞERBETÇİ TURAN

1 INTRODUCTION

1.1 Problem Definition

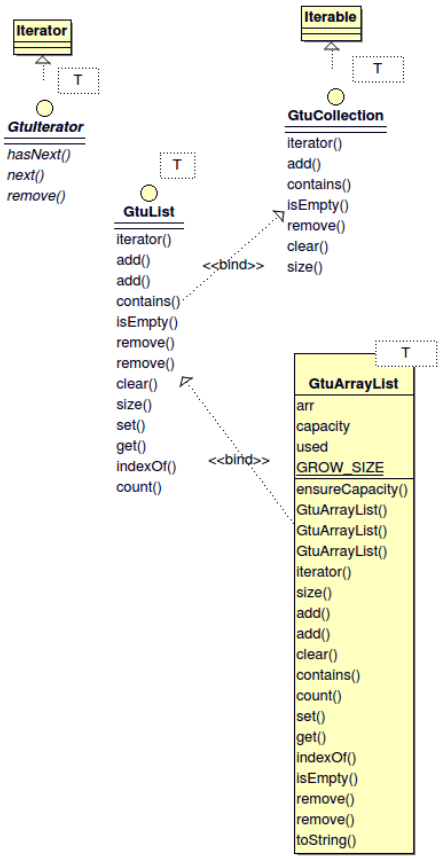
In this homework, the problem is design a program that finds the number of people who are considered popular by all other people. User of this program, writes the relations which describes the relations between people into a txt file and can easly find the result.

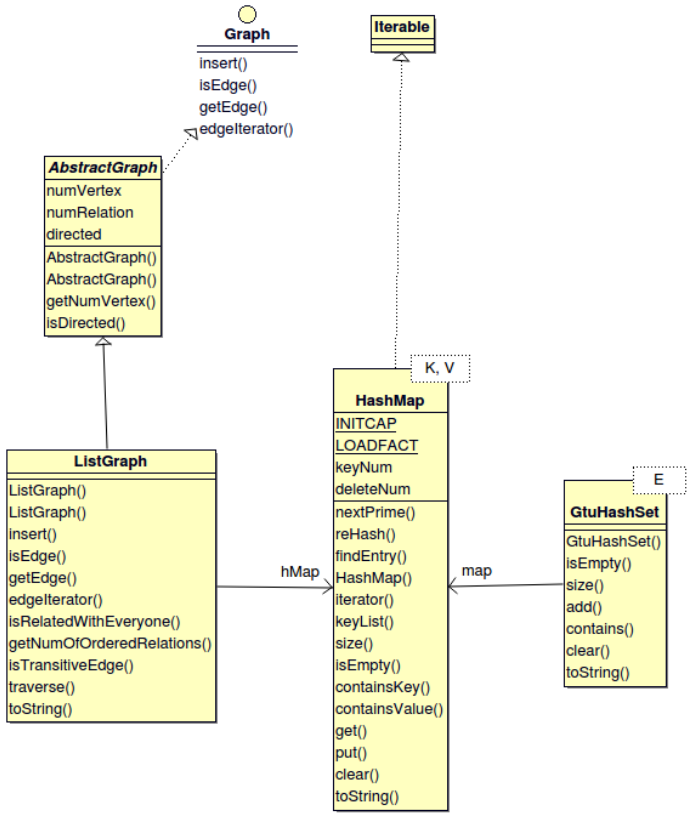
1.2 System Requirements

This program does not require specific piece of hardware,software library or any other plug-in to run. A user which have already installed **Java** into his/her computer, can run this program easily. There is no memory limitation for this program. If the size of the input is getting greater, the run-time of the program can be affect according the CPU speed, or memory size, but this will not exhaust the computer while program running.

2 METHOD

2.1 Class Diagrams





2.2 Use Case Diagrams

User of this program is supposed to write a txt file which contains relations between people. This relation is pair such as x,y that defines “x thinks y is popular”. User can add any number of relations into this file. The first stamement of the input file represents the number of people and number of relations. First number is number of people, second number is number of relations between people. After completing the input txt file, user can run this program by IDE that supports Java management since no runnable (JAR) file has not been generated currently yet. There is no Graphical User Interface designed for this program.

Example input file :

3 3

1 2

2 1

2 3

2.3 Problem Solution Approach

Representing the relations

People can think any number of people as a popular, so there is no limitation for this situation. Linked structure,tree structure etc does not give a property to link people each other with no limitation. Graph structure is useful to link people together according the popularity issue. So graph data structure used for the represent the relations and store the people and manipulate the data.

Designing a graph data structure

To design a graph, firstly *Graph* interface is designed to represent the behaviour of the graph data structure.

To collecting the common implementations and data fields in the Graph hierarchy, *AbstractGraph* class is implemented. By this, we can easily represent the abstraction of the graph terminology. In Part 2.1 Class diagrams, hierarcyh can be easly understand visually.

To implement a concrete and specific graph data structure, *ListGraph* class has been implemented. *ListGraph* class has been implemented by using *HashMap*. This procedure is like implementing graph by adjacency list. But using the HashMap class , is more easy to implement and maniuplation. The performance of the using the *HashMap* inside of the *ListGraph* class is better than using adjacency list or adjacency matrix. In *HashMap*, searching and adding new item is **O(1)** time complexity. So implementing with *HashMap* is effective.

Reading input from txt file

For reading the input txt file which contains relation pairs, *FileInputStream* and *BufferedReader* classes used in constructor of *ListGraph* class to construct the graph according to input txt file.

Construct the graph

In constructor of *ListGraph* class, inputs read by *FileInputStream* and *BufferedReader* objects. After reading a line from the input txt file, this line must be parsed into two

numbers by splitting the space. For splitting the space, split() method defined in String class used. After splitting the space, we have two strings that contains numbers that represents people in this homework. To convert them to integer, Integer.parseInt static method used. By this, two numbers can be easly reachable to adding to graph.

Searching the people who are considered as popular by everyone

Popularity is defined by directly or un-directly in this problem. Directly relation is defined by this syntax : 1,2. This meants that person 1 thinks that person 2 is popular. Un-directly popularity is as known as ordered relation in discrete mathematics. For example, 1,2 and 2,3 are exist, means that people 1 thinks that person 2 is popular, person 2 thinks that person 3 is popular. We have un-directly 1,3 here, means that person 1 thinks undireclty that person 3 is popular.

To find the number of the people who are considered as popular by everyone, we must first traverse the all graph and test if person is popularized by everyone.

To test if the person is considered as popular by everyone, we must traverse all the network of the person. **isRelatedWithEveryone(int number)** method used the check whether the person is considered as popular by everyone or not. This method calls **isTransitiveEdge** method for V-1 times to check all vertices. To narrow down the problem to easily understand it in the English, we can say that : “If there is a path between x and y in the graph,(undirectly or direclty), x thinks that y is popular.”. By looking this approach, the main point is, check whether there is a path between all numbers and specific person. **isTransitiveEdge(int source, int destination)** method checks there is a path between source and destination undirectly or directly. This method, first checks if there is a directly path by **isEdge(int source,int dest)** which is inherited from *Graph* interface. If it is, returns true. If it is not a edge(directly), it is time to check if it is a undirectly relation. To search undirectly relation, recursive **traverse(..)** method used. This method traverse all the network based on breadth-first search algorithm. And this method has **O(|V| + |E|)** time complexity. The method which returns the result after searching calls **isRelatedWithEveryone** method per every vertices in the graph. So the total time complexity is (V*(V-1)*(V+E)) = **O(V^3)**.

3 RESULT

3.1 Test Cases

I test this program with giving different input files with different types of relations. Examples of inputs and results will be given in the 3.2 Runnig Results part.

3.2 Running Results

Input file :

3 3
1 2
2 1
2 3

Result :

```
Number of people who are considered
as popular by everyone :
1
```

Inuput file :

3 3
1 2
2 3
3 1

Result :

```
Number of people who are considered
as popular by everyone :
3
```