Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

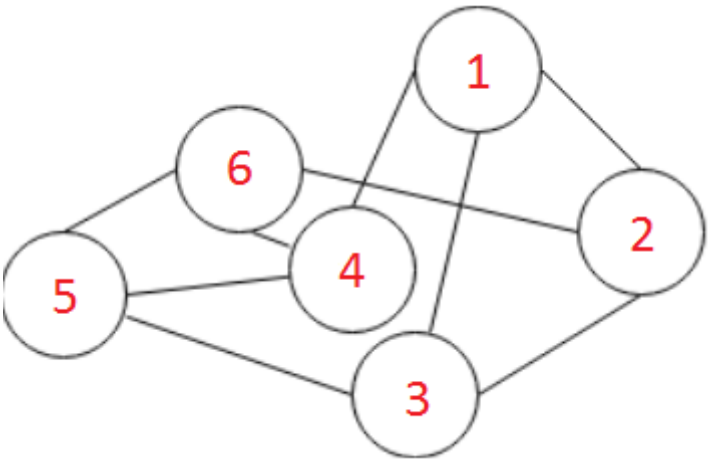
HOMEWORK 8 REPORT

YUSUF PATOGLU
151044027

1 INTRODUCTION

1.1 Problem Definition

We can call our problem as popularity problem. In popularity problem we will try to find the number of people that is known popular by all other persons. There is a condition also. If A thinks B is popular and B thinks C is popular, A will think C is popular too. This is called transitivity. If A thinks C is popular then A can think someone is popular which also known popular for C. So in our problem we can't simply say if A-B and B-C and A-C. This is true but there can be another relations therefore we need to use a special data structure like **graph.**



The graph will look like this. If relations are different connections will be different of course. In this example we have 6 people and all must have numbers like 1,2,3,4,5,6. We will try to connect these nodes with the help of graph.

1.2 System Requirements

This is a light project it doesn't make so many calculations. So default JVM settings will be enough.

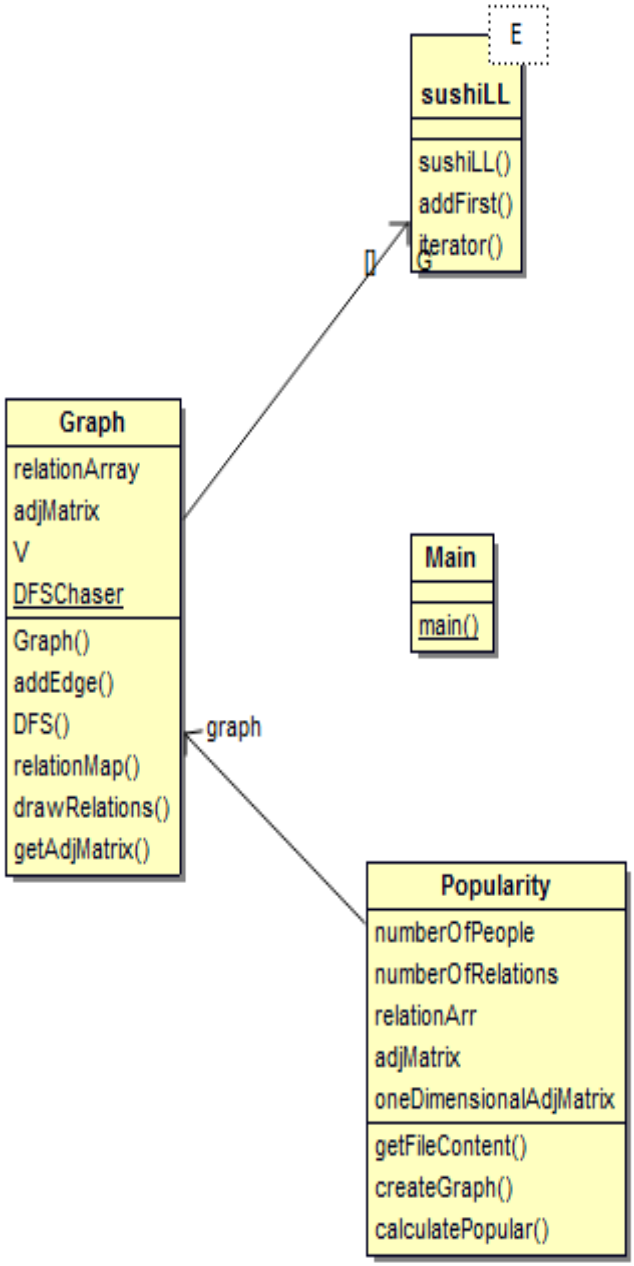
Just in case JVM system requirements are noted here on Java's website:

- Minimum memory 1 GB
- Recommended memory: 2 GB for Windows platforms, 1 GB for non-Windows platforms.
- Minimum disk space: 250 MB
- Recommended disk space: 500 MB

2 METHOD

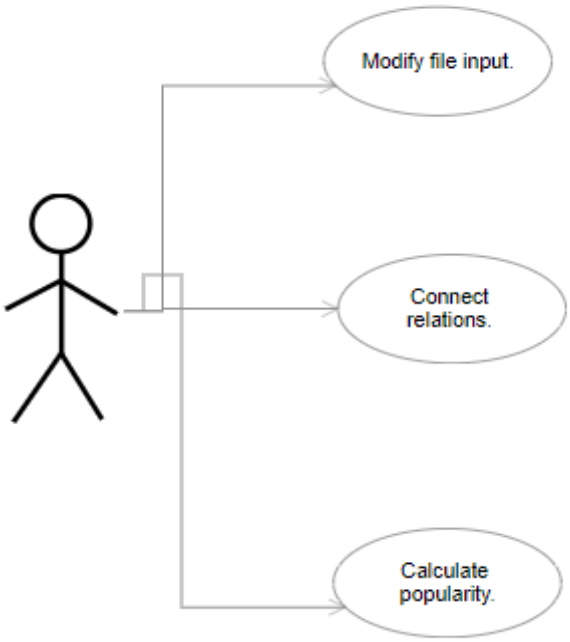
2.1 Class Diagrams

Graph holds linkedlist object for creating adjacency list. Popularity class holds Graph object for creating relations between people



2.2 Use Case Diagrams

User can use create Popularity object then, can get file content, create graph in order to draw relations and calculate the popular people.



2.3 Problem Solution Approach

In order to solve our problem like I said in problem definition, I used graph. There were two ways to implement graph data structure, adjacency matrix and adjacency list. I choosed adjacency list because I found traversing easier in adjacency list. I implemented my Graph data structure with adjaceny list so. In order to implement my graph data structure I needed a linked data structure. Therefore I implemented a linkedlist named with SushiLinkedList. Since I will using this linkedlist in order to connect relations inside graph. I only needed addfirst method and iterator. For example my linkedlist doesn't have addLast method. Because addFirst costs O(1) time complexity but addLast costs O(n) time complexity. Therefore I choosed to implement addFirst method. Linkedlist addFirst complexity O(1)

Graph class constructor constructs number of people times linkedlists. So if there are n people, space complexity will be O(n). Graph method as DFS method that performs the depth first traversal on list. Since we're visiting all the elements, the recursive call stack causes O(n) space complexity. Also O(n) time complexity. addEdge method of graph has O(1) time complexity because it only adds element to head of linkedlist. Relation map sets a boolean table for DFS operation. Therefore it takes O(n) space complexity. Draw relations creates 2D array to backup DFS traversal data. Therefore it takes O(number of people * number of people) space complexity. Also have two seperate loops that takes O(n) time complexity.

Popularity class' getFileContent stores all the file input in an array named with relationArray. Therefore space and time complexity is O(n). Create graph method of Populariy class takes O(n) space complexity and takes O(number of people * number of people) time complexity to store 2d array into 1d array. Calculate popular method of Popularity class takes takes O(number of people * number of people * number of people) time complexity and O(1) space complexity.

The crucial point of this project is calculating how many people is find popular by all other people. The solution was like this:

1- DFS the list.
2- While traversing the list store all of the visited nodes in a 2D array
3- Copy contents of 2D array to another 1D array
4- Count the occurences of each element if same element occurs number of people times than that means that people found popular by other people. Simply increment the popular counter.
5- After loop ends return the popular counter.

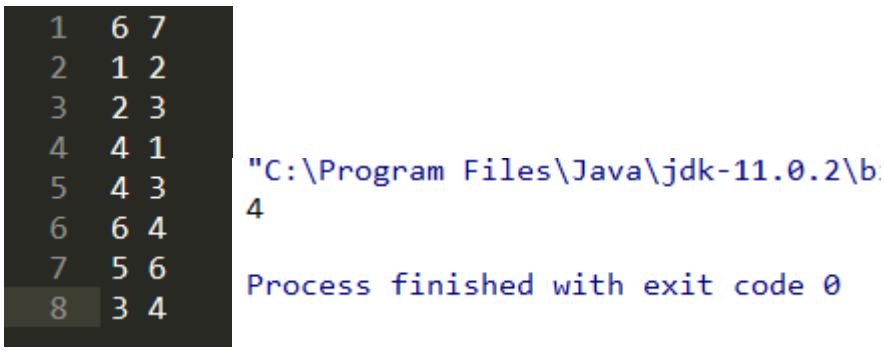
3 RESULT

3.1 Test Cases

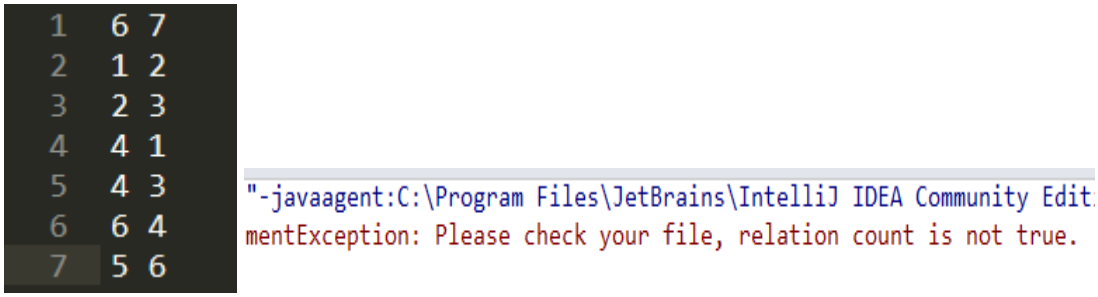
I tested my program with several inputs, while trying different scenarios I realised that I should throw an exception if file input is wrong. For example if first line has 3 3 which means 3 people and 3 relation. But file contains only 2 relation, I throwed exception here like "Check your file."

3.2 Running Results

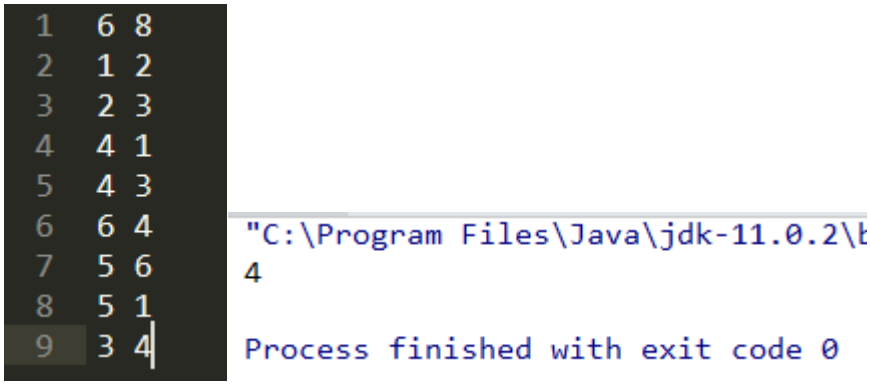
Input file 1: Output 1:



Input file 2: Output 2:



Input file 3: Output 3:



Input file 4: Output 4: