

**Gebze Technical University
Computer Engineering**

CSE 222 - 2019 Spring

HOMEWORK 8 REPORT

**ÖMER ÇEVİK
161044004**

Course Assistant: Ayşe Şerbetçi TURAN

1 INTRODUCTION

1.1 Problem Definition

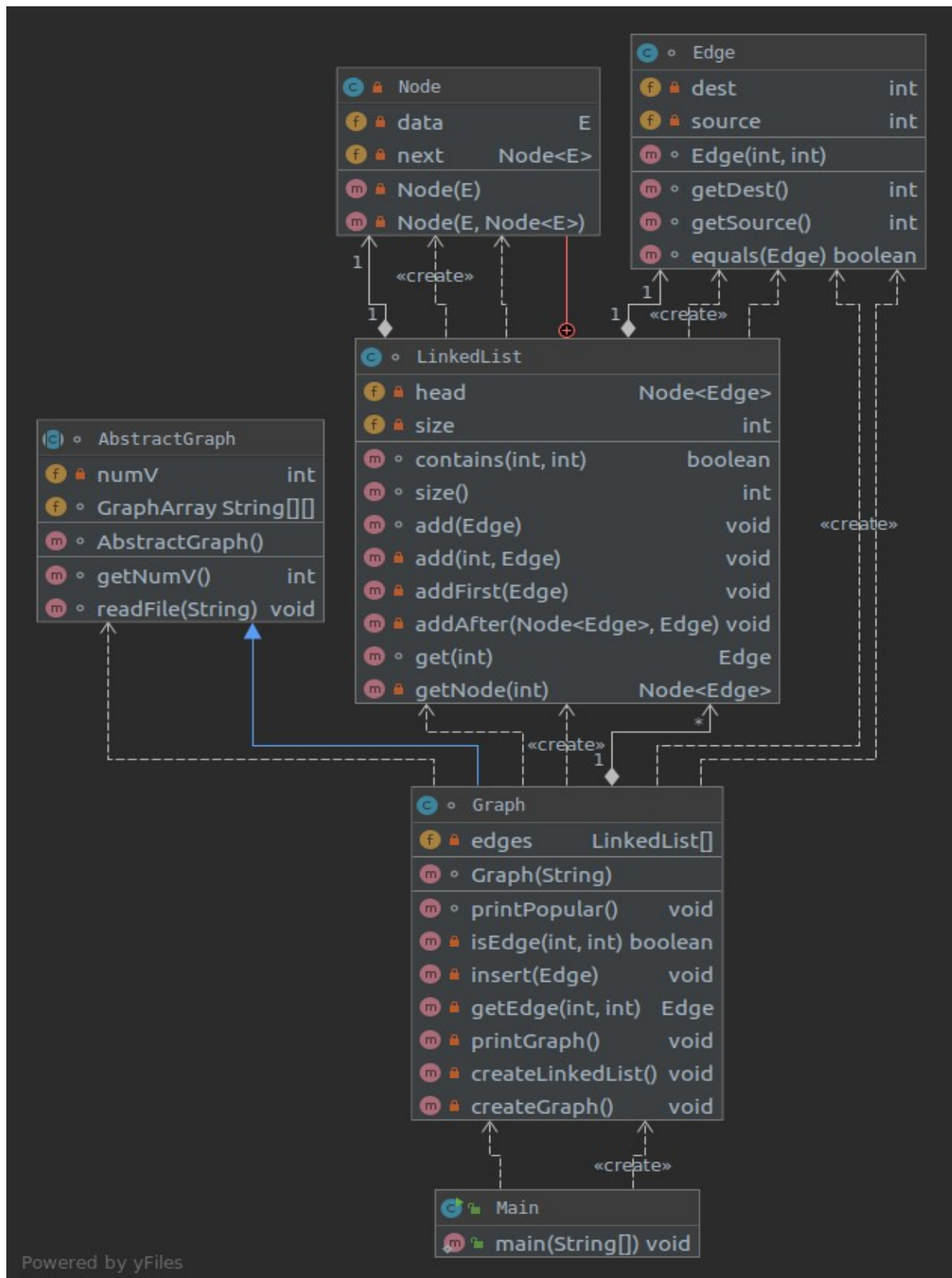
In that programme, there are given some datas in file. That datas are two space-separeted and in first line first data is node size second data is relation size. The wanted is that if a node has a relation to another node and the relationed node has another relation to another node you must create a relation between that first and third nodes. After all are done, you must find the popularity of each node. How many popular nodes are available find it.

1.2 System Requirements

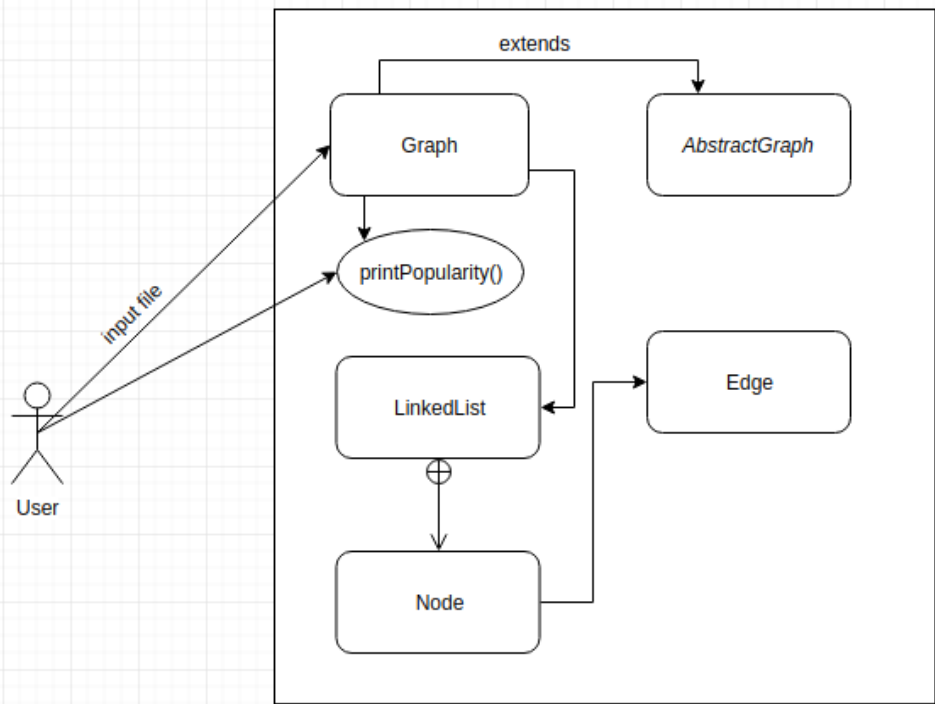
My solution doesn't require a specific piece of hardware, or maybe a certain minimal amount of memory, or a certain operating system, or a software library to be installed in order to run properly. It built by IntelliJ IDEA and used java programming language. It might run correctly if you use Java Virtual Machine and JDK is at least 8. Will my program work with 128KB of memory? I don't know, I didn't work with 128KB of memory but maybe it can work. If your nephew's smartphone's operating system is Android then it must run on it.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams



2.3 Problem Solution Approach

In that programme, the graph structure solves the problem. I used adjacency linked list to create a graph. There is a LinkedList array in graph. LinkedList keeps Node objects. The Node class is a generic class. It keeps the next object and data. Graph class extends the abstract graph named AbstractGraph. In AbstractGraph I read the file into a 2D string array. After that all datas are reserved in array. Then reading the datas from that array into file using insert() method. After I check the links that if P1 relations P2 and P2 relations P3 then P1 relations P3. Then insert it into graph. While searching the P1 – P3 relation I needed to move on graph two times. After that all My graph is ready. Finding the popularity in that graph is worked like that: created an integer array sized graph's linkedlist array size. Then I made a loop in graph. While moving on graph, I increment the array which index points to destination of edge. Then I make a loop to find the maximum popular. I make the last loop in that array to find how many popular edges in that graph. And print it to screen.

Time Complexity

AbstractGraph Class

- ◆ AbstractGraph() : No complexity.
- ◆ getNumV() : Returns numV, $O(1)$.
- ◆ readFile() : Reads the file into array, $O(n)$.

Edge Class

- ◆ Edge() : Initialize dest and source, $O(1)$.
- ◆ getDest() : Returns dest, $O(1)$.
- ◆ getSource() : Returns source, $O(1)$.
- ◆ equals() : Checks the dests and resources, $O(1)$.

Node Class

- ◆ Node() : Initialize data and next object, $O(1)$.

LinkedList Class

- ◆ contains() : Checks if edge is in graph, $O(n)$.
- ◆ size() : Returns size, $O(1)$.
- ◆ add() : Adds node in linkedlist, $O(1)$.
- ◆ addFirst() : Adds node on front, $O(1)$.
- ◆ addAfter() : Adds node on back, $O(1)$.
- ◆ get() : Calls getNode() and traverses the linkedlist, $O(n)$.
- ◆ getNode() : Traverses the linkedlist, $O(n)$.

Graph Class

- ◆ Graph() : Reads file, creates linkedlist, creates graph, $O(n^2)$.
- ◆ printPopular() : Traverse graph, $O(n)$.
- ◆ isEdge() : Calls contains(), $O(n)$.
- ◆ insert() : Calls add(), $O(1)$.
- ◆ getEdge() : Traverses graph, $O(n)$.
- ◆ printGraph() : Prints Graph, $O(n)$.
- ◆ createLinkedList() : Initialize linkedlist, $O(n)$.
- ◆ createGraph() : Reads into graph datas and finds the relations, $O(n^2)$.

Space Complexity

AbstractGraph Class

- $4 + 2 \times \text{numV}$

Edge Class

- $4 + 4 = 8$

Node Class

- $E + \text{Node}$

LinkedList Class

- $4 + \text{Node} = 4 + E + \text{Node}$

Graph Class

- $\text{numV} \times \text{LinkedList} = \text{numV} \times (4 + \text{Node}) = \text{numV} \times (4 + E + \text{Node})$

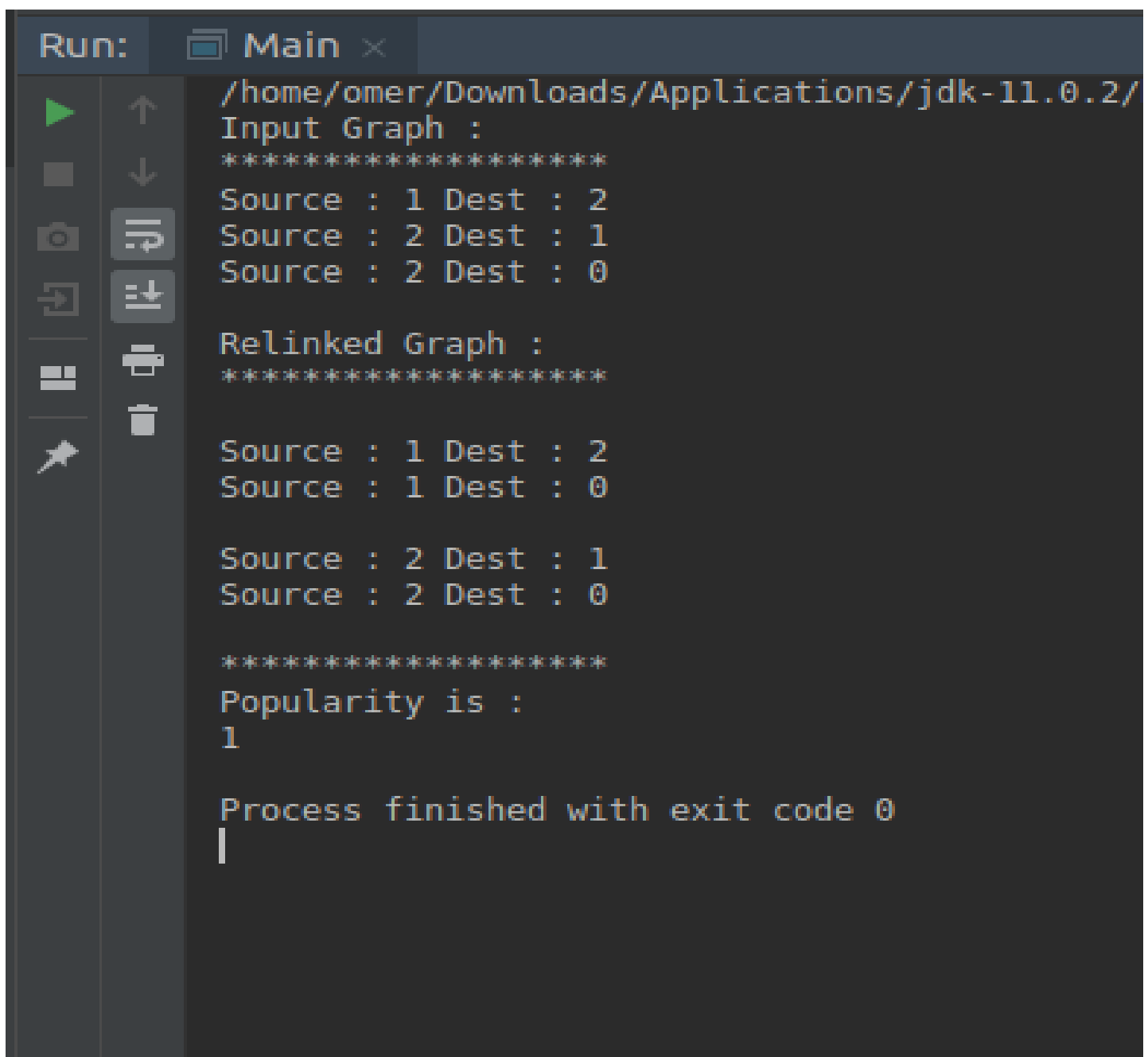
3 RESULT

3.1 Test Cases

Main Test

I created a driver class that called Main. Then I created the object of graph. After that I called printPopularity() method to calculate popularity. Results are in **3.2. Running Results**.

3.2 Running Results



```
Run: Main x
/home/omer/Downloads/Applications/jdk-11.0.2/
Input Graph :
*****
Source : 1 Dest : 2
Source : 2 Dest : 1
Source : 2 Dest : 0

Relinked Graph :
*****

Source : 1 Dest : 2
Source : 1 Dest : 0

Source : 2 Dest : 1
Source : 2 Dest : 0

*****
Popularity is :
1

Process finished with exit code 0
|
```