Gebze Technical University
Computer Engineering

CSE 222 – 2019 Spring

HOMEWORK 8 REPORT

RUMEYSA KARAKAVAK
141044063

Course Assistant:Ayse Serbetci Turan

1 INTRODUCTION

1.1 Problem Definition

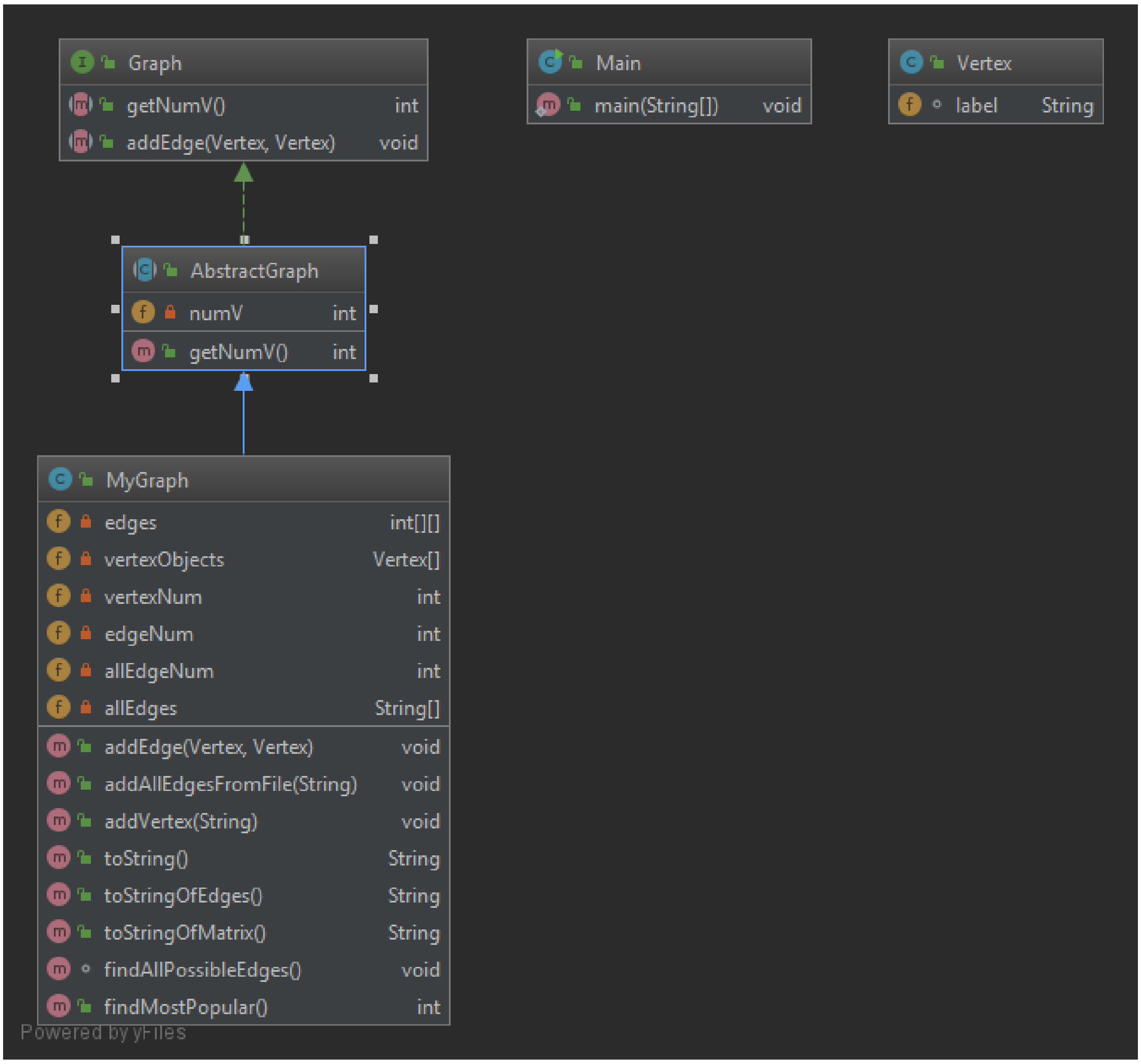
This homework for implementing graph data structure. You have vertices and edges between them that hold in an input file. You must find another edges between vertices cause of transitivi attribute. Then create an adjacency matrix that includes 0s and 1s. If there is a way from a vertex to another vertex, it represents 1 in matrix. After initialize that matrix, you should find most popular vertex. This means vertex which have maximum count of visited. If there is same max count of visited in vertices, output will be increase.

1.2 System Requirements

This program was coded and can running on IntellijIDEA(it can runs another platforms probably). Used latest version of JDK 11.0.2. .
Operating System (i.e. Windows 10, Linux or Mac OS X 10.3.8)
Processor Speed (i.e. Pentium 4, 3.2 GHz or Power PC G5, 2.0 GHz)
Memory, a.k.a. RAM (i.e. 512 MB)
Hard Disk Space (i.e. 80 GB available)

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

This software allows to user creating graphs accorfing an input file that includes edges and vertices. He/She should create an input file and create an graph object with this file.

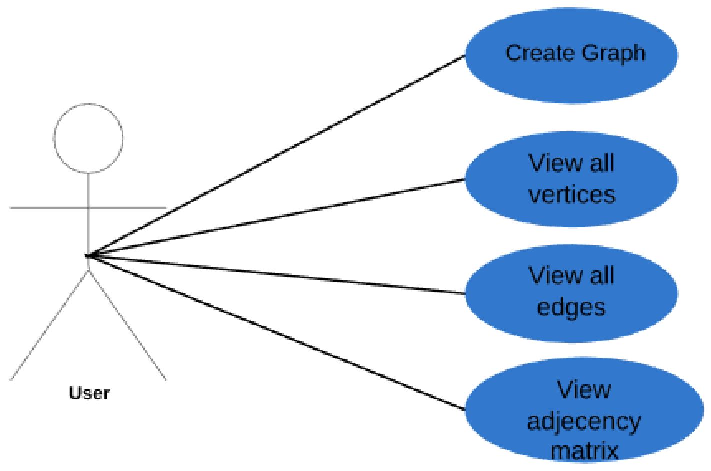
Then creating graph user allows to printing this grapsh’s edges, vertices and adjacency matrix.

The other methods are private for user.

MyGraph directedGraph = new MyGraph(fileName)

fileName : includes vertices and edges.

MyGraph : constructor for graph that takes file as an argument.
toString() : that gives all vertices in graph.
toStringOfEdges(): that gives all edges in graph.
toStringOfMatrix(): that gives matrix that contains 1s and 0s according to edges.



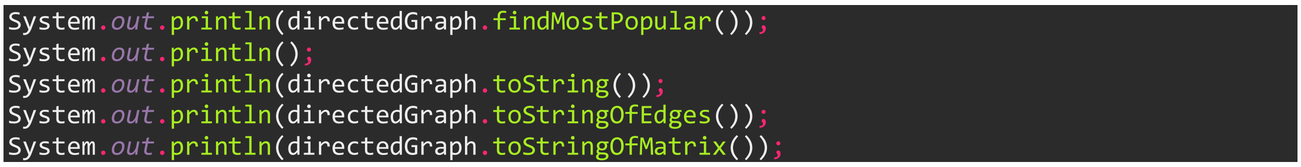
2.3 Problem Solution Approach

I chosed array primitive type for implementing graph in this homework. I created a new class that name is Vertex. This class have label member. For holding vertex objects, i used Vertex array. For holding edges of this graph, used string array.
For find transitive edge, graph traversed with depth first search method by algorithm. My graph is implemented as an adjacency matrix (a V x V array), then, for each node, i have to traverse an entire row of length V in the matrix to discover all its outgoing edges. So the complexity of DFS is **O(V * V) = O(V^2)**.
After finding all edges in graph, created 2D integer array for keep edges in a matrix.
In this way user can reach an element of graph with **O(1)** constant time.
For finding popular vertex count, check all matrix and its complexity is **V*V -> O(V^2).**

3 RESULT

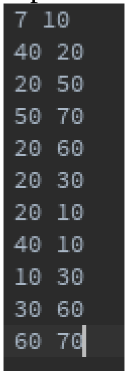
3.1 Test Cases

```
String fileName = args[0];
MyGraph directedGraph = new MyGraph(fileName);
System.out.println();
System.out.print("Popular vertex count is : ");
```



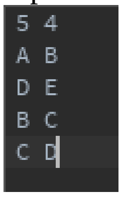
```
System.out.println(directedGraph.findMostPopular());
System.out.println();
System.out.println(directedGraph.toString());
System.out.println(directedGraph.toStringOfEdges());
System.out.println(directedGraph.toStringOfMatrix());
```

Input file 1



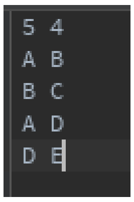
```
7 10
40 20
20 50
50 70
20 60
20 30
20 10
40 10
10 30
30 60
60 70
```

Input file 2



```
5 4
A B
D E
B C
C D
```

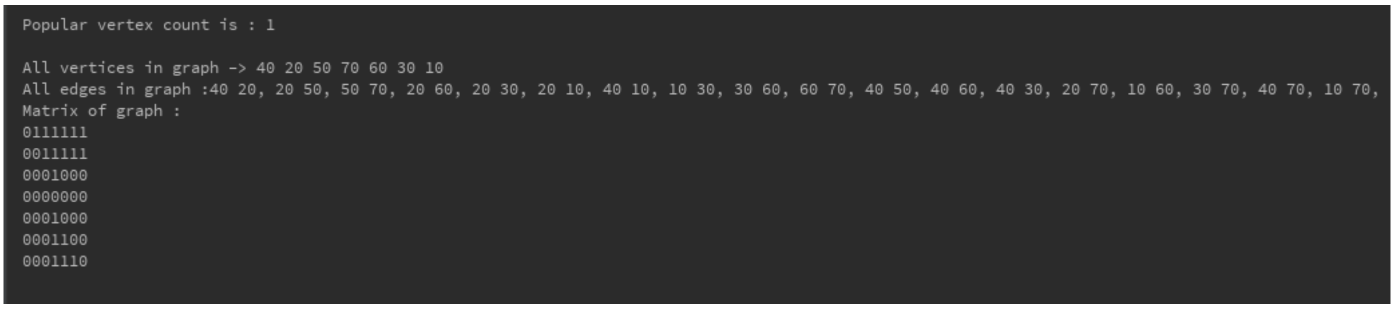
Input file 3



```
5 4
A B
B C
A D
D E
```

3.2 Running Results

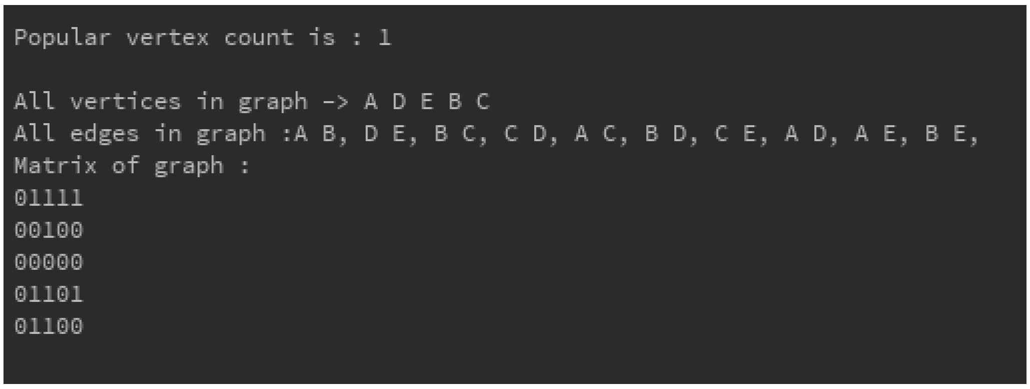
Output 1



```
Popular vertex count is : 1

All vertices in graph -> 40 20 50 70 60 30 10
All edges in graph :40 20, 20 50, 50 70, 20 60, 20 30, 20 10, 40 10, 10 30, 30 60, 60 70, 40 50, 40 60, 40 30, 20 70, 10 60, 30 70, 40 70, 10 70,
Matrix of graph :
0111111
0011111
0001000
0000000
0001000
0001100
0001110
```

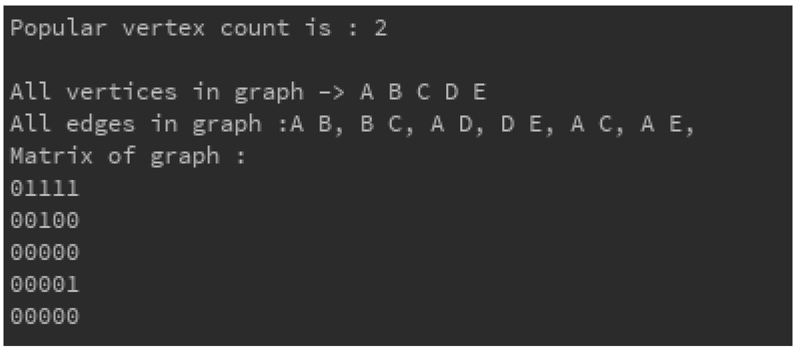
Output 2



```
Popular vertex count is : 1

All vertices in graph -> A D E B C
All edges in graph :A B, D E, B C, C D, A C, B D, C E, A D, A E, B E,
Matrix of graph :
01111
00100
00000
01101
01100
```

Output 3



```
Popular vertex count is : 2

All vertices in graph -> A B C D E
All edges in graph :A B, B C, A D, D E, A C, A E,
Matrix of graph :
01111
00100
00000
00001
00000
```