**Gebze Technical University**
**Computer Engineering**

**CSE 222 - 2019 Spring**

# HOMEWORK 8 REPORT

**GÖKHAN HAS**
**161044067**

Course Assistant: Ayşe Şerbetçi TURAN

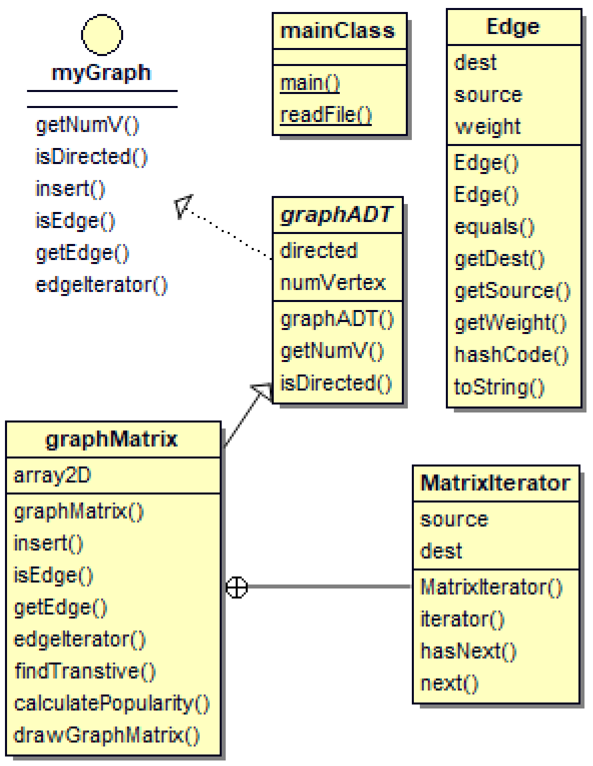
# 1 INTRODUCTION

## 1.1 Problem Definition

In this assignment, it is required to find the popular number of people using the graph structure. This requires a graph implementation. In this graph, the transitive ones must be found as extra. There's no way to be given in the file. Popularity is as follows: There is one source and one destination. Fx, 1 and 2. One thinks the two is popular. In other example, two thinks three is popular. Using the transition feature, indirectly, 1, thinks 3 is also considered to be popular. To keep this source and destination, we need to create a separate class called Edge.

# 2 METHOD

## 2.1 Class Diagrams



.

## 2.2 Problem Solution Approach

In this assignment, I used four classes and an interface to solve problems.

In the Edge class, there is a source, destination and a weight named variable if it is to be weighted. Constructor is just assigning and has O (1) time complexity. **I could override the equals and hashCode methods (HashMap,LinkedList etc.), since many different implications could be performed.** The equals method looks at whether the object reference that was originally submitted is Edge. If not return false. If the Edge reference is true or false according to the equality of its source and destination (O (1)). The hashCode method uses a toString of source and dest, and takes the hashCode of the result. O (1). Get methods in other methods. In them are (1).

myGraph is an interface. I've written to the Object Orianted principle to be fully appropriate. I think I get a better image in the Class diagram. At least it's more readable. As it is known, only methods are written and no implemantation is done in interfaces. So there is no need to calculate time complexity.

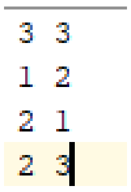
graphADT is an abstract class. It has two variables. One is holding the vertex number and the other is holding whether it is directed or not. There is a constructor and two get methods. It's all O (1). Classes derived from this class will now decide what to do with the implementation.

graphMatrix class where all operations are performed. As the name implies, I have implemantated the graph using a two-dimensional array. I've decided it's boolean. This class is derived from the graphAdt abstract class. I have to implement insert, isEdge, getEdge, Iterator methods in this class which I don't implement in graphAdt class. Constructor calls the constructor of graphAdt and then creates array2D. These operations O (1). The insert () method takes a Edge reference as a parameter. And I have to write -1 because arrays in indexes start from 0. These processes are also in constant time.O (1). The isEdge () method also takes source and destination parameters. IndexOutOfBoundsException throws if exiting the matrix. Returns true if it was previously added to the source and destination. O (1). The getEdge () method returns only Edge, such as the isEdge () method, otherwise returns null. O (1). In this class there is a separate class called MatrixIterator. According to the given source iterator is being used. The hasNext () method can be up to a vertex number. So time complexity is O (V).

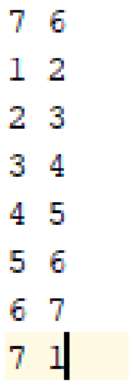
I wrote a method called findTranstive (). As his name suggests, he also finds and signs transtive ones. But it's a costly method. Time complexity is O (V ^ 3). Because there are three for loops (inner). This method will be used as a private method to help in another method. The calculatePopularity () method calls the findTranstive() method. So it becomes O (V ^ 3). Normally it was O (V ^ 2). Person, I do things by thinking that she/he is not popular. (1,1),(2,2).. vs not included. In the drawGraphMatrix () method, I print the matrix on screen. Time Complexity is O (V ^ 2).

## 3 RESULT

### 3.1 Test Cases

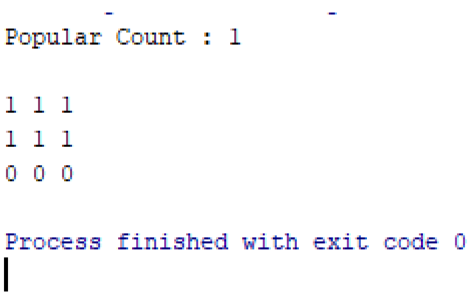


```
3 3
1 2
2 1
2 3
```



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

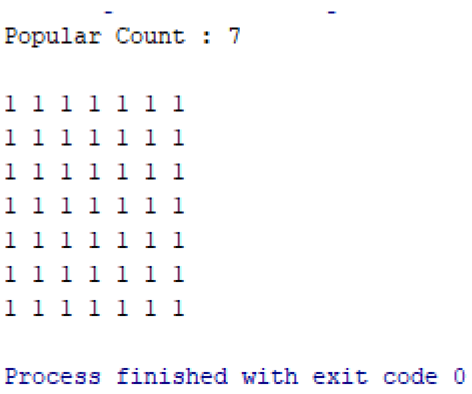
### 3.2 Running Results



```
Popular Count : 1

1 1 1
1 1 1
0 0 0

Process finished with exit code 0
```



```
Popular Count : 7

1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1

Process finished with exit code 0
```