Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 8 REPORT

EMRE KAVAK
151044085

Course Assistant:

1 INTRODUCTION

1.1 Problem Definition

In this home work we have to calculate most popular people according to given input.txt file. The problem was how many people thing each other popular and how mand people are popular in this group. For hold relationships between people we have to use own graph ADT and implement its methods. We should select methods between adjacency matrix, adjacency list or other methods. We didnt allowed to use java collections.

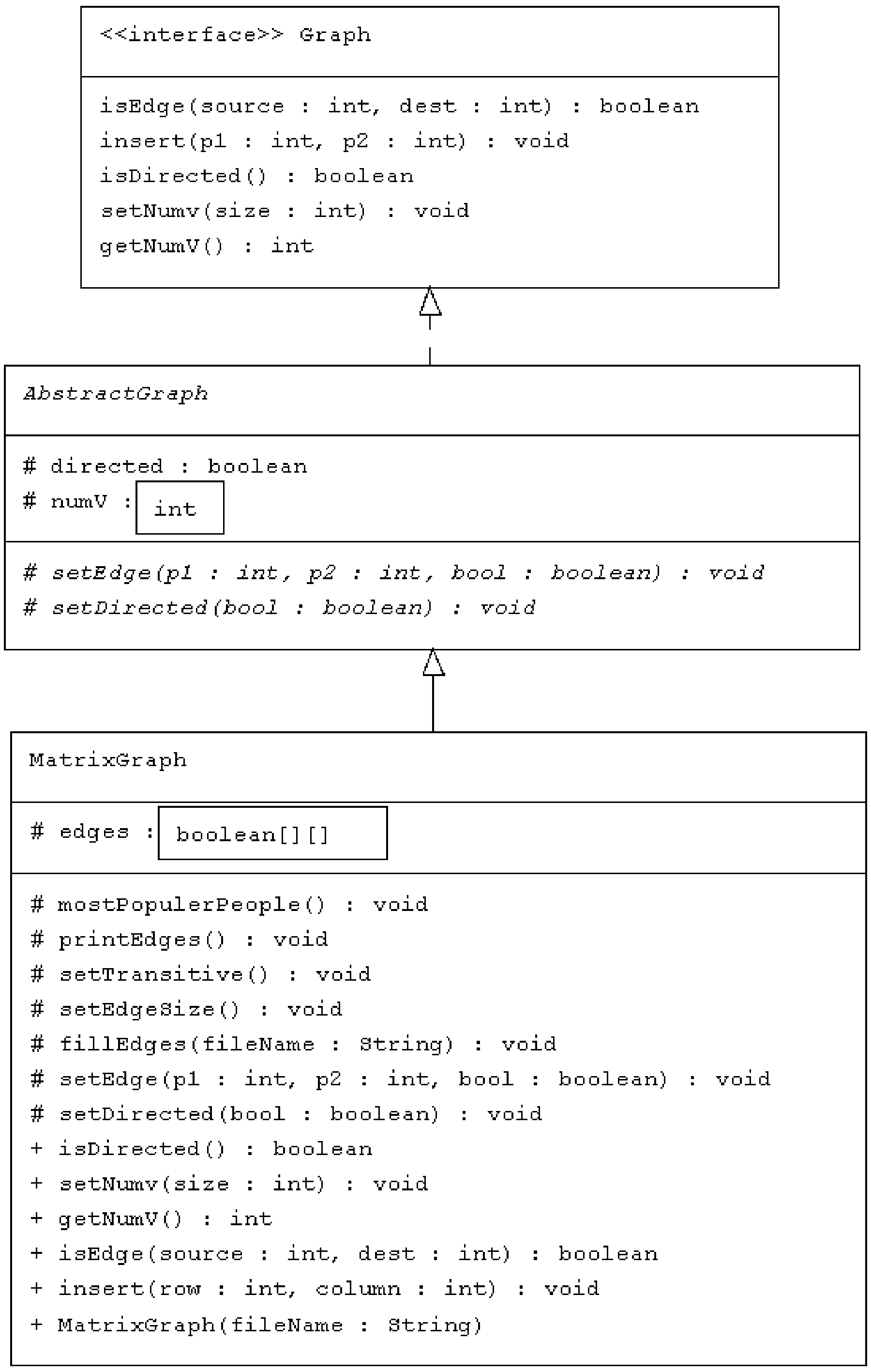
1.2 System Requirements

This program run on intelliJ IDE. I mean, Project file will be intelliJ file. So, for run this program there must be intelliJ IDE. So, you can just run this program on computer or laptop. Also **you have to give parameter full input.txt path in the created main class object**. After that, you just click run and you will see and integer number on the standart output.

IntelliJ IDE Reguirements : Minimum 2GB Ram, At least 2 GB harddisk space,Windows operating System.

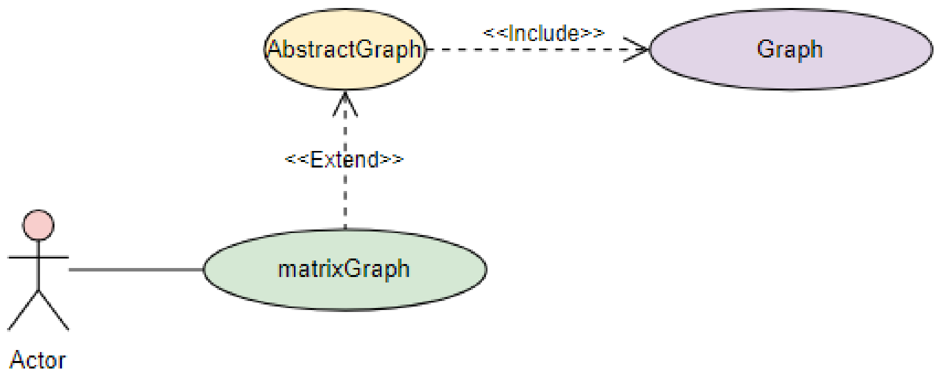
2 METHOD

2.1 Class Diagrams



<<interface>> Graph
isEdge(source : int, dest : int) : boolean
insert(p1 : int, p2 : int) : void
isDirected() : boolean
setNumv(size : int) : void
getNumV() : int
AbstractGraph
# directed : boolean
# numV : int
# setEdge(p1 : int, p2 : int, bool : boolean) : void
# setDirected(bool : boolean) : void
MatrixGraph
# edges : boolean[][]
# mostPopulerPeople() : void
# printEdges() : void
# setTransitive() : void
# setEdgeSize() : void
# fillEdges(fileName : String) : void
# setEdge(p1 : int, p2 : int, bool : boolean) : void
# setDirected(bool : boolean) : void
+ isDirected() : boolean
+ setNumv(size : int) : void
+ getNumV() : int
+ isEdge(source : int, dest : int) : boolean
+ insert(row : int, column : int) : void
+ MatrixGraph(fileName : String)

2.2 Use Case Diyagram



2.3 Problem Solution Approach

Java doesn’t have graph ADP. So, I created my own interface **graph** and **AbstractGrap** class. This class implement graph interface and has methods. AbstractGraph class holds number of edges in the data field **numV** and **directed** data member for hold this graph directed or not. (This graph will directed because of people relations). And, the last class is **matrixGraph**. This class extends from **AbstractGrap** class. I used to **Adjacency matrix** because of this method worst case complexity same as Adjacency list (according to our book chapter 12) and this method implementaion is easy. Our teacher always say take the easy one and implement it if it same as other according to complexity. I just have to hold boolean two dimentional array and number of edges.

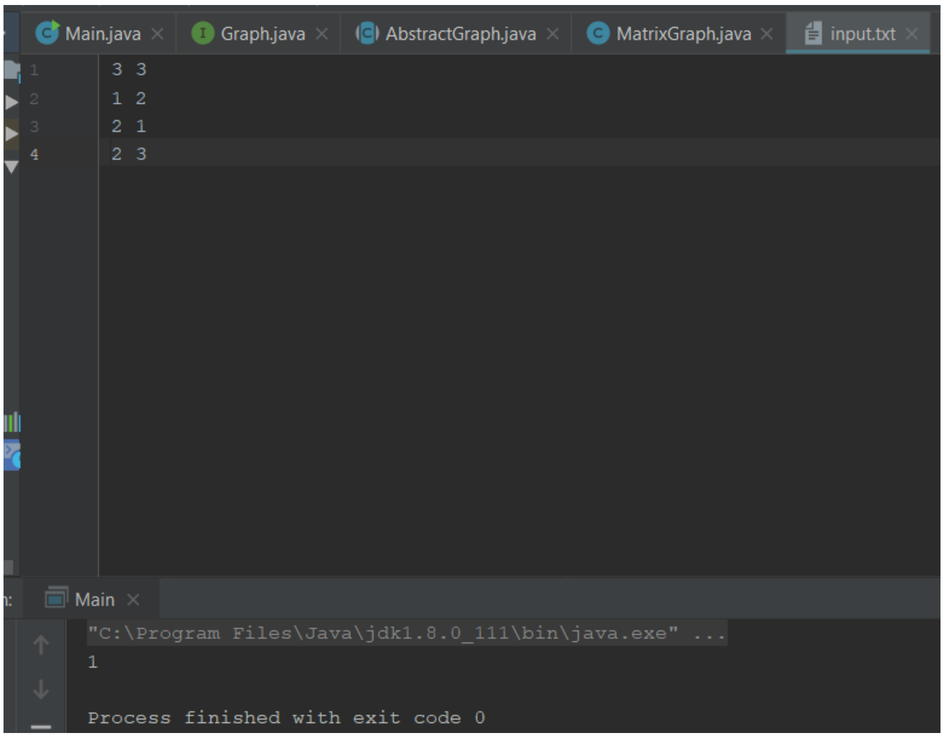
I read **input.txt** in main.java class. I created an matrixGraph object and put full file path this object constructer. This constructor send file name into fillEdges method and this method take memory allocation for two demontional array edges according to given number of people. When I take allocation, I fill all array false. İf two vertices (people) have an edge (relation), I just should change false to true. I set number of vertices (numV) and fill all array with given relation. I do it in while loop and this loop will continue according to given relations size. After that, I senkronize the array according to transitive rules in the given pdf.

I used to **breadth – first search algorithm** . Because of I have to calculate edges between given to people and for do this, breadth – first search better than others. Because of We must visit all nodes for which the shortest path from the start node is length *k* before we visit any node for which the shortest path from the start node is length *k+1*. For calculte most popüler people, I search all columns and calculate all trues. İf there is an false, I set max with 0. İf max will reach vertices size - 1, I increace popular people count. Finally, I printed popular people count

3 RESULT

3.1 Test Cases

Test 1

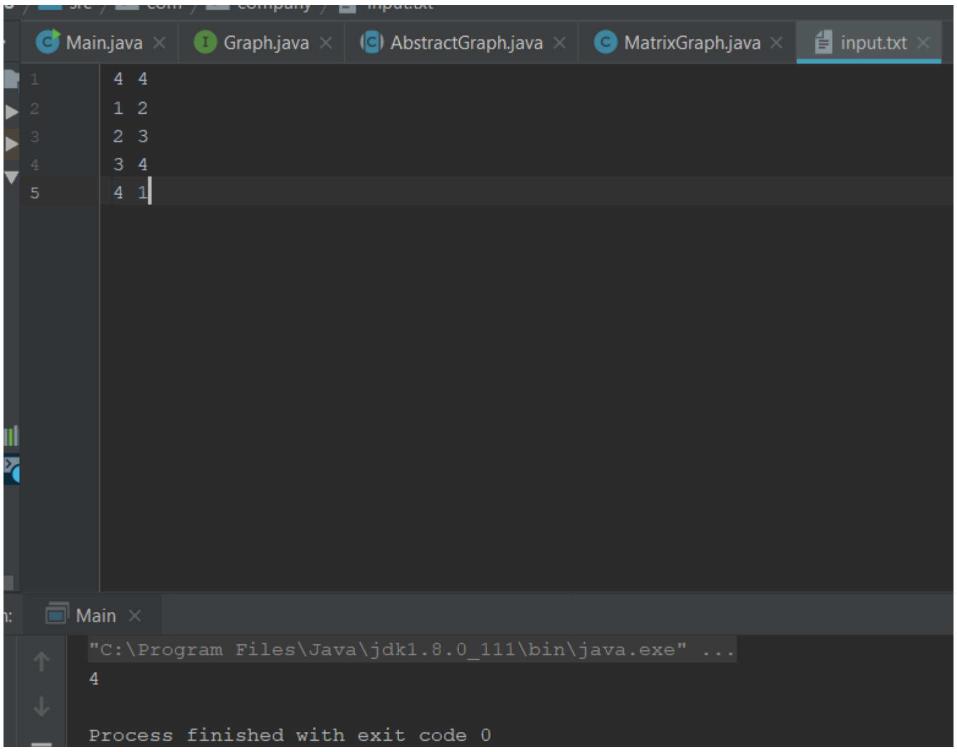


```
3 3
1 2
2 1
2 3
```

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
1

Process finished with exit code 0
```

Test 2

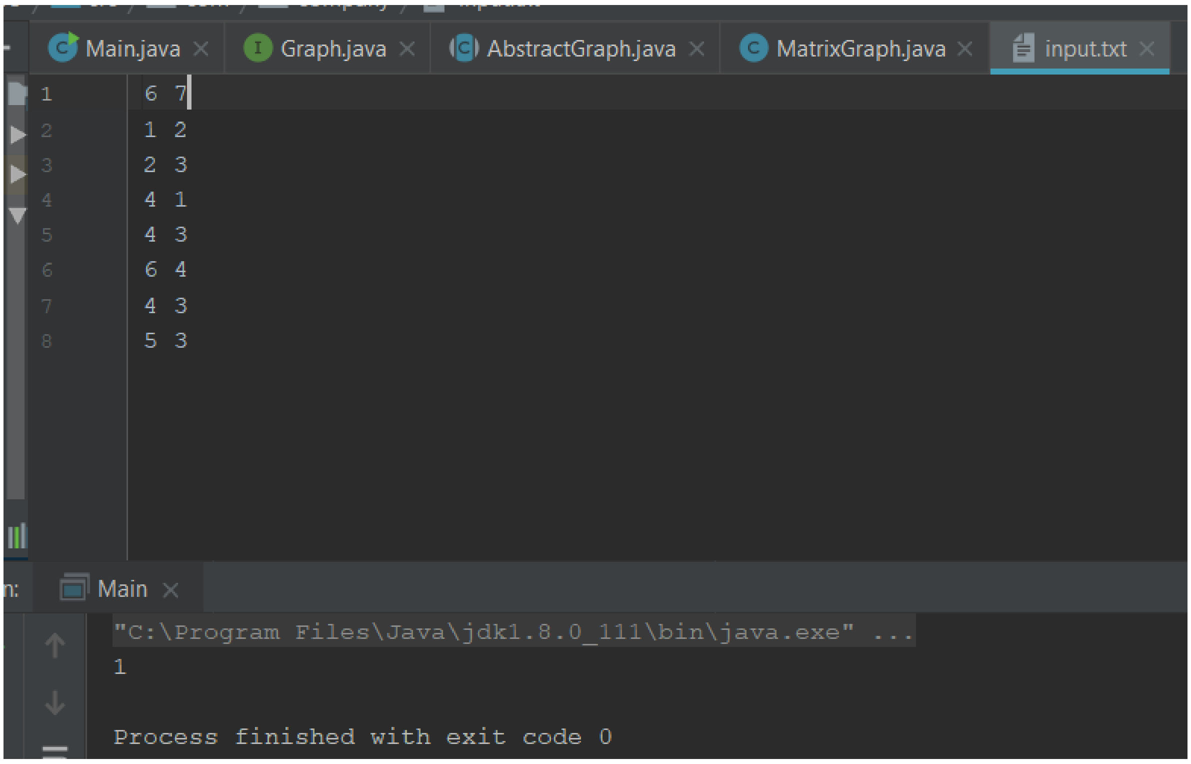


```
4 4
1 2
2 3
3 4
4 1
```

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
4

Process finished with exit code 0
```

Test 3



Main.java Graph.java AbstractGraph.java MatrixGraph.java input.txt

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

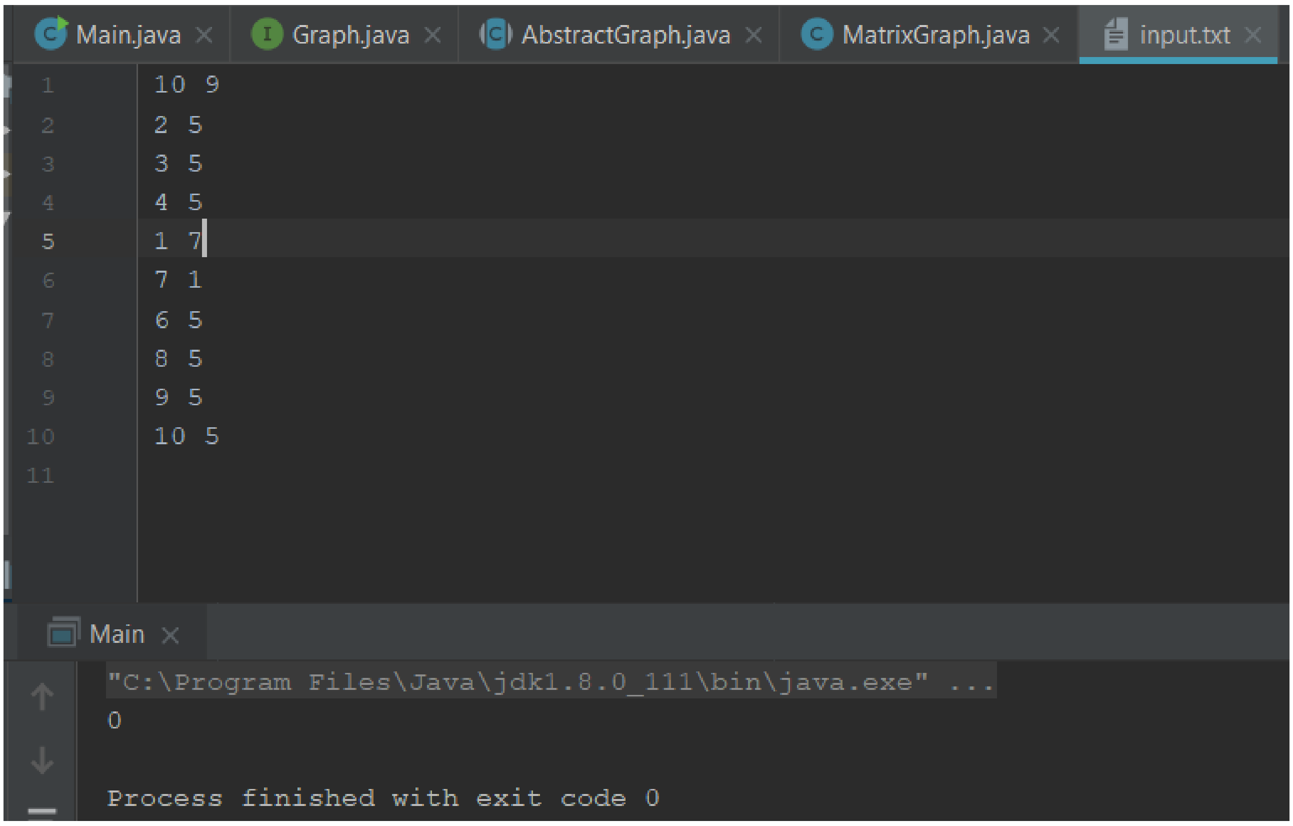
Main

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
1

Process finished with exit code 0
```

Test 4

Result is 0 because of all people should think famuous if a person will popular.



Main.java Graph.java AbstractGraph.java MatrixGraph.java input.txt

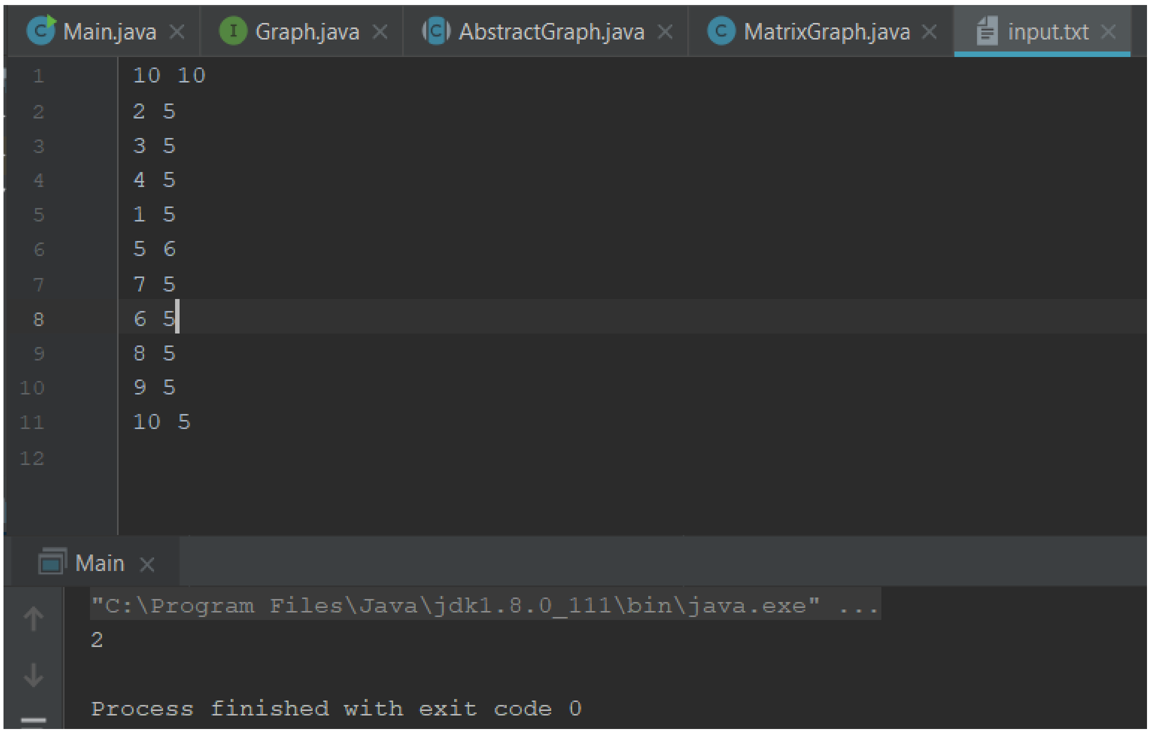
```
10 9
2 5
3 5
4 5
1 7
7 1
6 5
8 5
9 5
10 5
```

Main

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
0

Process finished with exit code 0
```

Test 5

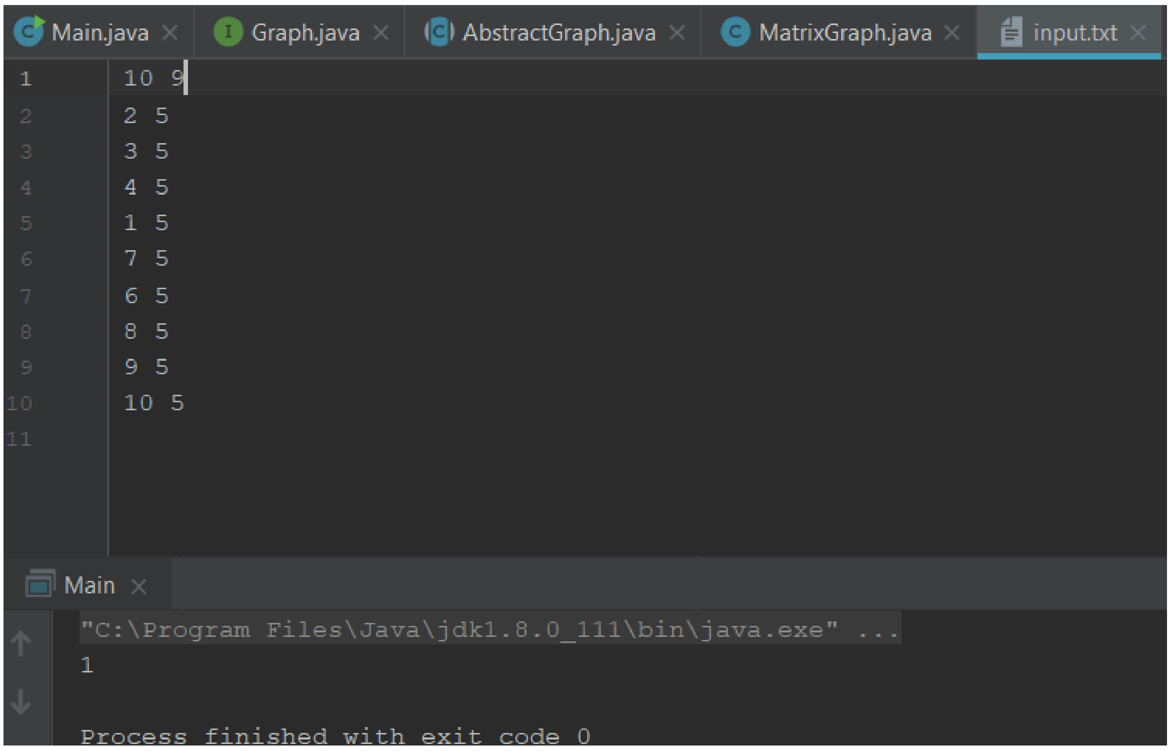


```
10 10
2 5
3 5
4 5
1 5
5 6
7 5
6 5
8 5
9 5
10 5
```

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
2

Process finished with exit code 0
```

Test 6



```
10 9
2 5
3 5
4 5
1 5
7 5
6 5
8 5
9 5
10 5
```

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
1

Process finished with exit code 0
```

3.2 Complexitys

3.2.1 Graph methods

- **int getNumV();** O(1)
- **void setNumv(int size);** O(1)
- **boolean isDirected();** O(1)
- **void insert(int p1,int p2**) O(1) , Because of I didnt search anyhing
- **boolean isEdge(int source, int dest)** O(1), Because of I didnt search anyhing

3.2.2 AbstractGraph Methods

- **abstract protected void setDirected(boolean bool);** O(1)
- **abstract protected void setEdge(int p1,int p2, boolean bool);** O(1)

3.2.3 MatrixGraph Methods

* **public void insert(int row, int column)** O(1)

* **public boolean isEdge(int source, int dest)** O(1)

* **public int getNumV()** O(1)

* **public void setNumv(int size)** O(1)

* **@Override**
public boolean isDirected() O(1)

* **@Override**
protected void setDirected(boolean bool) O(1)

* **@Override**
protected void setEdge(int p1, int p2, boolean bool) O(1)

* **protected void fillEdges(String fileName)**
O(logn) Because of I search all input line and there are 2 element in the line

* **protected void setEdgeSize()** O(1)

* **protected void setTransitive()**
O(V+E) because of I search all vertices and I check all Edges

* **protected void mostPopulerPeople()**
O(E) Because of I check all Edges