

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 7 REPORT

**Burak DEMİRCİ
141044091**

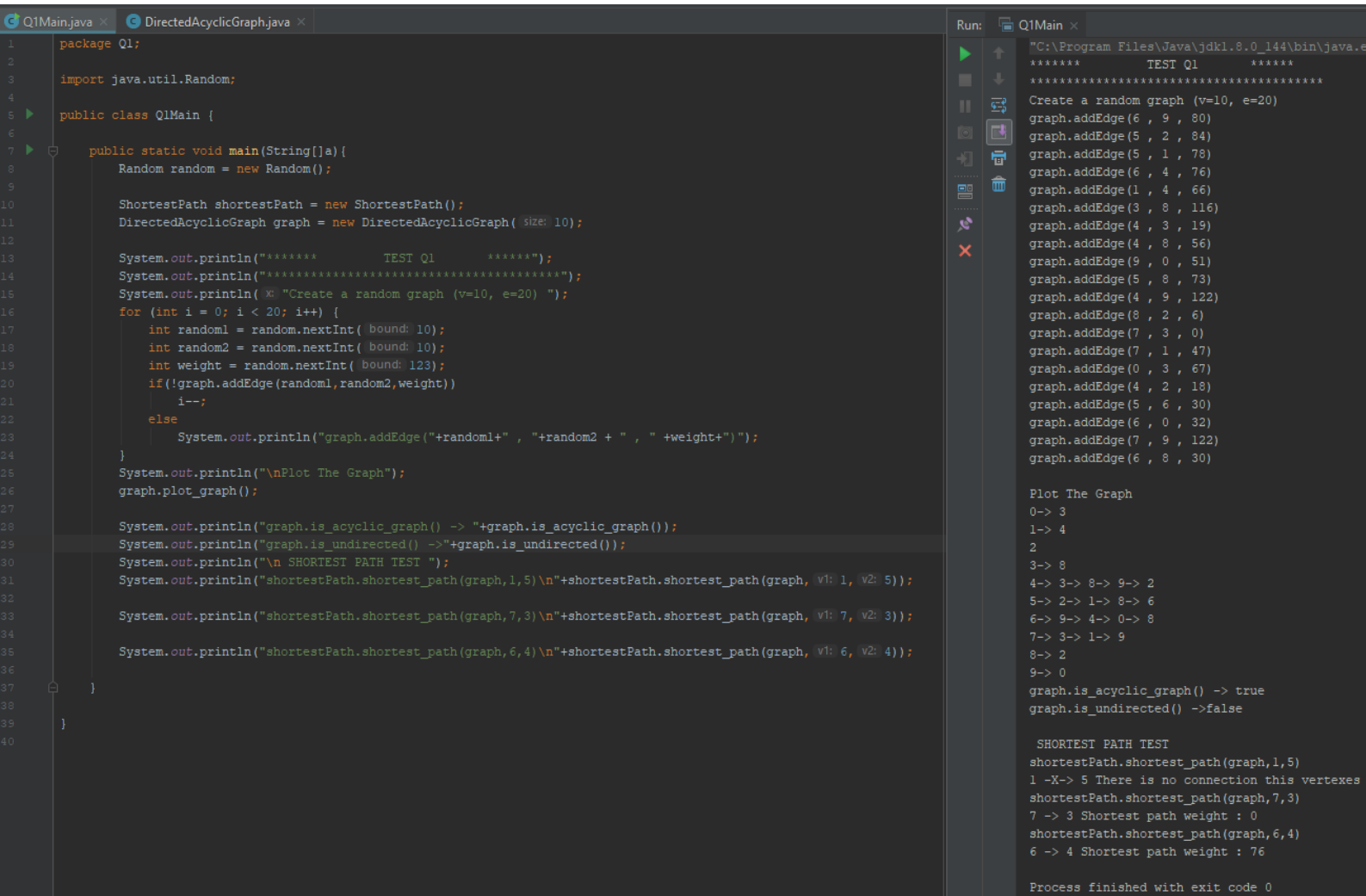
Course Assistant: Fatma Nur Esirci

1 Q1

1.1 Problem Solution Approach

Bu bölümde bizden istenilen Directed Acyclic Graph classını implement edip üzerinde plat_graph , is_undirected, is_acyclic_grapg ve shortest_path(g,v1,v2) metod ve yöntemlerinin imlement edilmesidir. Directed Graph oluşturukun öncelikle Graphımızın Node inner classı imlement edildi bu class bir vertex ve bir weight tutan bir node yapısıdır directed Graphın en önmeli özelliği yönünün olmasıdır bunu sağlayabilmek adına edges arrayinin her bir indexi aynı zamanda bir vertex olarak görev görür bu vertexe başka bir vertex bağlanılarak oluşturulan edges yapısı ise bu indexin elemanı olarak yapıya eklenir. is_acyclic metodunun kurulması şöyledir. Verilen graphın ilk indexinden başlayarak tek tek ona bağlı olanları dolanıp işaretler ve ilerler eğer işaretlenmiş bir birim gelirse burada cycle vardır denir ve acylist yapısını bozar. Shortest_path classında ise verilen graphın verilen ilk vertexinin bütün pathlerini bir stack yapısına atarak işe başlar edges sayısı ile eşit bir array oluşturulur ve içerisine maxInteger(Maksimum integer) atılarak devam edilir verilen v1 vertexinden v2 vertexine olan bütün pathler sıralanır ve en küçük olan meydana çıkar . Eğer verilen vertexler arası bağlantı yok ise buda belirtilir. Veya verilen vertexlerden herhangi biri verilen graphın elemanı değilse.

1.2 Test Cases



```
Q1Main.java x DirectedAcyclicGraph.java x
1 package Q1;
2
3 import java.util.Random;
4
5 public class Q1Main {
6
7     public static void main(String[]a){
8         Random random = new Random();
9
10        ShortestPath shortestPath = new ShortestPath();
11        DirectedAcyclicGraph graph = new DirectedAcyclicGraph( size: 10);
12
13        System.out.println("***** TEST Q1 *****");
14        System.out.println("*****");
15        System.out.println("❌ Create a random graph (v=10, e=20) ");
16        for (int i = 0; i < 20; i++) {
17            int random1 = random.nextInt( bound: 10);
18            int random2 = random.nextInt( bound: 10);
19            int weight = random.nextInt( bound: 123);
20            if(!graph.addEdge(random1,random2,weight))
21                i--;
22            else
23                System.out.println("graph.addEdge("+random1+ " , "+random2 + " , " +weight+"");
24        }
25        System.out.println("\nPlot The Graph");
26        graph.plot_graph();
27
28        System.out.println("graph.is_acyclic_graph() -> "+graph.is_acyclic_graph());
29        System.out.println("graph.is_undirected() ->"+graph.is_undirected());
30        System.out.println("\n SHORTEST PATH TEST ");
31        System.out.println("shortestPath.shortest_path(graph,1,5)\n"+shortestPath.shortest_path(graph, v1: 1, v2: 5));
32
33        System.out.println("shortestPath.shortest_path(graph,7,3)\n"+shortestPath.shortest_path(graph, v1: 7, v2: 3));
34
35        System.out.println("shortestPath.shortest_path(graph,6,4)\n"+shortestPath.shortest_path(graph, v1: 6, v2: 4));
36
37    }
38
39 }
40
```

```
Run: Q1Main x
"C:\Program Files\Java\jdk1.8.0_144\bin\java.e
***** TEST Q1 *****
Create a random graph (v=10, e=20)
graph.addEdge(6 , 9 , 80)
graph.addEdge(5 , 2 , 84)
graph.addEdge(5 , 1 , 78)
graph.addEdge(6 , 4 , 76)
graph.addEdge(1 , 4 , 66)
graph.addEdge(3 , 8 , 116)
graph.addEdge(4 , 3 , 19)
graph.addEdge(4 , 8 , 56)
graph.addEdge(9 , 0 , 51)
graph.addEdge(5 , 8 , 73)
graph.addEdge(4 , 9 , 122)
graph.addEdge(8 , 2 , 6)
graph.addEdge(7 , 3 , 0)
graph.addEdge(7 , 1 , 47)
graph.addEdge(0 , 3 , 67)
graph.addEdge(4 , 2 , 18)
graph.addEdge(5 , 6 , 30)
graph.addEdge(6 , 0 , 32)
graph.addEdge(7 , 9 , 122)
graph.addEdge(6 , 8 , 30)

Plot The Graph
0-> 3
1-> 4
2
3-> 8
4-> 3-> 8-> 9-> 2
5-> 2-> 1-> 8-> 6
6-> 9-> 4-> 0-> 8
7-> 3-> 1-> 9
8-> 2
9-> 0
graph.is_acyclic_graph() -> true
graph.is_undirected() ->false

SHORTEST PATH TEST
shortestPath.shortest_path(graph,1,5)
1 -X-> 5 There is no connection this vertexes
shortestPath.shortest_path(graph,7,3)
7 -> 3 Shortest path weight : 0
shortestPath.shortest_path(graph,6,4)
6 -> 4 Shortest path weight : 76

Process finished with exit code 0
```

2 Q2

2.1 Problem Solution Approach

Bu bölümde bizden Undirected Acyclic Graph oluşturmamız istenilmiştir . Bu yapı implemen edilirken öncelikle LinkedList arrayi olarak edges ler tanımlanmıştır yapıda yön olmadığı için her vertexin bağlı olduğu başka vertex her ikisinde de gösterilmiştir. Bu yapı da bizden istelen bazı metodlar plot_graph bu method birbirine bağlı verteşleri grafik olarak çizmeye yarayan metodur. Bunu implement ederken yapının bir yerinden başlayıp bağlı olanları bir stack yapısında tutarak son olarak ekrana bastırmaktır. Bu yapının undirected olduğunun sağlanması için yapılan is_undirected metodu şöyle tanımlanmıştır her edge iki vertexe sahiptir eğer vertex1 vertex2 ye bağlı ise vertex2 de vertex1 e bağlı olmak zorundadır bu koşulu sağladığı zaman yapı undirected olur. is_acyclic_graph bu method verilen graphın içerisinde herhangi bir cycle(döğü) olup olmadığını saptamaya yarayan metoddur bunun anlamak için yapının başı dediğimiz edges[0] konumundan başlayıp elemanları gezip işaretlemektir eğer bir tur sırasında ayışaretli eleman denk gelirse yapıda cycle (döğü) vardır diyip false döndürüyoruz .

2.2 Test Cases

```
Q2Main.java x DirectedAcyclicGraph.java x
1 package Q2;
2
3 import java.util.Random;
4
5 public class Q2Main {
6
7     public static void main(String []a){
8
9         Random random = new Random();
10        System.out.println("***** TEST Q2 *****");
11        System.out.println("*****");
12        UndirectedAcyclicGraph graph = new UndirectedAcyclicGraph( size: 15);
13        Connect con = new Connect();
14
15
16        System.out.println("Create a UndirectedAcyclicGraph ( v=15 ) ");
17
18        System.out.println("addEdge 10 for Testing ");
19        for (int i = 0; i < 10; i++) {
20            int random1 = random.nextInt( bound: 15);
21            int random2 = random.nextInt( bound: 15);
22            if(!graph.addEdge(random1,random2))
23                i--;
24            else
25                System.out.println("graph.addEdge("+random1+" , "+random2 + " ");
26        }
27
28        System.out.println("\nPlot The Graph");
29        graph.plot_graph();
30        System.out.println("graph.is_acyclic_graph() -> "+graph.is_acyclic_graph());
31        System.out.println("graph.is_undirected() ->"+graph.is_undirected());
32        System.out.println("\n IS_CONNECTED TEST ");
33        System.out.println("con.is_connected(graph,1,4) -> "+con.is_connected(graph, v1: 1, v2: 4));
34        System.out.println("con.is_connected(graph,2,7) -> "+con.is_connected(graph, v1: 2, v2: 7));
35        System.out.println("con.is_connected(graph,6,0) -> "+con.is_connected(graph, v1: 6, v2: 0));
36
37    }
38
39
40
41

```

```
Run: Q2Main x
"C:\Program Files\Java\jdk1.8.0_144\bin\jav
***** TEST Q2 *****
*****
Create a UndirectedAcyclicGraph ( v=15 )
addEdge 10 for Testing
graph.addEdge(0 , 3)
graph.addEdge(4 , 3)
graph.addEdge(1 , 11)
graph.addEdge(2 , 6)
graph.addEdge(0 , 2)
graph.addEdge(9 , 8)
graph.addEdge(4 , 5)
graph.addEdge(2 , 14)
graph.addEdge(8 , 10)
graph.addEdge(11 , 5)

Plot The Graph
0-> 3-> 2
1-> 11
2-> 6-> 0-> 14
3-> 0-> 4
4-> 3-> 5
5-> 4-> 11
6-> 2
7
8-> 9-> 10
9-> 8
10-> 8
11-> 1-> 5
12
13
14-> 2
graph.is_acyclic_graph() -> true
graph.is_undirected() ->true

IS_CONNECTED TEST
con.is_connected(graph,1,4) -> true
con.is_connected(graph,2,7) -> false
con.is_connected(graph,6,0) -> true

Process finished with exit code 0

```

3 Q3

3.1 Problem Solution Approach

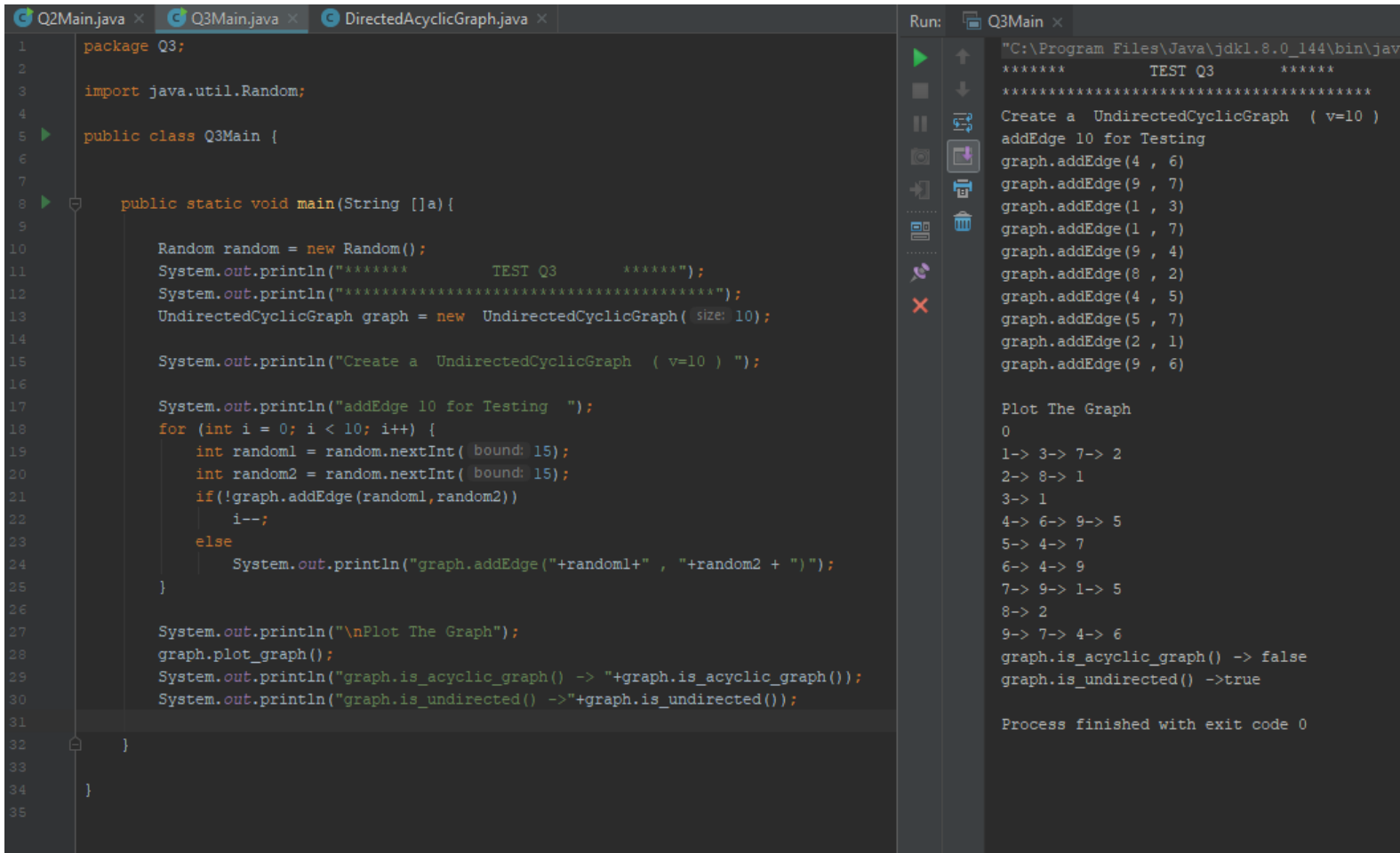
Bu yapı implemen edilirken öncelikle LinkedList arrayi olarak edges ler tanımlanmıştır yapıda yön olmadığı için her vertexin bağlı olduğu başka vertex her ikisinde de gösterilmiştir. Bu yapı da bizden istelen bazı metodlar plot_graph bu method birbirine bağlı verteşleri grafik olarak çizmeye yarayan metodur.

Bunu implement ederken yapının bir yerinden başlayıp bağlı olanları bir stack yapısında tutarak son olarak ekrana bastırmaktır. Bu yapının undirected olduğunun sağlanması için yapılan is_undirected metodu şöyle tanımlanmıştır her edge iki vertexe sahiptir eğer vertex1 vertex2 ye bağlı ise vertex2 de vertex1 e bağlı olmak zorundadır bu koşulu sağladığı zaman yapı undirected olur.

is_acyclic_graph bu method verilen graphın içerisinde herhangi bir cycle(döğü) olup olmadığını saptamaya yarayan metoddur bunun anlamak için yapının başı dediğimiz edges[0] konumundan başlayıp elemanları gezip işaretlemektir eğer bir tur sırasında ayışaretli eleman denk gelirse yapıda cycle (döğü) vardır diyip false döndürüyoruz .

Bu bölümün implemantasyonun çoğu yönü Q2 deki gibidir ayrılan yönleri ise şöyledir.Öncelikle edges eklenirken bu eklenilecek olan edges cycle durumu oluşturuyormu diye kontrol edilmeden eklenebilmektedir. Ve yapı gezilirken bu kurala göre gezilmektedir.

3.2 Test Cases



```
package Q3;

import java.util.Random;

public class Q3Main {

    public static void main(String []a){

        Random random = new Random();
        System.out.println("*****          TEST Q3          *****");
        System.out.println("*****");
        UndirectedCyclicGraph graph = new UndirectedCyclicGraph( size: 10);

        System.out.println("Create a UndirectedCyclicGraph ( v=10 ) ");

        System.out.println("addEdge 10 for Testing ");
        for (int i = 0; i < 10; i++) {
            int random1 = random.nextInt( bound: 15);
            int random2 = random.nextInt( bound: 15);
            if(!graph.addEdge(random1,random2))
                i--;
            else
                System.out.println("graph.addEdge("+random1+" , "+random2+" )");
        }

        System.out.println("\nPlot The Graph");
        graph.plot_graph();
        System.out.println("graph.is_acyclic_graph() -> "+graph.is_acyclic_graph());
        System.out.println("graph.is_undirected() ->"+graph.is_undirected());
    }
}
```

```
*****          TEST Q3          *****
*****
Create a UndirectedCyclicGraph ( v=10 )
addEdge 10 for Testing
graph.addEdge(4 , 6)
graph.addEdge(9 , 7)
graph.addEdge(1 , 3)
graph.addEdge(1 , 7)
graph.addEdge(9 , 4)
graph.addEdge(8 , 2)
graph.addEdge(4 , 5)
graph.addEdge(5 , 7)
graph.addEdge(2 , 1)
graph.addEdge(9 , 6)

Plot The Graph
0
1-> 3-> 7-> 2
2-> 8-> 1
3-> 1
4-> 6-> 9-> 5
5-> 4-> 7
6-> 4-> 9
7-> 9-> 1-> 5
8-> 2
9-> 7-> 4-> 6
graph.is_acyclic_graph() -> false
graph.is_undirected() ->true

Process finished with exit code 0
```

4 Q4

Bu bölüm yetiřmemiřtir.