

**Gebze Technical University  
Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 7 REPORT**

**Furkan AKTAŞ  
141044029**

Course Assistant: Fatma Nur Esirci

# 1 Q1

**General package**'ı altında ortak class'ları ,fonksiyonları topladım. Bunlar temel graph classları (AbstractGraph, Edge , Graph, MatrixGraph) ve benim eklediğim ödevde kullandığımız fonksiyonları içeren **MyGraph** classı ( **fonksiyonlar; makeAcyclicGraph, makeCyclicGraph, is\_acyclic\_graph, is\_undirected** ) içeriyor.

**makeAcyclicGraph:** directed veya undirected durumlarına göre acyclic graph üretir. Directed graph için base vertex tekrar eklenmemek üzere graph oluşturuldu(Örnek; 0-1 0-2 0-3, 1-2 1-3, 2-3). Undirected için 0 hariç tüm vertexler 0 eklendi. (0-1, 0-2, 0-3 ..., 0-n) Yani eklenmek istenen edge sayısına kadar vertexler sırasıyla eklenir.

**makeCyclicGraph:** her vertex döngü ile birbirine eklendi. 0-0, 0-1, 0-2, ... 1-0, 1-1, 1-2, ... 2-0, 2-1, 2-2 ... bu sayede cyclic graph elde edilir.

**is\_acyclic\_graph:** Dfs temel alınarak yapıldı. Undirected is ilk komşusu hariç diğer tüm vertexlerden başlangıç vertexine komşu olan bir vertex varsa false return eder. Directed ise başlangıç vertex'ine komşu olan vertex bulunduğunda false return eder.

**is\_undirected:** Bfs temel alınarak yapıldı. Verdiğiniz ipucuna göre yapıldı. Eğer ters edge aynı ağırlıkta değilse false return eder.

**plot\_graph:** edge iterator yardımı ile sadece komşu vertexleri bastırarak şekilde yazıldı.

## 1.1 Problem Solution Approach

### shortest\_path:

Graph üretilirken Genereal package daki fonksiyonlar kullanıldı. Temel olarak dijkstra algoritması kullanılarak yapıldı. Önce istenilenEğer bağlı ise disktra çalıştırıldı( dijkstra 'da bağlı olmayan graphlarda sıkıntı çıkıyordu ufak ekleme yaptım.) ve eld path in graph ta bağlı olup olmadığı is\_connected() ile kontrol edildi. e edilen pred ve dist array'leri kullanılarak path elde edildi. Dijkstra v1 ile çalıştırıldı. Daha sonra v2 'den pred ve dist ile greiye dogru distance 0.0 bulana kadar geri gidildi ve vertexler stack'te tutuldu. Son olarak stack ten elemanlar alıanark vector'e koyuldu.

## 1.2 Test Cases

Kitapta sayfa 507 deki figure 10.17 ve kendi fonksiyonumdan üretilen graph ile denendi. Kendi fonksiyonumda random weighted olduğundan her seferinde farklı sonuç veriyor.

```
cse222-hw7 ~/IdeaProjects
.idea
out
src
  General
    AbstractGraph
    Edge
    Graph
    MatrixGraph
    MyGraph
  Q1
    Main
    Q1Graph
  Q2
    Main
    Q2Graph
  Q3
    Main
    Q3Graph
cse222-hw7.iml
ternal Libraries

19 }
20
21 System.out.println(graph.shortest_path(graph, v1: 0, v2: 8));
22 System.out.println(graph.shortest_path(graph, v1: 1, v2: 9));
23 System.out.println(graph.shortest_path(graph, v1: 0, v2: 5));
24
25
26 // kitapta syf 507 , figure 10.17
27
28 System.out.println("\nfigure 10.17");
29 Q1Graph graph1 = new Q1Graph( numV: 10, directed: true);
30
31 graph1.insert(new Edge( source: 0, dest: 3));
32 graph1.insert(new Edge( source: 0, dest: 1));
33 graph1.insert(new Edge( source: 1, dest: 2));
34 graph1.insert(new Edge( source: 1, dest: 4));
35 graph1.insert(new Edge( source: 1, dest: 6));
36 graph1.insert(new Edge( source: 1, dest: 7));
37 graph1.insert(new Edge( source: 2, dest: 8));
38 graph1.insert(new Edge( source: 2, dest: 9));
39 graph1.insert(new Edge( source: 4, dest: 5));
40
41
42 if (!graph1.is_acyclic_graph(graph1)){
43     System.out.println("This is not acyclic graph !!");
44 }
45
46 if (graph1.isUndirected(graph1)){
47     System.out.println("This is not directed graph !!");
48 }
49
50 System.out.println(graph1.shortest_path(graph1, v1: 0, v2: 5));
51 System.out.println(graph1.shortest_path(graph1, v1: 0, v2: 2));
52 System.out.println(graph1.shortest_path(graph1, v1: 1, v2: 9));

Main > main()

/usr/lib/jvm/java-8-oracle/bin/java ...
[0, 1, 8]
[1, 9]
[0, 1, 5]

figure 10.17
[0, 1, 4, 5]
[0, 1, 2]
[1, 2, 9]

Process finished with exit code 0
```

## 2 Q2

This part about Question2 in HW7

### 2.1 Problem Solution Approach

**is\_connected:** Bfs temel alınarak yapıldı. V1 den travers etmeye başlar. Traverse edilirken herahngi bir komşu v2 ise true return eder.

### 2.2 Test Cases

İlk testte makeAcyclicGraph fonksiyonu kullanılarak oluşturulan graph üzerinde denendi.

is\_acyclic\_graph , is\_undirected testleri yapıldı. Daha sonra is\_connected testi yapıldı.

2. testte kitaptaki figure 10.17 temel alınarak oluşturulmuş graph ile aynı aşamaları uygulayarak denendi

The screenshot shows an IDE with a project named 'cse222-hw7'. The file explorer on the left shows a directory structure with 'Main.java' selected. The code editor displays the following Java code:

```
19 }
20
21 for (int i = 0; i < 5; i++) {
22     if (graph.isConnected(graph, v1: 0, i)) {
23         System.out.println("0 " + i + " are connected !!");
24     }
25 }
26
27
28
29 System.out.println("\nfigure 10.17");
30 Q2Graph graph1 = new Q2Graph( numV: 10, directed: false);
31
32 graph1.insert(new Edge( source: 0, dest: 3));
33 graph1.insert(new Edge( source: 0, dest: 1));
34 graph1.insert(new Edge( source: 1, dest: 2));
35 graph1.insert(new Edge( source: 1, dest: 4));
36 graph1.insert(new Edge( source: 1, dest: 6));
37 graph1.insert(new Edge( source: 1, dest: 7));
38 graph1.insert(new Edge( source: 2, dest: 8));
39 graph1.insert(new Edge( source: 2, dest: 9));
40 graph1.insert(new Edge( source: 4, dest: 5));
41
42
43 if (!graph1.isAcyclicGraph(graph1)) {
44     System.out.println("This is not acyclic graph !!");
45 }
46
47 if (!graph1.isUndirected(graph1)) {
48     System.out.println("This is not undirected graph !!");
49 }
50
51 System.out.println(graph1.isConnected(graph1, v1: 0, v2: 5));
52 System.out.println(graph1.isConnected(graph1, v1: 0, v2: 9));
```

The output window at the bottom shows the following execution results:

```
/usr/lib/jvm/java-8-oracle/bin/java ...
0 0 are connected !!
0 1 are connected !!
0 2 are connected !!
0 3 are connected !!
0 4 are connected !!

figure 10.17
true
true
true

Process finished with exit code 0
```

### 3 Q3

This part about Question3 in HW7

#### 3.1 Problem Solution Approach

BFS MatrixGraph içinde vardı. DFS recursion kullanarak kendim yazdım. Spanning Tree için kitaptaki prim algoritması kullanıldı.

#### 3.2 Test Cases

Package içindeki main de senaryo yapıldı.

İlk örnekte makeCyclicGraph fonksiyonu ile graph oluşturuldu. Ve bfs dfs ve prim algoritmaları çalıştırıldı.  
2. örnekte Q4 teki graph kullanılarak algoritmalar çalıştırıldı.

```
cse222-hw7 ~/IdeaProjects
.idea
out
src
  General
    AbstractGraph
    Edge
    Graph
    MatrixGraph
    MyGraph
  Q1
    Main
    Q1Graph
  Q2
    Main
    Q2Graph
  Q3
    Main
    Q3Graph
cse222-hw7.iml
External Libraries

22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

for (int i: arr
    ) {
    System.out.print(i+ " ");
}
System.out.println("");

System.out.println(graph.primAlgorithm(graph, start: 0)); // spanning tree

System.out.println("\nQ4 'teki oluşan graph'");

Q3Graph graph1 = new Q3Graph( numV: 7, directed: false);

graph1.insert(new Edge( source: 0, dest: 1));
graph1.insert(new Edge( source: 0, dest: 5));
graph1.insert(new Edge( source: 1, dest: 2));
graph1.insert(new Edge( source: 2, dest: 4));
graph1.insert(new Edge( source: 2, dest: 6));
graph1.insert(new Edge( source: 2, dest: 5));
graph1.insert(new Edge( source: 4, dest: 3));
graph1.insert(new Edge( source: 5, dest: 4));

arr = graph1.breadthFirstSearch( start: 1);

for (int i: arr
    ) {
    System.out.print(i+ " ");
}
System.out.println("");

Main > main()

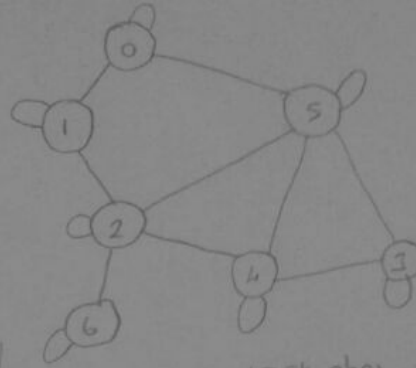
Main (1)
/usr/lib/jvm/java-8-oracle/bin/java ...
-1 0 0 0 0 0 0 0 0
-1 0 1 1 1 1 1 1 1
[[ (0, 1): 1.0], [(0, 9): 1.0], [(1, 8): 1.0], [(1, 7): 1.0], [(1, 6): 1.0], [(1, 5): 1.0], [(1, 4): 1.0], [(1, 3): 1.0], [(1, 2): 1.0]]
Q4 'teki oluşan graph'
1 -1 1 4 2 0 2
1 -1 5 4 2 0 2
[[ (0, 1): 1.0], [(0, 5): 1.0], [(1, 2): 1.0], [(5, 4): 1.0], [(2, 6): 1.0], [(4, 3): 1.0]]

Process finished with exit code 0
```

## 4 Q4

4-

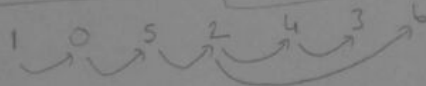
	0	1	2	3	4	5	6
0	1						
1		1					
2			1	1			
3					1		
4				1	1	1	
5	1			1	1	1	
6			1				1



Eğer 1'den fazla seçenek varsa  
kSütun olanından devam edildi.

→ Kitapta iteratör büyük olanı seçipminus  
sıradan çıkarttım.

DFS →

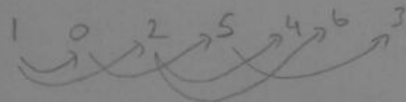


⇒ 3 işaret edildikten  
sonra recursion 2'ye  
döner b'yi işaret  
etti.

parent arr →

1	-1	5	4	2	0	2
0	1	2	3	4	5	6

BFS →



parent arr →

1	-1	1	5	2	0	2
0	1	2	3	4	5	6

⇒ Önceki kardeşler eklenir  
devam ediyor. Çıkan vertesin  
kardeşleri ekleniyor.

→ 0 ve 2 önce eklenir  
daha sonra 0'ın kardeşi 5  
eklenir. Daha sonra 2  
eklenir, 2'nin kardeşleri 4 ve 6  
eklenir. Daha sonra 5'in  
ve kardeşi 3 eklenir.  
Her zaman visited olmayanlar  
eklenir.

BFS

- Önce verilen root'un tüm kardeşleri ekler
- Eğer Adjacency matrix kullanılırsa  $O(V^2)$ , list kullanılırsa  $O(V \cdot \text{deg})$  çalışma süresine sahip
- Önce tüm kardeşleri ekler, sonra bunların 1'i üzerinden devam eder. Eğer bir özel bir şey arıyorsam, gerekirse hafıza kullanmam olurum.
- Shortest path probleminde tercih edilebilir.

DFS

- Verilen root'un 1 kardeşini ekler ve eklenen kardeşlerin recursive olarak devam eder.
- BFS ile aynı çalışma süresine sahip
- Sürekli eklenen yeni kardeşler üzerinden ilerlediklerinden hafızada sadece önceki kardeş eklenerek ilerler. Bu sayede daha az hafıza kullanır.
- Tüm graph üzerinde işlem yapılacaksa tercih edilebilir.