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 üretitr. 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 tüme 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 vertex lerden başlangıç vertexine komşu olan bir vertex varsa false return eder. Directed ise başlangıç vertex'ine komşu olan vertex bulundugunda false return eder.

is_undirected: Bfs temel alınarak yapıldı. Verdiğiniz ipucuna göre yapıldı. Eğer ters edge aynı agırlıkta değilse false return eder.

plot_graph: edge iterator yardımı ile sadece komşu vertexleri bastıracak şekilde yazıldı.

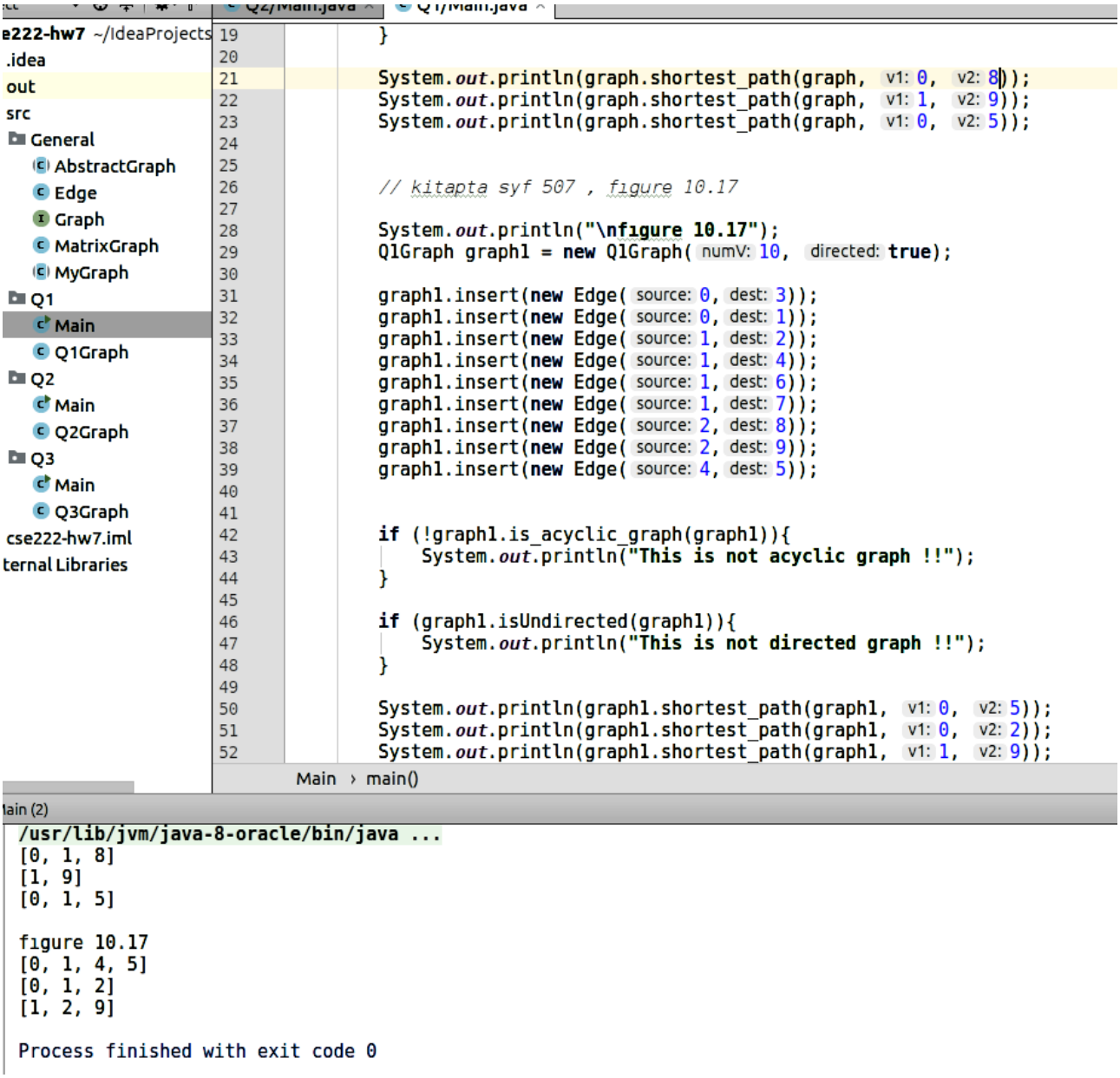
1.1 Problem Solution Approach

`shortest_path:`

Graph üretilirken Genereal package daki fonksiyonlar kullanıldı. Temel olarak dijkstra algoritması kullanırarak 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 oldugundan her seferinde farklı sonuç veriyor.


e222-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
Q2/Main.java
Q1/Main.java
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()
ain (2)
/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

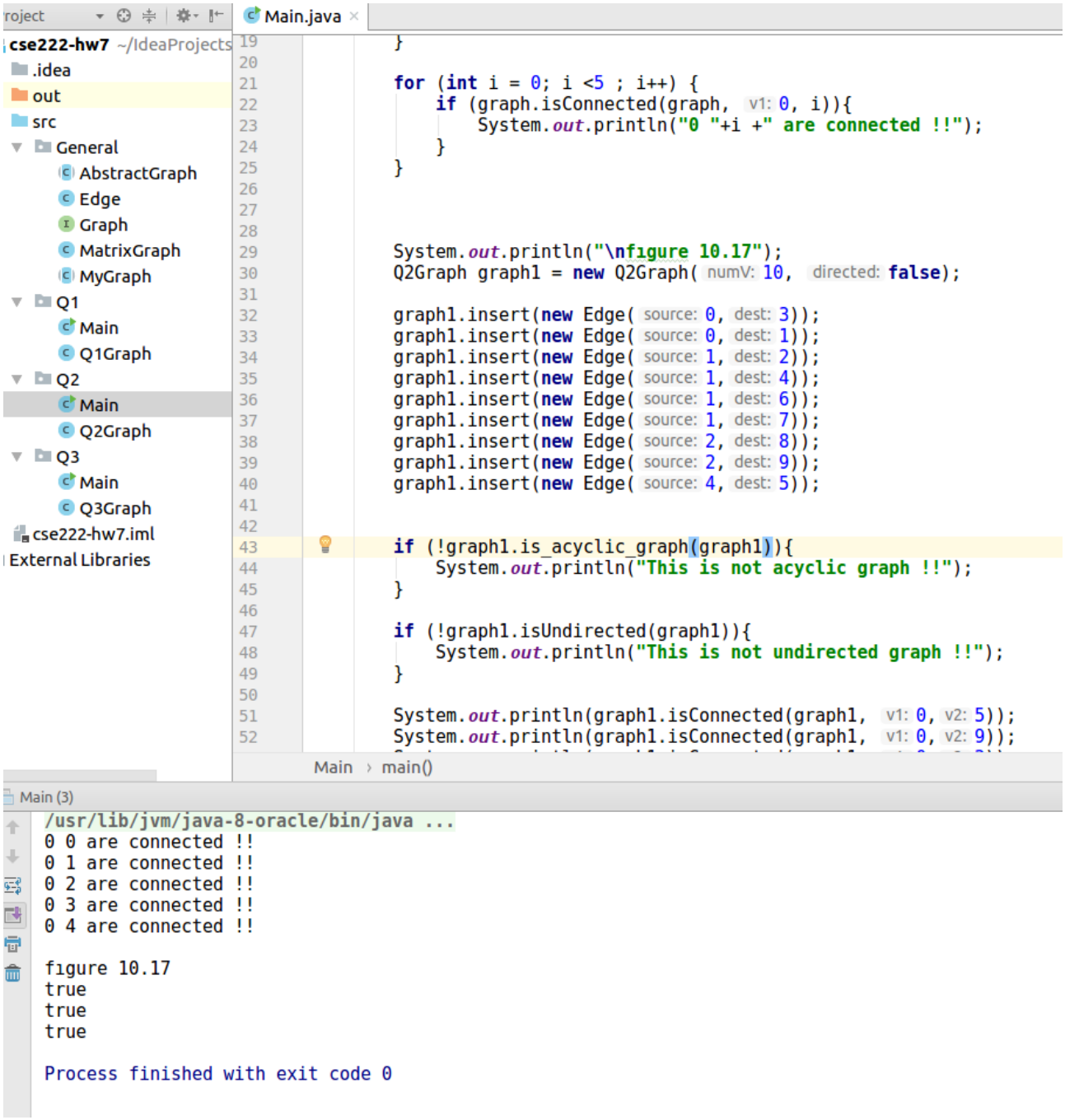
This part about Question2 in HW7

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



3 Q3

This part about Question3 in HW7

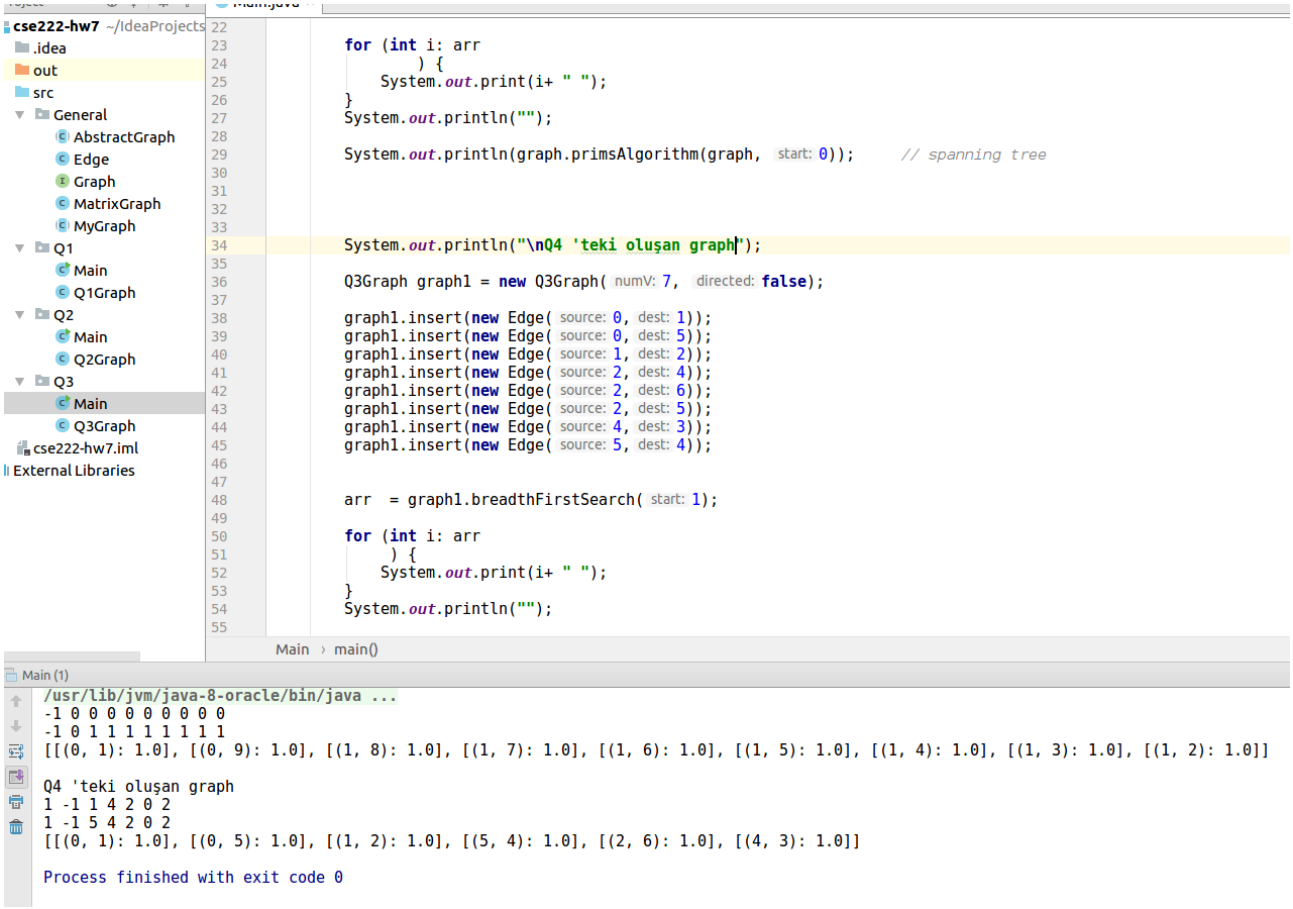
3.1 Problem Solution Approach

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

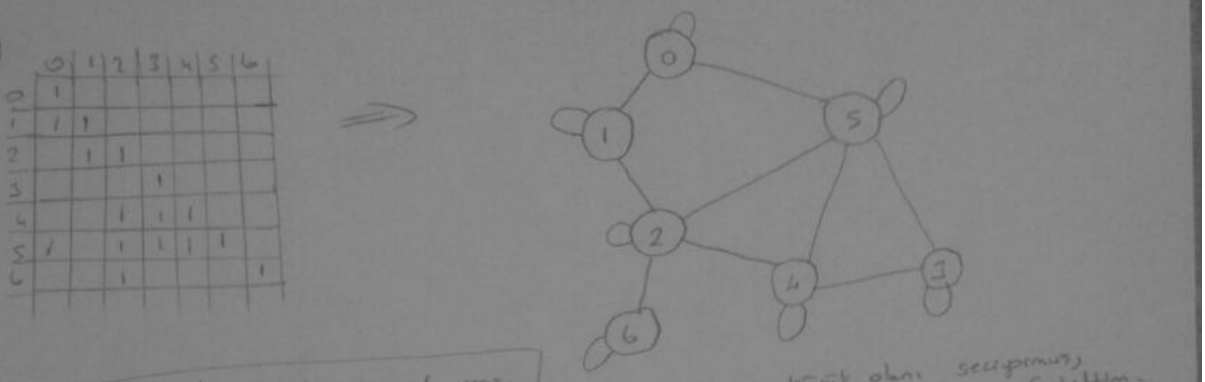
3.2 Test Cases

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.



4 Q4



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

→ Kitapta iterator büyük olanı seçiyormuş, sonradan farkettim.

DFS → 1 0 5 2 4 3 6

⇒ 3 ziyaret edildikten sonra recursion 2'ye düşüp 6'yı ziyaret etti.

parent arr →

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

BFS → 1 0 2 5 4 6 3

⇒ Queue'ya kardeşler eklenerek devam ediyor. Çıkan vertex'in kardeşleri ekleniyor.

→ 0 ve 2 önce eklendi daha sonra 0 çıktı 0'ın kardeşi 5 eklendi. Daha sonra 2 çıktı, 2'nin kardeşleri 4 ve 6 eklendi Daha sonra 5 çıktı ve kardeşi 3 eklendi.

Her zaman visited olmayanlar eklendi.

parent arr →

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

BFS

- Önce verilen root'un tüm kardeşlerini ekler
- Eğer Adjacency matrix kullanılırsa $O(V^2)$, List kullanılırsa $O(V+E)$ çalışma süresine sahiptir
- Önce tüm kardeşleri ekler, sonra bunlardan 1'i üzerinde devam eder. Eğer bu özel bir şey arıyorsam, gereksiz hafıza kullanmış olurum.
- Shortest path problemlerinde tercih edilebilir.

DFS

- Verilen root'un 1 kardeşini ekler ve eklenen kardeşi sanki recursif olarak devam eder.
- BFS ile aynı çalışma süresine sahiptir
- Sürekli eklenen geri kardeşler üzerinden ilerlediğinden hafızada sadece atılan kardeşler eklenerek ilerler. Bu sayede daha az hafıza kullanır.
- Tüm graph üzerinde işlem yapılacaksa tercih edilebilir.