Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 7 REPORT

EFKAN DURAKLI
161044086

Course Assistant: Fatma Nur Esirci

1 Q1

Bu bölümde düğüm sayısı 10, kenar sayısı 20 ağırlıkları random olarak belirlenen acyclic bir çizge oluşruruldu. Çizgenin acyclic ve directed olup olmadığı yazılan fonksiyonlarla test edildi. İki nokta arasındaki en kısa mesafe yazılan fonksiyonlarla bulundu.

1.1 Problem Solution Approach

Düğüm sayısı 10 olan 20 kenarlı bir çizge oluşturuldu. Bu çizge oluşturulurken ağırlık için 20 tane random sayıdan oluşan bir array oluşturuldu. Acylic olabilmesi için çizgedeki kenarlar ona göre ayarlandı.

1.2 Test Cases

Bu çizge üzerinde aşağıdaki fonksiyonlar çalıştırıldı.

- plot_graph
- is_undirected
- is_acyclic_graph
- shortest_path (use least 3 different label pair)

Bu fonksiyonları çıktıları aşağıdaki gibidir.

Not : Fonksiyonun görsel olarak gösteriminde her düğümün başında parantez içinde yazan double değerler. Düğümler arasındaki ağırlığı ifade eder.

```
QUESTION1 TEST
Created Directed Acyclic Graph have random weights
Number of vertices : 10
Number of edges : 20
Visual representation of graph
0 : ( 70,75 )1 -> ( 2,90 )9
1 : ( 13,00 )2 -> ( 40,82 )6 -> ( 83,60 )7 -> ( 34,19 )8
2 : ( 65,65 )3 -> ( 50,38 )4 -> ( 46,28 )5 -> ( 42,26 )7
3 : ( 10,33 )4 -> ( 19,93 )6 -> ( 60,67 )7
4 : ( 13,53 )5 -> ( 13,18 )6 -> ( 40,44 )9
5 : ( 80,53 )6
6 : ( 17,65 )7
7 : ( 66,60 )8
8 : ( 40,39 )9
9 :
Graph is directed
Graph is acyclic
--------------------------------------------------------------------------------
The shortest path between vertex 2 and vertex 6 = [2, 4, 6]
Distance of shortest path = 63.55994257653796
--------------------------------------------------------------------------------
The shortest path between vertex 3 and vertex 9 = [3, 4, 9]
Distance of shortest path = 50.771983155544255
--------------------------------------------------------------------------------
The shortest path between vertex 0 and vertex 4 = [0, 1, 2, 4]
Distance of shortest path = 134.1233325367034
--------------------------------------------------------------------------------
There is no path between vertex 8 and vertex 0
```

2 Q2

Bu bölümde düğüm sayısı 15, kenar sayısı 13 ve kenar ağırlıkları bütün kenarlarda eşit acyclic bir çizge oluşruruldu. Çizgenin acyclic ve directed olup olmadığı yazılan fonksiyonlarla test edildi. Herhangi iki düğümün bağlı olup olmadığı is_connected fonksiyonuyla teste edildi.

2.1 Problem Solution Approach

Düğüm sayısı 15, kenar sayısı 13 olan ağırlıksız bir çizge oluşturuldu. Çizgelerin kenarları bir dosyaya yazılarak bu dosyadan okundu. İs_connected fonksiyonunun test edilebilmesi için birbirine hiçbir şekilde bağlı olmayan düğümler oluşturuldu.

2.2 Test Cases

Bu çizge üzerinde aşağıdaki fonksiyonlar çalıştırıldı.

- plot_graph
- is_undirected
- is_acyclic_graph
- is_connected function (use least 3 different label pair)

Bu fonksiyonları çıktıları aşağıdaki gibidir.

```
QUESTION2 TEST
Created Undirected Acyclic Graph have no weights
Number of vertices : 15
Number of edges : 13
Visual representation of graph
0 : ( 1,00 )1
1 : ( 1,00 )0 -> ( 1,00 )2
2 : ( 1,00 )1 -> ( 1,00 )3
3 : ( 1,00 )2 -> ( 1,00 )4
4 : ( 1,00 )3 -> ( 1,00 )5
5 : ( 1,00 )4 -> ( 1,00 )6
6 : ( 1,00 )5 -> ( 1,00 )7
7 : ( 1,00 )6 -> ( 1,00 )8
8 : ( 1,00 )7 -> ( 1,00 )14
9 : ( 1,00 )14
10 : ( 1,00 )14
11 : ( 1,00 )14
12 : ( 1,00 )13
13 : ( 1,00 )12
14 : ( 1,00 )8 -> ( 1,00 )9 -> ( 1,00 )10 -> ( 1,00 )11
Graph is undirected
Graph is acyclic
------------------------------------------------------
Vertex 0 and vertex 8 is connected
Vertex 14 and vertex 5 is connected
Vertex 7 and vertex 10 is connected
Vertex 6 and vertex 13 is not connected
```

3 Q3

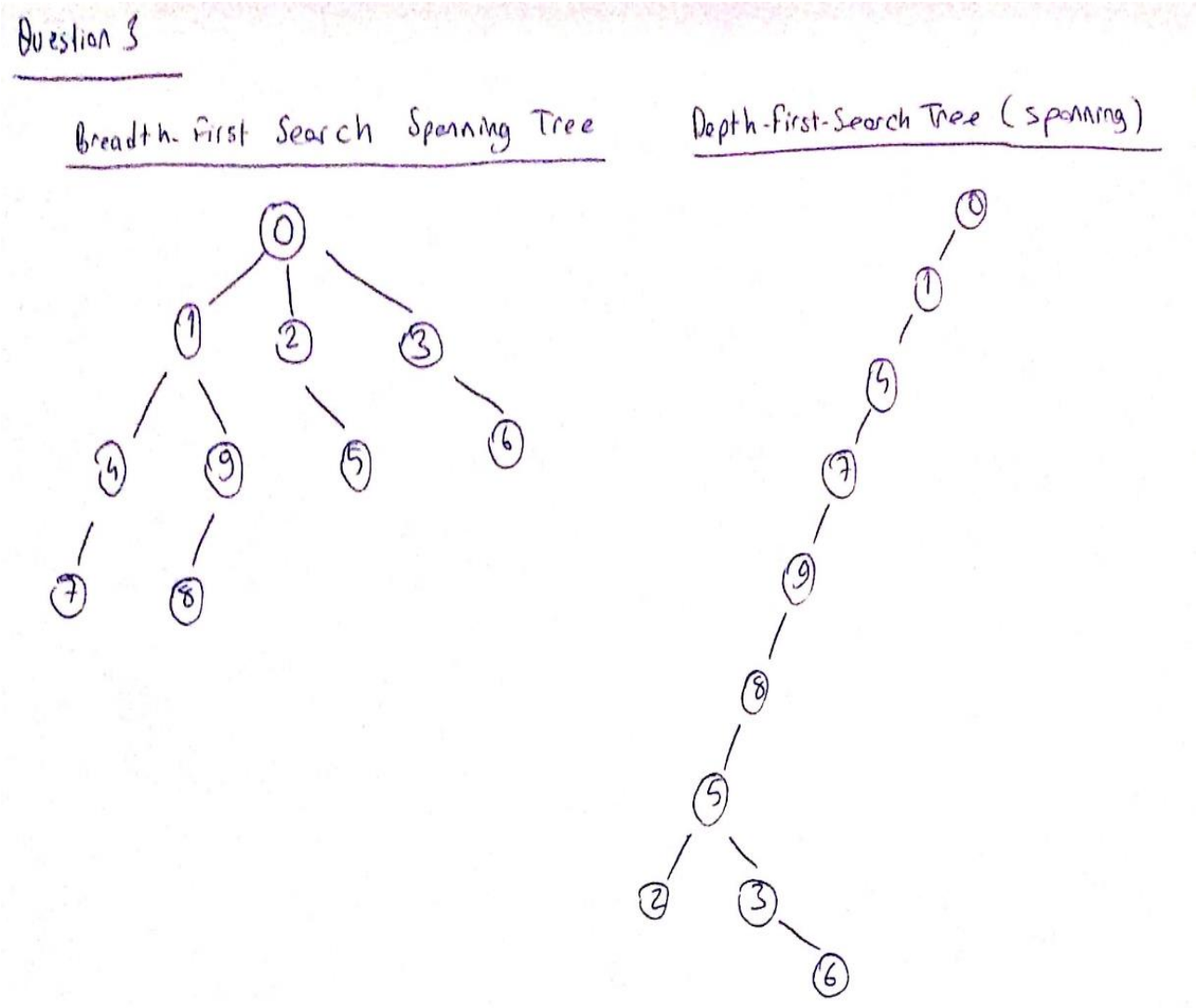
Bu bölümde düğüm sayısı 10, kenar sayısı 16 ve kenar ağırlıkları bütün kenarlarda eşit cyclic bir çizge oluşruruldu. Çizgenin acyclic ve directed olup olmadığı yazılan fonksiyonlarla test edildi. İki nokta arasındaki en kısa mesafe yazılan fonksiyonlarla bulundu. Bu çizge üzerinde Dept-First Search ve Breadth-First Search algoritmaları çalıştırılarak spanning tree'ler çizildi.Minimun spanning tree için Prism algoritması kullanıldı.

3.1 Problem Solution Approach

Düğüm sayısı 10, kenar sayısı 16 olan ağırlıksız bir çizge oluşturuldu. Çizgelerin kenarları bir dosyaya yazılarak bu dosyadan okundu. Breadth-First Search ve Depth-First Search alogoritmaları çalıştırılarak spanning tree çizildi. Prism algoritması kullanılarak minimum spanning tree bulundu.

3.2 Test Cases

Spanning tree'ler elle çizildi. Bunlar aşağıdaki gibidir.



Bu çizge üzerinde aşağıdaki fonksiyonlar çalıştırıldı.

- plot_graph
- is_undirected
- is_acyclic_graph
- DepthFirstSearch (Show that spanning tree)
- BreathFirstSearch (Show that spanning tree)

Bu fonksiyonları çıktıları aşağıdaki gibidir.

```
QUESTION3 TEST
Created Undirected Cyclic Graph have no weights
Number of vertices : 10
Number of edges : 16
Visual representation of graph
0 : ( 1,00 )1 -> ( 1,00 )2 -> ( 1,00 )3
1 : ( 1,00 )0 -> ( 1,00 )4 -> ( 1,00 )9
2 : ( 1,00 )5 -> ( 1,00 )0
3 : ( 1,00 )0 -> ( 1,00 )6 -> ( 1,00 )4 -> ( 1,00 )5
4 : ( 1,00 )1 -> ( 1,00 )7 -> ( 1,00 )3 -> ( 1,00 )6
5 : ( 1,00 )8 -> ( 1,00 )2 -> ( 1,00 )3
6 : ( 1,00 )3 -> ( 1,00 )8 -> ( 1,00 )4
7 : ( 1,00 )4 -> ( 1,00 )9 -> ( 1,00 )8
8 : ( 1,00 )9 -> ( 1,00 )5 -> ( 1,00 )6 -> ( 1,00 )7
9 : ( 1,00 )7 -> ( 1,00 )8 -> ( 1,00 )1
Graph is undirected
Graph is cyclic
```

```
BREADTH-FIRST SEARCH TEST
Parent Array for Starting point 0
[ -1 0 0 0 1 2 3 4 9 1 ]
-----------------------------------------------------
DEPTH-FIRST SEARCH TEST
Parent Array for Graph
[ -1 0 5 5 1 8 3 4 9 7 ]
Finish Order for for Graph
[ 2 6 3 5 8 9 7 4 1 0 ]
Discovery Order for for Graph
[ 0 1 4 7 9 8 5 2 3 6 ]
-----------------------------------------------------------
Minimum Spanning Tree of Graph for Starting point 0
0 -> 1
0 -> 3
1 -> 9
3 -> 5
9 -> 8
5 -> 2
8 -> 7
8 -> 6
7 -> 4
```

Metotlar

● is_directed

Bu metot yazılırken çizgedeki bütün kenarlar gezildi ve kenarların complementleri olup olmadığına bakıldı. Eğer herhangi bir kenarın bile comlementi yok ise bu çizge yönsüzdür ve doğru değer döndürülür. Eğer comlementi var ise complementlerin ağırlıklarına bakıldı. Eğer kenarın complementiyle ağırlığı farklı ise bu çizge yönsüzdür ve doğru değer döndürülür.

● is_acylic

Bu metot yazılırken iki adet yardımcı metot yazıldı. Yönlü ve yönsüz çizgeler için ayrı ayrı bakıldı. Eğer bir çizge yönsüzse aynı düğümden başlayarak en az 3 kenar gezerek geliyorsa bu çizge cyclic'tir. Yönsüz çizgede ise bir düğümden başlayarak aynı düğüme ez az bir kenar gezerek gelebiliyor ise bu çizge cylic bir çizgedir.

● shortest_path

Bu metot için Path adında yeni bir sınıf yazıldı. Bu sınıfın içerisinde yolun üzerinde bulunan düğümler ve bu yolun uzunluğu bulunur. En kısa mesafeyi bulmak için Dijkstras algoritmasından yaralanıldı.

● plot_graph

Bu metot çizgenin ekrana yazdırılabilmesi için yazıldı. Bu metodun çıktısında çizgede bulunan bütün düğümler için o düğüme bağlı diğer düğümler ok işareti kullanılarak gösterildi. Düğüme bağlı diğer düğümlerin o düğüme olan uzaklıkları parantez içinde gösterildi.

● is_connected

Bu metot iki düğüm arasında bir yol olup olmadığını bulmak için yazıldı. Bu metot yazılırken Breatdh-First Search algoritmasından faydalanıldı.

Answer of Question 4 :

Breadt-First Search Algoritması : Bu algoritmada ilk önce başlangıç nodundan başlanır. Daha sonra başlangıç nodunun komşuları, komşularının komşuları diye bütün nodelar ziyaret edilene kadar devam edilir. Ziyaret edilen nodlar ziyaret edildi diye işaretlenir. Bu algoritmada başlangıç noduna uzaklığı $k+1$ olan bir nodun ziyaret edilebilmesi için k nodun ziyaret edilmesi gerekir. Bu algoritma kullanılarak yönsüz çizgede en kısa yol bulunabilir.

Depth-First Search Algoritması : Bu algoritma derinlik bazlı bir algoritmadır. Bu algoritmada başlangıç noktasından başlanarak komşular ardından komşularının komşuları diyerek en derine kadar devam eder. En derine kadar gelindiğinde bu nod ziyaret edildi olarak işaretlenir. Ardından geriye doğru ziyaret edilerek devam edilir. Bu algoritmada iki türlü renk kullanılır. Biri keşfedilmiş nodlar için, diğeri ziyaret edilmiş nodlar içindir. Bu algoritmada başlangıç nodu en son ziyaret edilir.

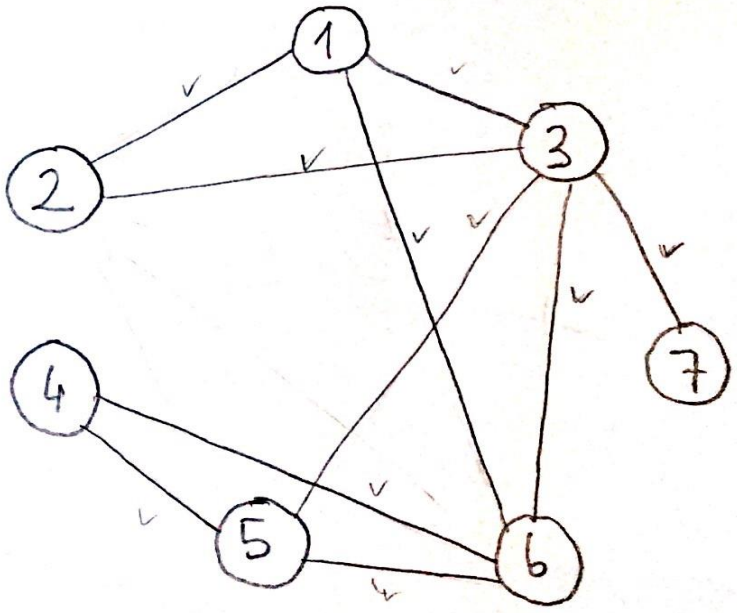
Bread-First Search Algoritmasının uygulama örnekleri:

- Bir node'de en kısa yolu bulma

Depth-First Search Algoritmasının uygulama örnekleri:

- Çizgelerin topolojik olarak sıralanması (Ders bağımlılıkları)

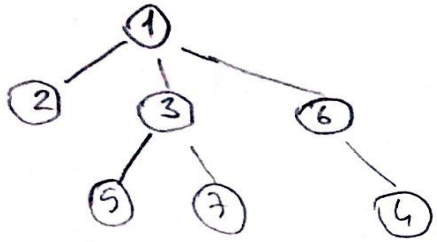
	1	2	3	4	5	6	7
1	1	1	0	0	0	1	0
2	1	1	1	0	0	0	0
3	0	1	1	0	1	1	1
4	0	0	0	1	1	1	0
5	0	0	1	1	1	1	0
6	1	0	1	1	1	1	0
7	0	0	1	0	0	0	1



a)

Visited Vertex	Que Contents after Visit	Visit Sequence
1	2 3	1
2	3	1 2
3	5 6 7	1 2 3
5	6 7 4	1 2 3 5
6	7 4	1 2 3 5 6
7	4	1 2 3 5 6 7
4	Empty	1 2 3 5 6 7 4

BFS Tree of Graph



b)

Operation	Adjacent Vertices	Discovery (visit) Order	Finish Order
Visit 1	2, 3	1	
visit 2	1, 3	1, 2	
Visit 3	1, 2, 5, 6, 7	1, 2, 3	
visit 7	3	1, 2, 3, 7	
Finish 7			7
Visit 6	1, 3, 4, 5	1, 2, 3, 7, 6	
visit 5	4, 6	1, 2, 3, 7, 6, 5	
Visit 4	5, 6	1, 2, 3, 7, 6, 5, 4	
Finish 4			7, 4
Finish 5			7, 4, 5
Finish 6			7, 4, 5, 6
Finish 3			7, 4, 5, 6, 3
Finish 2			7, 4, 5, 6, 3, 2
Finish 1			7, 4, 5, 6, 3, 2, 1

DFS Tree of Graph