Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 7 REPORT

EMİRHAN KARAGÖZOĞLU
151044052

Course Assistant: Fatma Nur Esirci

1 Q1

1.1 Problem Solution Approach

shortestPath methodunda Dijkstra algoritması kullanıldı. Dijkstra algoritması bir vertexten tüm vertexlere olan uzaklıkları bulmamızı sağlar. Biz ise bu algoritmadan yararlandık fakat bizim istediğimiz iki vertex arası en kısa mesafeyi bulmaktı. Bu yüzden bu algoritmadının sonucundan bize verilen bitiş noktasına olan uzaklık çekildi. Method shortest path deki vertexleri içeren bir vector return etmektedir.

1.2 Test Cases

Bu partta directed acyclic random weight 10 vertex ve 20 edge 'e sahip bir graph oluşturuldu. Bu graph plot_graph methodu ile adjacency list şeklinde ekrana bastırıldı. Bu graphın özellikleri is_undirected ve is_acyclic_graph methodları ile kanıtlandı. Daha sonra bu graph üzerinde 3 kez farklı vertexler arasında shortestPath methodu çalıştırıldı. Shortest path'ler ve weight değerleri ekrana bastırıldı.

```
          ---Graph---
|V:0 - D:0| -> |V:2 - D:2.0| -> |V:7 - D:7.0|
|V:1 - D:0| -> |V:2 - D:19.0|
|V:3 - D:0| -> |V:0 - D:18.0| -> |V:4 - D:12.0| -> |V:7 - D:13.0|
|V:5 - D:0| -> |V:1 - D:17.0| -> |V:4 - D:10.0| -> |V:8 - D:11.0|
|V:6 - D:0| -> |V:3 - D:6.0| -> |V:4 - D:5.0| -> |V:9 - D:7.0| -> |V:7 - D:1.0| -> |V:2 - D:6.0|
|V:7 - D:0| -> |V:2 - D:7.0|
|V:8 - D:0| -> |V:1 - D:24.0| -> |V:4 - D:25.0| -> |V:9 - D:22.0|
|V:9 - D:0| -> |V:2 - D:4.0| -> |V:4 - D:18.0|

-----------------------
Graph is directed.
Graph is acyclic graph.
-----------------------
Shortest path from 3 to 2
[3, 0, 2]
Distance from 3 to 2 = 20
-----------------------
Shortest path from 5 to 4
[5, 4]
Distance from 5 to 4 = 10
-----------------------
Shortest path from 4 to 2
There is no path between 4 and 2
-----------------------
```

2 Q2

2.1 Problem Solution Approach

is_connected methodunda amaç verilen graph üzerinde verilen iki vertex arasında herhangi bir path var mı yok mu onu kontrol etmemizdi. Bunu yapmak için DFS methodunu kullandım. Depth first search methodunun start vertexine bize verilen ilk vertexi, target vertexine de bize verilen ikinci vertexi vererek DFS methodunu çağırdım. DFS methodu arama işlemi başarılı olduğu takdirde true aksi halde false döndürmektedir. Buna göre start dan end e DFS yapıp buluyorsa bu iki vertex connected demektir.

2.2 Test Cases

Bu partta undirected acyclic no weight(default 1) 15 vertex 'e sahip bir graph oluşturuldu. Bu graph plot_graph methodu ile adjacency list şeklinde ekrana bastırıldı. Bu graphın özellikleri is_undirected ve is_acyclic_graph methodları ile kanıtlandı. Daha sonra bu graph üzerinde 3 kez farklı vertexler arasında is_connected methodu çalıştırıldı. Sonuçlar ekrana bastırıldı.

```
          ---Graph---
|V:0 - D:0| -> |V:3 - D:1.0|
|V:1 - D:0| -> |V:2 - D:1.0| -> |V:4 - D:1.0|
|V:2 - D:0| -> |V:1 - D:1.0| -> |V:3 - D:1.0|
|V:3 - D:0| -> |V:0 - D:1.0| -> |V:2 - D:1.0|
|V:4 - D:0| -> |V:1 - D:1.0| -> |V:7 - D:1.0|
|V:5 - D:0| -> |V:7 - D:1.0|
|V:6 - D:0| -> |V:7 - D:1.0|
|V:7 - D:0| -> |V:4 - D:1.0| -> |V:5 - D:1.0| -> |V:6 - D:1.0|
|V:8 - D:0| -> |V:9 - D:1.0|
|V:9 - D:0| -> |V:8 - D:1.0| -> |V:10 - D:1.0| -> |V:11 - D:1.0| -> |V:12 - D:1.0|
|V:10 - D:0| -> |V:9 - D:1.0| -> |V:13 - D:1.0|
|V:11 - D:0| -> |V:9 - D:1.0|
|V:12 - D:0| -> |V:9 - D:1.0|
|V:13 - D:0| -> |V:10 - D:1.0| -> |V:14 - D:1.0|
|V:14 - D:0| -> |V:13 - D:1.0|

----------------------
Graph is undirected.
Graph is acyclic graph.
----------------------
8 and 14 is connected!
----------------------
9 and 0 is not connected!
----------------------
5 and 2 is connected!
----------------------
```

3 Q3

3.1 Problem Solution Approach

Spannig tree bir graphtaki her vertex'e değmek koşulu ile minumum weight'e sahip sub graph oluşturmak demektir. Bunu gerçekleştirmek için BFS ve DFS methodlarından yararlandım. spanningTreeWithBFS methodunda BFS methodunu spanningTreeWithDFS methodun da ise DFS methodunu kullandım.

3.2 Test Cases

Bu partta undirected cyclic no weight(default 1) 10 vertex 'e sahip bir graph oluşturuldu. Bu graph plot_graph methodu ile adjacency list şeklinde ekrana bastırıldı. Bu graphın özellikleri is_undirected ve is_acyclic_graph methodları ile kanıtlandı. Daha sonra bu graph üzerinde spanningTreeWithBFS ve spanningTreeWithDFS methodları çağırıldı ve oluşan spanning treeler ekrana bastırıldı.

```
          ---Graph---
|V:0 - D:0| -> |V:1 - D:0.0| -> |V:4 - D:0.0| -> |V:7 - D:0.0| -> |V:13 - D:0.0|
|V:1 - D:0| -> |V:0 - D:0.0| -> |V:4 - D:0.0| -> |V:5 - D:0.0| -> |V:2 - D:0.0|
|V:2 - D:0| -> |V:1 - D:0.0| -> |V:3 - D:0.0| -> |V:6 - D:0.0|
|V:3 - D:0| -> |V:2 - D:0.0| -> |V:6 - D:0.0| -> |V:10 - D:0.0| -> |V:14 - D:0.0|
|V:4 - D:0| -> |V:0 - D:0.0| -> |V:1 - D:0.0| -> |V:7 - D:0.0| -> |V:8 - D:0.0|
|V:5 - D:0| -> |V:1 - D:0.0| -> |V:6 - D:0.0| -> |V:9 - D:0.0|
|V:6 - D:0| -> |V:2 - D:0.0| -> |V:3 - D:0.0| -> |V:5 - D:0.0| -> |V:10 - D:0.0|
|V:7 - D:0| -> |V:0 - D:0.0| -> |V:4 - D:0.0| -> |V:8 - D:0.0| -> |V:13 - D:0.0|
|V:8 - D:0| -> |V:4 - D:0.0| -> |V:7 - D:0.0| -> |V:11 - D:0.0| -> |V:9 - D:0.0|
|V:9 - D:0| -> |V:5 - D:0.0| -> |V:8 - D:0.0| -> |V:10 - D:0.0| -> |V:12 - D:0.0|
|V:10 - D:0| -> |V:3 - D:0.0| -> |V:6 - D:0.0| -> |V:9 - D:0.0| -> |V:14 - D:0.0|
|V:11 - D:0| -> |V:8 - D:0.0| -> |V:13 - D:0.0|
|V:12 - D:0| -> |V:9 - D:0.0| -> |V:14 - D:0.0|
|V:13 - D:0| -> |V:0 - D:0.0| -> |V:7 - D:0.0| -> |V:11 - D:0.0|
|V:14 - D:0| -> |V:3 - D:0.0| -> |V:10 - D:0.0| -> |V:12 - D:0.0|

-----------------------
Graph is undirected.
Graph is cyclic graph.
-----------------------
          ---Spanning Tree With BFS---
|V:0 - D:0| -> |V:1 - D:1.0|
|V:1 - D:0| -> |V:0 - D:1.0| -> |V:4 - D:1.0|
|V:2 - D:0| -> |V:5 - D:1.0| -> |V:8 - D:1.0|
|V:3 - D:0| -> |V:9 - D:1.0| -> |V:10 - D:1.0|
|V:4 - D:0| -> |V:1 - D:1.0| -> |V:7 - D:1.0|
|V:5 - D:0| -> |V:13 - D:1.0| -> |V:2 - D:1.0|
|V:6 - D:0| -> |V:11 - D:1.0| -> |V:9 - D:1.0|
|V:7 - D:0| -> |V:4 - D:1.0| -> |V:13 - D:1.0|
|V:8 - D:0| -> |V:2 - D:1.0| -> |V:11 - D:1.0|
|V:9 - D:0| -> |V:6 - D:1.0| -> |V:3 - D:1.0|
|V:10 - D:0| -> |V:3 - D:1.0| -> |V:12 - D:1.0|
|V:11 - D:0| -> |V:8 - D:1.0| -> |V:6 - D:1.0|
|V:12 - D:0| -> |V:10 - D:1.0| -> |V:14 - D:1.0|
|V:13 - D:0| -> |V:7 - D:1.0| -> |V:5 - D:1.0|
|V:14 - D:0| -> |V:12 - D:1.0|

          ---Spanning Tree With DFS---
|V:0 - D:0| -> |V:13 - D:1.0|
|V:1 - D:0| -> |V:4 - D:1.0|
|V:2 - D:0| -> |V:6 - D:1.0| -> |V:10 - D:1.0|
|V:3 - D:0| -> |V:14 - D:1.0| -> |V:6 - D:1.0|
|V:4 - D:0| -> |V:7 - D:1.0| -> |V:1 - D:1.0|
|V:5 - D:0| -> |V:10 - D:1.0| -> |V:7 - D:1.0|
|V:6 - D:0| -> |V:3 - D:1.0| -> |V:2 - D:1.0|
|V:7 - D:0| -> |V:5 - D:1.0| -> |V:4 - D:1.0|
|V:8 - D:0| -> |V:11 - D:1.0| -> |V:9 - D:1.0|
|V:9 - D:0| -> |V:8 - D:1.0| -> |V:12 - D:1.0|
|V:10 - D:0| -> |V:2 - D:1.0| -> |V:5 - D:1.0|
|V:11 - D:0| -> |V:13 - D:1.0| -> |V:8 - D:1.0|
|V:12 - D:0| -> |V:9 - D:1.0| -> |V:14 - D:1.0|
|V:13 - D:0| -> |V:0 - D:1.0| -> |V:11 - D:1.0|
|V:14 - D:0| -> |V:12 - D:1.0| -> |V:3 - D:1.0|
```

4 Q4

Differences Between BFS and DFS

1. BFS level level gezerken DFS derinlemesine gezer (leaf ' e ulaşana kadar yada visited olmayan bir vertex kalmayana kadar).
2. BFS unvisited vertexleri tutmak için Queue kullanırken DFS Stack kullanır.
3. BFS DFS 'den daha fazla memory kullanır.
4. BFS DFS 'den daha yavaş çalışır.
5. BFS 'nin kullanım alanlarına örnek olarak : P2P Networks, Social Networks ve GPS Navigation Systems verilebilirken ; DFS 'nin kullanım alanlarına da örnek olarak : Topological Sort , solving puzzle or maze which has one solution verilebilir.
6. Hem BFS hem DFS 'nin kullanım alanlarına örnek olarak shortest path ve minimum spanning tree bulma verilebilir.