Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 07 REPORT

CELAL TEMİZ

101044070

Course Assistant: Fatma Nur Esirci

1 Q1

1.1 Problem Solution Approach

Bu problemin çözümünde vertex sayısı 10, edge sayısı 20 ve random olarak verilmiş weight'leri olan bir directed – acyclic graf oluşturulup, istenen methodları testi sağlanmıştır.

Bu yapı Graph List yapısı üzerinde implement edilmiştir. Graph List yapısının oluşturulması için ders kitabından faydalanılmış, referans verilmiştir.

Problem çözümünde kullanılan AbstractGraph, DepthFirstSearch ve Graph interface ders kitabına aiitir. Edge classı da ders kitabına bakılarak implement edilmiştir.

plot_grap() methodu ile istenilen liste şeklinde graph yapısı görselleştirilmiştir.

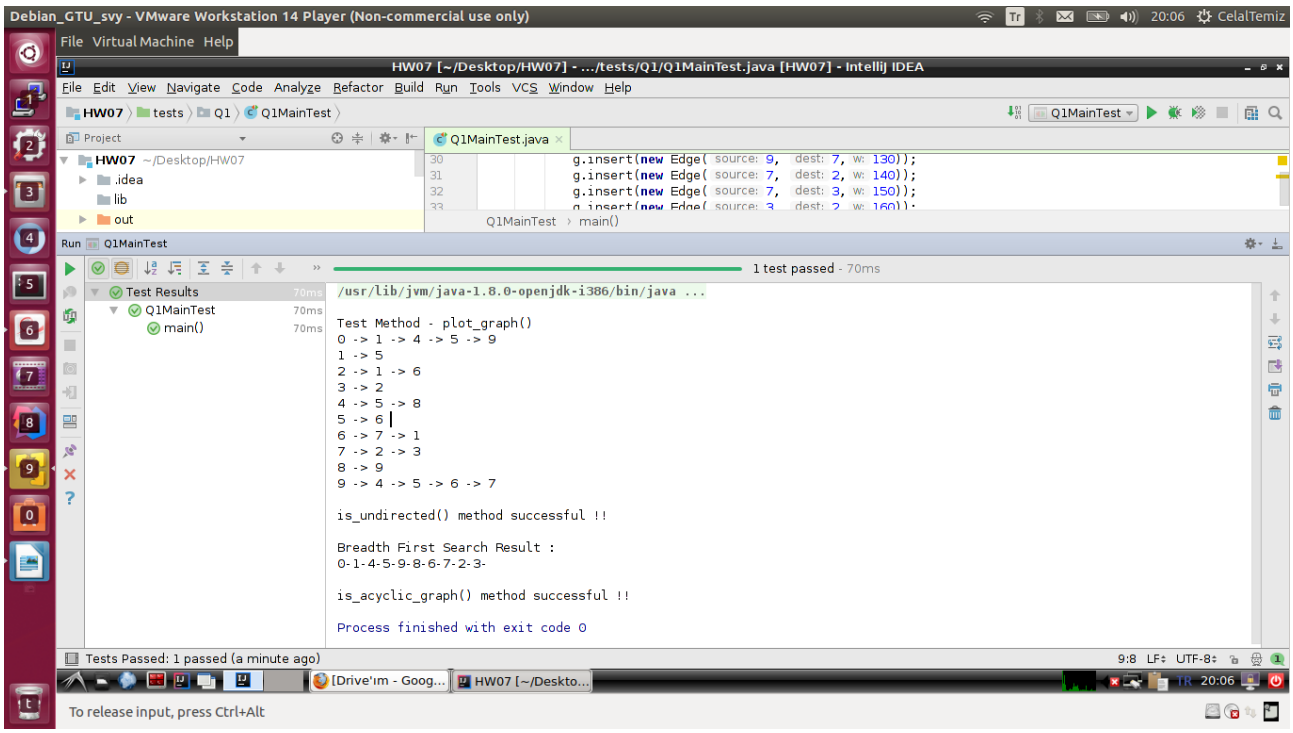
is_undirected() methodu ile graph üzerinde birbirine bağlı tüm edge'lerin sayısı hesaplanmış ve vertex sayısı ile aralarındaki bağıntı ile ilişkilendirilerek implement edilmiştir.

is_acyclic_graph() methodu içerisinde breadth-first search algoritmasını kullanılmıştır. Bu algoritmanın breadtg-first search işlemini yaparkeni parent bilgisi bir dizide tutulup, return edilmektedir. İki farkli vertex' in aynı parent'a sahip olması ve bu vertex' lerin birbirine bağlı olması halinde graph yapısında bir cycle olduğu tespit edilmektedir.
Breadth-first search algoritması ders kitabından alınmıştır

shortest_path() bu method için dijkstra's algoritması kitaptan alınmış, method tamamen implement edilmemiştir.

1.2 Test Cases

Bu problemim çözümü için main test ve JUNIT main test yazılmıştır.
Proje içerisinde tests klasöründe yer almaktadır.



2 Q2

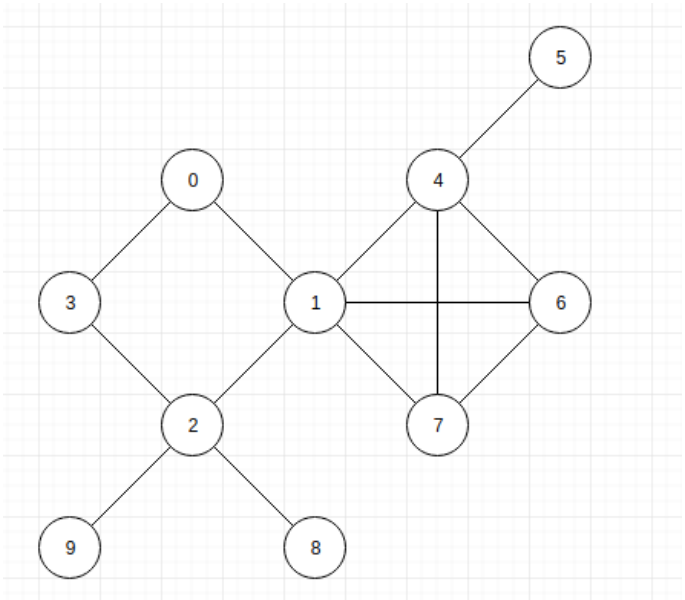
2.1 Problem Solution Approach

Bu problemin çözümünde vertex sayısı 15, weight bilgisi olmayan olan bir undirected – acyclic graf oluşturulup (Şekil 2.1.1), istenen methodları testi sağlanmıştır.

Bu prolemin çözümünde önceki problemden farklı olarak 1 method eklenmiştir.

is_connected() methodunda öncelikle verilen vertex'lerin graph içerisinde olup olmadığı bir checkVertexInGraph() helper methodu ile kontrol edilmektedir.

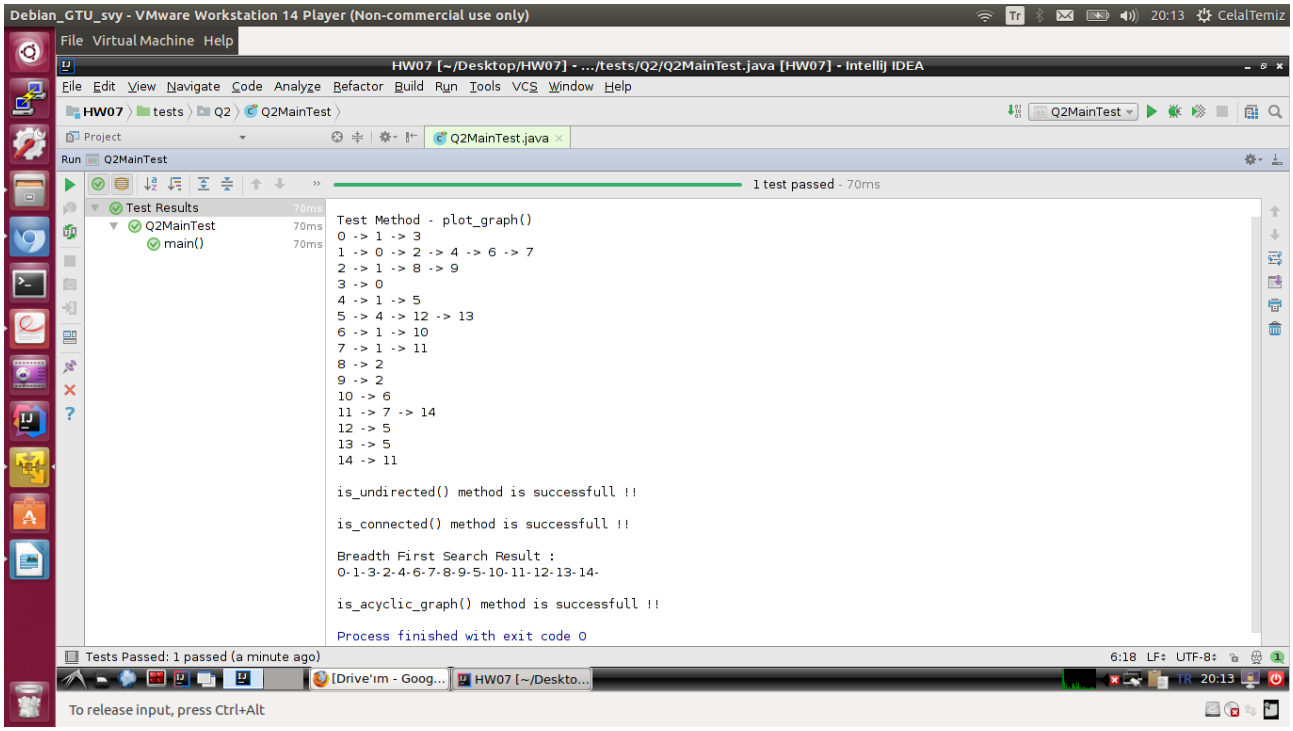
Eğer her iki vertex graph' ın elamanı ve 1. vertex' in destination'ı 2. vertex ise bu durumda bu vertex' lerin birbirine bağlı olduğu anlaşılmıştır.



Şekil 2.1.1

2.2 Test Cases

Bu problemim çözümü için main test ve JUNIT main test yazılmıştır. Proje içerisinde tests klasöründe yer almaktadır.



Şekil 2.2.1 – JUNIT Main Test

3 Q3

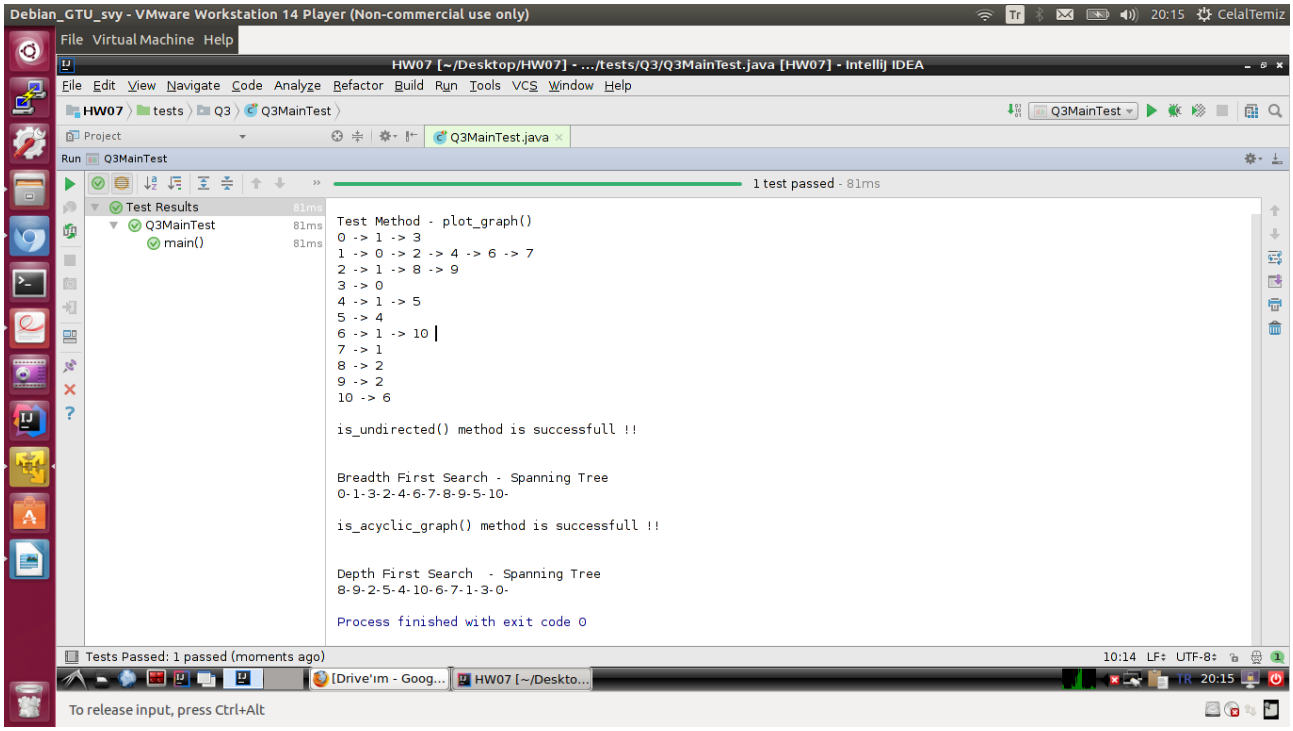
3.1 Problem Solution Approach

Bu problemin çözümünde vertex sayısı 10, weight bilgisi olmayan olan bir undirected – cyclic grapf oluşturulup, istenen methodları testi sağlanmıştır.

Breadth-first search ve depth-first search algoritmaları ders kitabında alınarak çalıştırılmış ve spanning tree ekrana yazdırılmıştır.

3.2 Test Cases

Bu problemim çözümü için main test ve JUNIT main test yazılmıştır. Proje içerisinde tests klasöründe yer almaktadır.

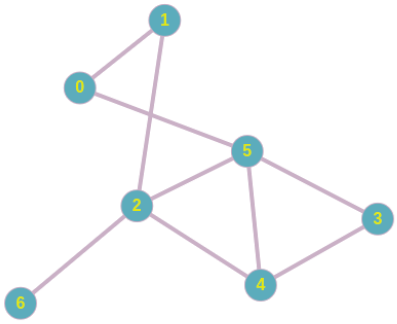


Şekil 3.2.1 – JUNIT Main Test

4 Q4

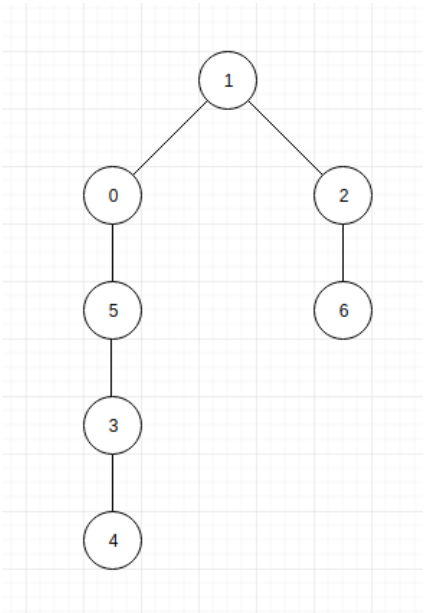
- BFS, Queue veri yapısını kullanıp, DFS, Stack veri yapısını kullanır.
- BFS, DFS algoritmasına göre daha fazla memory kullanır.
- BFS, yavaş bir algoritma olup DFS hızlı bir algoritmadır.
- BFS herhangi iki vertex arasında en kısa yolu bulma açısından elverişlidir. Optimal çözüm üretir. DFS en uzun yolu traverse ettiği için bu açıdan verimsizdir.
- Graph çok uzun ve vertex sayısı fazla ise, DFS çok uzun zaman alır. Bu durumda BFS daha iyi bir algoritmadır.
- Aranan vertex, graph' ın en uzak konumunda ise BFS algoritması bu durum için elverişli değildir.
- BFS en kısa yol bulma, peer to peer network' lerde komşulukları bulma, sosyal medyada belli bir uzaklıktaki arkadaşlıklara ait bilgiler tutulmasında, gps navigasyon yazılımlarında kullanılır.
- DFS, unweighted graph' larda minimum spanning tree bulma, graph içerisinde cycle bulma,bipartite graph durumuna bakma, verilen iki konum arasında bir yol bulma gibi uygulamarda kullanılır.

Verilen adjacency matrix' e ait graph şekildeki gibidir. Self loop' lar söylendiği gibi alınmamıştır.



Şekil 4-1

a) DFS Tree



b) BFS Tree