

**ELEKTRONİK VE OTOMASYON BÖLÜMÜ**

**MOBİL TEKNOLOJİLERİ**

**Yapay Zeka Final Raporu**

**Konu: En Kısa Yolu Bulma**

**Kullanılan Platform: NetBeans**

**Kullanılan Programlama Dili: Java**

**GitHub:** <https://github.com/kayrayilmazer/yapayzeka>

**Hazırlayan:**

**Kayra Yılmazer**

**18MY93021**

**Danışman: Öğr.Gör.Nilgün İNCEREİS**

**İçindekiler**

1. Genel Bilgiler (Proje Konusu)
   1. Projeye Genel Bakış
   2. Proje Yazılma Platformu
   3. Proje Kullanılan Dil
   4. Kullanılan algoritma
2. Dijkstra Algoritması Nedir?
   1. Belman Ford Algoritması
3. Proje Yapım Aşaması
   1. Algoritma
   2. Proje Anlatımı
   3. Proje Kodları
4. Kaynakça

**Proje Konusu**

Proje, En kısa yolun bulunmasını sağlayan bir program yapmak. Ben Netbeans ortamında java dilini kullanarak yazdım. Verilen yollardan en kısa yolu bulmasını dijkstra algoritmasını kullanarak yaptım.

**Dijkstra Algoritması Nedir**

Ağırlıklı bir grafikte bir başlangıç düğümünden bir hedef düğüme en kısa yolu bulmak için bir algoritma Dijkstra'nın algoritmasıdır. Algoritma, başlangıç köşesinden, kaynaktan grafikteki diğer tüm noktalara kadar en kısa yollardan oluşan bir ağaç oluşturur.

Algoritmayı kullanmanın bir şartı, grafiğin her kenarda negatif olmayan bir ağırlığa sahip olması gerektiğidir. Bir öğrencinin evden okula mümkün olan en kısa şekilde gitmek istediğini varsayalım. Bazı yolların yoğun şekilde sıkışık ve kullanımı zor olduğunu biliyor. Dijkstra'nın algoritmasında bu, kenarın büyük bir ağırlığa sahip olduğu anlamına gelir-algoritma tarafından bulunan en kısa yol ağacı, daha büyük ağırlıklara sahip kenarlardan kaçınmaya çalışacaktır. Öğrenci bir harita hizmeti kullanarak yol tarifleri ararsa, Dijkstra'nın algoritmasını ve diğerlerini de kullanabilirler.

Dijikstra algoritmasından başka Bellman Ford Algoritmasıda vardır. Bunun algoritması ise şöyle işler. Bütün düğmelere sonsuz mesafe atanır. Her düğme ve her kenar için. Düğümün üzerindeki mesafeden daha küçük mesafe bulduysa. Düğüm mesafesini ve düğüme gelinen düğümü günceller.

**Algoritma**

1) En kısa yol ağacında yer alan köşeleri takip eden bir sptSet (en kısa yol ağacı seti) oluşturuyorum, yani kaynaktan minimum uzaklığı hesaplanır ve sonuçlandırılır. Başlangıçta, bu set boştur.

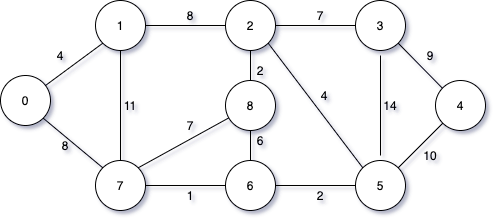
2) Giriş grafiğindeki tüm köşelere bir mesafe değeri atıyorum. Tüm mesafe değerlerini sonsuz olarak başlatıyorum. Kaynak köşe için mesafe değerini 0 olarak atıyorum, böylece önce seçilmesini sağlıyorum.

3) Sptset tüm köşeleri içermez.

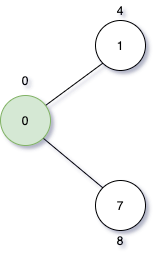
a) sptset'te olmayan ve minimum mesafe değerine sahip bir köşe u seçiyorum.

b) sptset'e U ekliyorum.

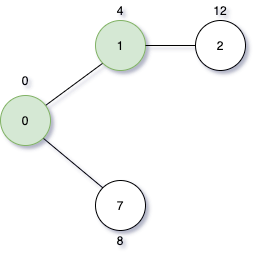
C) U'nun tüm bitişik köşelerinin mesafe değerini güncelliyorum. Mesafe değerlerini güncellemek için, tüm bitişik köşelerde yineleme yapıyorum. Her bitişik köşe v için, u (kaynaktan) mesafe değerinin toplamı ve u-V kenarının ağırlığı, V'nin mesafe değerinden daha az ise, V'nin mesafe değerini güncelliyorum.



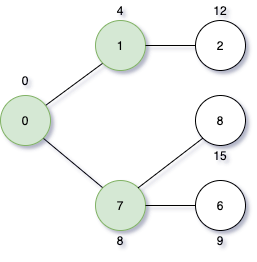
Küme sptSet başlangıçta boştur ve köşelere atanan mesafeler {0, INF, INF, INF, INF, INF, INF} burada INF sonsuz gösterir. Şimdi minimum mesafe değeri ile köşe seçiyorum. Vertex 0 seçilir, sptset'e dahil edilir. Böylece sptSet {0} olur. 0'dan sptset'e dahil edildikten sonra, bitişik köşelerinin mesafe değerlerini güncellenir. 0'ın bitişik köşeleri 1 ve 7'dir. 1 ve 7 mesafe değerleri 4 ve 8 olarak güncellenir. Aşağıdaki subgraph Köşeleri ve mesafe değerlerini gösterir, sadece sonlu mesafe değerlerine sahip köşeler gösterilir. SPT'DE yer alan köşeler yeşil renkte gösterilir.



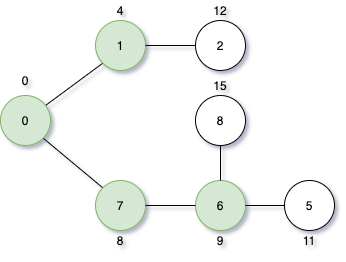
Minimum mesafe değeri olan tepe noktasını seçeriz ve zaten SPT'YE dahil edilmemiştir (sptset'de değil). Vertex 1 alınır ve sptset'e eklenir. Böylece sptSet şimdi {0, 1} olur. 1'in bitişik köşelerinin mesafe değerlerini güncelleriz. Vertex 2'nin mesafe değeri 12 olur.



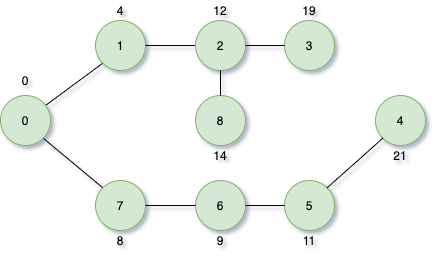
Minimum mesafe değeri olan tepe noktasını seçeriz ve zaten SPT'YE dahil edilmemiştir (sptset'de değil). Vertex 7 seçildi. Böylece sptSet şimdi {0, 1, 7} olur. 7 bitişik köşelerin mesafe değerlerini güncelleriz. Vertex 6 ve 8'in mesafe değeri sonlu olur (sırasıyla 15 ve 9).



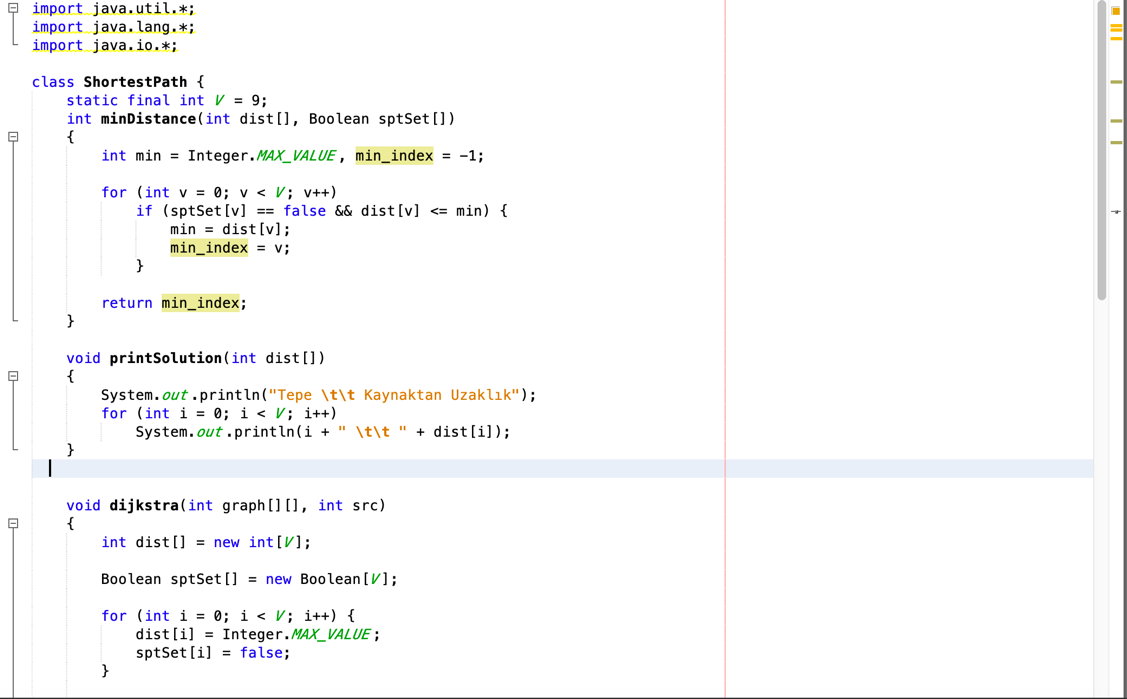
Minimum mesafe değeri olan tepe noktasını seçeriz ve zaten SPT'YE dahil edilmemiştir (sptset'de değil). Vertex 6 seçeriz. Yani sptSet şimdi olur {0, 1, 7, 6}. 6 bitişik köşelerin mesafe değerlerini güncelleriz. Vertex 5 ve 8'in mesafe değeri güncellenir.

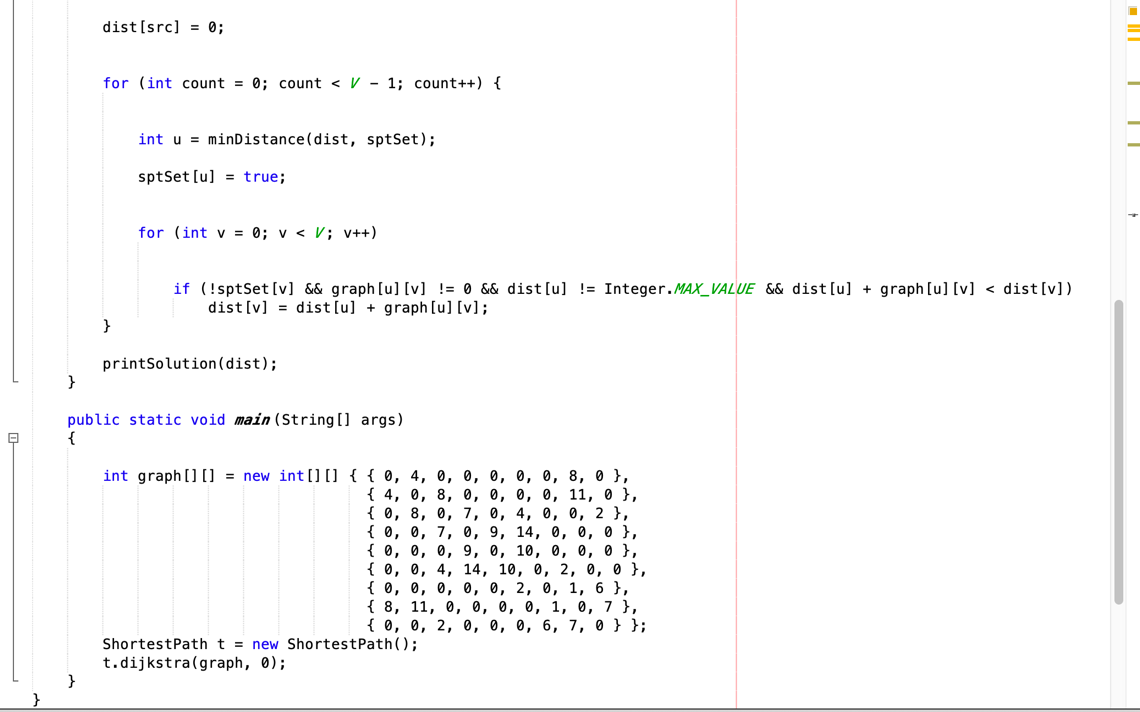


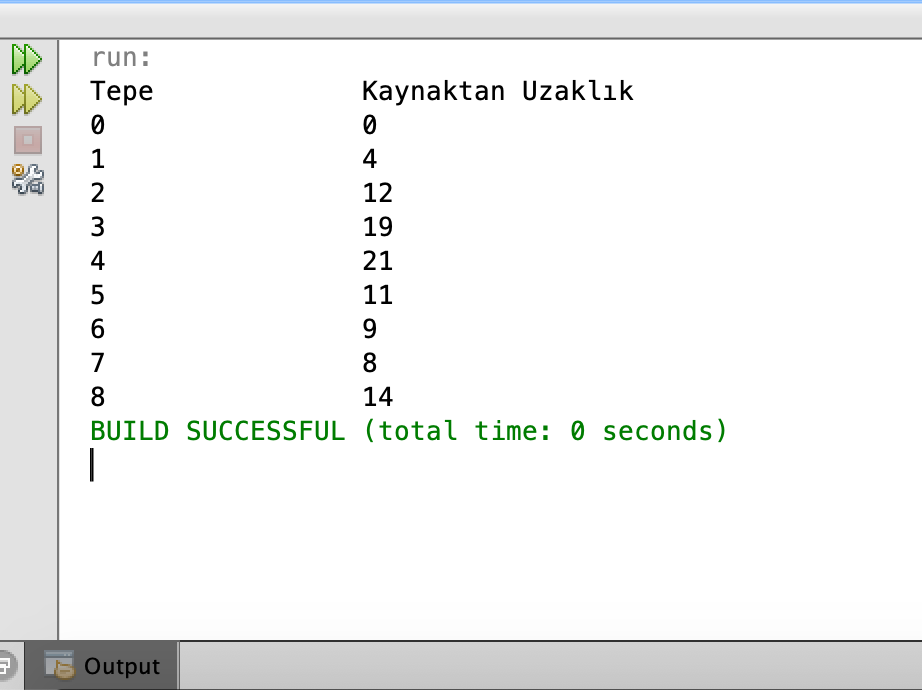
SptSet verilen grafiğin tüm köşelerini içerene kadar yukarıdaki adımları tekrarlıyoruz. Son olarak, aşağıdaki en kısa yol ağacını (SPT) elde edeceğiz.



**Kodlar**







**Kaynakça**

[www.geeksforgeeks.org/](http://www.geeksforgeeks.org/)