GEZGİN KARGO PROJESİ

Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği

Taha Yıldız
tahayildiz515@gmail.com
180201019

Ahmet Furkan Kaşıkçı
furkankasikci45@gmail.com
180201030

ÖZET

Bu projede amaç veri yapıları,graf yapısı, algoritma mantığını kullanarak probleme çözüm bulmaktır

Anahtar Kelimeler:

Veri Yapıları, En Kısa Yol ,Graf Yapısı

1.Giriş

1.1 Yapım Aşamaları:

Projede istenilen diller arasından JAVA seçildi daha sonra sonra hangi algoritmanın daha uygun olacağı belirlendi. Bu yönde Dijkstra Algoritması seçildi. İnternetten kod örnekleri arayışına geçildi. Örnek kod bulundu.Bu kodumuzda **"dijkstra", "yazCozum", "yazYol"** İsimli üç fonksiyon var**. "yazYol"** fonksiyonunda iki şehir arası giderken hangi illerden geçildiği yazdırılmakta ve bu fonksiyon **"yazCozum" '**da kullanılmakta. **"yazCozum"** fonksiyonu hangi iki şehir arası gidildiğini yazdırır ve bu iki şehir arası mesafeyi hesaplar,bunu döndürür. **"yazCozum"**ve **"yazYol"** yazma işlemlerini açtıkları **"output-all.txt"** dosyasına yaparlar.Bu dosyaya programın bulduğu bütün yollar yazdırılır. **"dijkstra"** ise dijkstra algoritmasını kullanarak hangi illerden gidileceğini belirler ve **"yazCozum"** kullanarak hem yolu yazdırır hem de güzergahın mesafesini hesaplar**.**

"Permutasyon" sınıfı oluşturuldu. Bu sınıf içerisinde **"permute1"** ve **"permute2"** adlı fonksiyon kodlanarak algoritmanın birden fazla şehre gidebilmesi sağlandı.Bu da permutasyon alınarak sağlandı. **"permute2" '** nin **"permute1" '**den farkı beş şehirden fazla şehir girildiği zaman belli sayıda güzergahı bulduktan sonra durması.İki fonksiyon da parametre olarak gidilecek şehirlerin plaka kodlarını içeren diziyi,bu dizide ilk gidilecek şehrin indisini ve güzergahların mesafelerini kaydedecek diziyi alıyor **."MatrisOkuma"** sınıfı oluşturuldu.Bu sınıf içerisinde **"matrisOku"** adlı fonksiyon oluşturularak **"sample.txt"** içindeki uzaklıkların okunup parametre olarak alınan matrisin içine atılması sağlandı. Bu fonksiyon **"permute1"** 'de ve **"permute2" '** de de çağrılarak **"permute1"** 'de ve **"permute2" '** de oluşturulan **"myArray"** adlı matrise uzaklıkların atılması sağlandı. **"DijkstraAlgorithm"** sınıfı içerisinde statik, integer 81 eleman kapasiteli **"yollar"** dizisi oluşturuldu. Güzergahda olan her il için yollar dizisinde o ilin plaka numarasının bir eksiğindeki indisde geçiş sayısı tutuldu ve o güzergâhtaki o il yazdırıldığında **"yazYol"** 'da parantezde geçiş sayısı da yazdırıldı. **"dijkstra" '**da o güzergahın yazımı bittikten sonra bi dahaki güzergahda aynı geçişlerin üstüne saymaması için yollar dizisi sıfırlandı.
Daha sonra **"iller.txt"** adlı dosya oluşturuldu. Bu dosyaya satır satır plaka sırasına göre il adları yazıldı.

Daha sonra **“DijkstraAlgorithm”** sınıfı içerisinde **“yazma”** adlı void fonksiyon oluşturuldu. Parametre olarak string bir dizi alan bu fonksiyon **“iller.txt”** içindeki il adlarını okuyup parametre olarak aldığı string dizisine atıyor. Bununla plaka numaralarının değil de şehir isimlerinin yazdırılması amaçlanıyor. Güzergah mesafelerinin tutulması için bir **“toplamDizi”** adlı bir dizi oluşturuluyor. Bu dizinin kapasitesi ne kadar şehir girilecekse onun faktöriyeli alınarak belirleniyor. Bunu hesaplamak içinse **“DijkstraAlgorithm”** sınıfı içinde **“fak”** isimli integer bir fonksiyon oluşturuluyor. Daha sonra indisi sayması için gene **“DijkstraAlgorithm”** sınıfı içinde static,integer,başlangıç değeri sıfır olan **“sayac”** adlı bir değişken tanımlıyoruz. Bu değişken **“toplamDizi”** nin uzunluğuna ulaştığında **“toplamDizi”** yi **“enKisa”** adlı fonksiyona gönderdi. Bu fonksiyonda **“toplamDizi”** sıralandı. Bundan sonra **“toplamDizi”** nin sıfırıncı indisi yani en kısa mesafe ve **“toplamDizi”** gene aynı sınıf içerisindeki **“okuBakiyim”** adlı fonksiyona gönderildi. **“okuBakiyim”** ’de **“EnKisaYol.txt”** **“EnKisa5Yol.txt”** isimli iki dosya açılıyor. Burda en kısa mesafe ve en kısa beş mesafe integerdan stringe dönüştürülüyor. Sonra bütün yolların yazılı olduğu **“output-all.txt”** dosyası okunuyor.Bundan sonra 2 eleman kapasiteli string bir dizi tanımlanıyor.Bu bir dizi döngü içerisinde anlık olarak **“output-all.txt”** içindeki verileri satır satır tutuyor.Alt alta yazıldığı için sıfırıncı indis yolu,birinci indis mesafeyi tutuyor. Eğer bu döngüde birinci indisle stringe dönüştürülen **“kisaMesafe”** eşleşirse o andaki sıfırıncı indis ve birinci indis **“EnKisaYol.txt”** dosyasına yazılıyor.Aynı işlemler en kısa beş yol için de yapılıyor.Fark olarak beş yol için beş kez dönen bir döngü daha kullanılıyor**.**

“Main” fonksiyona gelirsek ilk önce il adlarını string **“iller”** dizisine kaydediyoruz. Daha sonra kullanıcıdan plaka girmek için bir girmesini, şehir ismi girmesi için iki girmesini istiyoruz. Sonrasında kaç şehre gidileceğinin bilgisi kullanıcıdan isteniyor. Buna göre şehir plakaları **“istenen”** adlı integer dizide bulunurken, şehir isimleri **“istenenSehir”** isimli String dizide bulunuyor. Kocaeli’den başlayıp Kocaeli’ye dönmemiz gerektiği için **“istenen” ’**in ilk ve son elemanı plaka numarasının bir eksiği olan kırk olurken**, “istenenSehir” ‘**in ilk ve son elemanı “**Kocaeli**” olmaktadır bundan dolayı bu dizilerin kapasitesi de kullanıcının girdiği şehir sayısının iki fazlasıdır.Sonrasında kullanıcı neyi seçti ise ona göre şehir plakaları veya şehir isimleri alınıyor. Uzaklıkları tuttuğumuz matrisde sıfırıncı indisden başladığımız için uzaklıkları tutmaya kullanıcıdan plakaları alırken aldığımız plakaları bir azaltarak **“istenen”** dizisine atılıyor. Şehir isimlerini seçimine geldiğimizde ise şehir isimlerini kullanıcıdan aldıktan sonra **“istenenSehir”** dizisi içiçe döngüye sokuluyor.Bu döngüde ilk başta tüm şehir isimlerin kaydededilen **“iller”** dizisiyle bir kontrole tabi tutuluyor. Bu kontrolde şehir isimleri eşse **“iller”** ’in o şehrinin indisi **“istenen”** dizisine atılıyor. Daha sonrasında kullanıcının girdiği şehir sayısı beşe eşit veya beşten azsa mesafelerin tutulduğu **“toplamDizi”** dizisinin kapasitesi **“fak”** fonksiyonu sayesinde ne kadar şehirse onun faktöriyeli alınarak verilir ve **“permute1”** fonksiyonuna **“istenen”, “toplamDizi”,** ilk olarak istenendeki birinci indise gidileceği için bir de gönderilir. **“permute1”** değişim işlemleri rekürsif şekilde her seferinde başlangıç kısmı bir artırılarak yapılır en sonunda başlangıç **“istenen”** dizisinin kapasitesine eşit olduğu zaman **“dijkstra”** fonksiyonuna gönderilir. Beşten büyükse de **“permute2” ‘**ye gönderilir. Burdaki **“toplamDizi” ’**nin boyutu **“toplamDiziBoyutu”** fonksiyonula bulunur

1.2 Karşılaşılan Sorunlar

En kısa mesafenin yolunu yazdırmada sorunla karşılaştık. Bir bütün olarak yolun yazdırıldığı **“yazCozum”** fonksiyonu integer bir değer döndüren bir fonksiyon olduğu için böyle bir sorunla karşılaştık. Dediğimiz gibi program içinde integer olan bu fonksiyondan yazıları çekmenin yolunu bulamadık en sonunda bulduğumuz **“output-all.txt”** de bütün yollar yazdırıldıktan sonra daha önceden bildiğimiz en kısa mesafeyle dosya içindeki yazan mesafelerle karşılaştırıp onun yolunu yazdırmak oldu.Kırk beş saniye kuralından dolayı da beş şehirden fazla şehir girilen durumlar için **“permute2”** fonksiyonunu yazmak durumunda kaldık.

2.Çalışma Adımları

1-)

```
run:
Plaka girmek icin 1, Sehir ismi girmek icin 2
```

2-)

```
run:
Plaka girmek icin 1, Sehir ismi girmek icin 2
1
Kaç şehre gidilecek:
```

3-)

```
run:
Plaka girmek icin 1, Sehir ismi girmek icin 2
1
Kaç şehre gidilecek:
3
Gidilmek istenen il plakası girin:
```

4-)

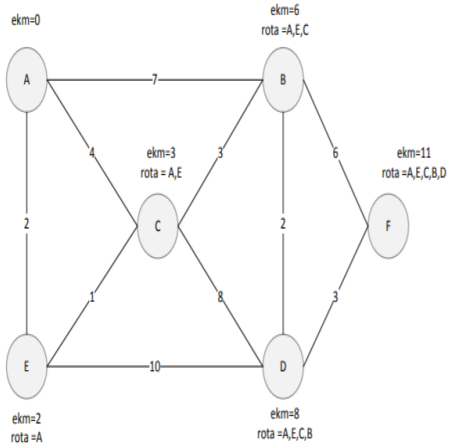
```
Gidilmek istenen il plakası girin:
67
18
26
*********************************
EN KISA YOL:
Kocaeli (0.gecis) => Sakarya (1.gecis) => Duzce  (1.gecis) =>
toplam mesafe: 1094
------------------------------------------------------------------

En kisa yol EnKisaYol.txt dosyasina yazdirildi.

En kisa 5 yol EnKisa5Yol.txt dosyasina yazdirildi.
```

3.Dijkstra Algoritması

Dijkstra algoritması, bir düğümden diğer tüm düğümlere en kısa yolları hesaplayan ve belirli bir başlangıç noktasına göre en kısa yolu belirleyen algoritmadır. Graflar için geliştirilmiş olup kenarların ağırlık değeri sıfır ya da pozitif olmalıdır. Eğer kenarların değerleri negatif değerse diğer algoritmalardan Bellman-Ford kullanılabilir. Dijkstra algoritmasının zaman karmaşıklığı $O(MlogN)$ şeklinde hesaplanmıştır. Dijkstra algoritması en kısa yolu belirlerken Greedy3 yaklaşımını kullanır. Yani bir düğümden diğer bir düğüme geçerken olası en iyi çözümü kullanır.Bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılır. Şekillerde Maliyetli graf ve Komşuluk matrisi yer almaktadır. Buna Dijkstra algoritması ile en kısa yol probleminin çözülmesi sağlanmıştır.



A	A	B	C	D	E	F
A	-1	7	4	-1	2	-1
B	7	-1	3	2	-1	6
C	4	3	-1	8	1	-1
D	-1	2	8	-1	10	3
E	2	-1	1	10	-1	-1
F	-1	6	-1	3	-1	-1

4.Sözde Kod

-Boyutu seksen bir olan iller adında dizi tanımla.

-Diziyi yazma fonksiyonuna gönder.

-Ekrana “Plaka girmek için 1,Şehir ismi girmek için 2” yaz.

-Kullanıcıdan girişi al.

-Ekrana “Kaç şehire gidilecek” yaz.

-Kullanıcıdan girişi al.

-“istenen” adında dizi tanımla.

-“istenenSehir” adında dizi tanımla.

-Eğer bir girdiyse ekrana “Gidilmek istenen il plakasını girin” yaz ve kullanıcıdan girilen şehir sayısı kadar giriş al ve “istenen” dizisine at.

-Eğer iki girdiyse ekrana “Gidilmek istenen il ismini girin” yaz ve kullanıcıdan girilen şehir sayısı kadar giriş al ve “istenenSehir” dizisine at.

-**“**iller” dizisiyle “istenenSehir” dizisini karşılaştır.

-Eğer eşit olan indisler varsa o “iller” indisini “istenen” dizisine at.

-Eğer girilen şehir sayısı beşten azsa “permute1” fonksiyonuna git.

-Değilse “permute2” fonksiyonuna git.

5.Kaynakça

1-)
https://www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/

2-)

http://www.zafercomert.com/Medya/2015_05_11_2_121_69f9a888.pdf

3-)

https://www.educative.io/edpresso/what-is-dijkstras-algorithm