Gezgin Kargo Problemi Proje Raporu

Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
Aybüke Özlem ULUTAŞ-Batuhan AKYOL
180202110-180202140

ÖZET:Nesneye yönelik programlama özelliklerini (dosya okutma,interface,extend) taşıyan ,graphları kullanarak en kısa yol algoritmasını geliştirerek bir JAVA uygulaması oluşturduk.

Anahtar Kelimeler – Graph,Jframe,Random

I. GİRİŞ

Programımızda bir klasör içinde bulunan .txt uzantılı dosyadan 81 şehrin komşuluklarını ve bunlara olan uzaklıklarını alıp ArrayList'e ekliyoruz.Daha sonra kullanıcının seçtiği illeri ArrayList'en çekip dijkstra algoritmasını kullanıp en kısa yolu buluyoruz.

TABLO I
DOSYALRIN FORMATI

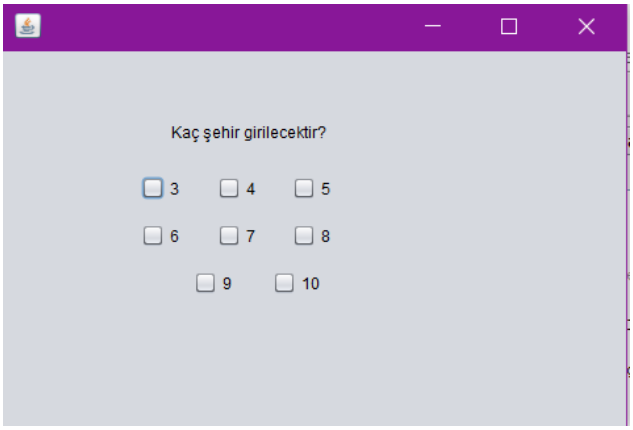
# Sehirler dosya formatı	#Noktalar dosya formatı
Adana,Hatay,191,Osmaniye,87,Kahramanmaras,192,Kayseri,335,Nigde,207,Mersin,69 Adiyaman,Sanliurfa,112, Diyarbakir,207,Malatya,187,Kahramanmaras,163, Gaziantep,150	Agri,884,173 Amasya,517,134 Ankara,367,180 Antalya,251,378 Artin,813,82 Aydin,103,308 Balikesir,113,190 Bilecik,221,163
Dosya 1	Dosya 2

II. YÖNTEM

Programımızı JAVA programlama dili ile geliştirdik. Kodumuzda ArrayList yapısı,for-while döngüleri, if-else koşul durumları,Jframe yapısı ve bazı özel fonksiyonlar kullanarak programımızı geliştirdik.

III. DENEYSEL SONUÇLAR

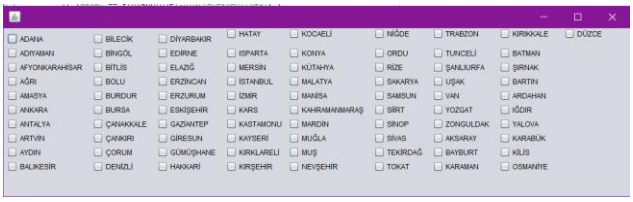
İlk başta kullanıcı gidilecek sehir sayısını girmektedir. Ardından şekil.1 de ki gibi bir menü karşısına çıkmaktadır.



Şekil.1 Kullanıcının karşısına çıkan pencere.

Kullanıcı sehir sayısını seçtikten sonra gidilecek yerleri seçmesi için karşısına 81 şehirlik bir pencere daha çıkar.

Seçtiği şehir sayısı kadar şehir seçebilir.Bu pencere Şekil.2 deki gibidir.



Şekil.2 Kullanıcının şehir seçimleri.

Örneğin eger kullanıcı 3 şehir seçerse arkada program rastgele bir rota oluşturur.Bu rakamların herbiri seçilen bir şehri temsil eder .Ancak 0 her zaman Kocaeli'ni temsil eder çünkü Kocaeli her zaman başlangıç ve bitiş noktasıdır.Arkada çalışan kodun muhtemel çıktısı Şekil.3 teki gibidir.Bu kısım Random(); methodu ile bulunuyor.

```
compile-single:
run-single:
03210
01320
02310
03120
01230
02130
```

Şekil.3 3 şehir seçilince hesaplanan rota

Kullanıcı n tane şehir seçtiğinde n! kadar farklı rota bulunur . Şekil 4 deki rakamlar ise n sehir için n! kadar yolların uzunluklarıdır..2şer tane aynı değer olma sebebi ise A şehrinden B şehrine giderken ve dönerkenki rota aynı uzunluktadır. Bu yüzden bizim en kısa 2 tane rotamız vardır .Kullanıcı 3 şehir seçtiğinde 3! 'den 6 rota oluşur ve bu 6 rotanın uzunluğu Şekil 4 deki gibi küçükten büyüğe doğru sıralanır.

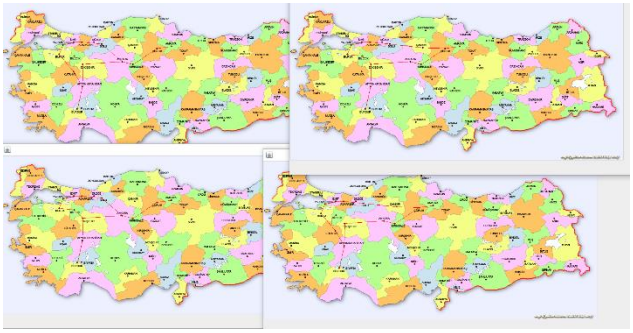
```
compile-single:
run-single:
[2465, 2465, 2550, 2550, 3609, 3609]
```

Şekil.4 Kullanıcının farklı n! uzaklıkları

Kullanıcı gidilecek sehirleri seçtikten sonra oluşturalan rotalar bu sefer şehir şeklinde en kısa rotadan en uzun rotaya doğru beş rota şeklinde şekil.5 teki gibi yazdırılır.

```
EN KISA ROTA
Kocaeli,Sakarya,Bilecik,Kutahya,Afyonkarahisar,Afyonkarahisar,Konya,Mersin,Adana,Adana,Kahramanmara

EN KISA 2.ROTA:
Kocaeli,Sakarya,Bolu,Ankara,Kirsehir,Nevsehir,Kayseri,Kahramanmaras,Adiyaman,Adiyaman,Kahramanmaras

EN KISA 3.ROTA:
Kocaeli,Sakarya,Bolu,Ankara,Aksaray,Nigde,Adana,Adana,Kahramanmaras,Adiyaman,Adiyaman,Kahramanmaras

EN KISA 4.ROTA:
Kocaeli,Sakarya,Bilecik,Kutahya,Afyonkarahisar,Afyonkarahisar,Konya,Aksaray,Nevsehir,Kayseri,Kahram

EN KISA 5.ROTA:
Kocaeli,Sakarya,Bolu,Ankara,Aksaray,Nigde,Adana,Adana,Mersin,Konya,Afyonkarahisar,Afyonkarahisar,Ko
```

Şekil.5 Kullanıcının girilen sehirlerle oluşturulan rotaların ekran çıktısı.



Şekil.6 5 farklı rotanın çizdirildiği pencere.

Rotalar belirlendikten sonra kullanıcı tarafından seçilen şehirler arasındaki en kısa mesafeli 5 rota 5 farklı pencere üstünde görüntülenir.

IV. YALANCI KOD

Sehirler ve komsulukların bulunduğu dosyayı okuma kodu:

```
File dosya2 = new File("prolab.txt");
Scanner dosya = null;

try {
        if (!dosya2.exists())
{          System.out.println("dosya yok");
            dosya2.createNewFile();          } else {
            dosya = new Scanner(dosya2);
            while (dosya.hasNextLine())                String
cumle = dosya.nextLine();String[] s = cumle.split
              sehirler81.add(new Sehir(s[0]));
              while (cift < s.length) {
           String isim=s[i].trim();
              String [] a=isim.split(",");
                 String komsuisimleri = s[cift];
                 int uzaklik = Integer.parseInt(s[cift + 1];
   sehirler81.get(sayi).komsular81.add(new
Kenar(sehirler81.get(sayi), komsuisimleri, uzaklik));
                 cift += 2;
              }     sayi++;
            }
            dosya.close();  }} catch (IOException e) {  }
```

Random ile rotaları hesaplayan kod;

```
ArrayList<String> gdsRota = new ArrayList<>();
      ArrayList<Integer> farkliRandom = new
ArrayList<>();
 Random random = new Random();
      while (gdsRota.size() != sehirsayisiFaktoriyel) {

         while (farkliRandom.size() != sehirSayisi) {

            rnd = random.nextInt(sehirSayisi) + 1;
                  if (farkliRandom.contains(rnd) == false)
{
               farkliRandom.add(rnd);
            }
         }
         String tmpRota = "";
         tmpRota += 0;
         for (int i = 0; i < farkliRandom.size(); i++) {

            tmpRota += farkliRandom.get(i);
         }
         tmpRota += 0;

         if (gdsRota.contains(tmpRota) == false) {

            gdsRota.add(tmpRota);
         }

         farkliRandom.clear();
```

En kısa yolu bulan kod:

```
  for (int i = 0; i < sehirler81.size(); i++) {
        ugranildi_mi = false;
        uzaklik = Integer.MAX_VALUE;
        oncekisehir = null;
      }
uzaklik = 0;
      PriorityQueue<Sehir> priorityQueue = new
PriorityQueue<>();

      priorityQueue.add(sourceVertex);

      sourceVertex.ugranildi_mi = true;

      while (!priorityQueue.isEmpty()) {

         Sehir actualVertex = priorityQueue.poll();

         for (Kenar edge : actualVertex.komsular81) {

            Sehir v = edge.gidileceksehir;
            if (!v.ugranildi_mi) {

               int newDistance = actualVertex.uzaklik +
edge.uzaklik;

               if (newDistance < v.uzaklik) {

                  priorityQueue.remove(v);

                  v.uzaklik = newDistance;

                  v.oncekisehir = actualVertex;

                  priorityQueue.add(v);
               }
            }
         }
         actualVertex.ugranildi_mi = true;
      }

   }

   public List<Sehir> getShortestPathTo(Sehir
targetVertex) {
      List<Sehir> path = new ArrayList<>();

      for (Sehir vertex = targetVertex; vertex != null;

vertex = vertex.oncekisehir) {

         path.add(vertex);
      }

      Collections.reverse(path);

      return path;

   }
```

Bulunan uzaklıkları küçükten büyüğe sıralan kod:

```
  ArrayList<Integer> uzaklikdizisitmp =
(ArrayList<Integer>) uzaklikDizisi.clone();
      for (int i = 0; i < tmp.size() - 1; i++) {

         for (int j = i + 1; j < tmp.size(); j++) {

            if (uzaklikdizisitmp.get(i) >=
uzaklikdizisitmp.get(j)) {
               int tmpuzaklik = uzaklikdizisitmp.get(i);
               int tmpsayi = tmp.get(i);
               uzaklikdizisitmp.set(i,
uzaklikdizisitmp.get(j));
               uzaklikdizisitmp.set(j, tmpuzaklik);
               tmp.set(i, tmp.get(j));
               tmp.set(j, tmpsayi);
            }
         }
      }
```

Çizdirme işini yapan kod parçacığı:

```
public void paint(Graphics g) {
      super.paint(g);
      ImageIcon i = new ImageIcon("harita.png");
      g.drawImage(i.getImage(), 0, 0, 1015, 479, null);
      for (int ia = 0; ia < rotaa2.size()-1; ia++) {
         g.setColor(Color.red);
         for(int j=0; j<ksehirler.size();j++){

if(ksehirler.get(j).ksehirismi.equals(rotaa2.get(ia))){
               x1=ksehirler.get(j).x;
               y1=ksehirler.get(j).y;
               break;
            }
         }
         for(int j=0; j<ksehirler.size();j++){

if(ksehirler.get(j).ksehirismi.equals(rotaa2.get(ia+1))){
               x2=ksehirler.get(j).x;
               y2=ksehirler.get(j).y;
               break;
            }
         }
         g.drawLine(x1, y1, x2, y2);

      }
   }
```

Çizdirme için gerekli dosyayı okuyan kod parçacığı:

```
 public void dosyaoku() {

      File sehirkordinatlar = new File("sehirkordinatlari.txt
Scanner dosya3 = null;
      try {
         if (!sehirkordinatlar.exists())
{            System.out.println("dosya yok");
            sehirkordinatlar.createNewFile();          } else {
            dosya3 = new Scanner(sehirkordinatlar
int sayi = 0;
            while (dosya3.hasNextLine())
       String cumle = dosya3.nextLine();
               String[] a = cumle.split(",");// .
               ksehirler.add(new kordinatsehir(a[0]));
```

```
            while (cift < a.length) {

                int xler = Integer.parseInt(a[cift]);
                int yler = Integer.parseInt(a[cift + 1]);

                ksehirler.get(sayi).x=xler;
                ksehirler.get(sayi).y=yler;
        }   }
          dosya3.close();  }

    } catch (IOException e) {   }
```

V. SONUÇ

Bu projeyi geliştirirken dosya okumayı,Jfrrame yapısını,graph yapısını,en kısa algoritmalarını ,bu algoritmalarla ilgili bazı özel fonksiyonları ve veri yapıları ile nesneye yönelik programlamayı birleştimeyi öğrendik.. Projede en zorlandığım kısımlardan biri rota hesaplaması yaparken kodumuzdaki eksik algoritmadan dolayı 9 10 şehir için gerekli zamanda hesaplama yapamamak oldu.

VI. KAYNAKÇA

1.Dijkstra's shortest path algorithm in Java using PriorityQueue. (2019, April 25). GeeksforGeeks.

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-in-java-using-priorityqueue/

2.En Kısa Yol Problemleri ve Dijkstra Algoritmasının Uygulanması. (n.d.). Zafer Cömert Kişisel Web Sayfası.

https://www.zafercomert.com/IcerikDetay.aspx?zcms=70

3.Graphs in Java: Dijkstra's algorithm. (2020, May 1). Stack Abuse.

https://stackabuse.com/graphs-in-java-dijkstras-algorithm/

4.Graphs in Java: Dijkstra's algorithm. (2020, May 1). Stack Abuse.

https://stackabuse.com/graphs-in-java-dijkstras-algorithm/

5.Java Applet | Draw a line using drawLine() method. (2019, January 18). GeeksforGeeks.

https://www.geeksforgeeks.org/java-applet-draw-a-line-using-drawline-method/

6.Java ile Dökümandaki Cümle ve Kelimeleri Saydırma. (2015, December 3). Emre Bektaş | Kişisel Web Sayfası.

https://www.emrebektas.com/java/java-ile-dokumandaki-cumle-ve-kelimeleri-saydirma/

7. Real's how-to. (n.d.). Real's HowTo.

https://www.rgagnon.com/javadetails/java-0250.html

8.Topuz, M. (2017, 15). Java - Dosya İşlemleri - Mert Topuz. Mert Topuz.

https://merttopuz.com/kodlama/java/java-dosya-islemleri