Gezgin Kargo Problemi Proje Raporu

Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Sefa ÖZTÜRK – Faruk ARIĞ
180202036-180202009

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,LinkedList

I. GİRİŞ

Programımızda bir klasör içinde bulunan .txt uzantılı dosyadan 81 şehrin komşuluklarını ve bunlara olan uzaklıklarını alıp LinkedList'e ekliyoruz. Kgm dosyasından alınan şehirler arası uzaklık verilerini de uzaklıklar dizisinde tutuyoruz. Daha sonra kullanıcının seçtiği illeri LinkedList'ten çekip algoritma kullanıp en kısa yolu buluyoruz.

TABLO I
DOSYALARIN FORMATI

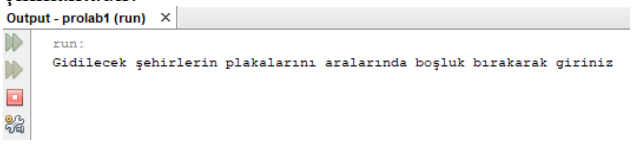
# Sehirler dosya formatı Plaka-Komşumu?-İl 1,0….(komşuluk)..,Adana 2,0….(komşuluk)..,Adıyaman 3,0….(komşuluk)..,Afyon . . . 81,0…(komşuluk)..,Düzce	#Noktalar dosya formatı Matris Yapısı 0,335,575,966,…,735 335,0,910,648,…,981 .,……., 0 , …… . .,……., 0 ,……. . .,……., 0 , …… . .,……., 0 , …… 735,981,375,1190,..,0
Dosya 1	Dosya 2

II. YÖNTEM

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

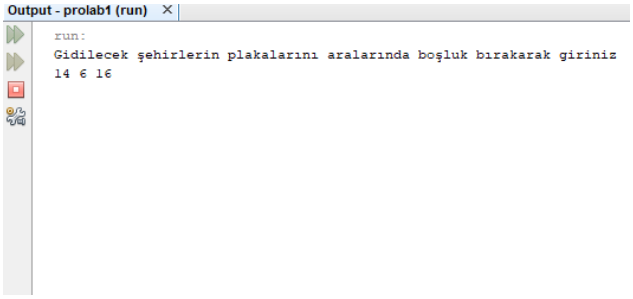
III. DENEYSEL SONUÇLAR

İlk başta kullanıcı gidilecek sehir sayısını kendi belirlemektedir. Ardından şekil.1 de ki gibi bir konsol çıkmaktadır.



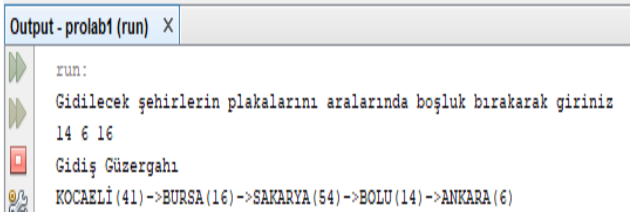
Şekil.1 Kullanıcının karşısına çıkan pencere.

Kullanıcı şehir sayısını kendi belirleyecek şekilde şehir plakalarının aralarında boşluk bırakarak gidilecek yerleri seçmesi için karşısına 81 şehir verili bir konsol çıkar.



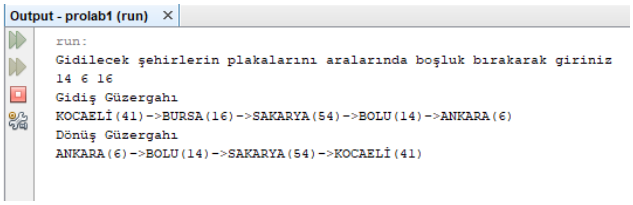
Şekil.3 Şehir sayısını ve plaka girilen ekran çıktısı.

Örneğin eğer kullanıcı 3 şehir girerse arkada program Sirala komutunu çalıştırır ve bir rota oluşturur.Başlangıç konumu olarak 0 her zaman Kocaeli'ni temsil eder çünkü Kocaeli her zaman başlangıç ve bitiş noktasıdır.Arkada çalışan kodun muhtemel çıktısı.



Şekil.3 şehir seçilince hesaplanan rota

Kullanıcı gidilecek sehirleri seçtikten sonra oluşturalan rotalar bu sefer şehir şeklinde en kısa rotayı bulur ve gidiş-geliş güzergahını konsola yazdırır.



Şekil.4 Kullanıcının girdiği sehirlerle oluşturulan rotaların ekran çıktısı.

Rotalar belirlendikten sonra kullanıcı tarafından seçilen şehirler arasındaki en kısa mesafeli rota pencere üzerinde görüntülenir.

Şekil.5 rotanın çizdirildiği pencere.

IV. YALANCI KOD

Sehirler ve komsulukların bulunduğu dosyaları okuma kodu:

```
File sehir = new File("sehir.txt");
File kgm = new File("kgm.txt");
File kordinat = new File("kordinat.txt");

int[][] uzakliklar = new int[81][81];
int[] kordinatY = new int[81];
int[] kordinatX = new int[81];
list sehirler = new list(81);

BufferedReader reader = null;
reader = new BufferedReader(new FileReader(sehir));
int i = 0;
String satir = reader.readLine();
while (satir!=null) {
    String[] line = satir.split(",");
    for (int j = 1; j < line.length-1; j++) {
        int k = Integer.parseInt(line[j].toString());
        sehirler.addKomsu(i, j, k,line[82]);
    }
    i++;
    satir = reader.readLine();
}

i = 0;
reader = new BufferedReader(new FileReader(kgm));
satir = reader.readLine();
while (satir!=null) {
    String[] line = satir.split(",");
    for (int j = 0; j < line.length; j++) {
        uzakliklar[i][j] = Integer.parseInt(line[j]);
    }
    i++;
    satir = reader.readLine();
}

i = 0;
reader = new BufferedReader(new FileReader(kordinat));
satir = reader.readLine();
while (satir!=null) {
    String[] line = satir.split(",");
    kordinatX[i] = Integer.parseInt(line[0]);
    kordinatY[i] = Integer.parseInt(line[1]);
    i++;
    satir = reader.readLine();
}
```

sirala ile komşuları hesaplayan kod:

```
static void sirala(int kms[], int ids[], int uzakliklar[][]) {
    int c = 0;
    for (int a = 0; a < ids.length; a++) {
        for (int b = ids.length - 1; b > a; b--) {
            if (kms[b - 1] > kms[b]) {
                int temp = kms[b-1];
                kms[b-1] = kms[b];
                kms[b] = temp;

                temp = ids[b-1];
```

```
                ids[b-1] = ids[b];
                ids[b] = temp;
            }
        }
        for (c = a + 1; c < kms.length; c++) {
            kms[c] = uzakliklar[ids[a]][ids[c]];
        }
    }
}
```

Gidişte En kısa yolu bulan kod:

```
for (int a = 0; a < ids.length; a++) {
        while (uzakliklar[i][ids[a]] != 0) {
            LinkedList<komsu> city = sehirler.getCity(i);

            uzaklik = uzakliklar[i][ids[a]];
            for (int j = 0; j < city.size(); j++) {
                if (city.get(j).maliyet != 0) {
                    if (uzaklik > uzakliklar[j][ids[a]]) {
                        uzaklik = uzakliklar[j][ids[a]];
                        plaka = j;
                        name = sehirler.getCity(j).get(j).name;
                        i = plaka;
                    }
                }
            }

            gidis[k] = plaka;
            k++;
            System.out.print("->"+name+"("+(plaka+1)+")");
        }
        i = ids[a];
    }
```

Dönüşte En kısa yolu bulan kod:

```
while (uzakliklar[i][40] != 0) {
        LinkedList<komsu> city = sehirler.getCity(i);
        uzaklik = uzakliklar[i][40];
        for (int j = 0; j < city.size(); j++) {
            if (city.get(j).maliyet != 0) {
                if (uzaklik > uzakliklar[j][40]) {
                    uzaklik = uzakliklar[j][40];
                    plaka = j;
                    i = plaka;
                }
            }
        }
            donus[k] = plaka;
        System.out.print("->"+sehirler.getCity(plaka).get(0).name+"("+(plaka+1)+")");
            k++;
        }
```

Çizdirme işini yapan kod parçacığı:

```
  public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        ImageIcon bg = new ImageIcon("harita3.png");
        image = bg.getImage();
        g.drawImage(image, 5, 35, null);



        int k = 0;
        int j = 0;
        while (gidis[k + 1] != 99) {
            g2.setColor(Color.BLUE);
            g2.setStroke(new BasicStroke(4f));
            g2.drawLine(x[gidis[k]], y[gidis[k]], x[gidis[k + 1]], y[gidis[k + 1]]);

            if (gidis[k + 1] == ids[j]) {
                ImageIcon pin = new ImageIcon("pin.png");
                pinImage = pin.getImage();
                g2.drawImage(pinImage, x[gidis[k + 1]]-8, y[gidis[k + 1]]-16, null);
                j++;
            }

            k++;
        }
        k = 0;
        float[] dashingPattern2 = {10f, 4f};
        Stroke stroke1 = new BasicStroke(2f, BasicStroke.CAP_BUTT,
                BasicStroke.JOIN_MITER, 1.0f, dashingPattern2, 0.0f);

        g2.setStroke(stroke1);
        g2.setColor(Color.BLACK);
        if (ids.length != 1) {
            while (donus[k + 1] != 99) {
                g2.drawLine(x[donus[k]], y[donus[k]], x[donus[k + 1]], y[donus[k + 1]]);
                k++;
            }

        }
    }
```

Graph yapısını oluşturan kod parçacığı:

```
class list {
    int cityCount;
    LinkedList<komsu> [] cities;

    list(int cityCount) {
```

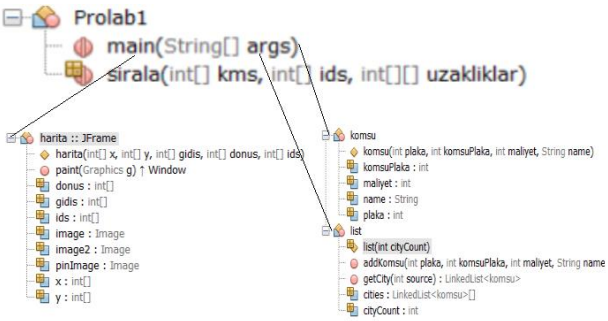
```
        this.cityCount = cityCount;
        cities = new LinkedList[cityCount];
        //initialize adjacency lists for all the vertices
        for (int i = 0; i <cityCount ; i++) {
            cities[i] = new LinkedList<>();
        }
    }

    public  void addKomsu(int plaka, int komsuPlaka, int
maliyet, String name) {
        komsu tmp_komsu = new komsu(plaka, komsuPlaka,
maliyet,name);
        cities[plaka].add(tmp_komsu);
    }

    public LinkedList<komsu> getCity(int source) {
        return this.cities[source];
    }
}
class komsu {
    int plaka;
    int komsuPlaka;
    int maliyet;
    String name;

    public komsu(int plaka, int komsuPlaka, int
maliyet,String name) {
        this.plaka = plaka;
        this.komsuPlaka = komsuPlaka;
        this.maliyet = maliyet;
        this.name = name;
    }
}
```

V. ALGORİTMA VE KULLANILAN METOTLAR

Öncelikle programı çalıştırdığımızda kullanıcının göreceği bir konsol açılır ve gidilecek şehir sayısını, şehir plakalarının aralarına boşluk bırakarak kendi belirlenmesi istenir.

Program çalışmaya başladıktan sonra; İlk olarak girilen şehirleri sirala fonksiyonunda kgm sitesinden alınan veriyle Kocaeli'ne uzaklıklarına göre sıralar. Sonra for döngüsü ile sıralanmış gidilecek şehirleri döner. "while (uzakliklar[i][ids[a]] != 0)" döngüsünde sıralanan şehirlerin arasının hangi şehirler üzerinden gidileceğini belirler. Burada a ile b şehirlerinin arasını bulmaya çalışıyor ise a şehrini city değişkenine atar ve maliyeti 0 olmayan kontrolü ile komşularını gezer, bu komşuluklardan b şehrine uzaklığı en küçük olanı gidiş güzergahına ekler ve yeni a şehrini b şehrine en yakın komşusu ile değiştirir. Dönüş algoritmasında da aynı mantık ile çalışır, son gidilen şehirden Kocaeli'ne ilk şehrin komşularından hedef şehre en yakın olanları seçerek ilerler. Sonuç olarak gidişte ve dönüşte ekrana en kısa yolu yazdırır.

Son olarak ise yapılan işlemlerden sonra harita çizimi için kullandığımız jframe kod parçası aktif hale gelir ve ekrana haritada rota çizilir.



Şekil.6 Kullanılan metotları gösteren diyagram.

VI. SONUÇ

Bu projeyi geliştirirken dosya okumayı,Jframe 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 gidiş ve dönüşteki en kısa yolu aynı anda yazdıramama sorunu oldu onun içinde ayrı ayrı işlem yaptık.

VI. KAYNAKÇA

[1] Bilgisayar Kavramları. (n.d.). Retrieved from http://bilgisayarkavramlari.sadievrenseker.com

[2] Theano, Flutter, KNime, Mean.js, Weka, Solidity, Org.Json, AWS QuickSight, JSON.Simple, Jackson Annotations, Passay, Boon, MuleSoft, Nagios, Matplotlib, Java NIO, PyTorch, SLF4J, Parallax Scrolling, Java Cryptography. (n.d.). Retrieved from https://www.tutorialspoint.com

[3] Where Developers Learn, Share, & Build Careers. (n.d.). Retrieved from https://stackoverflow.com

[4] 5. Java Applet | Draw a line using drawLine() method. (2019, January 18). GeeksforGeeks.
[5] https://www.geeksforgeeks.org/java-applet-draw-a-line-using-drawline-method/

[6] 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]