Gezgin Zeplin

Kocaeli Üniversitesi Bilgisayar Mühendisliği Programlama Laboratuvarı 2 - 1. Proje

Ata Gülalan
160202034
xavaneo@gmail.com

Oğuzhan Türker
160202015
oguzturker8@gmail.com

Gezgin Zeplin, görüntülenen Türkiye Haritası üzerinde kullanıcının seçmiş olduğu iki şehir arasındaki zeplinle gidilebilecek en kısa yolu yine kullanıcının belirlediği yolcu sayısıyla hesaplar.

I. GIRIŞ

Gezgin Zeplin, şehirler, zeplin ve arayüz olmak üzere üç sınıftan oluşur:

- Şehir sınıfı, harita üzerinde bulunan şehirlerin **isim, plaka, enlem, boylam, rakım** ve harita üzerindeki **x** ve **y** koordinatlarını tutar.
- Zeplin sınıfı, şehir bilgilerinin dosyalardan çekildiği, en kısa yol algoritmasının çalıştığı, genel tüm işlemlerin yapıldığı ana sınıftır.
- Ekran sınıfı, kullanıcının belirlediği girdileri alan, bu girdileri ana sınıfta işleme sokup sonuçları Türkiye Haritası ile birlikte kullanıcıya gösteren arayüz sınıfıdır.

Program sonucunda belirlenen iller arası zeplinle gidilebilen en kısa mesafe ölçülür. Rota çizilir ve haritada gösterilir. Belirlenen kriterlere göre **%50 kar** ve **maksimum kar** problemleri çözülür.

II. TEMEL BILGILER

Program **Java** programlama dilinde geliştirilmiş olup, geliştirilme ortamı olarak **Jetbrains IntelliJ IDEA** kullanılmıştır.

III. TASARIM

Gezgin Zeplin programının geliştirilme aşamaları altta belirtilen başlıklar altında açıklanmıştır.

A. Algoritma

Gezgin Zeplin'in Türkiye Haritası üzerinde iller arasında en kısa mesafeyle dolaşması için kullanılan algoritmanın adı "**Dijkstra Algoritması**" dır.
[**https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm**]

B. Yöntem

Gezgin Zeplin, yazılması kolaylaştırılmak adına üç sınıfta ayrılarak yazılmıştır. Şehirleri, dosyalardan okunan ve kendilerine ait olan bilgileri saklayabilecekleri şekilde birer nesneler olarak oluşturduk.

Her şehir, kendi komşularını "**ArrayList**" olacak şekilde saklayarak, oluşacak 81 şehirlik **graf** modelinin düğümlerini oluştururlar. Şehirlerin sahip oldukları bilgileri, proje içerisindeki metin belgelerinden okunarak düzgün ve sistematik düzenlenmesi zeplin sınıfında yapılmaktadır.

Programda bulunan %50 kar ve maksimum kar hesaplama problemleri ana sınıf olan zeplin sınıfının içinde hesaplanmaktadır. Her sınıf, diğer sınıf ve nesnelerle gerekli iletişimini sağlayabilmek adına, ilgili değerlerini diğer sınıf ve nesnelere döndürebilecek metotlara sahiptir.

C. Metodlar

- putGlobals(rC, mnY, …)
  Metoda giren değişkenleri Global değişkenlere atar.

- getSehir(plaka)
  Plakaya bağlı olan şehir objesini döndürür.

- plakadanIsim(plaka)
  Plaka koduna ait şehrin ismini döndürür.

- plakadanXY(plaka)
  Plaka koduna ait şehrin harita üzerindeki konumunu döndürür.

- dosyaOkuma(dosya)
  Verilen dosyayı okuyup bir kopyasını daha sonraki kullanımlar için belleğe alır.

- sehirlerArasiUzaklik(lat1, lon1, lat2, lon2)
  İki konum arasındaki yatay uzaklığı bulur

- deg2rad(deg)
  Girilen dereceyi radyana çevirip döndürür.

- init()
  Bellekteki graf'ı siler ve yenisini oluşturur.

- ucmaUzakligiBul(s1, s2)
  İki şehir arasındaki uçma uzaklığını bulur.

- dijkstra(baslangic)
  Şehirler arası en kısa mesafe algoritmasını çalıştırır.

- enKisaYol(baslangic)
  Bulunmuş en kısa yolu döndürür.

- gidilebilirMi(x, s1, s2, ekstra, yolcu)
  Zeplinin, verilen iki şehir arasında uçup uçamayacağını kontrol eder.

- flyWithCapasity(baslangic, bitis, yolcu)
  Verilen yolcu sayısına bağlı olarak başlangıç ve bitiş şehirleri arasında dijkstra algoritmasını çalıştırır.

- yuzdeElliKarHesapla(bs, bt, yl)
  %50 kar problemini çözer ve sonucunu döndürür.

- maksimumKarHesapla(bs, bt)
  Maksimum kar problemini çözer ve sonucunu döndürür.

D. Sözde kod

1. Dosyadan enlem-boylam-rakım verilerini oku.
2. Bu verilerden bir **Şehir** nesnesi oluştur.
3. Dosyadan **komşuluk** verilerini oku.
4. Bu verilerden şehirleri birbirine **bağla**.
5. Harita görüntüsünü dosyadan oku.
6. Harita üzerinde tüm şehirlerin butonlarını oluştur.
7. Kullanıcıdan bir **girdi** bekle.
8. **Birinci** şehri oku.
9. **İkinci** şehri oku.
10. Yolcu **sayısını** oku.
11. Geri kalan tüm yardımcı **parametreleri** oku.
12. Parametreleri **Zeplin'e** gönder.
13. İstenen özellik **maksimum kâr** ise; minimum yolcu sayısı ve maksimum yolcu sayısı arasındaki tüm yolcu sayılarını denemek suretiyle 15. maddeyi çalıştır.
14. İstenen özellik **%50 Kâr** ise; verilen yolcu sayısına göre 15. maddeyi çalıştır.
15. Verilen parametrelere göre, iki şehir arasındaki **en kısa** mesafeyi bul.
16. Bulunan yoldaki şehirleri eğime göre kontrol et.
17. İki şehir arasındaki eğim, verilen parametrelere uymuyor ise bu iki şehir arasındaki mesafeyi sonsuz yap ve 15'ye geri dön.
18. En kısa mesafeyi **haritada** göster.
19. Sonuçları kullanıcıya döndür.

IV. SONUÇLAR

Gezgin zeplin programı şehirler arası en kısa mesafeyi, şehirlerin enlemlerini, boylamlarını ve rakımlarını göz ardı etmeden, yolcu sayısının eğime etkisini de hesaba katarak, doğru bir şekilde ―Dijkstra Algoritması yardımı ile― hesaplar.

V. KAZANIMLAR

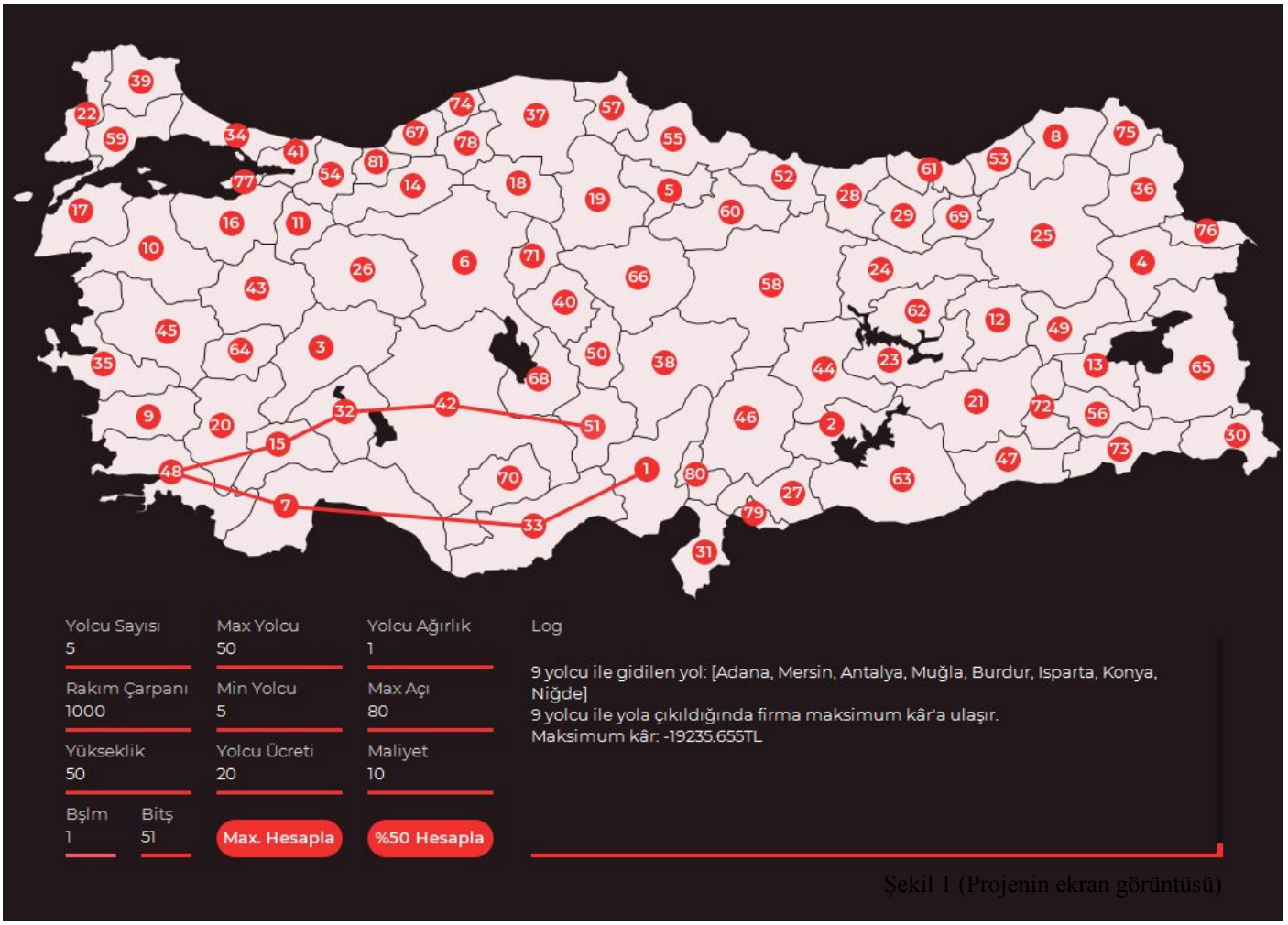
Gezgin Zeplin'in bize kattıkları;

- Çoklu ve formatlı dosya okuyabilmek
- Nesneler arası işlem yapabilmek
- Nesneler ile graf oluşturabilmek
- En kısa yol algoritmalarını koda implement edebilmek
- Arayüz oluşturabilmek
- Kullanıcıya göre sonuç üretebilen arayüz tasarlayabilmek
- Kullanıcı deneyimini geliştirebilmek

Gezgin Zeplin'in kullanıcılara katacakları;

- En kısa yol algoritmaların işleyişini anlayabilmek
- Yolcu sayısının, zeplinin hareketine ve elde edilen kara etkisini görebilmek
- Veriler, görsel ve dinamik bir şekilde kullanıcıya gösterilerek, kullanıcıdan kolay bir şekilde verilerin alınıp işlendikten sonra, kullanıcıya hitap edecek şekilde gösterilmesi.

VI. EKRAN GÖRÜNTÜSÜ



Şekil 1 (Projenin ekran görüntüsü)

VII. Karşılaşılan Sorunlar ve Çözüm Yöntemleri

1. Komşuluk dosyasındaki tutarsızlık

Program, X ve Y şehirleri arasındaki en kısa yolu bulmaya çalışırken farklı, Y ve X şehirleri arasında en kısa yolu bulmaya çalışırken farklı sonuçlar döndürdü. Bu olayın, komşuluk dosyasındaki tutarsızlıktan meydana geldiğini anladık ve platformda bu konuyla alakalı bir başlık açtık.

2. Şehirler arasındaki rakım farkının hesaplanması

Şehirler arasında, zeplinin yaptığı yükselme ve alçalma hareketi, ilk şehir ve son şehir için geçerli, ancak bu iki şehir komşu ise geçerli değil. Şehirler arası yolculukta birden fazla şehre gidip gidememe kontrol edileceğinden bu kontrolün graf dışına çıkarılması gerekiyordu. Bu yüzden buna ait bir metot yazıldı ve rakım farkı için genel bir formül bulundu.

3. En kısa yolun resimde gösterilmesi

İşlemler sonucu bulunan en kısa yolun görsel olarak kullanıcıya gösterilmesi, programın erişebilirliği açısından gerekliydi. Bir Türkiye haritasının üzerinden tüm şehirlere tıklayarak şehirlerin koordinatlarını aldık. Haritadaki 81 ilin hepsinin üzerine buton koyarak, programın gösterimini güçlendirdik.

4. Butonların ve Kaydırma çubuğunun stillendirilmesi

Varsayılan temanın yeteri kadar albenili ve kullanıcı dostu gözükmemesinden dolayı, kullanıcı dostu bir tasarım yapmak için tasarımı baştan oluşturduk.

Kaynakça

[1] **Variable is accessed within inner class. Needs to be declared final**
https://stackoverflow.com/questions/14425826/variable-is-accessed-within-inner-class-needs-to-be-declared-final

[2] **Is it possible to have a java swing border only on the top side?**
https://stackoverflow.com/questions/2174319/is-it-possible-to-have-a-java-swing-border-only-on-the-top-side

[3] **Colors for the project**
https://coolors.co/f45b69-f6e8ea-22181c-5a0001-f13030

[4] **How to set JFrame to appear centered, regardless of monitor resolution?**
https://stackoverflow.com/questions/2442599/how-to-set-jframe-to-appear-centered-regardless-of-monitor-resolution

[5] **Find shortest path between 2 points in a distance weighted map**
https://stackoverflow.com/questions/17480022/java-find-shortest-path-between-2-points-in-a-distance-weighted-map

[6] **Physical and Logical Fonts**
https://docs.oracle.com/javase/tutorial/2d/text/fonts.html

[7] **Find shortest path between 2 points in a distance weighted map**
https://stackoverflow.com/questions/423950/rounded-swing-jbutton-using-java