Gezgin Kargo Problemi

Furkan Aydoğan
Bilgisayar Mühendisliği, Kocaeli Üniversitesi
180202085
18020285@kocaeli.edu.tr

Sefa Berke Kara
Bilgisayar Mühendisliği, Kocaeli Üniversitesi
180202086
180202086@kocaeli.edu.tr

Giriş

Gezgin Kargo Problemi uygulaması "sehirler.txt","koordinatlar.txt","mesafeler.txt" dosyalarının her birini okur. Bu dosyalar içindeki satır bilgilerine program tarafından erişim sağlanır. Dosyalardaki her bir satırda bulunan bilgiler belleğe kayıt edilir.Kuulanıcı tarafından şehirlerin girilmesiyle birlikte program en kısa rotanın olduğu rotaları görsel ve yazılı olarak bastırır. Program sonlandırıldıktan sonra en kısa yollar "output.txt" dosyasına bastırılır.

Özet

Gezgin Kargo Problemi "sehirler.txt" dosyasından her şehrin bilgilerini(plaka kodu ,ismi ve komşularının ismi ve plaka kodlarını), "mesafeler.txt" dosyasından her şehrin kendi komşularına olan mesafelerini ve "koordinatlar.txt" dosyasından da "4lturkiye-siyasi-haritasi2.png" resim dosyasından alınmış her bir şehrin koordinatlarının x ve y değerlerinin bulunduğu değerler bulunmaktadır ve bu bütün değerler okunarak program tarafından belleğe kaydedilir.

Okuma tamamlandıktan sonra proje tarafından istenilen bölüme geçiş yapılır. Kullanıcı en kısa yolunu hesaplamak istediği şehirlerin plaka kodlarını girer Program başlangıç ve bitiş noktası Kocaeli olmak üzere algoritma ile oluşturulabilecek tüm rotaların şehirlerini ve mesafelerini hesaplar. Hesaplanan bu şehirler arasından en kısa mesafeye sahip olan şehir başta olmak üzere en kısa alternatif 5 yol sıralma algoritması ile sıralanıp çıktı olarak kullanıcıya Türkiye haritasında görsel olarak terminal ekranın ve "output.txt" dosyasında da yazılı olarak bastırır.

Temel Bilgiler

Windows 10 işletim sistemine sahip bilgisayar ile geliştirme yapılmıştır. Program Java programlama dilinde geliştirilmiş olup, tümleşik geliştirme ortamı olarak "Netbans" kullanılmıştır. Ek program olarak ise verilen verilerin daha net bir şekilde anlaşılabilmesi için "NotePad++" kullanılmıştır.

Tasarım

Gezgin Kargo Problemi projesinin programlanma aşamaları altta belirtilen başlıklar altında açıklanmıştır

Algoritma

Program biri şehirler için diğer biri ise komşular için olmak üzere 2 tane class ve bir tane de "Main" class'dan oluşmaktadır. Şehirler için oluşturulan class da şehirlere ait şehir adı, plaka kodu, komsu ve komşu maliyetleri tutulur. Haritalar için oluşturulan ve jframe tarafından "extend "edilen haritalar class'ında ise gösterilecek olan haritanın pencere boyutu, düzeni, Türkiye haritası ve şehirlerin harita üzerindeki x ve y koordinatları tutulur. Main class içinde program 3 farklı

“.txt” uzantılı dosyayı okur. İlk önce “koordinatlar.txt ”dosyasını okuyarak harita üzerindeki x ve y koordinatlarını belleğe kaydeder. İkinci olarak “sehirler.txt ” dosyasındaki şehirlere ait şehir ismi, şehir plaka kodu ve şehirlere ait komşular okunarak belleğe kayıt edilir. Üçüncü olarak ise “mesafeler.txt” dosyasındaki her bir şehrin kendi komşularına olan mesafeleri okunarak belleğe kayıt edilir. Program bu dosyalardan birisinin dahi okunamaması veya dosya yolunun hata ermesi durumunda “Dosya bulunamadı” uyarısını bastırır. Dosya okuma işlemi tamamlandıktan sonra “ plaka_bul “fonksiyon ile komşu şehirlerin plakaları bulunur alınan plakalar için “komsu_maliyet” matrisinde komşuluk matrisi oluşturulur.” enkisa_yol “fonksiyonuna şehirleri göndererek onların hangi şehre komşu olduklarını dizi değişkene atanır.

“” enkisa_yol” Fonksiyonunun Algoritması

” enkisa_yol” fonksiyonu hangi şehirden başlayacağını ve komşuluk maliyet matrisini parametre olarak alır. Başlangıçta her bir düğümün uzaklığı MAX değer olarak atama yapılır ve MAX değer atadıktan sonra başlangıç olarak referans alınan plakanın dizisine 0 değeri atanır. Ardından fonksiyona hangi şehir gönderiliyorsa dizinin o indexine -1 değeri atanır. Algoritma uğranmamış olan ve sonsuz değerden daha küçük bir değer arayana kadar kontrol eder ardından sonsuz değer değiştikten sonra en kısa gidebilecek mesafeyi bulur. Uğranılan şehirler “ugrandi_Mi” adlı dizi de “true” değer olarak işaretlenir. Uğranılan şehir için komşu maliyet matrisinde” komsu_mesafe”ye atama işlemi yapılır ve eğer “komşı_mesafe” matisinden 0 ‘dan farklı bir değer varsa (yani komşusu ise) kontrol edilir. Olması durumunda ilk gelen şehirden komşusu olan diğer şehirlerin dizisini ve maliyetini oluşturur. Sonuç olarak fonksiyon mesafe dizisini döndürmüş olur.

Bu fonksiyon tamamlandıktan sonra program tarafından döndürülen mesafe bir matriste tutulur ve sıradaki “maliyet” adlı fonksiyona geçiş yapılır .“maliyet” fonksiyonuna da “enkısa_yol” fonksiyonundaki gibi “komsu_maliyet” ve “sehirler” dizileri parametre olarak gönderilir ve bir şehrin diğer şehirlere göre olan en kısa maliyetini döndürür. ”maliyet” ve gönderilen fonksiyonunun algoritması “enkisa_yol” algoritması ile aynıdır sadece döndürülen değer farklıdır.

Girilen şehir sayısı kadar basamak sayısı oluşturulur ve 2 katına kadar döndürülür. Rakamları birbirinden farklı ise girilen şehirlere ait bütün rota olasılıkları ve bu rotaların maliyetleri hesaplanır. Hesaplamalar tamamlandıktan sonra algoritma maliyetlerine göre bütün hesaplanmış rotaları maliyetlerine göre sıralma yaparken şehirler de bu sıralamaya eklenir.

Sıralanmış yolların bastırılması için “output.txt” dosyası oluşturulur. Ve oluşturulan rotalar Türkiye haritasında en kısa rotanın olduğu bir pencere ve ekstra 5 farklı rotanın bulunduğu bir pencere bastırılır.

Sözde Kod

1.Şehirler için sehirler adında bir class oluştur.
2.Class içinde şehir biligleri için String tipinde “sehir_adi” ,“komsu “ ve int tipinde “plaka_kodu” ,” komsu_maliyet” adlı değişkenler oluştur
3.Haritalar adlı JFrame tarafından extend edilmiş Class oluştur
4.Haritanın üzerinde çizim yapılması için pencere oluştur ve Jframe ayarlamaları yap
5.Trükiye haritasını arka plan olarak ayarla
6. “koordinatlar.txt ”dosyasını okuyarak harita üzerindeki x ve y koordinatlarını belleğe kaydet.
7. “sehirler.txt ” dosyasındaki şehirlere ait şehir ismi, şehir plaka kodu ve şehirlere ait komşular oku ve belleğe kayıt et
8. “mesafeler.txt” dosyasındaki her bir şehrin kendi komşularına olan mesafeleri oku ve belleğe kayıt et.
9. Bu dosyalardan birisinin dahi okunamaması veya dosya yolunun hata ermesi durumunda “Dosya bulunamadı” uyarısını bastır
10. sonra “ plaka_bul “fonksiyon ile komşu şehirlerin plakaları bul.
11. Alınan plakalar için “komsu_maliyet” matrisinde komşuluk matrisi oluştur.

12. .” enkisa_yol “fonksiyonuna şehirleri göndererek onların hangi şehre komşu olduklarını dizi değişkene ata.
13. ” enkisa_yol” fonksiyonu hangi şehirden başlayacağını ve komşuluk maliyet matrisini parametre olarak alır.
14. Başlangıçta her bir düğümün uzaklığı MAX değer olarak atama yap
15. Atanan MAX değeri başlangıç olarak referans alınan plakanın dizisine 0 değeri ata.
16. Fonksiyona hangi şehir gönderiliyorsa dizinin o indexine -1 değeri ata
17. Uğranmamış olan ve sonsuz değerden daha küçük bir değer arayana kadar kontrol et.
18.Sonsuz değer değiştikten sonra en kısa gidebilecek mesafeyi bul
19. Uğranılan şehirler “ugrandi_Mi” adlı dizi de “true” değer olarak işaretle
20. Uğranılan şehir için komşu maliyet matrisinde” komsu_mesafe”ye atama işlemi yap
21. Eğer “komşı_mesafe” matrisinden 0 ‘dan farklı bir değer varsa (yani komşusu ise) kontrol et.
22. Olması durumunda ilk gelen şehirden komşusu olan diğer şehirlerin dizisini ve maliyetini oluştur
23.Mesafe dizisini return et.
24. Döndürülen mesafe dizisini bir matriste tut.
25. Sıradaki “maliyet” adlı fonksiyona geçiş yap.
26. “komsu_maliyet” ve “sehirler” dizileri parametre olarak gönder.
27. ” enkisa_yol” fonksiyonunda yapılan işlemlerin aynısını yap.
28. Bir şehrin diğer şehirlere göre olan en kısa maliyetini döndür
29. Program tarafından döndürülen mesafeyi bir matriste tut.
30. Girilen şehir sayısı kadar basamak sayısı oluştur ve 2 katına kadar döndür.
31. Döngü de artan i değerini her seferinde string değişkenine ekle.
32. Diziye çevrilmiş olan String değişkenini sırala.
33. Kullanıcı kaç şehir girdi ise ona göre rakamları al karşılaştırama yap.
34. Bütün sayılar birbirinden farklı oluyorsa o sayıların modunu alarak gezilecek şehirlere aktar.
35. Kullanıcıdan aldığım şehirlerin plakalarını “sehir_plaka” değişkenine ata
36. Eğer plaka -1 değil ise arraylist’e ilk plakayı ata.
37. Yukarı da oluşturulan komşu dizisinde gezilecek şehir “ i+1” diyerek olası kombinasyonlar da Kocaeli’nden sonra hangisi geliyorsa onu şehir plakasına ata.
38. Bu sırada da maliyetleri toplayıp bütün olası maliyetleri değişken dizisine ata.
40. Gidilen en son yerdeki plakayı al ve “sehir_plaka” da tut.
41. Alınmış olan şehirlerin maliyetlerini küçükten büyüğe doğru sıralama işlemini yap
42. Maliyetleri sıralma yaparken yollarda da aynı şekilde sırlama yap.
43. Harita da çizeceğim noktalar için yeni bir point nesnesi oluştur.
44.” Output.txt” adlı bir dosya oluştur.
45.Olutşurulan dosyaya sıralanan rotaları ve maliyetleri yazdır.
46. Yazdırma işleminin yapılaması durumunda "Dosya yazma işlemi başarısız " uyarısı verir.
47.Harştaya en kısa yolun rotasını bastır
48. Ekstra 5 tane en kısa yol için de bir şehre gelip o şehirden çıkarken plakasını 2 kere tekrar edene bakıyorum (yani o şehre uğranmış mı diye referans alıyorum).
49. Eğer o şehre uğrandıysa “koordinatlar.txt”den okunan koordinatları noktalar değişkenine ata.
50.Atanan bu değişkenleri harita da çizdir.
51.Son

Karşılaştığımız sorunlar:

En kısa yol algoritmasında bir şehrin diğer şehirlere en kısa mesafelerini matriste ile mi dize ile mi tutacağımızı karar veremedik araştırmalar sonucunda bunun çözümüne vardık.

Şehirlerin komşularını(C dilindeki struct yapısı gibi) nasıl tutacağımız konusunda zorlandık ve bu sorunun çözümünü ayrı bir şehirler class’ı oluşturarak her şehir için nesne üreterek komşularını ve plakasını tutmasını sağladık.

Gidilecek olası yollar için bütün varyasyonlar bulmak ve hem de kısa sürede algoritmanın sonuçlanmasını amaçladık. Bunun için araştırmalar yaptık pek bir fikir sahibi olamadık fikir sahibi olduğumuz kadar belli bir algoritma geliştirerek Kocaeli dışında en fazla 8 şehir uğrayacak bir algoritma geliştirdik.

İstenilen şehirleri harita üzerinde nasıl çizeceğimiz konusunda bir miktar zorlandık. En sonunda şehirlerin koordinatlarını alarak hangi şehre uğradıysa o şehrin koordinatını çizdirerek bu isteri de tamamladık.

Jar oluşturma kısmında okuduğumuz txt uzantılı dosyaları tam yolunu verdiğimiz için başka bir platform da çalışmadığından bunu her platform da nasıl çalıştıracağımızı bulmakta bir miktar sorunlar yaşadık

Projenin Bize Kattığı Yararlar:

Teorik olarak bildiğimiz en kısa yol algoritmasını koda dökerek bunu uygulama şansımız oldu ve sadece teorik olarak değil de pratik olarak da fikir sahibi olduk.

*Gidilebilecek bütün olası yollar normalde uzun sürerken algoritmada belli değişiklikler yaparak hem bütün olası yolları bulup hem de kodun daha kısa sürede sonuç vermesini sağladık.

*Harita üzerinde noktalara göre çizim yaparak hem çizim fonksiyonunu hem de Java GUI ortamını daha iyi kavradık

Kullanılan Methodlar

public static int[] enkisa_yol() :

Her gönderilen şehir için en kısa mesafeyi “dijikstra algoritması” ile bulur.

public static int[] maliyet() :

Her gönderilen şehir için en kısa maliyeti “dijikstra algoritması” ile bulur

static int plaka_bul() :

String gönderilen şehrin plaka kodunu döndürür.

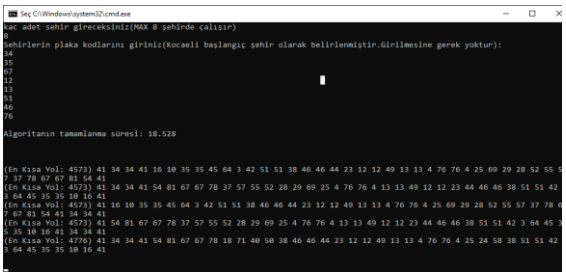
static int faktöriyel() :

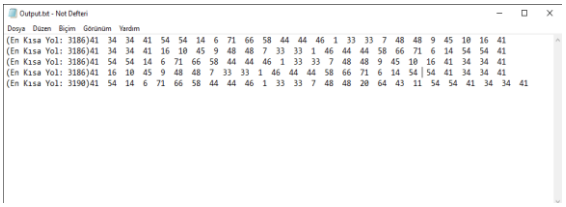
Girilen şehir sayısına göre kaç farklı sıralama yapılacağının sayısının bulur.

Sonuçlar

Gezgin Kargo Problemi uygulaması kendisine verilmiş olan “koordinatlar.txt”,”sehirler.txt” ve ”mesafeler.txt” dosyalarını okur. . Bu okuma gerçekleştikten sonra isterler bölümündeki işlemler yapılarak sonuçlar ekrana ve Türkiye haritası üzerine bastırılır. Ek olarak bütün rotalar ve kısa yollar çıktı olarak “output.txt” dosyasına yazdırılır.

Ekran Görüntüleri





www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/?ref=rp

Kaynakça

Java Applet | Draw a line using drawLine() method

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

Class java.awt.Point

https://courses.cs.washington.edu/courses/cse341/98au/java/jdk1.2beta4/docs/api/java/awt/Point.html

getproperty() and getproperties() methods of System Class in Java

https://www.geeksforgeeks.org/getproperty-and-getproperties-methods-of-system-class-in-java/

Printing Paths in Dijkstra's Shortest Path Algorithm