Kocaeli Üniversitesi
Bilgisayar Mühendisliği Bölümü
Programlama Laboratuvar II Proje I

Gezgin Kargo Problemi

Ramazan Kaan YARAYAN 190201138
Kocaeli, Türkiye
rknyryn@gmail.com

***Özet*—Merkezi Kocaeli/Türkiye olan ve Türkiye'nin şehirlerine kargo dağıtımı yapılmasını amaçlayan. Dağıtım yapılırken;veri yapıları,veri modelleri, graf yapısı ve çeşitli algoritmalar kullanılarak gidilmesi gereken şehirleri en az yol maliyetli olacak şekilde bir rota oluşturarak ve buna ek alternatif rotalarında oluşturulmasını gerçekleştiren bir yol bulma uygulamasının geliştirilmesi.**

I. GIRIŞ

Kargo dağıtım probleminde; başlangıç noktası belirlenmiş olup, başlangıç noktasından itibaren gidilecek (maksimin 10 şehir) olunan şehirlerin hepsine komşuluk ilişkilerinden hareket ederek uğraması ve tüm hedef şehirler ziyaret edildikten sonra başlangıç noktasına geri dönülmesini amaçlayan ve bunu gerçekleştirmek için sezgisel arama algoritmalarını kullanan, bulunan rotayı son kullanıcının kolaylıkla anlayabileceği şekilde gösteren bir rota oluşturma uygulamasıdır.

II. YÖNTEM

A. Veri Toplama

Uygulamanın geliştirilmesinde hangi algoritmaların kullanılabilinir olduğu hakkında internet üzerinden araştırma yapıldı.

Bir şehirden başka bir şehire giden en kısa yolu bulmak amaçlandığı için; yapılan araştırma sonucunda bulunan algoritmalardan sezgisel olarak çalışanlara öncelik verilip, bu sezgisel algoritmalar (örneğin: minimax ağaçları, arı sürüsü arama algoriması, a* algoritması, tepe tırmanma algoritması…) arasından işlev bakımından en uygun olduğu düşünülen "a* algoritması" tercih edildi.

Uygulama C# ile geliştirildiğinden; C# ile dosya işlemleri hakkında da araştırma yapıldı. Dosya okuma-yazma işlemleri için araştırma sonucunda elde edilen FileStream, StreamReader ve StreamWriter sınıfları kullanıldı.

B. A Algoritması*

A* algoritması; başlangıç noktasından hedefe ulaşılmasını sağlayan en kısa yolun bulunmasını amaçlayan sezgisel ve gerçek maliyet ile hesap yapan bir algoritmadır.

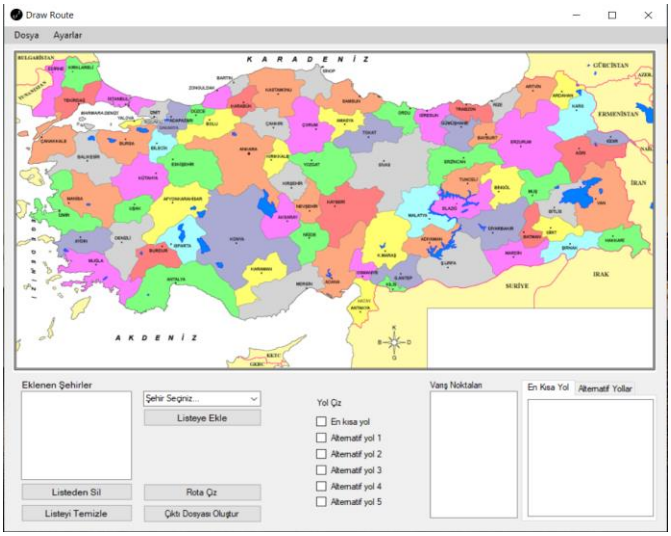
Temel olarak, başlangıç noktasından gidilebilinecek alternatif yolların gerçek maliyetleri ile gidilen alternatiften hedefe olan sezgisel(örnek: bir şehirden başka bir şehire için; kuş uçuşu mesafe) maliyetin toplanması sonucunda elde edilen hesaplanmış maliyeti, diğer alternatiflerin de hesaplanmış maliyeti ile kıyaslayıp, en küçük maliyete sahip olana öncelik verilip, en küçük maliyetli olana hareket edilmesi ve devamında da hareket edilen noktadan alternatiflere bakılıp, maliyetlerinin hesaplanması, tekrardan en küçük maliyetli olana hareket edilmesi ve hedefe ulaşana kadar bu şekilde çalışarak ilerleyen bir algoritmadır.

Hesaplamalar yapılırken oluşabilecek bazı durumlar için; a* algoritmasını açık list eve kapalı liste olarak iki liste kullanılarak yapılmaktadır. Açık liste; hareket edilecek noktadan gidilebilinecek alternatifleri ve onların maliyetlerini tutar. Açık listeye eklenmiş olan en küçük maliyetli noktaya hareket edilmesi halinde gidilen nokta kapalı listeye eklenir, hareket edilen nokta kapalı listeye eklenir. Yeni noktadan gidilebilinecek alternatifler hesaplanır ve açık listeye eklenir, ancak listeye eklenirken, eklenen alternatif noktanın kapalı listede olmaması kontrolü yapılır, bu şekilde bir döngü oluşması durumu engellenmiş olunur. Eğer yeni eklenecek nokta zaten açık listede var ise de hesaplanan maliyetler karşılaştırılır ve eğer yeni hesaplanan maliyet öncekinden daha az ise açık listede olan nokta güncellenir. Hedef noktası kapalı listeye eklendiğinde arama işlemi son bulur.

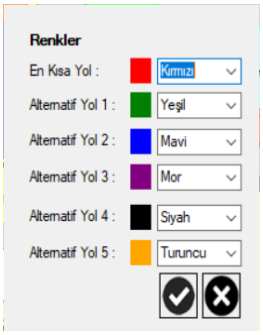
C. İşlev

Uygulamanın çalışırken kullanması gereken veriler için "sehirler.txt" ve "mesafeler.txt" şeklinde iki dosya oluşturuldu. Şehirler dosyasında; tüm şehirler ve komşuları tutulurken, mesafeler dosyasında; sezgisel olarak hesaplama yapılırken kullanıması amaçlanan her bir şehrin diğer tüm şehirlere olan uzaklığı tutuldu.

Kaynak dosyaları da hazır hale getirildikten sonra, formun tasarımı*(örnek: Şekil 1.0 , Şekil 1.1)* gerçekleştirildi.



(Şekil 1.0)



(Şekil 1.1)

1) FilePath Sınıfı

Hazırlanan kaynak dosyalarınaki verilere erişim yollarını tutmayı sağlamaktadır.

OpenFileDialog nesnesini kullanarak kullanıcıdan belirtilen formatta olan ".txt" uzantılı dosyaları seçmesi istenmektedir. Ardından seçilen bu dosyların, dosya yolları uygulama klasörünün içerisinde ".txt" uzantılı dosyalarda tutulmaktadır. Bu şekilde kaynak dosyalar farklı yerlede olsa bile dosyaların yollarını uygulamaya kayıt ettikten sonra; eğer kaynak dosyası silinmez veya değiştirilmez ise dosyalara erişim sağlanmakta ve tekrar bu işlemin yapılmasına gerek duyulmamaktadır.

2) Colors Sınıfı

Ekrana çizilecek yolların renklerinin değiştirilebilir olması için; kullancının kolaylıkla değiştirebileceği bir ayar ekranı ile kullanıcıdan seçtiği renklerin kayıt edilmesini sağlar. Böylelikle son kullanıcı belirlenen seçenekler arasında, rotayı ekrana çizerken kullanılacak rengi kendi belirlemiş olur.

Renkler bit ".txt" uzantılı dosyada uygulamanın keni klasörü içerisinde tutulmaktadır. Uygulama başlatıldığında renkler bir ArrayList'e alınarak, çizim işlemi yapılırken bu Arraylist kullanılmaktadır.

3) FileData Sınıfı

Belirli bir formatta hazırlanmış olunan bu kaynak dosyalarının içerisindeki verilerin okunması ve uygulama içerisinde tutulmasını sağlamaktadır.

FileData sınıfı içerisinde FileStream, StreamReader sınıflarından türetilmiş nesneler ile verilerin okunması ve uygulama içerisine aktarılması *(örnek: Şekil 1.2)* işlemi gerçekleştirilmektedir.

```
public static void getData()
{
    try
    {
        FileStream file = new FileStream(FilePath.citiesFilePath, FileMode.Open, FileAccess.Read);
        StreamReader reader = new StreamReader(file);

        string line = reader.ReadLine();
        while (line != null)
        {
            addCity(line);
            line = reader.ReadLine();
        }

        reader.Close();
        file.Close();
    }
    catch (Exception)
    {
        System.Windows.Forms.MessageBox.Show("Veriler alınırken bir hata oluştu!\nLütfen dosyaları tekrar içeri aktarınız ve tekrar işleyiniz.", "Draw Route");
    }
}
```

(örnek: Şekil 1.2)

4) Location Sınıfı

A* algoritması kullanılarak rota araması yapılırken, şehrin plaka kodunu, gerçek mesafesini, sezgisel mesafesini, hesaplanmış mesafesini ve hangi şehirden geldiğini tutmak için oluşturulmuştur.

```
class Location
{
    public int plateCode;
    public int realDistance;
    public int heuristicDistance;
    public int calculatedDistance;
    public Location previousCity;
```

(Şekil 1.3)

Şekil 1.3'te belirtilen değişkenlere sahip olup, tüm değişkenlerine değer ataması yapan bir kurucu metoda da sahiptir.

5) FindRoute Sınıfı

Rota bulma işleminin yapılmasını gerçekleştiren sınıftır.

FileData sınıfı kullanılarak alınan veriler işlenmek için burada kullanılır. Gidilecek rotanın hangi şehirleri kapsadığını tutmak için bir dizi kullanılır. Bu dizi kullanıcının ListBox' a eklediği şehirler ile doldurulur. Gidilecek şehirler belirlendikten sonra a* algoritması çalışmaya başlar. drawRoute fonksiyonu çağrılarak arama işlemi başlatılır. drawRoute fonksiyonu öncelikle kullanıcını eklediği sırada şehirlerin hepsinin ziyaret edilip başlangıç noktasına geri dönebileceği bir yol bulur. Ardından bulduğu yoldaki şehirlerin sırayla o şehire gitmeden gidilebilinecek alternatif yolları bulur. Ardından gidilecek şehirleri başlangıç noktasına olan sezgisel uzaklıklarına göre küçükten büyüğe sıralayarak, sırasıyla gidilebilinecek diğer yollar bulunur. Tüm bu bulunan yolları "log.txt" adında bir dosyaya kayıt eder. Kayıt ettiği yollar içerisinden en küçük maliyetli olandan başlıyacak şekilde bir sıralama yapar. Oluşturulan bu sıralamayı

foundedRoute adında tanımlanmış bir Arraylist'e ekler. Böylelikle gidilecek en kısa yol ve alternative yollar bulunup, kullanılmak üzere bir ArrayList'e atanmış olur.

6) PaintRoute Sınıfı

```
private int a, b;
private Graphics graphic;
private Pen pen;
private System.Windows.Forms.PictureBox picBox;
private bool[] option = new bool[6];

private System.Collections.ArrayList shortestPath = new System.Collections.ArrayList();
private System.Collections.ArrayList alternativePath1 = new System.Collections.ArrayList();
private System.Collections.ArrayList alternativePath2 = new System.Collections.ArrayList();
private System.Collections.ArrayList alternativePath3 = new System.Collections.ArrayList();
private System.Collections.ArrayList alternativePath4 = new System.Collections.ArrayList();
private System.Collections.ArrayList alternativePath5 = new System.Collections.ArrayList();
```

(Şekil 1.4)

Şekil 1.4 ile gösterilmiş olunan değişkenlere sahip olan ve bu değişkenler ile kullanıcının bulunan rotayı görsel olarak görmesini sağlayan çizimi gerçekleştiren sınıftır.

Sınıf içerisinde tüm şehirler koordinat olarak PointF türünde bir dizi içerisinde plaka kodu ile erişilmesi mümkün olacak şekilde tutulmaktadır. Bulunan rotadaki şehirlere maksimum 5 alternatif ve en kısa yol olarak tutulan listelerin içerisindeki şehirlere Graphic sınıfında türetilmiş nesne kullanılarak, kayıtlı olan renkler ile rotanın çizimi gerçekleştirilmektedir.

D. Kullanım

Eğer dosya yolları ile ilgili bir uyarı aldıysanız; sol üstte bulunan dosya sekmesi altındaki "Şehirleri içeri aktar" ve "Mesafeleri içeri aktar" seçenekleri kullanılarak belirlenen formatta hazırlanmış kaynak dosyalarını seçniz. Ardından yine Dosya sekmesi altında bulunan "Dosyaları işle" seçeneğine tıklayarak içeri aktardığınız kaynak dosyaları içerisindeki verileri uygulama için kullanılır hale getirmiş olursunuz.

Bir rota çizilmesi için; öncelikle gidilmesini istediğiniz şehri rotaya eklemek için bir şehir seçiniz ve listeye ekle seçeneğine tıklayınız. Eğer bir şehri silmek isterseniz; silmek istediğiniz şehri listede seçiniz ve "Listeden Sil" seçeneğine tıklayınız veya seçmiş olduğunuz şehre sağ tıklayarak listeden sil diyerek te silebilirsiniz. Listeyi temizlemek isterseniz "Listeyi Temizle" seçeneğine tıklayınız.

Oluşturulacak rota için şehirleri girdikten sonra "Rota Çiz" seçeneğine tıklayınız.

Bulunan en kısa yol ve alternative yollar listelerde gösterilecektir. Listeler arasında başlıklarına tıklayarak geçiş yapabilirsiniz.

Harita üzerinde çizdirmek isterseniz; "Yol Çiz" seçenekleri arasından hangisini çizdirmek isterseniz o seçeneğe tıklayarak rotayı çizdirebilirsiniz. Eğer rotayı silmek isterseniz seçeneği kaldırınız.

Ayarlar sekmesi altından "Renk Ayarları" seçeneğine tıklayarak açılan pencerede, yolların hangi renkle gösterilmesini istiyorsanız onu seçip, seçenekleri kayıt ederek gösterilecek renkleri değiştirebilirsiniz.

Bulunan rotayı çıktı dosyası olarak basmak isterseniz; "Çıktı Dosyası Oluştur" seçeneğine tıklayarak veya Dosya sekmesi altından "Çıktı Dosyası Oluştur" seçeneğine tıklayınız, açılan pencerede dosyasyı nereye kayıt etmek istediğinizi seçip, dosyanın adını veriniz ve kaydet diyerek çıktı dosyası oluşturabilirsiniz.

III. SONUÇLAR

- Sezgisel arama algoritmalarının neden kullanıldığı öğrenildi.
- Sezgisel arama algoritmalarının neler olduğu öğrenildi.
- Sezgisel arama algoritmalarının farkları öğrenildi.
- Sezgisel bir algoritma olan A* algroitmasının çalışması ve kodlanması hakkında fikir sahibi olundu.
- Liste elemanı olarak bir sınıf nesnesi kullanılması öğrenildi.
- Graphic sınıfının kullanımı hakkında bilgi sahibi olundu.
- Dosya oluşturma,slime,okuma ve yazma işlemleri hakkında bilgi edinildi.

KAYNAKLAR

Sezgisel Algoritmalar

http://bilgisayarkavramlari.sadievrenseker.com/2008/12/22/sezgisel-algoritmalar-bulussal-algoritmalar-heuristic-algorithms/

A* algoritması kodlanması

https://gigi.nullneuron.net/gigilabs/a-pathfinding-example-in-c/

https://www.raywenderlich.com/3016-introduction-to-a-pathfinding#toc-anchor-007

Dosya İşlemleri

https://www.kodlamamerkezi.com/c-net/c-ile-dosya-ve-klasor-islemleri/

C# Özellikler ve Metotlar

https://docs.microsoft.com/tr-tr/dotnet/csharp/programming-guide/

List<> Özellikler ve Metotlar

https://docs.microsoft.com/tr-tr/dotnet/api/system.collections.generic.list-1?view=netcore-3.1

C# Colors

https://docs.microsoft.com/tr-tr/dotnet/api/system.drawing.color?view=netcore-3.1

Save File Dialogs Kullanımı

https://www.hikmetokumus.com/makale/31/csharp-ile-savefiledialog-kullanimi