Algoritma Analizi ve Tasarımı Dersi

2022-2023 Bahar Dönemi

**ÖDEV 1 RAPORU**

Teslim Tarihi: 08.04.2023

Öğrenci Adı: Mustafa Alp Yanıkoğlu

Öğrenci Numarası: 222802005

1. Giriş

Ödevin amacı; (x,y) koordinatları belli olan şehirlerin her birini gezmek koşulu ile en kısa yolu bulmak.

Program.cs’de kullanılan veriler;

void LoginScreen(): Giriş menüsünü içerir, kullanıcıya dosya seçtirir.

void ReadFileAndFindShortesPath(string ): Verilen dosya uzantısı ile dosyayı okur daha sonra en yakın komşu algoritması ile temel bir kısa yol hesaplar. Son olarak Simulated Annealing algoritması ile yolun daha kısasını bulmaya çalışır.

void PrintList(List<Point> ): Listeyi yazdırır.

FilePath.cs’de kullanılan veriler;

\_filePathWithoutFileName :Dosya uzantısı olan string bir property.

String GetFilePath(TspFilePaths ): Dosya yoluna seçilen enum değerini ekler.

Enum TspFilePaths(): Bu bir enum’dır ve uygulama dosyalarındaki dosyaları bulundurur.

Point.cs’de kullanılan veriler;

class Point(double x, double y): Noktaların X ve Y koordinatlarının bilgisini tutan sınıf.

ReadFile.cs’de kullanılan veriler;

List<Point> ReadXandYCordinates(string filePath): Gönderilen dosya yolunu satırlarını okuyarak bir listeye ekler ve geriye liste döner

SimulatedAnnealing.cs’de kullanılan veriler;

List<Point> Adjacent(List<Point> route, Random random): Gönderilen listeden 2 adet eleman seçerek yerlerini değiştirir ve geriye güncellenmiş listeyi döner.

Double Error(List<Point> points): Hata değeri hesaplamak için kullanılıyor. Daha küçük bir hata değeri tespit edilirse daha iyi yol olduğu çıkarımı elde edilir.

List<double> Path(List<Point> points): Verilen listedeki elemanlar o an ki en kısa yolun sırasını belirtir ve şehirler arasındaki uzaklıkları hesaplar. Geriye şehirler arasındaki mesafeleri döner.

double Dist(Point point1, Point point2): Verilen iki nokta arasındaki uzaklığı hesaplar. Geriye uzaklığı döner.

List<Point> FindShortesPath(List<Point> points): En yakın komşu algoritmasını kullanarak en kısa yol rotasını oluşturur.

double AcceptanceProbability(double error, double adjError, double currTemperature): Kabul olasılığını hesaplamak için kullanılır. Parametre olarak sırasıyla; hata değeri, önceki adımda hesaplanmış hata değeri ve mevcut sıcaklık değerlerini alır. Geriye double bir değer döner. Bu hesaplama, daha kötü bir çözümün bile bazen kabul edilebileceği şekilde tasarlanmıştır. Ancak, sıcaklık düştükçe, kabul olasılığı da düşer ve algoritma daha iyi bir çözüm için aramaya devam eder.

List<Point> SimulatedAnnealingStart(List<Point> points): Bir nokta listesi alır ve en iyi rotayı hesaplamak için ‘Simulated Annealing’ algoritmasını uygular. İlk olarak, \_randomNumber değişkeni, noktaların sayısına bağlı olarak bir rastgele sayı üretmek için kullanılır. \_maxIter değişkeni, algoritmanın en fazla kaç adım atabileceğini belirtir. \_minCurrTemperature, algoritmanın duracağı en düşük sıcaklık değerini belirler. \_startTemperature, algoritmanın başlangıç sıcaklığıdır. \_alpha, sıcaklığın ne kadar hızla düşeceğini belirler. \_iteration değişkeni, algoritmanın şu ana kadar kaç adım attığını izler. \_currTemperature, mevcut sıcaklık değerini tutar. Daha sonra, \_error değişkeni, Error yöntemi kullanılarak verilen nokta listesi için bir hata değeri hesaplanır. Sonra, algoritma, belirtilen maksimum adım sayısına ulaşana veya bir çözüm bulana kadar döngü içinde çalışır. Her adımda, Adjacent adlı başka bir yöntem kullanılarak mevcut nokta listesiyle benzer bir rotaya sahip bir \_adjRoute listesi oluşturulur ve bu rotanın hatası \_adjErr hesaplanır. Daha sonra, \_adjErr değeri, \_error değerinden daha iyi ise, \_adjRoute rotası kabul edilir ve bu rotanın hata değeri \_error olarak güncellenir. Ancak, \_adjErr değeri \_error değerinden daha kötü ise, kabul olasılığına bağlı olarak \_adjRoute rotası kabul edilir veya reddedilir. \_currTemperature değeri, her adımda bir faktörle azaltılır ve \_iteration değeri arttırılır. Son olarak, algoritma, en iyi rotayı içeren nokta listesini döndürür.

\_maxIter -> Algoritmanın kaç adımda sonlanacağını belirtir.

\_\_minCurrTemperature -> Algoritmanın çalışmasının durdurulacağı en düşük sıcaklığı belirler.

\_ startCurrTemperature -> Algoritmanın başlangıç sıcaklığıdır.

\_iteration -> Algoritmanın kaç adım attığını izler.

\_ currTemperature -> Mevcut sıcaklığın değerini tutar.

\_error -> Hata değerini tutar.

1. Geliştirme ve Çalıştırma Ortamları

Visual Studio, Visual Studio Code

1. Algoritma Akışı

Geliştirdiğim algoritmanın sözde kodu (Pseudo Code).

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Sonuç
   * + - TCP (Transmission Control Protocol) problemi, bir ağ bağlantısı üzerindeki iki nokta arasındaki veri transferindeki gecikme süresini ölçmek için kullanılan bir yöntemdir. Bu yöntemde, bir noktadan diğerine bir paket gönderilir ve alıcı nokta bu paketi aldığını teyit eder. Teyit alma işlemi gerçekleşinceye kadar geçen süre ölçülür ve buna göre ağdaki gecikme süresi tahmin edilir.
       - İnternet hizmet sağlayıcıları, müşterilerine sunacakları internet hizmetinin kalitesini ölçmek ve iyileştirmek için TCP problemini kullanabilirler.
       - Ağ yöneticileri, ağdaki gecikme süresi ve veri transfer hızını ölçmek için TCP problemini kullanabilirler. Bu sayede ağda meydana gelen problemleri tespit edip çözebilirler.
       - Bulut bilişim sağlayıcıları, müşterilerine sunacakları bulut hizmetlerinin kalitesini ölçmek ve iyileştirmek için TCP problemini kullanabilirler. Video konferans uygulamaları gibi gerçek zamanlı iletişim uygulamaları, kullanıcılar arasındaki bağlantının kalitesini ölçmek için TCP problemini kullanabilirler.
       - Web siteleri, ziyaretçilerin deneyimini iyileştirmek için sayfa yükleme sürelerini ölçmek için TCP problemini kullanabilirler.