

İSTANBUL OKAN ÜNİVERSİTESİ

MESLEK YÜKSEKOKULU

BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

Yapay Zeka

Minimum Yolun Bulunması Problemi

Hazırlayan

Barış Aydoğdu

18MY03023

DANIŞMAN

ÖĞR.GÖR. NİLGÜN İNCEREİS

İçindekiler

Giriş

Genel Bilgiler…..…..

Yapay Zeka Nedir?3

Minimumu Yolun Bulunması Problemi Nedir?4

Graph Theory Nedir ?4

Nerelerde Kullanılır?5

Kaba Kodu6

Proje Bilgileri..….….

Proje Konusu 6

Minimum Yolun Bulunması Uygulama Tasarımı6

Proje Yapım Aşamaları7

**Özgeçmiş9**

Kaynakça10

Ek

Github Link11

Kaynak Kodları12

Yapay Zeka Nedir ?

Yapay zeka; insan gibi davranışlar sergileme, sayısal mantık yürütme, hareket, konuşma ve ses algılama gibi birçok yeteneğe sahip yazılımsal ve donanımsal sistemler bütünüdür. Başka bir deyişle yapay zeka; bilgisayarların insanlar gibi düşünmesini sağlar.

Yapay zekaya X sorusu yönetildiğinde daha önce verilen veya tanımlanan X sorusu cevapları arasından en rasyonel olanını seçip sunar. Bu nedenden dolayı X sorusu gelince her defasında yapay zeka X sorusunun cevaplarını süzer ve en rasyonel olanını sunar. Yapay zeka günümüzde X sorusu örneğinden daha fazlasını yapmaktadır.

Yapay Zekanın Uygulama Alanları:

1-)Ses tanıma

2-)Görüntü işleme

3-)Doğal dil işleme

4-)Muhakeme

Ses tanıma; Ses tanıma teknolojisi günümüzde level atlamış; Yapay zekalar sesi tanımakla kalmayıp sesi anlayıp cevap verme seviyesine gelmiştir.Bilgisayarın sesi tanıması, sesleri harflere dönüştürmesi, harfleri anlayıp cevap vermesi hiç de kolay değil sadece buradan baktığımızda bile bu teknolojinin ne kadar geliştiğini görmekteyiz. Ses tanıma teknolojisi sadece akıllı telefonlarımız da bulunan siri,cortana gibi ya da diğer sanal asistanlarda kullanılmamaktadır. Hayatımızın her noktasında ses tanıma teknolojisine dair ürünler görmek mümkün.

Görüntü işleme; Görüntü işleme hayatımızın birçok noktasında kullanılmaktadır. Sürücüsüz araçların daha güvenli hareket etmeleri için etrafını iyi anlamaları gerekmektedir.Görüntü işleme sayesinde pikseller koda dönüşüyor. İnsanları,hayvanları, trafik ışıklarını, uyarı levhalarını anlayan ve buna göre hareket eden sürücüsüz araç görüntü işlemeye verilecek en iyi örneklerden biri.

Doğal dil işleme; Diğer yapay zeka uygulama alanlarında olduğu gibi doğal dil işleme alanında birçok alt başlık ve kullanım alanı mevcut. En basit örneğiyle Google gibi arama motorlarında “Bunu mu demek istediniz” gibi cümlelerle arama algoritmaları geliştiriliyor. Arama motoru örneğinden daha fazlasının yapıldığını hatırlatmak da fayda var.

Muhakeme; Öncelikle muhakemenin tanımını yapmakta fayda var. Muhakeme;”Birbirine karşı savları olan iki yanı dinleyerek bir yargıya varma, yargılama.” Yapay zeka da muhakeme en zor alanlardan biridir. Oluşturulan yapay zekanın birkaç olayı süzüp en mantıklı en rasyonel kararı vermesi çok önemlidir. Örneğin; Bir dava da hakimin suçluyu suçsuzu ayırt etmesi muhakeme yeteneğine bağlıdır. Burada yapay zeka delilleri ve olayı süzerek muhakeme yeteneği ile suçluyu suçsuzu ayırt edilebilmektedir.

**Graph Theory Nedir ?**

Graph Theory yani Çizge Kuramı olarak adlandırılır. Çizgeleri, yani grafları inceleyen matematik dalıdır. Graflar ,node (yada vertice) yani düğümler ve edge yani bağlantılardan oluşan bir çeşit ağ yapısıdır.

Bağlantıların her birinin değeri vardır ve tek yada çift yönlü olabilirler. Bir G grafı, G = (V,E) şeklinde ifade edilir. Burada V düğümler kümesi , E ise bağlantılar kümesini ifade eder.

Minimum Yolun Bulunması Problemi Nedir?

Minimum yolun bulunması problemi kısaca , iki düğüm arasında en maliyetle gidilebilen bir yolun varlığını belirleme yöntemidir. Graph Theory içerisinde önemli bir yere sahiptir. Graf içerisinde belirtilen node’lar arasındaki en kısa yolu bulmayı amaçlar. Bu özelliğiyle Gezgin Satıcı Problemi’ne de (Travelling Salesman) benzer.

Ancak Gezgin Satıcı’dan iki noktada ayrılır. Birincisi, hedef düğüme giderken tüm düğümlere uğramak zorunda değil ve ikincisi de hedefe vardıktan sonra başlangıca dönmez.

En kısa yollar döngüler içeremez, ancak negatif bağlantı değerleri bulundurabilirler. Bununla birlikte n tane düğümün olduğu bir grafta en kısa yol n-1 kenardan daha fazla kenara sahip olamaz.

Kaynak nodun diğer tüm node’lara olan en kısa yollarını bularak Graph Theory içerisinde yer alan En Kısa Yol Ağacı’nı da (Shortest Path Tree) oluşturulabilir.

Kısa Yol Problemleri Nerelerde Kullanılır ?

* Navigasyon cihazlarında güzergah ayarlaması sırasında
* Oyun Programlamada
* Dynamic Routing kullanan Network’lerde En ucuz yada hızlı hat ayarlaması yaparken
* Limancılık, havacılık, postacılık güzergah ayarlaması esnasında
* Rubik Cube’ü en az hamlede çözmek için.
* Robot hareket planlamasında kullanılmaktadır.

**Navigasyonda Kısa Yolun Bulunması:**

Navigasyon cihazları iki nokta arasındaki kısa yolu hesaplayabilmek için Hollanda’lı Matematikçi Dijkstra tarafından bulunan Dijkstra Algoritması kullanılır.

Bu algoritmanın temeli çizge teoremine ([Graf Teorisi](https://www.muhendisbeyinler.net/graf-teorisi-nedir/" \t "_blank)) dayanır. Kısa yol probleminde harita üzerindeki yerleşim yerleri (iller/ilçeler/köyler vb.) nokta olarak ve bu noktalara giden yollar çizgiler halinde ifade edilmektedir.

**En Kısa Yol Problemleri**

**En Kısa Yol Problemleri 3 Algoritmadan oluşmaktadır.**

**Bunlar:**

1-)Dijkstra Algoritması

2-)Bellman ve Ford

3-)Floyd Algoritmasıdır.

**Dijkstra Algoritması:**

Dijkstra algoritması bir düğümden diğer tüm düğümlere en kısa yolları hesaplar. Yani bir başlangıç noktasına göre en kısa yolu belirleyen algoritmadır. Ağırlıklı ve yönlü graflar için geliştirilmiş olup kenarların ağırlık değeri sıfır ya da sıfırdan büyük bir değer olmalıdır. Eğer kenarların değerleri sıfırdan küçük oluyorsa daha genel algoritmalardan Bellman-Ford kullanılabilir. Dijkstra algoritmasının zaman karmaşıklığı genel olarak O(MlogN) şeklinde hesaplanmıştır. Dijkstra algoritması en kısa yolu belirlerken Greedy yaklaşımını kullanır.Yani bir düğümden diğer düğüme geçerken olası en iyi yerek çözümü göz önüne alır.

Her seferinde bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılır.

**Dijkstra algoritmasının zayıf yönü**

Algoritma ne yazık ki eksi (-) değer taşıyan bir kenar bulunması halinde başarılı çalışmaz. Bunun sebebi eksi (-) değerdeki kenarın sürekli olarak mevcut durumdan daha iyi bir sonuç üretmesi ve algoritmanın hiçbir zaman için kararlı hale gelememesidir.

**Bellman ve Ford**

Dijkstra gibi bir düğümden diğer tüm düğümlere olan en kısa yolu belirler.

Bellman Ford algoritması, Dijkstra farklı olarak negatif değere sahip grafların için de

doğru olarak çalışır. Ancak burada da eksi maliyetli çevrimler olmamalıdır.

**Floyd Algoritması**

Graf üzerindeki her bir düğüm için diğer tüm düğümlere olan en kısa yolları belirleyen algoritmadır. En kısa yolu bulmak için kullanılan en genel algoritmadır.Genellikle yoğun graflarda kullanılır.Grafın komşuluk matrisi şeklinde tutulması durumunda Floyd Algoritması O(N3) karmaşıklığı olmaktadır.

**Dijkstra Algoritmasının sözde kodu** (pseudocode):

1. Başlangıç noktasını belirle.
2. Başlangıç noktasından diğer noktalara olan maliyeti belirle ve düşük maliyetli  
   noktayı işaretle.
3. İkinci adımda işaretlenen noktadan gidilebilen diğer noktalar arasında da  
   aynı işlemi tekrarla.

**Projem Hakkında**

Projem bir Minimum Yolun Bulunması ProblemininDijkstra Algoritması örneğidir.

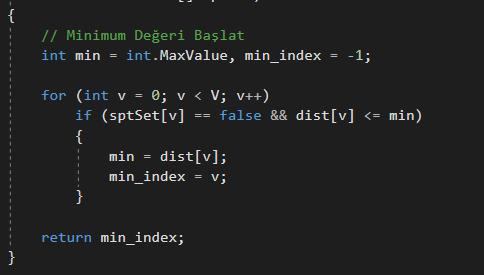
Projemi VisualStudio 2017 kod derleyicisini kullanarak, C# dilinde yazdım.

**Uygulama Tasarımı:**

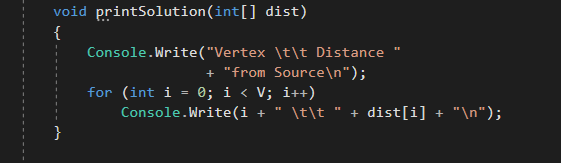
Uygulamam bir konsol projeyi olup sadece oluşturduğum bir graph üzerinde ki en kısa yolu bulmaktadır. Konsol projesi içerisindeki çıktım bir Tepe(Vertex) değeri ve Kaynaktan Uzaklık (Distance From Source) değerlerin içermektedir.

**Proje Yapım Aşamaları:**

Projeyi kodlamaya başlarken öncelikli olarak minimum değeri başlatan bir kod yazdım. Bunu for döngüsü kullanarak gerçekleştirdim. “Başta min\_index”’i, -1 olarak belirledim. Fakat for döngüsü içerisine girdikten sonra belirlenen değeri int türünde atadığım V değişkenini almış bulundu.

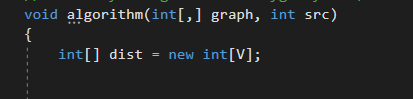


Oluşturulan mesafe dizisini yazdırmak için printSolution adında bir fonksiyon yazdım. Uygulama Tasarımında bahsetmiş olduğum Tepe ve Kaynaktan Uzaklık değerlerini yazdıran fonksiyon aşağıda belirttiğim fonksiyondur.

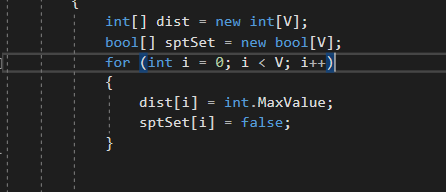


Uygulamam için temelleri oluşturduktan sonra asıl kısa yolu bulacak olan fonksiyonu yazmaya başladım. Bu fonksiyon içerisinde int tipinde iki tane parametre olarak değişken tanımladım.

Ardından dizi tanımladım.”src” ile başlangıç değeri olan 1 arasında ki mesafeyi kısa tutacak bir çıktı dizisi.

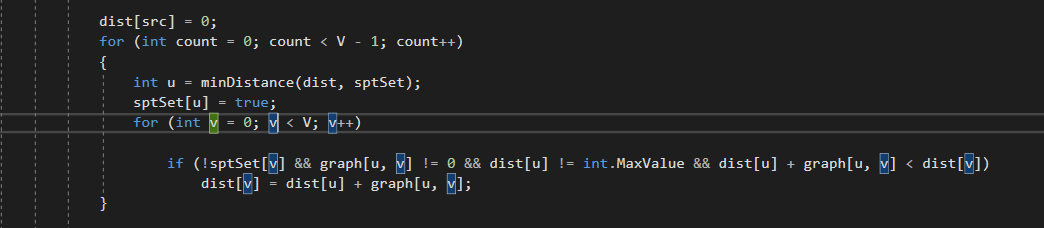


Hemen ardından bool tipinde sptSet isminde bir değişken tanımladım. Bu değişken eğer bir tepe noktası varsa geçerli oluyor. Sonrasında bir for döngüsü kullanarak bool tipinde tanımladığım değişkeni “false” olarak seçip bir “”dist adında ki dizimi MaxValue olarak başlattım.



Kaynak tepe noktasının mesafesini kendisinden itibaren her zaman 0 olarak belirledim. Daha sonra bir for döngüsü kullanarak, tüm köşeler için en yakın bulmayı hedefledim.

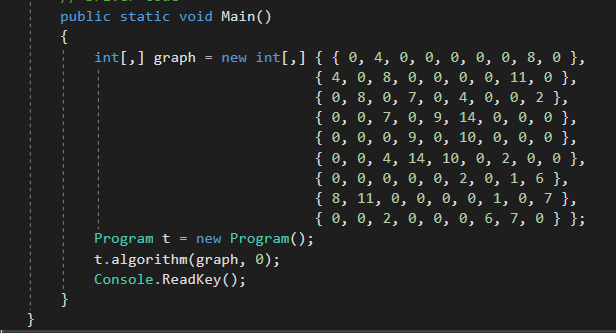
Döngü içerisinde seçilmiş tepe noktasını daha önce bool değişkenimi false olarak tanımlamıştım. Bu kod bloğunda döngü içerisinde “true” olarak gelmekte.



“printSolution” adında yazdığım fonksiyonu içerisine “dist” değişkenini alarak inşa edilen mesafe dizisini yazdırdı.

C:\Users\baris\OneDrive\Masaüstü\Ekran Alıntısı.PNG

Son olarak bu işlemleri gerçekleştirdiğim Graph’ı tanımladıktan sonra programım çalışır duruma geldi.



**Özgeçmiş**

22 Mart 1999 tarihinde İstanbulda doğdum. İlk öğrenimimi Eğe Sanayi İlköğretim okulunda tamamladım. Daha sonrasında lise hayatıma Süleyman Demirel Lisesinde devam ettim. Lisenin son yılında Maltepe Uğur Temel Lisesinde lise hayatımı tamamladım. Okan Üniversitesi Bilgisayar Programcılığı bölümünde öğrenim görmekteyim. Üniversite sürecimde kendimi geliştirmek için staj yaptığım kurumda halen proje geliştirmeye devam etmeyim. Projelerimi Xamarin ile Native mobil uygulamalar yazarak devam ettiriyorum. Okul dışında hobilerimden biri yüzmek. Lisansli uzun mesafe yüzücüsüyüm. Ayrıca yeni diller öğrenmek ve resim çizmeyi çok severim.

**İngilizce Seviyesi:**

Okuma: İyi,

Yazma: İyi,

Dinleme:İyi

Konuşma:İyi

**Bilgisayar Yetenekleri:**

Word, Powerpoint, Outlook: Orta  
C# Nesne Tabanlı Programlama:İyi

JavaScript Programlama:Orta

ASP.NET:Başlangıç Seviyesi

MSSQL Veritabanı:İyi

HTML5:Çok iyi

Xamarin: Çok İyi

**Tamamlanan Projeler:**

Öğrenciler için Ders Takip Program (C# Programlama ile)

Yük Gemisi Yönetim Sistemi (C# Programlama ile)

Yemek Tarifleri Web Sitesi (HTML5,CSS)

Seyahat Gemisi Veritabanı (MSSQL)

JavaScript Örnekleri Web Sitesi (HTML,CSS,JavaScript)

Bekçi Tur(Xamarin)

BioNet(Xamarin)

BusinessTalk(Xamarin)

**Sertifikalar:**

**09/2016 İngilizce Kursu / Maltepe Amerikan Kültür**

İngilizce okuma,yazma,dinleme ve konuşma üstüne 12 aylık bir kursun ardından sertifika

**01/2019 Udemy**

Veri Bilimine Giriş 3 saatlik bir eğitim üstüne sertifika

**Udemy**

**04/2019** Linux: Herkes için Özgür İşletim Sistemi 1.5 saatlik bir eğitim üstüne sertifika

,  **Udemy**

**04/2019** Algoritma ve Programlama Mantığı Eğitimi 6.5 saatlik bir eğitim üstüne sertifika

**01/2020** **Bahçeşehir Üniversitesi**

Türkiye Siber Kümelenmesi ve Siber Klüpler Birliği Siber Güvenliği Kış Kampı Web Uygulama Güvenliği katılım sertifikası

**Kaynakça**

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

<https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/>

<http://www.zafercomert.com/IcerikDetay.aspx?zcms=70>

<https://slideplayer.biz.tr/slide/2742392/>

<http://endustri.eskisehir.edu.tr/nila/IST328/duyuru/EN%20KISA%20YOL.pdf>

<https://www.muhendisbeyinler.net/dijkstra-algoritmasi/>

**EK:**

**GithubLink:**

**Program Kodları:**

class Program

{

static int V = 9;

int minDistance(int[] dist,

bool[] sptSet)

{

int min = int.MaxValue, min\_index = -1;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

{

min = dist[v];

min\_index = v;

}

return min\_index;

}

void printSolution(int[] dist)

{

Console.Write("Vertex \t\t Distance "

+ "from Source\n");

for (int i = 0; i < V; i++)

Console.Write(i + " \t\t " + dist[i] + "\n");

}

private void algorithm(int[,] graph, int src)

{

int[] dist = new int[V];

bool[] sptSet = new bool[V];

for (int i = 0; i < V; i++)

{

dist[i] = int.MaxValue;

sptSet[i] = false;

}

dist[src] = 0;

for (int count = 0; count < V - 1; count++)

{

int u = minDistance(dist, sptSet);

sptSet[u] = true;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u, v] != 0 && dist[u] != int.MaxValue && dist[u] + graph[u, v] < dist[v])

dist[v] = dist[u] + graph[u, v];

}

printSolution(dist);

}

public static void Main()

{

int[,] graph = new int[,] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

Program t = new Program();

t.algorithm(graph, 0);

Console.ReadKey();

}

}

}