****

**MESLEK YÜKSEKOKULU**

**Bölüm Adı:** Bilgisayar Teknolojileri

**Program Adı:** Bilgisayar Programcılığı

**Ders Adı:** Yapay Zeka

**Proje Adı:** Kısa Yol

**Hazırlayan**

**Adı - Soyadı:** Altar Buğra BOĞAZ

**Okul Numarası:** 18MY03016

**Öğretim Görevlisi:** Nilgün İNCEREİS

İçindekiler

[Yapay Zeka Nedir? 3](#_Toc38655740)

[Yapay Zekanın Sınıflandırılması 3](#_Toc38655741)

[Dijkstra Algoritması 3](#_Toc38655742)

[Proje Konusu 4](#_Toc38655743)

[Projem 5](#_Toc38655744)

[Proje Kodlarım 8](#_Toc38655745)

[Kaynakça 10](#_Toc38655746)

# 

# Yapay Zeka Nedir?

Yapay zeka, insanlar gibi düşünmek ve eylemlerini taklit etmek için programlanmış makinelerde insan zekasının simülasyonunu ifade eder. Terim, öğrenme ve problem çözme gibi bir insan zihniyle ilişkili özellikler sergileyen herhangi bir makineye de uygulanabilir.Yapay zekanın ideal özelliği, belirli bir hedefe ulaşmak için en iyi şansa sahip eylemleri rasyonalize etme ve alma yeteneğidir.

# Yapay Zekanın Sınıflandırılması

Yapay zeka iki farklı kategoriye ayrılabilir: zayıf ve güçlü. Zayıf yapay zeka, belirli bir işi gerçekleştirmek için tasarlanmış bir sistemi içerir. Zayıf AI sistemleri, yukarıdaki satranç örneği gibi video oyunlarını ve Amazon'un Alexa ve Apple'ın Siri'si gibi kişisel asistanları içerir. Asistana bir soru soruyorsun, sana cevap veriyor.

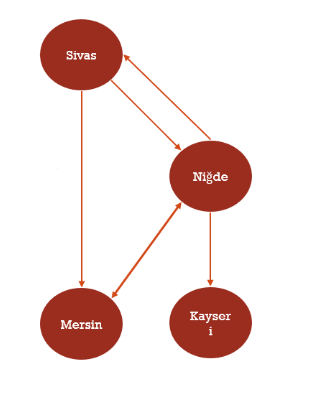
Güçlü yapay zeka sistemleri, insan benzeri olduğu düşünülen görevleri yerine getiren sistemlerdir. Bunlar daha karmaşık sistemler olma eğilimindedir. Bir kişinin müdahale etmesine gerek kalmadan problem çözmeleri gerekebilecek durumları ele almak üzere programlanırlar.

# Dijkstra Algoritması

Dijkstra algoritması , bir grafikteki düğümler arasındaki, örneğin yol ağlarını temsil edebilecek en kısa yolları bulmak için bir algoritmadır. 1956 yılında bilgisayar bilimcisi Edsger W. Dijkstra tarafından tasarlandı ve üç yıl sonra yayınlandı.

Algoritma birçok varyantta bulunur. Dijkstra'nın orijinal algoritması verilen iki düğüm arasındaki en kısa yolu buldu , ancak daha yaygın bir varyant tek bir düğümü "kaynak" düğümü olarak düzeltir ve kaynaktan grafikteki diğer tüm düğümlere en kısa yolları bulur ve en kısa yolu üretir ağacı.

# Proje Konusu



Oluşturduğum şemada şehirler arasında ki yollar belirtilmiştir. Bir başlangıç şehri oluşturup yola çıkılacaktır. Şema yönlü graf örneğidir. Yönlü graflarda yalnızca okun yönüne hareket edebiliriz. Geriye dönüş olmaz, çift yönlü gidiş geliş için iki yönlü ok kullanılır. (Mersin-Sivas arası gibi mesela)

# Projem

Dijkstra algoritmasını uygularken bir tane başlangıç noktası belirleriz. Belirlediğimiz ilk noktanın maliyeti sıfır olur. Diğer belirlediğimiz tüm noktaların maliyeti sınırsızdır.

Başlangıç olarak Konya’yı seçiyorum ve komşularına maliyetlerini belirliyorum.



Konya düğümü başlangıç olduğu için hep sıfır kalacaktır. Konya’ya komşu olan Eskişehir, Mersin, Antalya’nın fiyatlarını belirleyelim. Diğer tüm düğümlerin maliyeti hala sınırsızdır.

Devamında 1. Satırdaki en az maliyetli düğümü seçeriz. Ancak dikkat etmemiz gereken konu seçtiğimiz düğümün daha önce ziyaret edilmemiş olmasıdır. En az maliyetli Eskişehir’i seçelim.

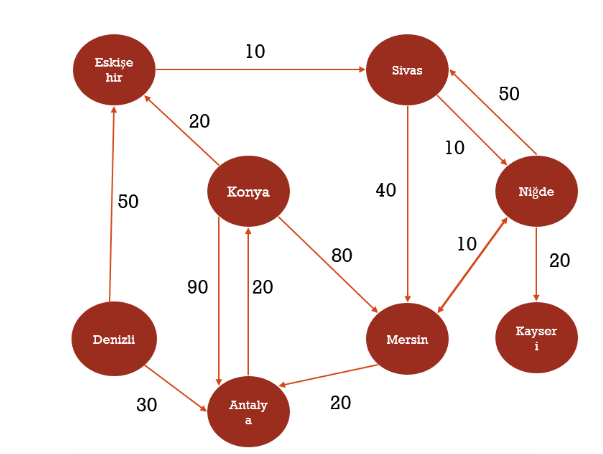


Eskişehir’den gidilecek tek yer Sivas. Sivas’a gidiş maliyeti 10 iken 30 yazdık. Çünkü Eskişehir’e gidişin maliyeti 20, Sivas’a gidişin maliyeti 10. İkisini topladığımız zaman 30 oluyor

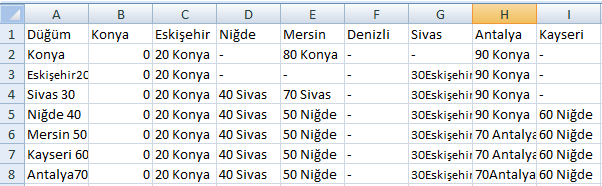
Yola en düşük maliyetli düğümle devam edeceğiz. En düşük maliyetli düğümler Sivas ve Eskişehir. Ancak bu iki düğümü de ziyaret ettiğimiz için haricindeki en düşük maliyetli düğüme yol alacağız. Bütün işlemleri bu şekilde tablomuza göre devam ettireceğiz. Şuana kadarki ilerlediğimiz yolu tabloya aktarıyorum.



Sivas’tan çıkan iki ok var. Niğde ve Mersin. Niğde’ye Sivas’tan maliyet 10, Sivas’ın da 30 maliyeti var. Bu yüzden maliyeti 40 olarak artırırız. Mersin’e olan maliyet ise 40, Sivas’ın maliyetini de ekleyince 70 olur. Önceki maliyet ise 80’di, 70 sayısı 80’den küçük olduğu için Mersin’in maliyetini değiştiririz. Dijkstra algoritmasının olayını burada net olarak görüyoruz. Daha kısa bir yol bulduk.



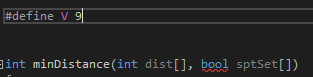
Tabloya göre ilerlediğimizde geriye kalan en düşük maliyetli düğüm Antalya oldu. Diğer tüm düğümleri ziyaret ettik. Denizli’nin ise maliyeti sonsuz kaldı.



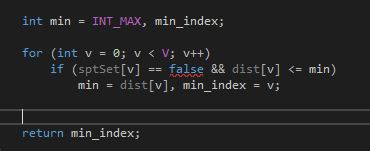
Antalya’dan tek yol Konya’ya var. Fakat Konya’nın maliyeti zaten sıfır. Başladığımız düğüm olduğu için. İlerlediğimiz yolda en küçük maliyetli düğüm Denizli’dir. Ziyaret edilmiş düğümleri saymadığımız için Denizli’nin maliyeti sınırsız olduğu için maliyetini değiştiremeyiz. Neden bu şekilde olduğunu sorarsanız Denizli’ye yol olmadığı için. Yol olmayan yere gidemeyiz. Bu sebepten Denizli sınırsız maliyet olarak tabloda yer alır.

# Proje Kodlarım

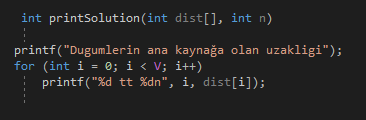
\*Başlangıç olarak graftaki düğüm sayısını V değişkeni ile ifade ediyoruz. En kısa yolu bulan yardımcı fonksiyon.



\*İlk başta sonsuz bir değer atıyoruz. Aşağıda bulunan döngü ziyaret edilmeyen düğümler içerisinde minimum yolu tespit ediyor. En az olanı buluyor.

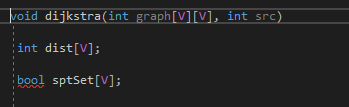


\*Başlangıç noktasından diğer düğümlere olan maliyeti yazdıran fonksiyonumuz.

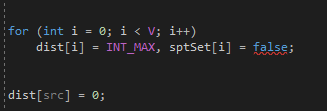


\*Başlangıç düğümünden bir başka düğüme olan uzaklığı burada tutuyoruz.

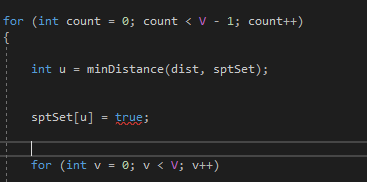
  Düğüm ziyaret edildiğinde bu diziye aktarıyoruz. Bu sayede ziyaret edilmiş düğümleri kontrol edebiliriz.



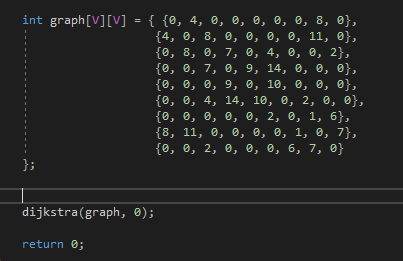
\*Bütün düğümleri ziyaret edilmemiş olarak gösteriyoruz. Birde başlangıç noktasının maliyetini her zaman sıfır olarak gösteririz.



\*Bütün düğümlere olan uzaklığın bulunacağı döngüye giriş yapıp henüz ziyaret edilmemiş en az maliyetli düğümü u değişkenine aktarıyoruz. Aynı zamanda diğer komşu düğümleri de tarıyoruz.



\*Sıfır düğümünü başlangıç noktası olarak belirliyoruz ve ağırlıklı graf yapımızı aşağıdaki gibi oluşturuyoruz.



# Kaynakça

\*https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

\*https://compro.com.tr/AC922/index.html?gclid=EAIaIQobChMI1\_zI5NaB6QIVhOd3Ch23gQ56EAAYASAAEgKievD\_BwE