**-**

**MESLEK YÜKSEKOKULU**

**Bölüm Adı:** Bilgisayar Teknolojileri

**Program Adı:** Bilgisayar Programcılığı

**Ders Adı:** Yapay Zekâ

**Proje Adı:** En kısa yol bulunma uygulaması

**Hazırlayan**

**Adı - Soyadı:** Onur İNCİK

**Okul Numarası:** 18MY03029

**Öğretim Görevlisi:** Nilgün İNCEREİS

İçindekiler

[GENEL BİLGİLER 3](#_Toc38560067)

[Yapay zekâ nedir? 3](#_Toc38560068)

[Yapay zekâ tarihi: 3](#_Toc38560069)

[Yapay Zekâ İsmi Nereden Geliyor? 3](#_Toc38560070)

[Yapay Zekâ Nasıl Yapılır? 3](#_Toc38560071)

[Proje Konusu: 4](#_Toc38560072)

[Projemde kullandığım algoritma: 4](#_Toc38560073)

[Warshall Algoritması Nedir? 4](#_Toc38560074)

[Warshall Algoritmasının Kullanım Alanları 4](#_Toc38560075)

[Uygulama geliştirme ortamı ve kullanılacak programlama dili: 5](#_Toc38560076)

[Projede izlenilen yol: 5](#_Toc38560077)

[Projemin Konusu: 6](#_Toc38560078)

[Projemde Kullandım özelliklerin açıklaması: 8](#_Toc38560079)

[Kod Kısmı: 8](#_Toc38560080)

[Uygulamamın konsolda ki tasarımı bu şekilde: 12](#_Toc38560081)

[Kaynakça: 12](#_Toc38560082)

# GENEL BİLGİLER

## Yapay zekâ nedir?

Gelen [bilgisayar bilimleri](https://en.wikipedia.org/wiki/Computer_science) , yapay zekâ ( AI bazen de denir), makine istihbarat, bir [istihbarat](https://en.wikipedia.org/wiki/Intelligence) gösterdiği [makinelere](https://en.wikipedia.org/wiki/Machine) aksine, doğal istihbarat tarafından görüntülenen [insanlarda](https://en.wikipedia.org/wiki/Human_intelligence) ve [hayvanlarda](https://en.wikipedia.org/wiki/Animal_cognition) . Önde gelen AI ders kitapları, alanı " [akıllı ajanlar](https://en.wikipedia.org/wiki/Intelligent_agent) " çalışması olarak tanımlar çevresini algılayan ve hedeflerine başarılı bir şekilde ulaşma şansını en üst düzeye çıkaran herhangi bir cihaz. Konuşma diline göre, "yapay zekâ" terimi genellikle "öğrenme" ve "problem çözme" gibi insanların [insan zihniyle](https://en.wikipedia.org/wiki/Human_mind) ilişkilendirdiği "bilişsel" işlevleri taklit eden makineleri (veya bilgisayarları) tanımlamak için kullanılır.

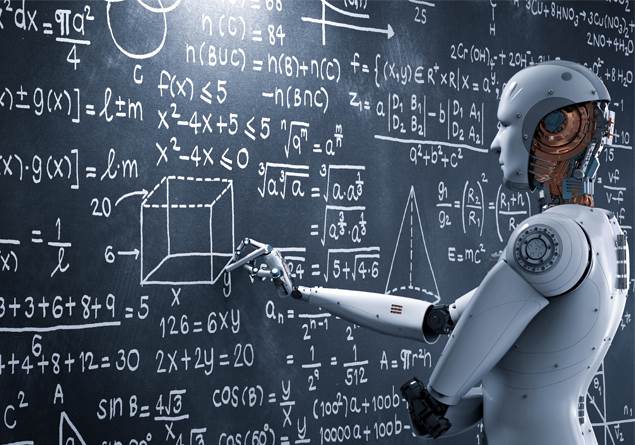
## Yapay zekâ tarihi:

Yapay zekâ (AI), bir insanın bilişsel yeteneklerini taklit etmeyi amaçlayan bir dizi bilim, teori ve teknik (matematiksel mantık, istatistik, olasılıklar, hesaplamalı nörobiyoloji, bilgisayar bilimi dâhil) olan altmış yıllık genç bir disiplindir. İkinci Dünya Savaşı'nın nefesiyle başlatılan gelişmeleri, bilgisayarla yakından ilişkilidir ve bilgisayarların daha önce sadece bir insana devredilebilecek giderek daha karmaşık görevleri yerine getirmesine yol açmıştır.

Bununla birlikte, bu otomasyon, tam anlamıyla insan zekâsından uzak durmaktadır, bu da adı bazı uzmanlar tarafından eleştiriye açık hale getirmektedir. Araştırmalarının nihai aşaması ("güçlü" bir AI, yani çok farklı uzmanlık sorunlarını tamamen özerk bir şekilde bağlamsallaştırma yeteneği) mevcut başarılarla ("zayıf" veya "orta" AI'lar, eğitimlerinde son derece verimli) kesinlikle karşılaştırılamaz alan). Henüz bilim kurguda gerçekleşen "güçlü" yapay zekâ, dünyayı bir bütün olarak modelleyebilmek için temel araştırmalarda (sadece performans iyileştirmelerinde değil) ilerlemeler gerektirecektir.

## Yapay Zekâ İsmi Nereden Geliyor?

1940'larda ve 50'lerde, çeşitli alanlardan (matematik, psikoloji, mühendislik, ekonomi ve siyaset bilimi) bir avuç bilim adamı yapay bir beyin yaratma olasılığını tartışmaya başladı. [Yapay zekâ](https://en.wikipedia.org/wiki/Artificial_intelligence) araştırması alanı 1956 yılında akademik bir disiplin olarak kuruldu.

****

Proje Konusu: Programı kullanan kişilerin bulmak istedikleri graf uzunluklarının en kısa yolu bulmasını sağlayan bir uygulama program kullanıcı aralarındaki en kısa yolu bulmak istediği başlangıç düğümünü isteyeceğim ve aralarındaki en kısa yolu bulmak istediği varış düğümünü girdikten uygulama ona göre kısa yolu çizecek.

## Projemde kullandığım algoritma:

İlk önce projem için araştırmalar yapmaya başladım hangi platformda yazmalıyım ve hangi algoritmalardan yararlanmalıyım bunun için internette kaynak arayışına girdim araştırmalarım sonucunda Floyd-**Warshall** algoritmasını buldum ve biraz araştırma yaparak ne olduğunu öğrendim araştırmam sonucunda bazı veriler buldum ve hangisini ne için kullanmam gerektiğine karar verdim. Bu kavramların özellikleri kısaca şu şekilde.

# Warshall Algoritması Nedir?

Bilgisayar bilimlerinde, Warshall algoritması, pozitif veya negatif kenar ağırlıklarına sahip ağırlıklı bir grafikte en kısa yolları bulmak için bir algoritmadır. Warshall algoritmasının tek bir uygulaması tüm köşe çiftleri arasındaki en kısa yolların uzunluğunu bulacaktır.

Bilgisayar bilimlerinin önemli konularından olan algoritma analizi sırasında sıkça bahsi geçen bir algoritmadır. Algoritmanın ana amacı belirli bir graf üzerinde bir başlangıçtan(source) bir bitiş düğümüne (sink, end, target)  en kısa yoldan (shortest path) ulaşmaktır.

İlgimi çeken bir özellik ise algoritmayı bulan iki kişinin ismi ile anılması.

# ****Warshall Algoritmasının Kullanım Alanları****

* Yönlü graflarda en kısa yolun bulunması için. (Yukarıda bu durumu gösteren bir örnek bulunmakta)
* Bir düğümden seyahat edilebilecek diğer düğümlerin bulunmasında (transitive closure). Örneğin matrisin ilk halinde gidilemeyen düğümler sonsuz değerine sahiptir. Şayet matris işlendikten sonra hala sonsuz değerine sahip matris elemanı bulunursa bunun anlamı o satır ve sütundaki düğümler arasında gidişin aktarmalı da olsa mümkün olmadığıdır.
* Ağ programlamasında özellikle yönlendirici algoritmalarında (routing algorithms) en kısa yolun tayin edilmesinde kullanılabilir.
* Yönsüz bir grafın (undirected graph), iki parçalı graf (bipartite graph) olup olmadığının bulunmasında kullanılabilir. Basitçe graftaki mesafelerin tamamını 1 uzunluğunda kabul edersek (veya mesafe olarak atlanan düğüm sayısını (hop count) kabul edersek) bu durumda bir düğümden gidilebilen bütün komşu düğümlerin mesafesi ya tek ya da çift olmalıdır. Bir düğümün hem tek hem de çift komşusu varsa bipartite değildir denilebilir.

## Uygulama geliştirme ortamı ve kullanılacak programlama dili:

Uygulamam C# konsol üzerinden çalışacak.

## Projede izlenilen yol:

**1.Aşama=** Kullanıcıdan Grafında bulunan düğüm sayısını doğru bir şekilde girmesini isteyeceğim kullanıcı girdikten sonra uygulamada graf oluşturuluyor.

Grafı dizi üzerinde tutarak dizide oluşturacağım iki matriside farklı dizilerde kaydedilecek.

Ve kullanıcıdan matris değerleri istenilecek.

Kullanıcının verdiği değerler for döngüsü bloğu içerisinde girilen matrise göre floyd ile hedef matris oluşturulur

**2.Aşama=** Graf üzerinde eğer sonsuz değer tanımı varsa o değerleri girmesini isteyeceğim.

**3.Aşama=** Kullanıcı aralarındaki en kısa yolu bulmak istediği başlangıç düğümünü girmesini isteyeceğim ve uygulama ona göre yol çizilecek.

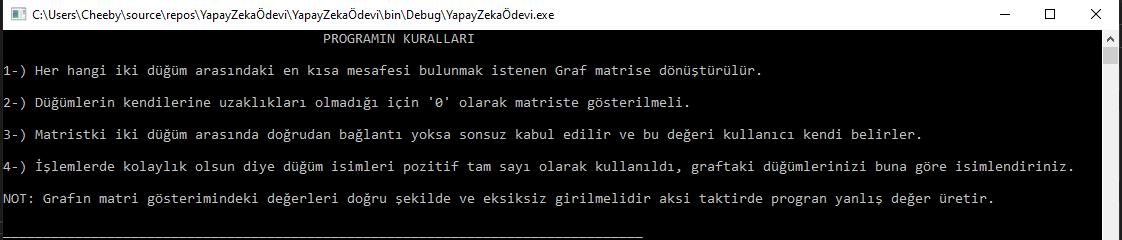
**4.Aşama=** Aralarındaki en kısa yolu bulmak istediği varış düğümünü girdikten sonra ona göre bir yol çizilecek.

**5.Aşama=** Kullanıcının işlemi bittikten sonra başka iki düğüm arası uzaklık bulmak istersen programı en baştan başlatması için bir tuş ataması yapacağım ve o tuşa bastığında en baştan yapabilecek her şeyi.

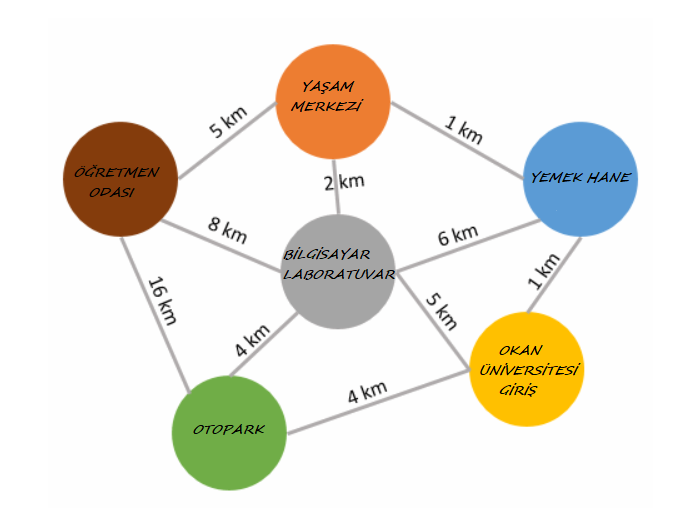
**Kullanıcıdan istenilen veriler şu şekilde:**

* Düğüm sayısını girilmesi.
* Sonsuz değer tanımı değerini girilmesi.
* En kısa yolu bulmak istediğiniz Başlangıç düğümünü girilmesi.
* Aralarındaki en kısa yolu bulmak istediğiniz Varış düğümünü girilmesi.
* En son olarak kullanıcı başka bir yol bulmak isterse uygulama başa döndürülecek.

Tasarım kısmında bunu şu şekilde yansıttım konsol olduğu için kullanıcıya kuralları belli etmem gerekiyordu bu yüzden oyun kurallarını yazdım ki kullanıcı doğru cevap alabilsin.

****

## Projemin Konusu:

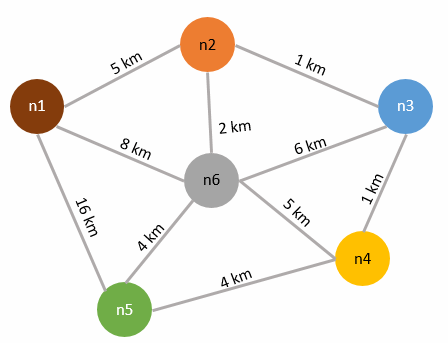


Örneğim sınıfta duruyoruz ve Okan üniversitesi girişe gitmemiz gerekiyor. Direkt bir yol kullanarak gidersek 5 km yol gitmek gerekiyor. Başka türlü ilk Yaşam merkezine, Yemek Haneye ve oradan’ da Okan üniversitesi giriş bölümüne geçmiş kabul edersek toplamda

4 km yol kat etmiş oluyoruz. 1 km kazancımız var bu güzergâhı takip ederek gidersek. Eğer önce Yemek haneye oradan Okan üniversitesi girişe gidersek 7 km yol kat etmiş oluyoruz. Senaryoyu biraz daha farklı yapalım. Diyelim ki evden Otoparka gideceğiz. Arabamızı alacağız. Doğrudan geçiş yaparsak 4 km yol gitmemiz gerekecek. Farklı seçeneklerde tercih edebiliriz. Örnek vermek gerekirse Okan üniversitesi giriş üzerinden geçiş yaparsak 9km, Öğretmenler odası üzerinden geçip gidersek 24km yol. Bunun gibi bir yerden diğer bir yere giderken pek çok yol denenebilir ve mesafe değiştirilebilir.

**Tam olarak Floyd-Warshall** algoritması bir boğumdan diğer bir boğuma gidebilmek için kullanılabilir en kısa gidiş yolunu çıkartılmasında kullanılarak karar vermemizi sağlar.

Bu mantığı biraz’ da matematiksel açıdan yapacak olursak. Yollar arasındaki en kısa mesafeleri bulmak istersek şu şekilde olacak.

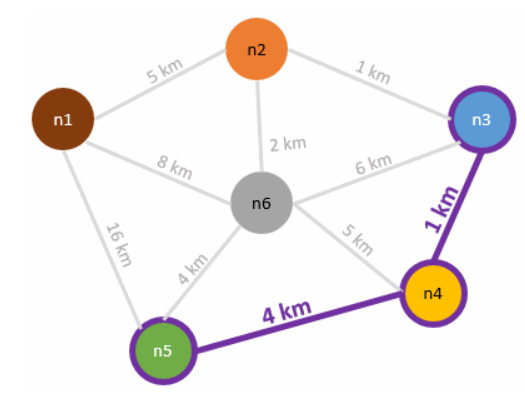


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | N1 | N2 | N3 | N4 | N5 | N6 |
| N1 | 0 | 5 | ∞ | ∞ | 16 | 8 |
| N2 | 5 | 0 | 1 | ∞ | ∞ | 2 |
| N3 | ∞ | 1 | 0 | 1 | ∞ | 6 |
| N4 | ∞ | ∞ | 1 | 0 | 4 | 5 |
| N5 | 16 | ∞ | ∞ | 4 | 0 | 4 |
| N6 | 8 | 2 | 6 | 5 | 4 | 0 |

Hücrelerin bazılarında sonsuzluk sembolü, bazılarında ise sıfır değeri var. İki boyutlu matris boğumların birbirlerine en yakın diğer boğuma olan gidiş mesafelerini göstermektedir. Boğumun kendisiyle arasındaki mesafe 0, direkt bağlı olmadığı boğumlar arasındaki mesafesi sonsuz sembolü ile işaret edilir. Mesela n1 boğumundan n3 ve n4 boğumlarına doğrudan bir hat olmadığı için sonsuz sembolü kullanıldı. Algoritmanın becerisi sonsuz sembollerini eritmek ve hatta sayısal değer alan hücrelerde olabilecek daha kısa mesafeler var ise bunları matris üzerinde güncellemektir.

Örneğin n1’den n3’e direkt gidişimiz olmadığından sonsuz olarak işaretlenmiş durumda. Ya da n1->n2->n3 şeklinde bir ulaşımda kullanılabilir. Yani n2 üzerinden geçiş yaparak n3’e varabiliriz. Elbette n3’e varmak için n6 üzerinden de hareket edebiliriz. Yani n1->n6->n3 şeklinde bir güzergâh da söz konusu olabilir. Hatta n1->n5->n4->n3 şeklinde de gidebiliriz.

Buna göre bir noktadan bir noktaya gidilebilecek en kısa mesafeler bulunmuştur. Örneğin **n3 noktasından n5 noktasına gitmek istediğimizde en kısa güzergâh 5km uzunluğunda olup n3->n4->n5 gidiş yolu** şeklindedir. Diğer kullanılabilir yollara bakıldığında gerçekten’ de en az yolun bu olduğunu görebiliriz.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | N1 | N2 | N3 | N4 | N5 | N6 |
| N1 | 0 | 5 | 6 | 7 | 11 | 7 |
| N2 | 5 | 0 | 1 | 2 | 6 | 2 |
| N3 | 6 | 1 | 0 | 1 | 5 | 3 |
| N4 | 7 | 2 | 1 | 0 | 4 | 4 |
| N5 | 11 | 6 | 5 | 4 | 0 | 4 |
| N6 | 7 | 2 | 3 | 4 | 4 | 0 |

Görsel olarak anlatımı bu şekilde program aynı şekilde kullanıcının girdiği değerler üzerinden. Benim uygulamamda ki kodlar floyd algoritmasına göre programa girilen her  hangi ağırlıklı ya da ağırlıksız graf için her hangi bir düğümden her hangi bir düğüme olan en kısa mesafeyi hesaplayacak bir program.

## Projemde Kullandım özelliklerin açıklaması:

**Nesne:** İçinde veri saklayan ve bu veriler üzerinde işlem yapacak olan metodlar bulunduran bileşenlerdir.

**Dizi:** Dizi bir kümedir. Aynı tipte verilere tek bir isimle erişmek için kullanılır. Bir dizinin bütün elemanları bellekte peş peşe saklanırlar.

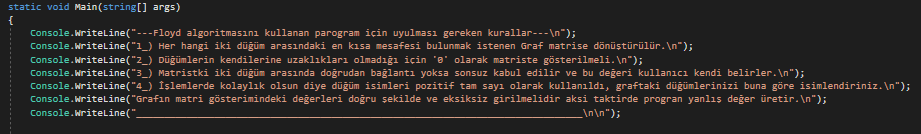
**Liste:** listelerde de veriler tutulur. String bir değişkende verilerimizin karakter olmasının aksine listelerde bu veriler herhangi bir değişken tipindeki veriler olabilir.

**Döngüler:** Döngü bir kod bloğunun X defa tekrar edilmesini sağlar. Temel anlamda 4 farklı döngü(loop) vardır.

**Toarray:** yöntemi, veri kümesinden select bölümündeki seçilen elemanlardan oluşan bir dizi döndürür.

**ReadKey:** Kullanıcı tarafından basılan bir sonraki karakteri veya işlev tuşunu alır. Basılan tuşu, isteğe bağlı olarak konsol penceresinde görüntülenir.

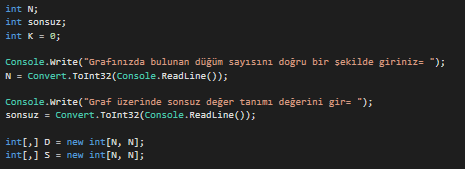
## Kod Kısmı:



Kullanıcıdan bir floyd üçgeni oluşturmak için veriler almamız gerekiyor bunun için ilk önce 3 adet nesne oluşturuyoruz

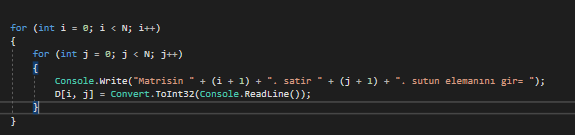
N nesnesi için graf düğüm sayılarını istiyoruz

Sonsuz nesnesi içinse grafta’ki sonsuz değer tanımını girmesini istiyoruz.

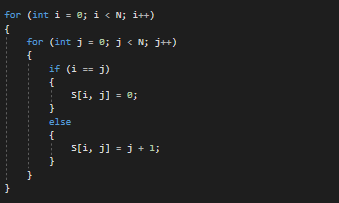


Kullanıcıdan aldığımız verileri ise D ve S atlı dizilerde tutuyoruz. Böylece matrisimizi oluştururken kullanıcının girdiği verilere göre floyd ve matrisimizi oluşturuyoruz.

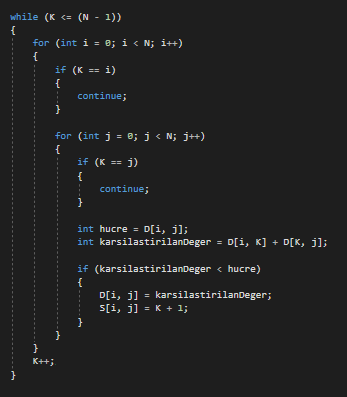
Kullanıcılardan ilk önce matris değerlerini alıyoruz kullanıcın her girdiği değeri diziye gönderiyoruz.



Daha sonra S matrisinin değerlerini atama işlemini gerçekleştiriyoruz.



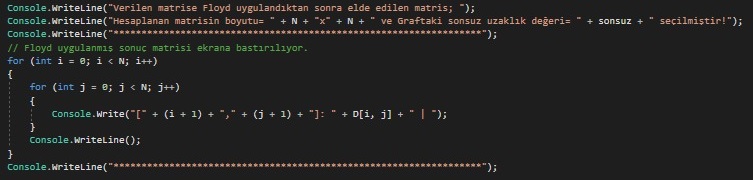
Kullanıcıdan aldığımız matris bilgilerine göre floyd ve hedef matrisi oluşturuyoruz bunu 3 döngü ile gerçekleştirdim.



Hücre olarak atadığım D nesnesi kullanıcıdan alınan matrisin işlem yapılan hücresi olarak tutulur. Karşılaştırılan değerde ise En kısa yolu göstermek için hücrenin kıyaslandığı yolu gösteriyor ve S matrisi güncelleniyor

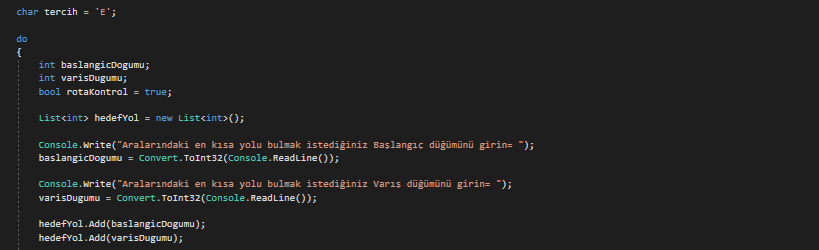
**K++ :** Matris her dizide baştan sona tarandığında k değerini 1 arttırmasını söylüyor böyle gidilen yol en son hangisiyle eşleşirse k nesnesinde o değer kalmış oluyor.

Verilen matrisi floyd uygulandıktan sonra elde edilen matris ekrana yazdırmak için tasarım kısmında şu şekilde bir fikirden yararlandım yorum ekleme kısmına parantezler koyarak alınan matrisleri o parantezlerin içine yazdırdım böylece konsolda kullanıcı girdiği verileri düzenli bir şekilde tasarımda görmüş oluyor ve karışıklık ortadan kalkıyor.



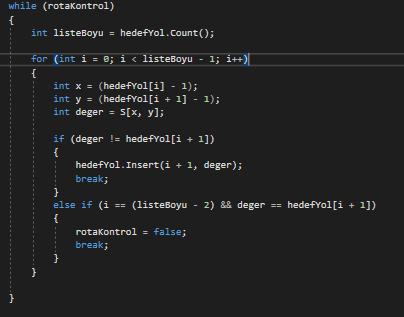
Kullanıcının oluşturmuş olduğu graf içinde bulmak istediği kısa yolu sordurma işlemi burada başlıyor böylece kullanıcıdan aldığımız matris verileri ile grafı oluşturmuş oluyoruz.

Char ile E adında bir tercih nesnesi yaratıyoruz bunun sebebi ise kullanıcı tekrardan en kısa yol bulmak isterse programı kapat aç yapmak zorunda kalmadan döngüye tekrar yönlendirilsin diye tercih sunuyoruz.

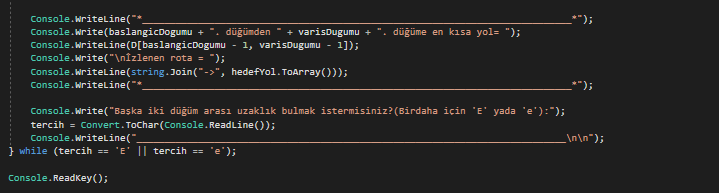


Kullanıcıdan en kısa yolu bulması istenilen matristeki başlangıç düğümü ve varış düğümünü girmesi isteniliyor. S matrisinden grafın başlangıç düğümünden itibaren yolu almak amacıyla bir liste oluşturuyoruz.

Rota kontrol nesnesini kullanarak bir döngü oluşturuyor böylece kullanıcıya geçilen yolları gösteriyoruz.

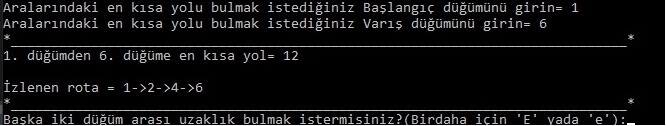
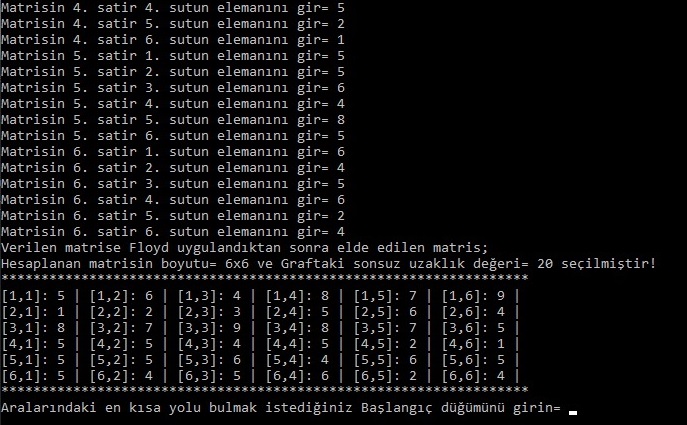
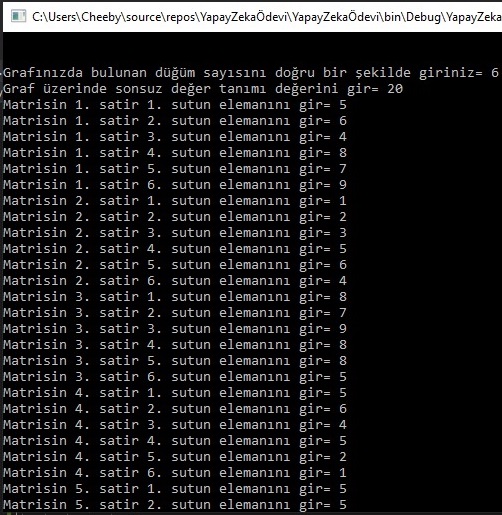
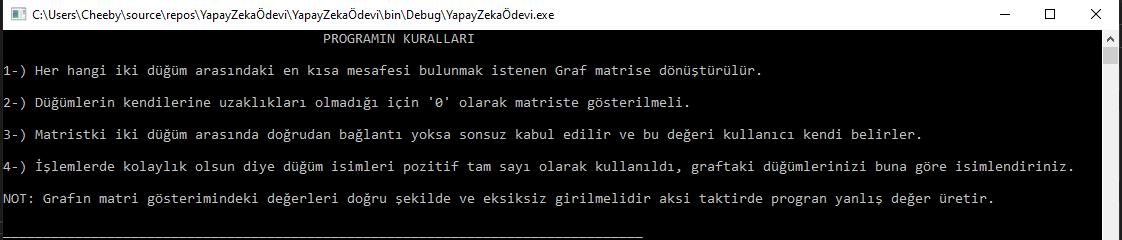


Kullanıcının verdiği bilgilere göre rota kontrol ile listede gidilecek yollar tek tek bakılıyor Başlangıç noktasından bitiş noktasına bütün matrislerden geçerek en kısa yol bulunuyor ve bu yol console yazdırılıyor.



Kullanıcı eğer başka bir kısa yol bulmak isterse tekrar bakmak isterseniz e tuşuna basmasını söylüyoruz yukarda oluşturduğumuz char “E” nesnesi sayesinde döngüye tekrar girerek baştan tercih yapabiliyor.

Uygulamamın konsolda ki tasarımı bu şekilde:



## 

## Kaynakça:

<https://www.neoldu.com/yapay-zeka-teknolojisi-nedir-hakkinda-bilgi-38130h.htm>

<http://www.balikesir.edu.tr/~ieee/index.php/2019/10/29/yapay-zeka-artificial-intelligence/>

<http://bilgisayarkavramlari.sadievrenseker.com/2009/05/29/floyd-warshall-algoritmasi/>