

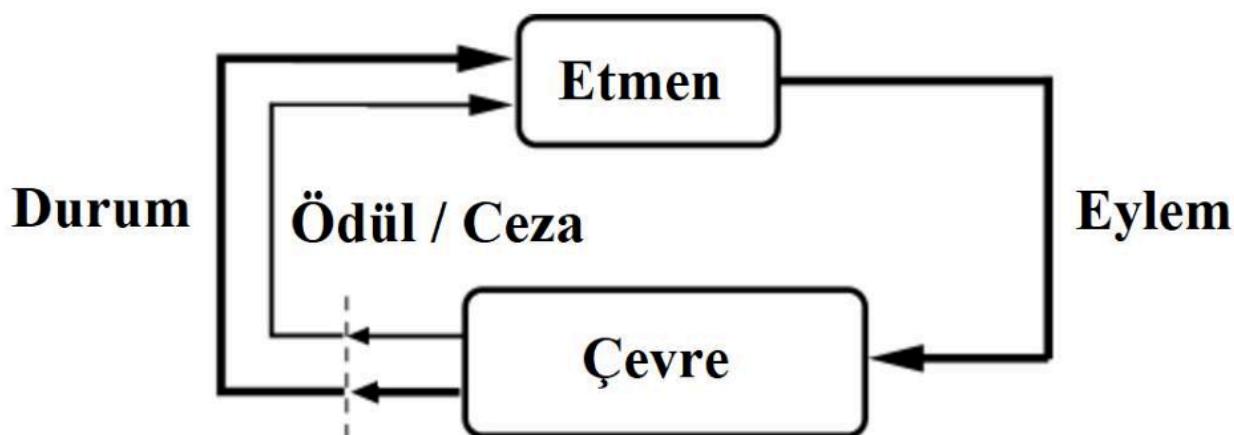
# Etmen Tabanlı Yapay Zekâ Modelleme Temelleri

Agent yapay zeka

- YAPAY ZEKA NASIL AKSİYONA ÇEVİRİR
- SADECE CEVAP ÜRETME İNTERNET KULLAN KARMAŞIK GÖREVLERİ YAP

Yapay Zekâ kapsamında çevresel faktörlerle etkileşim gerektiren problem çözümlemeleri için tasarlanmış çeşitli yaklaşımalar bulunmaktadır. Bu yaklaşımaların özünde şu ana dek dephinilen farklı Yapay Zekâ algoritmalarının etkileşimsel fonksiyonları içerecek şekilde tasarlanması yer almaktadır. Buna göre, çevreyle etkileşen Yapay Zekâ unsurları kısaca etmen ya da İngilizce karşılığında esinlenerek ajan (agent) olarak adlandırılmaktadır.

Görselen anlaşılmak üzere bir etmen, çevresiyle etkileşime girip girdiği etkileşim sonucunda elde ettiği bulgulara göre dönütlerde (yani eylemlerde) bulunabilecek şekilde tasarlanmaktadır. Bir etmenin eylemlerine karar verme aşamasında içerisinde bulunduğu durum ve çevreden aldığı ödül/ceza dönütleri etkili olmakta, bunlar gelecek durumları da yönlendirmektedir. Buradan hareketle, özellikle kolektif, çok sayıda unsurun bir araya gelerek çözüm üretmesi gereken problem durumlarında birden fazla etmen de kullanılabilmektedir. Genel olarak bu yöndeki Yapay Zekâ tabanlı çözüm yaklaşımı etmen tabanlı Yapay Zekâ, bu yöndeki çözüm süreçlerinin tasarımına da etmen tabanlı modelleme adı verilmektedir.



Şekil 6.1. Tipik bir etmenin genel yapısı (Web Kaynağı 6.1)

**Örnek: Sıcak Nesne Deneyimi** Bir çocuğun sıcak bir nesneye dokunması bir eylemdir. Canının yanması (ceza/negatif geri bildirim), çocuğun "sıcağa dokunmamalıym" bilgisini öğrenmesini sağlar. Bu, etmen tabanlı modellemenin temelidir.

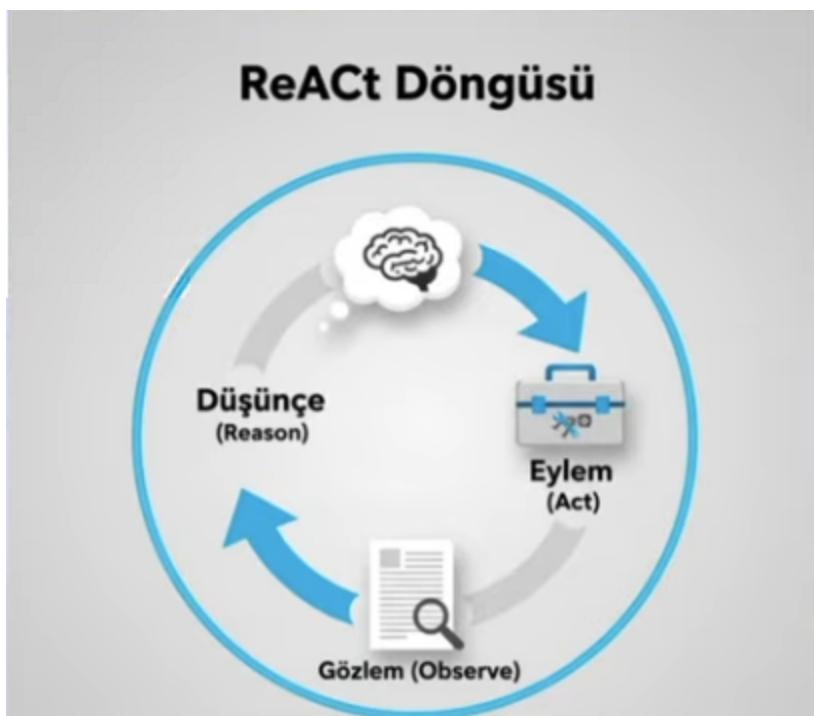
## LLM ler (KAHİN)

- Kendi mağarasında (eğitim verisi ile oturur)
- Sadece bildiklerini (statik bilgileri söylerler)
- Bugün hava nası ne bilim 2024 den sonrasında ulaşamıyorum

## Generatif yapay zeka AI agent lar

- Beyin ve Araç kutusu güç daha çok araç kutusu neleri olduğuna bağlı robotlar için sensör chat gpt için hangi siteler hangi bilgiler
- Bu gün hava nası mağradan çıkar hava durumu aracını kullanır bilgiyi alır eleştirir yorumlar cevap verir.

Temel döngü :





## Etmen Tabanlı Modellemede Problem Tasarımı

Etmen tabanlı çözümlerin çalışacağı problemlerde, etkin çözüm elde etmek adına birtakım faktörlerin/bileşenlerin dikkate alınması ve bunlar altında tanımlamalar yapılması gerekmektedir. İlgili faktörleri/bileşenleri genel olarak şöyle ifade edebiliriz:

Robot süpürgeyi bir etmen olarak kabul edersek; kamera, kızılıötesi gibi sensörler ile çevre algılama sağlanırken, dinamik tepki vericiler olarak çeşitli

motorlar, fırçalar, mekanik kollar vs. ile problem çözümü (çevreyi temizlemek/süpürmek) sağlanır.

- Etmen Sayısı: Çözümde kullanılacak etmenlerin sayısı, tek bir etmenin mi yoksa çok sayıda etmenin mi (kolektif bir şekilde) çalışacağı belirlenmelidir. Bazı problemler tek bir etmen ile kolayca çözülebilirken, bazı problemler de çok sayıda etmenin hem çevreyle hem de birbirleriyle iletişim halinde olarak çalışmasını ve çözüme ulaşmasını gerekliliğinden kılmalıdır.

#### YAZAR - ELEŞTİRİMEN - ARAŞTIRMACI - PROJE



- Etmen Nitelikleri: Çözümde yer alacak etmenlerin sahip olacakları muhtemel parametreler aynı zamanda etmen nitelikleri olarak bilinmektedir. Bu parametreler, etmenlerin eylemlerine, muhakemelerine ve çevreyle ya da diğer unsurlarla (örneğin diğer etmenler) etkileşimlerine yön verecek, düzenlenebilir değerler olmaktadır.
- Etmen Eylemleri: Etmen eylemleri, ilgili etmenlerin çevreyle etkileşimleri ve geçerli nitelik değerleri üzerinden karar süreçlerini işletmelerini ve hatta eylemde bulunmalarını içermektedir. Bu eylem yapıları aslında birer fonksiyon olarak tanımlanmaktadır.
- Çevre: Etmenlerin içerisinde bulunduğu çevrenin, en iyi etmen etkileşimleri için sınırları ve karakteristik nitelikleri ile tanımlanmaları gerekmektedir.

Bileşen	Açıklama	Örnek (Robot Süpürge)
<b>Etmen Sayısı</b>	Tek bir etmen mi yoksa kolektif çalışan çoklu etmenler mi?	Tek bir süpürge veya sürü halinde çalışan robotlar.
<b>Etmen Nitelikleri - Tool</b>	Etmenin sahip olduğu sabit veya değişken parametreler.	Pil seviyesi, emiş gücü, koordinat bilgisi.

Bileşen	Açıklama	Örnek (Robot Süpürge)
<b>Etmen Eylemleri</b>	Etmenin yapabileceği fonksiyonel hareketler.	İleri gitme, sağa dönme, fırçayı çalışma.
<b>Çevre</b>	Etmenin hareket ettiği sınırları belirlenmiş alan.	Evin odaları, duvarlar, halılar.

Benzer şekilde; bir insan olarak da çevreyle etkileşimlerimiz neticesinde kararlar alabilen ve eylemler gerçekleştiren unsurlar olarak etmen tabanlı modellemeyle olan benzerliklerimiz konusunda örnekler (örneğin markette kasıyer ile olan iletişimimiz, bir çocuğun sıcak bir unsura elini değiirdiği zaman ulaştığı tecrübe: 'sıcağa dokunmamalıym'...vs.) vererek, etmen kavramına ilişkin pekiştirici örnekleri de vurgulayarak, öğrencilerin kavramları daha iyi anlamasını sağlayacaktır.

İfade edilen unsurlar ile etmenlerin içerisinde bulunduğu problemin de tanımlanması gerekmektedir. Problemle ilgili olarak şu tanım faktörleri önemlidir:

**Problem Kısıtları:** Problem ile ilgili çözümü temsil eden parametrelerin hangi kısıtlar altında olacağı, etmen davranışları neticesindeki çıktıların da gidişatını belirlemektedir.

**Çevredeki Dinamik Unsurlar:** Etmenlerin eylemleri neticesinde durumları değiştirebilecek dinamik çevrenin söz konusu olacağı gibi, çevrede yer alacak ve etmenlerin gelecek davranışlarını şekillendirecek başka dinamik unsurlar da söz konusu olabilmektedir. Özellikle çok etmenli modellemelerde başka etmenler de dinamik unsurlar olarak kabul edilmektedir. Yine tek veya çok etmenli problem çözümlerinde çevreyle bağlantılı değişken parametreli unsurlar da problem çözümünü benzetim odaklı problemler için daha uyumlu hale getirebilmektedir.

**Ödül/Ceza Fonksiyonları:** Etmenlerin eylemleri sonrasında gelecek kararlarını ve eylemlerini düzenleyecek birtakım çevresel ödüller veya cezalar dönüt olarak verilebilmektedir. Bu ödül ve cezalar çevredeki bazı unsurlarla etkileşimden doğabilecegi gibi, etmenler için tanımlanan kurallarla ya da etmenin çevrede hareket halinde olduğu her aşamada uygulanabilmektedir.

- Problem Kısıtları:** Etmenin hareket alanını ve davranış sınırlarını belirler (Örn: Duvarlar, merdiven boşlukları).
- Dinamik Unsurlar:** Çevrede zamanla değişen objeler (Örn: Hareket eden insanlar, yer değiştiren eşyalar).
- Ödül/Ceza Fonksiyonları:** Etmenin "doğu" yolu bulması için kullanılan puanlama sistemidir.

## Q-Öğrenme ile Öğrenen Etmen Modelleme

Etmen tabanlı çözümlerde yapay zekânın makine öğrenmesi alanında kullanılan öğrenme yöntemi takviyeli öğrenme (reinforcement learning) olarak bilinmektedir. Takviyeli öğrenme

gerçek hayatı yaşıyarak öğrenme gibidir. Bu öğrenmede genel olarak şu hususlar söz konusudur: Öğrenmede ödül/ceza mantığı vardır.

Ödül olarak algılanacak değerler daha yüksekkken, ceza olarak algılanan değerler düşük değerler olarak tasarlanır.

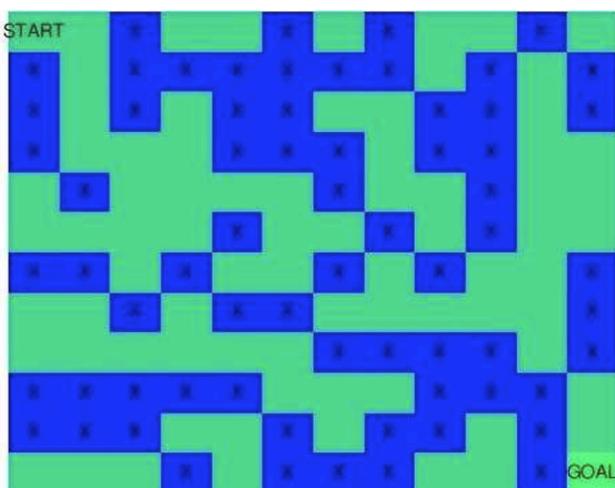
Takviyeli öğrenme yapan unsurun ödül/ceza değerini belirleyen belli eylemler vardır (duvara çarparsan eksi 1 puan, çarpmazsan -yoluna devam edersen- her zaman artı 1 puan gibi).

Takviyeli öğrenme yapacak unsur içerisinde bulunduğu ortam/problem için rastgele eylemlerde bulunur ve çok sayıda eylemlerle en iyi tecrübe kazanılması sağlanır. Etmen tabanlı çözümlerde takviyeli öğrenme için kullanılan temel tekniklerden biri de Q-Öğrenme (Q-Learning) olarak bilinmektedir.

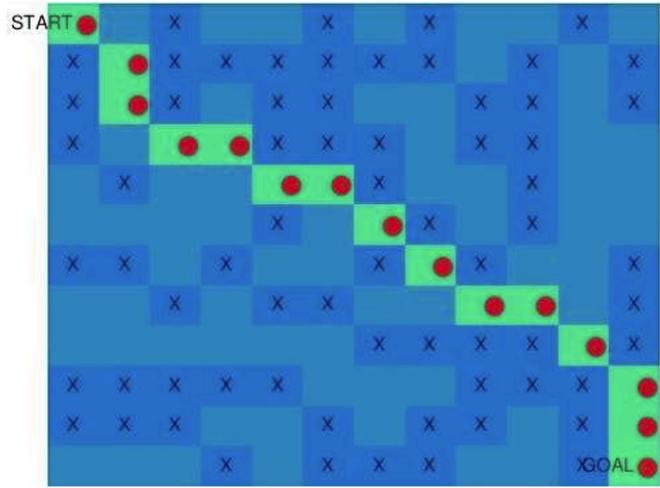
Q-Öğrenme'de daha önceden tasarlanmış ödül/ceza puanlarının yer aldığı bir tablo kullanılmak suretiyle, başlangıçta içi boş olan yeni bir Q tablosunun en uygun değerlerle doldurulması sağlanır.

Şekil 6.2a'da gösterildiği gibi; başlangıç noktası (start) ve bitiş noktası (goal) gösterilen bir problemde, engellerin olduğu her kareye negatif, engelsiz karelere ise pozitif puan verdiği düşünülürse; Q-Öğrenme ile gerçekleştirilen işlem sonucunda, etmenin Şekil 6.2b'de görüldüğü gibi kendisini amaca götüren en uygun yolu bulması sağlanmaktadır.

Bu örnekte soldaki görüntü önceden tasarlanmış ödül/ceza (puan) tablosuken, sağdaki görüntü başlangıçta içi boş olan ama daha sonra en uygun yoldaki değerlerin diğer karelere göre daha yüksek olduğu Q tablosudur.



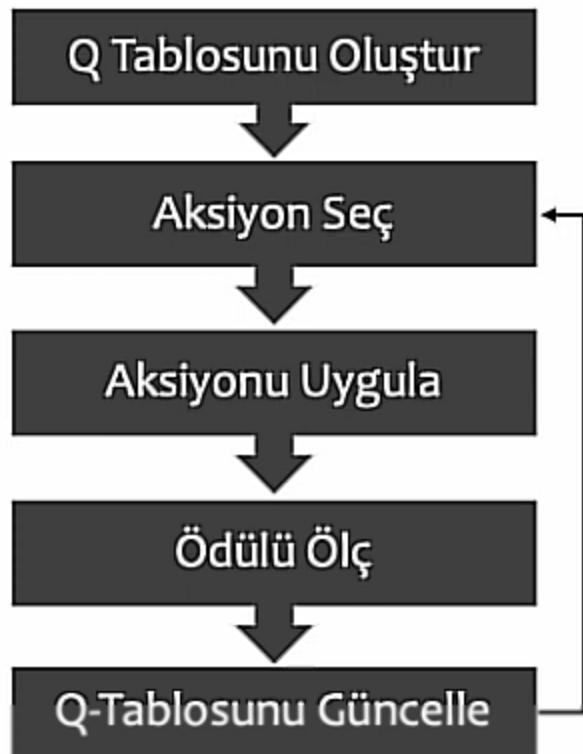
(a)



(b)

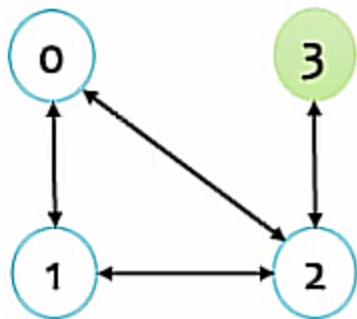
Şekil 6.2. Q-Öğrenme ile örnek bir problem akışı.

Q-Öğrenme'de etmen her seferinde rastgele adımları denemekte ve bu paragrafı takiben verilen formül sayesinde mevcut ödül/ceza tablosunu kullanmak suretiyle gelecek eylemlerinden gelebilecek maksimum değerleri (tipki satrançta gelecek hamleleri düşünmek gibi) dikkate almakta; yine öğrenmesini etkileyen diğer parametreleri ve yeni tablodaki ödül/ceza değerini harmanlayarak adım attığı eylemlerdeki değerlerin güncellenmesini sağlamaktadır. Böylelikle iyi eylemler zamanla daha fazla değer kazanırken, kötü eylemlerin değerlerinin zamanla azalması sağlanmaktadır.



**Q-Tablosu uygun hale gelene kadar bu işleme devam eder.**

$$Q^{\text{yeni}}(s_t, a_t) \leftarrow (1 - \alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{eski değer}} + \underbrace{\alpha}_{\text{öğrenme oranı}} \cdot \underbrace{\left( \underbrace{r_t}_{\text{ödül}} + \underbrace{\gamma}_{\text{azalma değeri}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\substack{\text{eylem sonucunda öğrenilen değer} \\ \text{gelecekteki tahmini maks. değer}}} \right)}_{\text{gelecekteki tahmini maks. değer}}$$



Gidilecek yolları belirleyen  
düğüm modeli

1. Q-Öğrenme teknigi çalışmadan önce elimizde eylemlerin puanlarını taşıyan bir tablomuz olur. Bununla birlikte içi boş değerlerden (sıfır) oluşan puan tablosuyla aynı satır-sütundan oluşan bir Q tablosu tasarılarız.

		Aksiyonlar			
		0	1	2	3
Durumlar	0	-1	0	0	-1
	1	0	-1	0	-1
	2	0	0	-1	100
	3	-1	-1	0	-1

Ödül tablosu

		Aksiyonlar			
		0	1	2	3
Durumlar	0	0	0	0	0
	1	0	0	0	0
	2	0	0	0	0
	3	0	0	0	0

Başlangıç Q-Table

2. Q-Öğrenme için öğrenme oranı Learning Rate ve azalma (gamma) değeri olarak iki reel sayı belirleriz. Bu sayılar öğrenme gücünü ve şans faktörünü etkiler.

3. Etmenimiz probleme göre ilk adımını atar.
4. Formülde eylem sonucunda öğrenilen değer kısmını hesaplamak için etmenin gelecek olası adımlarının bütün kombinasyonlarından (ilk başta boş tasarladığımız ama zamanla dolacak Q tablosundan) gelecek en büyük puanı azalma değeri ile çarpar ve eylem puanlarını taşıyan tablodan etmenin attığı adıma tekabül eden ödül değeriyle topladıktan sonra elde ettiğimiz değeri öğrenme oranı ile çarparız.
5. Dördüncü adım ile elde ettiğimiz değeri Q tablosunda mevcut olan değerin üzerine ekler, ardından  $(1 - \text{ öğrenme oranı})$  değeriyle çarparız.
6. Yeni elde ettiğimiz değer, etmenin adım attığı Q tablosu hücresindeki yeni değer olur.

### İlk İterasyon

- Ajan rastgele bir yere konumlanır.(0)
- Yapabileceği hareketlerin Q-tablosundaki değerine göre maksimum olanı seçer.(2)
- Q Fonksiyonu ile bu seçimi değerlendirilir.
- Q-Table güncellenir.

$$Q(0,2)=Q(0,2) + 0.7*(0+0.8*\max(Q(2,1), Q(2,3))-Q(0,2))$$

$$Q(0,2)=0+0.7*(0+0.8*0)-0$$

$$Q(0,2) \leftarrow 0$$

7. Üçüncü adımdan itibaren bütün işlem adımlarını bir durma sayısı/kriterine kadar tekrarlar, böylece Q tablosunu döngüsel/iteratif bir şekilde güncelleriz.

### Aksiyonlar

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

İlk İterasyon  
Sonucu Q-Table

## İkinci İterasyon (2,3)

- İlk iterasyondan 2 durumuna konumlannmıştık.
- Yapabileceği hareketlerin Q-tablosundaki değerine göre maksimum olanı seçer. (3e gideceğimizi varsayıyalım)
- Q Fonksiyonu ile bu seçimi değerlendirilir.
- Q-Table güncellenir.

$$Q(2,3)=Q(2,3) + 0.7*(100+0.8*\max(Q(3,2))-Q(2,3))$$

$$Q(2,3)=0+0.7*(100+0.8*0)-0$$

$$Q(2,3) \leftarrow 70$$

		Aksiyonlar			
		0	1	2	3
Durumlar	0	0	0	0	0
	1	0	0	0	0
	2	0	0	0	70
	3	0	0	0	0

İkinci İterasyon  
Sonucu Q-Table

## Uygulama ve kod

```
import numpy as np

ortam_satir_sayisi = 11
ortam_sutun_sayisi = 11      # 11 e 11 lik bir kare matris

q_degerleri = np.zeros((ortam_satir_sayisi, ortam_sutun_sayisi, 4))
# başlangıç tablosunun yazımı 3 de matris 11 satır 11 stün 4 derinlik

hareketler = ['yukari', 'sag', 'asagi', 'sol'] # hareketler bir string listede

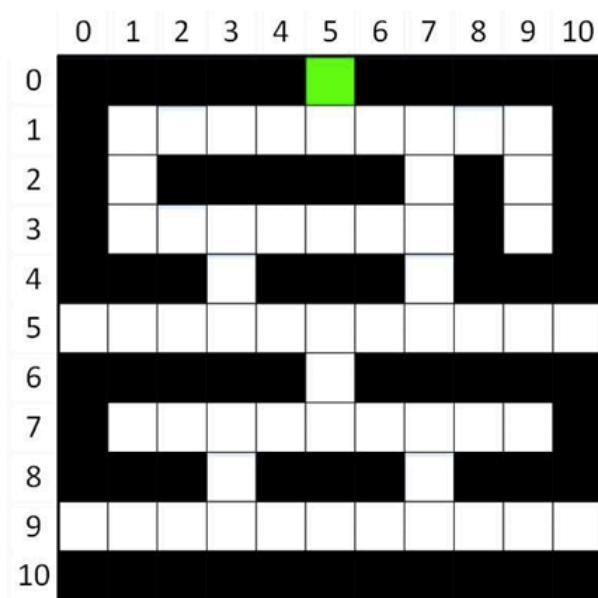
oduller = np.full((ortam_satir_sayisi, ortam_sutun_sayisi), -100.)
# ödül matrisi yazımı 11 satır 11 sutun haritada her yeri - 100 yaptım ki
baslangicta her yer siyah

oduller[0, 5] = 100.
# harimanın büyük ödül noktasını belirledim yani ödül tablomda her yer -100
bir tek bu nokta 100

gecitler = {}
```

```
# bu kod satiri ile güvenli yollar belirlenir
gecitler[1] = [i for i in range(1, 10)]
gecitler[2] = [1, 7, 9]
gecitler[3] = [i for i in range(1, 8)]
gecitler[3].append(9)
gecitler[4] = [3, 7]
gecitler[5] = [i for i in range(11)]
gecitler[6] = [5]
gecitler[7] = [i for i in range(1, 10)]
gecitler[8] = [3, 7]
gecitler[9] = [i for i in range(11)]
"""
{
1: [1, 2, 3, 4, 5, 6, 7, 8, 9],
2: [1, 7, 9],
3: [1, 2, 3, 4, 5, 6, 7, 9],
4: [3, 7],
5: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
6: [5],
7: [1, 2, 3, 4, 5, 6, 7, 8, 9],
8: [3, 7],
9: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
}
"""

```



```

for satir_indeks in range(1, 10):      # satır indeks 1 den 9 a kadar git

    for sutun_indeks in gecitler[satir_indeks]: # dictionarynin içindeki
listelere ulaşır mesela 2 key için sutun indeks [1, 7, 9] u kullanır ve
buradaki yerlere -1 atanır

        oduller[satir_indeks, sutun_indeks] = -1.

for satir in oduller:
    print(satir)
"""

[-100. -100. -100. -100. -100.  100. -100. -100. -100. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1. -100.]
[-100.   -1. -100. -100. -100. -100.   -1. -100.   -1. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1. -100.   -1. -100.]
[-100. -100. -100.   -1. -100. -100.   -1. -100. -100. -100.]
[-1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.]
[-100. -100. -100. -100. -100.   -1. -100. -100. -100. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1. -100.]
[-100. -100. -100.   -1. -100. -100.   -1. -100. -100. -100.]
[-1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.]
[-100. -100. -100. -100. -100. -100. -100. -100. -100. -100]

"""

```

```

# bool değer döndürü bulunan indeks odullerde neye karşılık geliyor
def engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
    if oduller[gecerli_satir_indeks, gecerli_sutun_indeks] == -1.:
        return False
    else:
        return True

```

```

def baslangic_belirle():

    gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
    gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)
    # random bir oduller haritasında yer ata ottam_satir_sayisi global
degiskenler

    while engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks): # koridora mı

```

```
denk geldim duvaramı eger duvarsa tekrar random bir noktaya at

gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)

return gecerli_satir_indeks, gecerli_sutun_indeks # duvar degilse bu döner
```

```
def sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks,
hareket_indeks):

    yeni_satir_indeks = gecerli_satir_indeks # hiçbir hareket olmassa eski
    yerinde kal
    yeni_sutun_indeks = gecerli_sutun_indeks# hiçbir hareket olmassa eski
    yerinde kal

    if hareketler[hareket_indeks] == 'yukari' and gecerli_satir_indeks > 0:
        # köşedemiyim kontrolü yapmak icin 0 . satırda yukarıya gitmesi eger haerek
        istegim yukarıya ise
        yeni_satir_indeks -= 1 # satırımı azalttım yani

    elif hareketler[hareket_indeks] == 'sag' and gecerli_sutun_indeks <
        ortam_sutun_sayisi - 1:

        yeni_sutun_indeks += 1

    elif hareketler[hareket_indeks] == 'asagi' and gecerli_satir_indeks <
        ortam_satir_sayisi - 1:

        yeni_satir_indeks += 1

    elif hareketler[hareket_indeks] == 'sol' and gecerli_sutun_indeks > 0:

        yeni_sutun_indeks -= 1

    return yeni_satir_indeks, yeni_sutun_indeks
```

```
def en_kisa_mesafe(basla_satir_indeks, basla_sutun_indeks): # maşıyet etkin
bir sonraki noktam neresi olsun

if engel_mi(basla_satir_indeks, basla_sutun_indeks):
    return [] # zaten engelin ustundeysem bos dizi gönder
```

```

else:

    gecerli_satir_indeks, gecerli_sutun_indeks = basla_satir_indeks,
basla_sutun_indeks

    en_kisa = []

    en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks]) # en kısa
listesine suanki konumu jayıt ettim

    while not engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
# engel noktası bulana kadar tara

        hareket_indeks = sonraki_hareket_belirle(gecerli_satir_indeks,
gecerli_sutun_indeks, 1.)

        gecerli_satir_indeks, gecerli_sutun_indeks =
sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks, hareket_indeks)

        en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])

    return en_kisa

```

```

# Epsilon-Greedy kullanılır
def sonraki_hareket_belirle(gecerli_satir_indeks, gecerli_sutun_indeks,
epsilon):

    if np.random.random() < epsilon: # np.random.random() 0 ile 1 arasında
rastgele bir sayı üretir.

    # np.argmax(...) Mevcut konumdaki Q-tablosuna bakar ve 4 hareket (yukarı,
aşağı, sağ, sol) arasından en yüksek puana (değere) sahip olan hareketin
indeksini döndür

    return np.argmax(q_değerleri[gecerli_satir_indeks, gecerli_sutun_indeks])
else:

    return np.random.randint(4) # 0 ile 4 arası sayı

```

Epsilon değerini nasıl seçtiğin, etmenin karakterini belirler:

- epsilon = 1.0 Etmen her zaman bildiği en iyi yolu seçer (Hiç risk almaz).
- epsilon = 0.0 Etmen her zaman rastgele hareket eder (Öğrenemez, sadece gezinir).

- **Gerçek Uygulama:** Genelde eğitim başında epsilon düşük tutulur (çok keşif), eğitim ilerledikçe epsilon artırılır (öğrenileni kullanma).

```

epsilon = 0.9 # haereket seçiminde randomluk vermek için
azalma_degeri = 0.9
ogrenme_orani = 0.9

for adim in range(1000): # bin iterasyonla q tablomuzun yazımı
    # gerçekleştirmektedir.

        satir_indeks, sutun_indeks = baslangic_belirle() # duvar olmayan random bir
        # nokta

        while not engel_mi(satir_indeks, sutun_indeks): # engele gidene kadar

            hareket_indeks = sonraki_hareket_belirle(satir_indeks, sutun_indeks,
            epsilon)

            eski_satir_indeks, eski_sutun_indeks = satir_indeks, sutun_indeks

            satir_indeks, sutun_indeks = sonraki_noktaya_git(satir_indeks,
            sutun_indeks, hareket_indeks)

            odul = oduller[satir_indeks, sutun_indeks]

            eski_q_degeri = q_degerleri[eski_satir_indeks, eski_sutun_indeks,
            hareket_indeks]

            fark = odul + (azalma_degeri * np.max(q_degerleri[satir_indeks,
            sutun_indeks])) - eski_q_degeri

            yeni_q_degeri = eski_q_degeri + (ogrenme_orani * fark) # q tablosunda
            # güncellenecek değer bulundu

            q_degerleri[eski_satir_indeks, eski_sutun_indeks, hareket_indeks] =
            yeni_q_degeri # q tablosu güncellendi

print('Eğitim tamamlandı.') # okuma ve ekrana yazdırma

ogr_sonrasi_satir = input('Robotun harekete başlayacağı satır indeksini
giriniz:')

```

```

ogr_sonrasi_sutun = input('Robotun harekete başlayacağı sutun indeksini giriniz:')

print('Kargo noktasına giden rota:',
      en_kisa_mesafe(int(ogr_sonrasi_satir), int(ogr_sonrasi_sutun)))
# okunan değerler string sen burda int e cast ediyorsun

```

## Açıklamasız kod

```

import numpy as np

ortam_satir_sayisi = 11

ortam_sutun_sayisi = 11

q_değerleri = np.zeros((ortam_satir_sayisi, ortam_sutun_sayisi, 4))

hareketler = ['yukarı', 'sağ', 'aşağı', 'sol']

oduller = np.full((ortam_satir_sayisi, ortam_sutun_sayisi), -100.)

oduller[0, 5] = 100.

gecitler = {}

gecitler[1] = [i for i in range(1, 10)]

gecitler[2] = [1, 7, 9]

gecitler[3] = [i for i in range(1, 8)]

gecitler[3].append(9)

gecitler[4] = [3, 7]

gecitler[5] = [i for i in range(11)]

gecitler[6] = [5]

gecitler[7] = [i for i in range(1, 10)]

gecitler[8] = [3, 7]

```



```
gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)

return gecerli_satir_indeks, gecerli_sutun_indeks

def sonraki_hareket_belirle(gecerli_satir_indeks, gecerli_sutun_indeks,
epsilon):
    if np.random.random() < epsilon:
        return np.argmax(q_degerleri[gecerli_satir_indeks, gecerli_sutun_indeks])
    else:
        return np.random.randint(4)

def sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks,
hareket_indeks):
    yeni_satir_indeks = gecerli_satir_indeks
    yeni_sutun_indeks = gecerli_sutun_indeks
    if hareketler[hareket_indeks] == 'yukari' and gecerli_satir_indeks > 0:
        yeni_satir_indeks -= 1
    elif hareketler[hareket_indeks] == 'sag' and gecerli_sutun_indeks <
ortam_sutun_sayisi - 1:
        yeni_sutun_indeks += 1
    elif hareketler[hareket_indeks] == 'asagi' and gecerli_satir_indeks <
ortam_satir_sayisi - 1:
        yeni_satir_indeks += 1
    elif hareketler[hareket_indeks] == 'sol' and gecerli_sutun_indeks > 0:
```

```
yeni_sutun_indeks -= 1

return yeni_satir_indeks, yeni_sutun_indeks

def en_kisa_mesafe(basla_satir_indeks, basla_sutun_indeks):

    if engel_mi(basla_satir_indeks, basla_sutun_indeks):

        return []

    else:

        gecerli_satir_indeks, gecerli_sutun_indeks = basla_satir_indeks,
basla_sutun_indeks

        en_kisa = []

        en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])

        while not engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):

            hareket_indeks = sonraki_hareket_belirle(gecerli_satir_indeks,
gecerli_sutun_indeks, 1.)

            gecerli_satir_indeks, gecerli_sutun_indeks =
sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks, hareket_indeks)

            en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])

    return en_kisa

epsilon = 0.9

azalma_degeri = 0.9

ogrenme_orani = 0.9

for adim in range(1000):
```

```
satir_indeks, sutun_indeks = baslangic_belirle()

while not engel_mi(satir_indeks, sutun_indeks):

    hareket_indeks = sonraki_hareket_belirle(satir_indeks, sutun_indeks,
epsilon)

    eski_satir_indeks, eski_sutun_indeks = satir_indeks, sutun_indeks

    satir_indeks, sutun_indeks = sonraki_noktaya_git(satir_indeks,
sutun_indeks, hareket_indeks)

    odul = oduller[satir_indeks, sutun_indeks]

    eski_q_degeri = q_degerleri[eski_satir_indeks, eski_sutun_indeks,
hareket_indeks]

    fark = odul + (azalma_degeri * np.max(q_degerleri[satir_indeks,
sutun_indeks])) - eski_q_degeri

    yeni_q_degeri = eski_q_degeri + (ogrenme_orani * fark)

    q_degerleri[eski_satir_indeks, eski_sutun_indeks, hareket_indeks] = yeni_q_degeri

print('Eğitim tamamlandı.')

ogr_sonrasi_satir = input('Robotun harekete başlayacağı satır indeksini giriniz:')

ogr_sonrasi_sutun = input('Robotun harekete başlayacağı stun indeksini giriniz:')

print('Kargo noktasına giden rota:',

en_kisa_mesafe(int(ogr_sonrasi_satir), int(ogr_sonrasi_sutun)))
```