

KOCAELİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

YAZILIM LAB.II

PROJE III

Q-Learning ile Yol Planlaması

MÜRVET NUR ŞEN -ÜMMÜHAN

TEPEBAŞ

190201097 – 180201088

ÖZET:

Bizden Q-Learning algoritması kullanarak engeller matrisinde ajanın oluşturacağımız arayüz ile kullanıcıdan alınan başlangıç ve bitiş koordinatları kullanarak en kısa ve en az maliyetle bir yol bulması ve bunu başka bir arayüz ile veri görselleştirilmesi isteniyor.

Projemiz “Python” dilinde “Numpy”, “Matplotlib” ve “Tkinter” kütüphanelerinden yararlanılarak “Visual Studio Code” derleme ortamında geliştirilmiştir.

GENEL BİLGİLER:

Ajan Q-Learning algoritmasını kullanarak engel sütunlarından kaçması ve beyaz alanlardan geçerek doğru yol alması gerekiyor. Ajanımız belirtilen beyaz kareden başlayıp kırmızı kutulara çarpmadan bitiş kısmına en kısa(maliyetle) yoldan ulaşarak başarılı sayılmaktadır.

Q-learning pekiştirmeli bir öğrenme algoritmasıdır. Ortam hakkında hiçbir şeyin bilinmediği durumlarda, Q-learning algoritması ortamı brute-force şeklinde, her ortam için olası tüm aksiyonları takip ederek, problem çözümü için en karlı yolu bulmaya çalışır. Q-learning algoritmasının girdileri kazanç matrisi

olarak adlandırılan R matrisidir. Bu matrisin satır ve sütunları ortamları temsil etmekte, $R[i][j]$ değeri ise i durumundan j durumuna geçtiğinde elde

edilen anlık kazanç değeridir. Eğer i durumunda j durumuna bir geçiş yoksa $R[i][j]$ değeri -1, geçiş var ancak j durumu hedef durum değilse değeri 0, j hedef durum ise değeri kullanıcı tarafından belirlenen bir kazanç değeridir. Q-learning algoritmasının çıktısı ise öğrenmenin kalitesini gösteren Q matrisidir. Q-learning iteratif bir algoritmadır ve tüm değerleri başlangıçta 0 olan Q matrisi optimum değerlere yakınsadığı da sona erer. Algoritma her iterasyonda rastgele bir durumdan öğrenmeye baslar, A'ya göre durum değiştirir ve Q matrisini günceller. A'ya göre hedef duruma ulaşıldığında iterasyon sona erer. A'ya göre bir durumdan birden fazla duruma geçiş olabilir. Böyle bir durumda, olası geçişler den biri rastgele seçilir. Eğer seçilen durum hedef duruma ulaştırmıyorsa, durum rastgele olacak durum olarak belirlenir. Hedef duruma ulaşılan kadar iterasyon devam eder.

Projemizde görselde belirtildiği gibi (Ek-1) ilk olarak kullanıcıdan başlangıç ve bitiş koordinatları istenmektedir. Devamında bu bilgilere göre engeller kırmızı olmak üzere matris görseli tasarlanan grafiksel arayüzde ekrana verilmektedir ve engeller koordinatlarla birlikte “engel.txt” belgesine yazdırılmaktadır. Q-Learning algoritmasına göre kısa yol hesaplandıktan sonra yeniden ekrana gelen yeni arayüzde mavi renkle gidilen yol çizdirilmektedir. Engellerin maliyeti -100, gidilebilecek yolların maliyeti -1 ve target maliyeti 100 olarak belirlenmiştir. Ajan herhangi bir beyaz kareden başlayarak sağa, sola, aşağı, yukarı ve çapraz hareket edebilmektedir. Herhangi bir başlangıç noktasından hedef noktaya ulaşınca kadar ajanın yaptığı bölüm adım sayısı (episode via step) grafiği yine tasarlanan grafiksel arayüzde çizdirilmektedir. İndirim faktörü $\gamma = 0.9$, kırmızıya çarparsa 100 ödül puanı, bitiş noktasına 100, diğer geçişlere -1 ödül puanı olarak hesaplanacaktır.

KULLANILAN KÜTÜPHANELER:

- import numpy as np
- import random
- from matplotlib import pyplot
- from matplotlib import colors

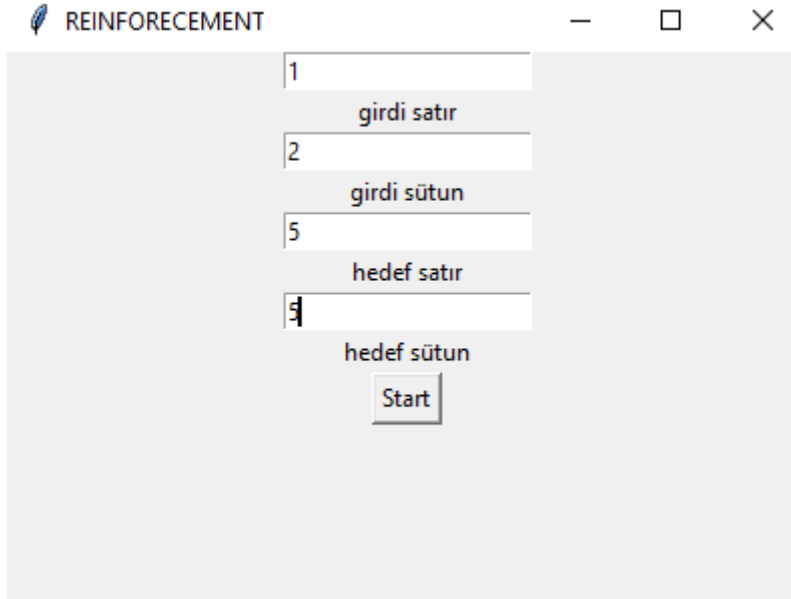
KULLANILAN FONKSİYONLAR:

- def input()
- def is_terminal_state()
- def get_starting_location()
- def get_next_action()
- def get_next_location()
- def get_shortest_path()

KAZANIMLAR:

Python veri bilimi ve veri bilimi kütüphaneleri, grafiksel arayüz tasarım kütüphanesi kullanımı öğrenilip projede efektif kullanılmıştır. Q-Learning ve pekiştirmeli öğrenme (reinforcement learning) algoritmaları detaylıca araştırılıp kazanım edinilmiştir

PROJE GENEL GÖRÜNÜMÜ:



REINFORCEMENT

1
girdi satır

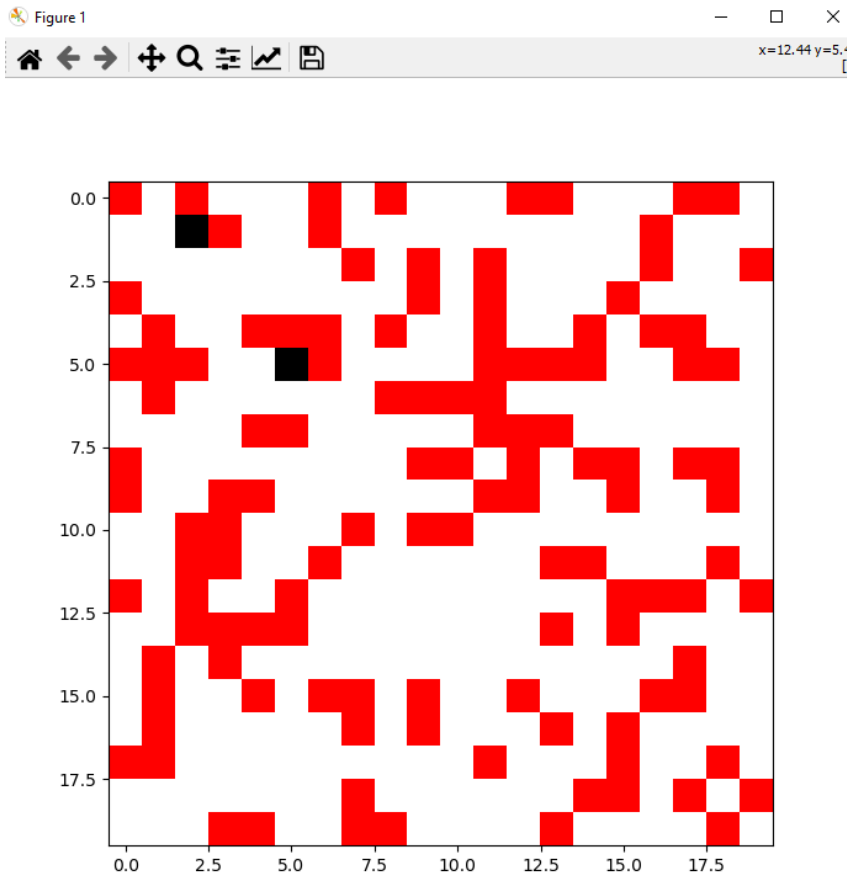
2
girdi sütun

5
hedef satır

5
hedef sütun

Start

Kullanıcıdan başlangıç ve hedef koordinatları alındı. (Ek-1)



Kullanıcıdan alınan koordinatlara göre random bir şekilde engeller ve yollar çizildi.

```

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

```

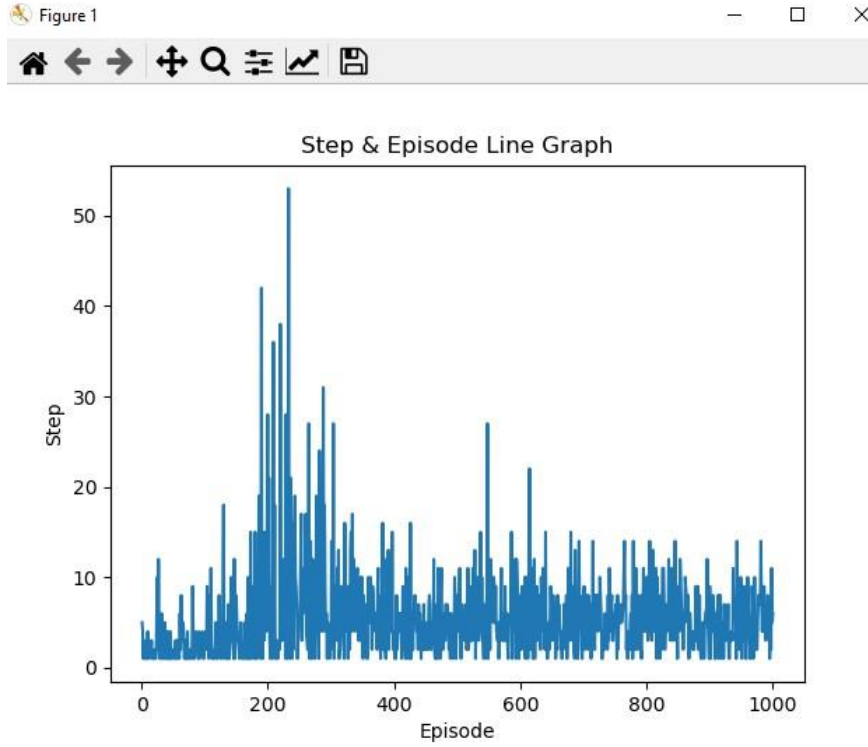
Çizilen matrisin R-tablosu.

```

Training complete!
[[1, 2], [2, 3], [3, 3], [4, 2], [4, 3], [5, 4], [5, 5]]

```

Girilen koordinatlar arasında ajanın bulduğu en kısa yol.



Episode via steps grafiğı çizdirildi.

KAYNAKÇA:

- Kaynakça 1:
- <https://numpy.org>
- Kaynakça 2:
- <https://matplotlib.org>
- Kaynakça 3:
- [Q-Learning: A Complete Example in Python](#)
- Kaynakça 4:
- [Foundations of Q-Learning](#)
- Kaynakça 5:
- <https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c>
- Kaynakça 6:
- Python Tkinter Dersl