Yazılım Laboratuvarı Q-Learning

Özge POYRAZ
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
180202025@kocaeli.edu.tr

Burak Can TEMİZEL
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
180202024@kocaeli.edu.tr

***Özet*—Bu çalışmada, Java programlama dili ile pekiştirmeli öğrenme yöntemi olan Q-Learning algoritması kullanılarak, bir ajanın yol, engel ve hedefin bulunduğu bir ortamda engellere çarpmadan, yolları kullanarak, hedefe ulaşması sağlanmıştır. Q-Learning algoritması ve ortam için çeşitli konfigürasyon ve parametre ayarlarının gerçekleştirilebileceği bir arayüz tasarlanmış, algoritmanın işleyişi gerçek zamanlı çizdirilmiş, elde edilen veriler grafik haline getirilmiş ve sonuç olarak bulunan yolun çizimi gerçekleştirilmiştir.**

Anahtar kelimeler—Pekiştirmeli Öğrenme, Q-Learning, Makine Öğrenmesi Yol Bulma Algoritmaları, Matris, Liste, Veri yapıları, Java, Swing, Grafik programlama .

I. GIRIŞ

Projeyi gerçekleştirmeden önce Q-Learning algoritması hakkında çeşitli araştırmalar gerçekleştirdik. Q-Learning, pekiştirmeli öğrenmenin en çok bilinen algoritmalarından olduğu için elimizde çok fazla kaynak bulunmaktaydı. Teorik bilginin yanında kaynaklardan algoritmanın farklı implementasyonlarını da inceledik. Q-Learning algoritması teşvik-ödül ile çalışmaktadır. R matrisi ve Q matrisi olmak üzere iki yapı üzerine kuruludur. Ajanımız ödüllerin bulunduğu R matrisinden faydalanarak, Q matrisini doldurmaktadır. Belirli iterasyonlar sonrasında Q matrisi bize optimal bir sonuç vermektedir. Raporun ilerleyen kısımlarında algoritmanın işleyişi ve implementasyonumuz hakkında daha detaylı bilgi verilecektir.

II. YÖNTEMLER,YAKLAŞIMLAR VE PROGRAM MIMARISI

Bu kısımda programın farklı özelliklerini oluşturmak için kullandığımız araçlar ve yöntemler üzerinde durularak ayrıntılı olarak bilgi verilecektir. Program mimarisi daha detaylı bir şekilde açıklanacaktır.

A. Q-Learning Algoritması

Algoritmayı gerçeklerken ilk olarak ajanın hareket edeceği ortamın matematiksel olarak oluşturulması gerekiyordu. Ödül matrisinin boyut bilgisi ve üç tam sayı olarak engel, yol ve hedef hücrelerin maliyet değerleri arayüz üzerinden kullanıcıdan alındı. Daha sonrasında tüm matris geçiş değerlerine göre oluşturuldu. Matris üzerindeki engeller yine kullanıcı tarafından girilen bir engel miktarına göre rastgele olarak maliyet değerleriyle beraber matrise eklendi. Bu aşamadan sonra engel ve yol bilgilerinin olduğu matris oluşmuş oldu. Daha sonraki aşamada kullanıcıdan ajanın başlangıç konumu, hedefin konumu, algoritma boyunca kullanılacak parametreler vs. alındı. Ajan başlangıç konumu, daha sonra durumlar arası geçişlerde kullanılmak üzere koordinat bilgisi olarak değişkene aktarıldı. Matriste hedef konuma, hedef maliyeti işlendi. Bu şekilde ödül matrisi tam olarak oluşmuş oldu. Daha sonra Q-Learning algoritmasını gerçekleştireceğimiz Q matrisimizi oluşturmamız gerekiyordu. Bu noktada n matris boyutu olacak şekilde [n*n][4] ve [n*n][8] olmak üzere kullanıcı tercihine göre 4 yönlü hareket ve 8 yönlü hareketi gerçekleyebileceğimiz, Q matrisini bir sıfır matrisi olarak oluşturduk. Ajanın hareketlerini seçerken kullanacağımız bir HashMap tanımladık ve içerisine String olarak gerçekleşebilecek hareket durumlarını ve tam sayı olarak bir indis karşılığı verdik. Daha sonrasında her hücre arası geçişi bir durum olarak kabul edeceğimiz için matris içerisindeki hücreleri 0'dan başlayarak yeni bir durum dizisinde numaralandırdık. Bu şekilde her hücrenin koordinatına karşılık gelen bir indisi olmuş oldu. Q-Learning algoritması iteratif bir algoritma olduğu için bir güncelleme fonksiyonu tanımladık ve algoritmanın her adımından sonra canlı önizlemeye karar verdik. Bu güncelleme işlemi için yeni bir Thread aracılığı ile program döngüsü oluşturduk ve güncelle fonksiyonunu sürekli olarak bu döngü içerisinde çağırarak iteratif bir hale getirdik. Güncelle fonksiyonu ilk olarak daha önce oluşturduğumuz durum dizisini ve ajan konumunu kullanarak hangi hücrede olduğunu mevcut_durum değişkenine aktarıyor. Algoritmanın işleyebilmesi için ajanın bir durum geçişi gerçekleştirmesi gerekiyor. Bu yüzden Ajanın nasıl hareket edeceğini belirleyecek bir hareket_sec metodu tanımladık.

Hareket belirlenirken ilgili metot mevcut durumu parametre olarak alıyor. Hareket metodu temel olarak iki farklı şekilde hareket gerçekleştiriyor. Bunlardan ilki rastgele olarak gerçekleştirilen hareket ikincisi ise Q Matrisindeki durumların maksimum değerlerinden faydalanılarak seçilen hareket. Bu iki hareketin kullanılmasının sebebi ajanın keşif sürecini arttırmak. Bunu kontrol edecek yöntem olarak Epsilon Greedy yöntemini tercih ettik. Program boyunca kullanıcıdan alınan ya da çeşitli faktörlerle değişebilen ve 0-1 arasında değer alan Epsilon değerimiz algoritmanın hangi hareketi seçeceğine karar veriyor. Bu yüzden hareket_sec metodu ilk olarak düzgün dağılımlı rastgele bir sayı seçiyor. Eğer bu sayı belirlenen Epsilon değerinden küçükse rastgele hareketi eğer Epsilon değerinden büyükse Q-Matrisi üzerinden ilgili durumun maksimum Q değerine sahip aksiyonunu seçerek hareketi gerçekleştiriyor. Rastgele hareket için yapılabilecek tüm hareketlerin eklenebileceği bir liste oluşturuluyor. Daha sonrasında ajanın matris sınırları içerisinde kalması garanti edilerek Q matrisindeki durum sayısı kadar hareket eğer gerçekleşebiliyorlar ise bu listeye HahsMap'ten faydalanılarak ekleniyor. Daha sonrasında bu listeden rastgele bir hareket seçilerek harekete karşılık gelen sayısal değer fonksiyondan döndürülüyor. Eğer Q-Matrisi üzerinden hareket gerçekleştirilecekse yine yapılabilecek hareketler için bir liste oluşturuluyor fakat bu noktada Q-Matrisi üzerindeki durumların aksiyon değerlerinden faydalanacağımız için listemiz sayısal bir ifade oluyor. Daha sonra yine matris sınırlarında kalmayı garantileyerek ajanın bulunduğu durumdan hareket edebileceği yönlerin Q matrisindeki değerleri listeye sırasıyla ekleniyor. Eğer bir hareket gerçekleştirilemiyorsa listedeki indis yapısının

korunması için Q Matrisindeki mevcut durumundaki Q değerlerinin en küçüğü alınarak bundan büyük bir sabit çıkartılarak değerin seçilmeyeceği garanti edilerek bu listeye ekleniyor. Daha sonrasında aynı Q değerlerine sahip hareketler olabileceği için bir iterator aracılığı ile bu liste dönülerek maksimum Q'ya sahip hareketler ayrı bir listeye alınıyor. Daha sonrasında bu listeden rastgele bir hareket seçiliyor ve bu harekete karşılık gelen indis fonksiyondan geri döndürülüyor. Bu noktadan sonra hareket_sec, epsilon değerine göre iki farklı şekilde hareketi gerçekleştirebilecek hale geliyor.

Algoritmanın temel fonksiyonu olan güncelle fonksiyonuna geri dönersek, hareket_sec metodu ile bir hareket gerçekleştirilerek hareketin indisi elde ediliyor. Daha sonra bu indis kontrol edilerek hareketin hangi yöne olduğu bulunuyor ve daha sonrasında hareketi gerçekleştirecek şekilde pozisyonlar güncelleniyor. Daha sonra durum matrisinden ajanın bulunduğu yeni hücrenin indisi elde ediliyor. Atılan adım bilgisi, ve ödül matrisinden bulunulan hücrenin ödül bilgisi gibi veriler grafikler için işleniyor. Yol çizimi için listeye ekleniyor. En başta her Episode'u sonlandıran engel, hedef gibi yapılar sonlandiricilar isimli bir listeye eklenmişti. Bu noktada yeni_durumumuzun bu listede olup olmadığını kontrol ediyoruz. Eğer bu listede değilse Q-Learning formülümüzün aracılığı ile hareketi ilk gerçekleştirdiğimiz durumun ilgili hareketinin Q değerine alfa ile çarpılarak ödül matrisindeki yeni pozisyonumuzun ödül değerine gama ile çarpılmış yeni durumumuzdaki maksimum Q değerini ekliyoruz ve son olarak tüm değerden Q matrisimizdeki ilk durumumuzun ilgili hareketinin Q değerini çıkartıyoruz. Burada alfa öğrenme katsayısı gama ise indirim faktörüdür. Bu şekilde Q Matrisi üzerinde formül aracılığı ile yakınsama işlemini gerçekleştiriyoruz. Eğer yeni durum sonlandiricilar listesinde bulunuyor ise bu durumda yapmamız gereken bazı ek şeyler daha bulunuyor. Öncelikle bir engele ya da hedefe varmak Episode sayacını güncelleyerek arttırıyor. Daha sonrasında Bir engel ve hedeften sonra olası bir sonraki hareket olmadığı için maksimum Q olmadan ödül tablosu üzerinden optimizasyonu gerçekleştiriyoruz. Her Episode sonunda algoritmaya daha önceden verilen sonlandirma adetine göre yolun kaç iterasyon boyunca değişip değişmediğini kontrol ederek gerekirse algoritmayı sonlandırarak, grafiklerin çizilmesini tetikliyoruz. Burada önemli bir nokta algoritma sonlanmadan bir adım önce Epsilon değeri 0'a eşitlenerek son gezilen yoldaki rastgelelik ortadan kaldırılarak sadece matristen faydalanılarak yolun çizilmesi sağlanıyor. Daha sonrasında ajanın konumu başlangıç konumuna eşitleniyor. Güncelleme fonksiyonunun en sonunda Epsilon değişkeni de güncelleniyor. Bu şekilde ikinci bir güncelleme çağrımında algoritma iteratif bir şekilde ilerlemiş oluyor. Ve zamanla yakınsayarak yolu ortaya çıkartıyor.

B. Kullanıcı Arayüzü ve Grafik Programlama

Uygulama çeşitli arayüz elemanlarından oluşmaktadır. Uygulama ilk çalıştırıldığında matris önizlemesini oluşturmak için kullanıcıdan çeşitli girdilerin alındığı bir pencere ile başlamaktadır. Bu ve daha sonra bahsedeceğimiz arayüz elemanlarını tasarlarken Netbeans UI Editoru kullanılmıştır. Kullanıcı girdileri verdikten sonra aynı pencere içerisinde yeni bir panel ile matris üzerinde ajan başlangıcı ve hedefinin seçildiği, diğer algoritma ayarlarının yapıldığı yeni bir pencere kullanıcıyı karşılamaktadır. Bu pencere içerisinde bir adet daha panel barındırmaktadır. Bu panel ile önceki aşamadaki bilgilerle oluşturulan matrisin görsel bir ifadesi bulunmaktadır. Her pencerenin kendine ait bir sınıfı vardır. Bu sınıfta içerideki panelin paint metodu override edilerek ve MouseListener Interface'i kullanılarak kullanıcının panel üzerinden Mouse ile konumları seçmesi sağlanmıştır. Çeşitli textfield ve checkboxlarla ayar bilgileri kullanıcıdan alınmıştır. Daha sonra tüm konfigürasyonlar tamamlanınca asıl algoritmanın yürütüldüğü ana pencereye geçilir. Burada ayar paneli üzerinde döngü hızını kontrol etmek için bir adet slider ve mevcut episode bilgisini görüntülemek için bir adet label bulunmaktadır. Panelin içinde ise yine algoritmanın çalıştığı süre boyunca canlı önizlemesinin görüntülenebileceği bir panel daha bulunmaktadır. Bu panel Q-Learning algoritmasının güncelleme fonksiyonun her iterasyonundan sonra yeniden çizdirilir. İçerisindeki resim dosyalarını Q-Learning algoritmasındaki ajan konumu, yol bilgisi ve matris bilgisi gibi ögeleri kullanarak ekrana çizdirir. Görselliğin iyileştirilmesi için darcula kütüphanesi ile programa karanlık tema özelliği uygulanmıştır.

C. Grafiklerin ve Yolun Çizdirilmesi

Q-Learning algoritması optimum değerlerine yakınsayıncaya kadar yol bilgisi dinamik olarak oluşturulmaktadır. Her episode başında yol için bir liste oluşturulmakta ve yolDugum adında tanımlanan koordinat bilgisi tutan düğüm sınıfı aracılığı ile her hareketi bir düğüm olarak bu listeye eklenmektedir. Epsilon değerine bağlı olarak ajan rastgele hareketler sergilediği için anlık yol değişiklik göstermektedir. Bizden algoritmanın bulduğu en kısa yol istendiği için nihai yolu çizdirirken Epsilon değerimizin 0 olması yani sadece Q matrisindeki her durum için en iyi aksiyonun takip edildiği yol gerekmektedir. Bu yüzden daha önce oluşturduğumuz sonlanma yapısında kontrol ederek algoritma sonlanmadan bir önceki adımda Epsilon değerini manuel olarak 0'a eşitleyerek ajanın son Episode'u bu şekilde tamamlamasını sağlayarak yol listesini oluşturmuş oluruz. Daha sonrasında bu yol listesi arayüzde çizdirilmektedir.

Benzer şekilde algoritma sonlandığında episode via steps ve episode via cost grafiklerini çizdirebilmek için her episode boyunca ajanın attığı adımlar sayılmakta ve pozisyonuna karşılık gelen ödül matrisi değerleri bir değişkende toplanmaktadır. Daha sonra tüm episodelar için liste haline getirilmektedir. Algoritma sonlandığında Grafik sınıfı kullanılarak, bu verilerin liste haline getirilmiş veri yapılarından, grafikleri çizdirmek için kullandığımız JFreeChart kütüphanesinin dataset yapısına dönüşüm gerçekleştirilip, grafikler çizgi grafiği olacak şekilde çizdirilmektedir

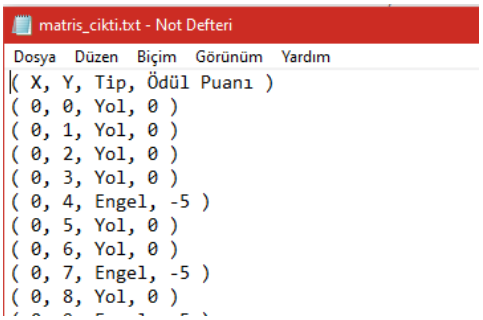
D. Kullanılan Temel Veri Yapıları

Programın geneli aksine algoritmayı gerçekleştirirken sınıfsal bir yapı yerine matrislere dayanan bir yaklaşım tercih edilmiştir. İşlemler matrisler üzerinde gerçekleştirilmiştir. Hareketlerin belirlenmesinde HashMap kullanılmıştır. Ağırlıklı olarak diziler kullanılmakta olup, dinamik işlemler ArrayListler ile gerçeklenmiştir.

E. Dosya İşlemleri

Projenin çıktı isteri için, yerleştirme penceresinden sonra kullanıcı hedefi belirleyince, FileWriter sınıfı ile projenin dosya yolunda yeni bir txt dosyası oluşturulmaktadır ve

içerisine her matris hücresinin koordinat bilgileri, engel mi yol mu yoksa hedef mi olduğuna dair tip bilgisi ve ödül değerleri yazdırılmıştır.



Matris Çıktısı

III. KAZANIMLAR VE EKSIKLIKLER

Projeyi gerçekleştirirken, makine öğrenmesi, pekiştirmeli öğrenme gibi çok çeşitli kavramlar hakkında yeni bilgiler edinme ve araştırma yapma fırsatı yakaladık. Programı esnek bir şekilde gerçeklediğimiz için çeşitli parameter ayarlarıyla farklı konfigürasyonları deneyerek test etme fırsatı bulduk. Projeyi gerçekleştirirken yaşadığımız tek eksiklik optimum sonucu geçiş ödülü 0 iken bulabilmemizdi. Geçiş ödülünde farklı değerlerde çok uzun çalışma süreleri ya da ajanın iki durum arasında takılması gibi problemler yaşadık ama bunları farklı parametreler ve konfigürasyonları uygulayarak aştık.

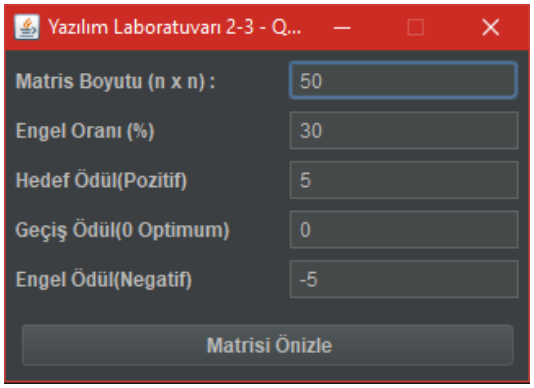
IV. GELIŞTIRME ORTAMI VE KULLANILAN DIL

Projeyi Java programlama dilinde Windows işletim sistemi üzerinde gerçekleştirirken, geliştirme ortamı olarak JDK 15 konfigürasyonlu Netbeans idelerini kullandık. Proje Java Maven projesi olup ilgili iki adet bağımlılıkları pom.xml dosyasında girilidir. Bu bağımlılıklar darcula ve JFreeChart kütüphaneleridir.

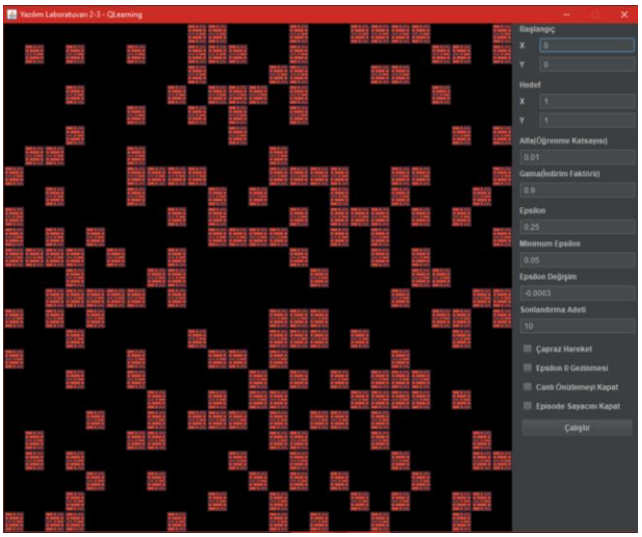
V. PROGRAMIN GENEL YAPISI VE TASARIMI

Program temel olarak başlatma ekranı, ayar ekranı, canlı önizleme ekranı ve grafik ekranları olmak üzere 5 adet pencere içeriyor. Her pencere farklı bir tasarıma sahip ve basit bir pencere yöneticisi aracılığıyla pencereler arası geçiş gerçekleştiriliyor.

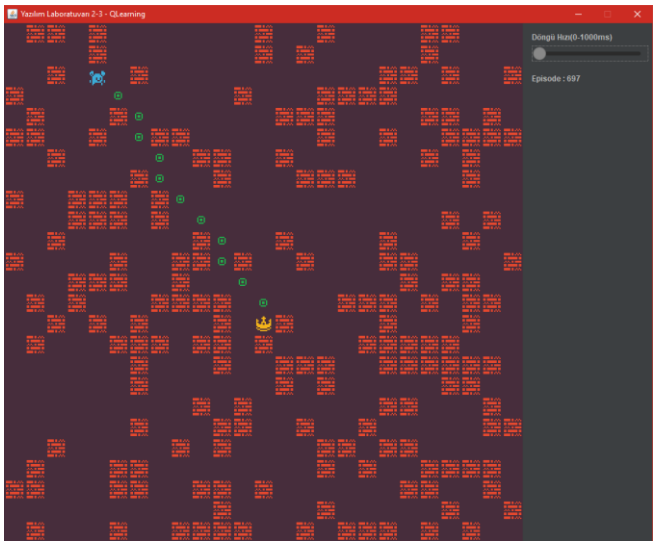
.



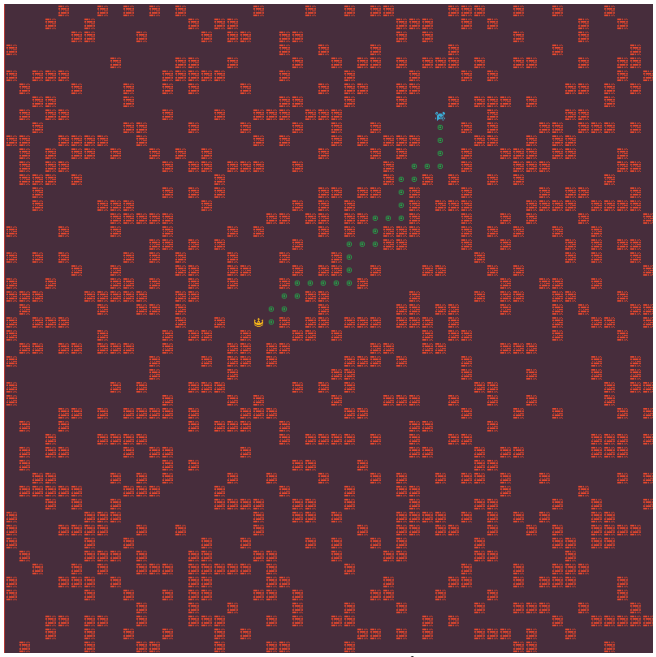
Başlatma Ekranı



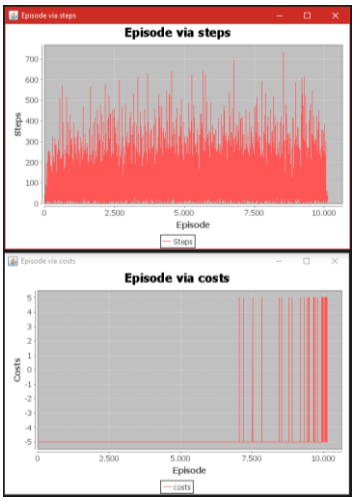
Konfigürasyon Ekranı



25x25 Matriste 8 Yönlü Hareket ile Elde Edilen Yol



50x50 Matriste 4 Yönlü Hareket İle Elde Edilen Yol



Grafik Çizimleri

VI. DENEYSEL SONUÇLAR

Algoritma içerisindeki parametrelerin değerlerinin tümüne ulaşımını arayüzden sağlayabildiğimiz için bu değerleri farklı senaryolarda farklı şekillerde uygulayarak çalışma koşullarını sürekli gözlemleme fırsatı elde ettik. Farklı koşullarda yol bilgisini, grafikleri vs. inceledik. Bu şekilde farklı senaryolar için optimum değerleri gözlemleyerek, keşfettik.

VII. SONUÇ

Q-Learning algoritmasını uygulayarak ajanın hedefe ulaşmasını, yol bilgisinin elde edilmesini, episode bilgilerinin grafikleştirilmesini, ödül matrisinin çıktısının oluşturulmasını sağladık. Bu şekilde tüm proje isterlerini gerçekleştirdik.

VIII. KAYNAKLAR

(1) https://medium.com/deep-learning-turkiye/python-i%CC%87le-q-learning-ef6413aa896e
(Erişim Tarihi: 06.05.2020)
(2) https://medium.com/@sddkal/python-ve-makine-%C3%B6%C4%9Frenmesi-q-learning-temelleri-181d29326782
(Erişim Tarihi: 06.06.2020)
(3) https://medium.com/analytics-vidhya/introduction-to-reinforcement-learning-q-learning-by-maze-solving-example-c34039019317
(Erişim Tarihi: 06.06.2020)

EK 1 -AKIŞ DİYAGRAMI

Q-Learning algoritmasının güncelleme fonksiyonun basitleştirilmiş genel akış şeması aşağıdaki gibidir.