Python Sayı Yok Etme Oyunu

PYTHON VİZE ÖDEVİ

***Abstract*—Bu raporda python üzerinde kodlanmış ve içerisinde matrisleri,rekürsif fonksiyonları ve dosya işlemlerini barındıran Sayı Yok Etme Oyunu anlatılmaktadır.**

Keywords—Giriş,Uygulama İçindeki Metodlar,Uygulama Detayları

I. GİRİŞ

Bu raporda Elemanları rastgele seçilmiş ve boyutları bizim tarafımızdan belirlenmiş bir matris üzerinde oynanan Sayı Yok Etme Oyununun yapım aşamasından bahsedilecektir.Öncelikle matrisin her bir elemanı yine aralığını bizim belirlediğimiz şekilde 0-9 arasındaki sayılardan rastgele seçilecektir.Oyuncu matrisin her bir elemanını seçme özgürlüğüne sahiptir.Oyuncu seçimini yaptıktan sonra seçtiği elemanla aynı değere sahip tüm komşularını(sağ,sol,üst,alt) matristen yok eder.Bir elemanın yok edilmesi aynı değerdeki tüm komşu elemanların yok edilmesiyle sonuçlanır. Oyunun amacı tek seferde c*fibonacci(n) puan denklemine göre en yüksek puanı almaktır.Bu puan denkleminde c seçilen değeri n ise yok edilen eleman sayısını temsil etmektedir.Python kodumuz oyuncuya oyunun ilk girdi olan matrisi aynı zamanda da çıktı olan matrisi txt uzantılı dosyalar sayesinde takip etmesini sağlarken oyun aşamasında da kullanıcıya çeşitli kod yönergeleriyle yardımcı olmaktadır..

II. UYGULAMA İÇINDEKİ METODLAR

```
def sayi_yok_etme_oyunu(satirlar, sutunlar):
```

Metotların en başında üstteki ana metodumuz olan sayi_yok_etme_oyunu() adlı metot çalışmaktadır.Metot parametreleri oynayacağımız oyun tahtası olan matrisin kaç satır ve sütundan oluşacağını değer olarak almaktadır.

```
def fibonacci(n):
```

Fibonacci metodumuz sayi_yok_etme_oyunu() adlı metodun içinde puan hesabında kullanılacağı için bu metodun içinde implement edilmiş olarak gelmektedir.Daha sonra oyunun oynandığı aşamalarda anlık olarak puan hesabı bu metod üzerinden yapılmaktadır.Parametre olarak hesaplanacak fibonacci değeri olan n'i almaktadır bu n yok edilen komşu sayısını temsil etmektedir.

```
def komsu_kontrol(satir_no, sutun_no, deger, sayac):
```

Uygulamamızın en önemli metodu olan komsu_kontrol() metodumuz da fibonacci gibi oyun içinde tekrar tekrar kullanılacağı için sayi_yok_etme_oyunu() metodu içinde implement edilmiştir.parametre olarak satır ve sütun numaralarını,kontrol edeceğimiz değeri ve puan hesaplamasında hesaba katmak için yok edilen eleman sayılarını tutacak olan sayaç değerini almaktadır.

III. UYGULAMA DETAYLARI

```
def fibonacci(n):
      if n <= 1:
            return n
      else:
            return fibonacci(n-1) +
fibonacci(n-2)
```

Fibonacci fonksiyonumuz şekildeki gibi rekürsif olarak çalışmaktadır . Bir adet base case e sahip olan metodumuz değer olarak Fibonacci toplamımızı ilerleyen işlemlerde kullanılmak üzere döndürmektedir.

```
def komsu_kontrol(satir_no, sutun_no,
deger, sayac):
        satirlar, sutunlar = nptahta.shape
        nptahta[satir_no, sutun_no] = -1
        sayac += 1
       #komsu kontrol
        for i, j in [(satir_no+1,
sutun_no), (satir_no-1, sutun_no),
(satir_no, sutun_no+1), (satir_no,
sutun_no-1)]:
            if i < 0 or i >= satirlar or j
< 0 or j >= sutunlar or nptahta[i,j] !=
deger:
                continue

            nptahta[i,j] = -1
            sayac = komsu_kontrol(i, j,
deger, sayac)

        # sütunları kontrol etme
    for j in range(sutunlar):
        if np.all(nptahta[:, j] == -1):
                # sütun elemanlarını
kaydırma
        nptahta[:, j:sutunlar-1] =
nptahta[:, j+1:]
                # en sağdaki sütunun
elemanlarını -1 yapma
        nptahta[:, sutunlar-1] = -1

        return sayac
```

Komşu kontrol fonksiyonumuz aldığı parametrelerin yardımıyla klavyeden girdiğimiz değerin komşularını

control edip yok etme işlevini gerçekleştirmektedir.Yok ederken sayıların alamayacağı bir değer olan -1 yapılması tercih edilmiştir.
Son olarak sütunları control eder ve yine uygulamamızın en kilit işlemlerinden biri olan sütunların boş kalması yani yok edilmesi durumunda sağdaki elemanların sütuna kaydırılması işlemini başarıyla gerçekleştirir.
Dönüş değeri olarakta puanlama hesabında kullanılmak üzere sayac değerini döndürür.

```
tahta = []
puan = 0
    # tahtayı oluşturma

for i in range(satirlar):
       row = []
       for j in range(sutunlar):
            # rastgele sayılar ekleme
           row.append(random.randint(0,9))
       tahta.append(row)

nptahta = np.array(tahta)

 # tahtayı girdi.txt dosyasına yazma

with open("girdi.txt", "w") as f:
      for row in nptahta:
            f.write(' '.join(map(str,
row)) + '\n')

print("Matris:")
print(nptahta)
print("Puan: ",puan)
```

Ardından en baştaki metodumuza gelen satır ve sütun değerlerine göre oyun tahtamızın oluşturula işlevi gerçekleşmektedir.Tahtamız oluşturulduktan sonra üzerinde çeşitli matris işlemlerini daha Rahat gerçekleştirmek üzere Numpy kütüphanesi kullanarak nptahta adlı matrise dönüştürülmektedir.
Devamında dosya işlemleri ile oyun tahtamızın ilk hali olan matris girdi.txt adlı dosya oluşturulup içine yazılmaktadır.
Bu işlemlerin tümü tamamlandıktan sonra oyunumuz matrisin ve puanımızın print işlevi ile ekrana yazılmasıyla başlamaktadır.
Programın O sıradaki çıktısı aşağıdaki gibidir:

```
Matris:
[[4 1 7 6 0 2 3 9 7 8]
 [8 9 6 5 2 8 0 0 8 2]
 [5 8 5 9 3 3 6 9 3 8]
 [5 8 8 7 1 7 3 0 0 4]
 [7 5 7 3 6 1 0 3 7 8]
 [2 7 4 0 4 2 8 3 7 5]]
Puan:  0
Hangi satırı seçmek istersiniz (1-6):
```

```
while(True):
        # kullanıcının satır ve sütun
seçimi
        satir_no = int(input("Hangi satırı
seçmek istersiniz (1-{}):
".format(satirlar)))-1
        sutun_no = int(input("Hangi sütunu
seçmek istersiniz (1-{}):
".format(sutunlar)))-1

        #seçilen değer satır,sğtun boyutu
dışındaysa döngü tekrarlıyor
        if satir_no >= satirlar or
sutun_no >= sutunlar:
            print("Lütfen geçerli bir
satır ve sütun numarası giriniz.")
            continue

        deger = nptahta[satir_no,
sutun_no]

        #seçilen değer önceden yok
edilmişse döngü tekrarlıyor
        if deger==-1:
            print("Lütfen geçerli bir
değer girin.")
            continue

        sayac = 0
        sayac = komsu_kontrol(satir_no,
sutun_no, deger, sayac)
        if(sayac<=1):
            print("Seçtiğiniz elemanın
etrafında bir komşusu yok.")
            print("Lütfen tekrar
deneyin.")
            continue
        adim_puan=deger*fibonacci(sayac)
        puan+=adim_puan

        print("Puan: ",puan)
```

Ardından oyunumuzun adımlar halinde bitene biz çıkış yapana dek çalışmasını sağlayacak while döngümüz başlamaktadır.

İlk olarak kullanıcıdan matris üzerinde bir satır ve sütun seçmesini ister ve gerekli inputları alır.Ardından koyduğumuz base caseler ile hatalı giriş olup olmadığını saptar ve eğer boyut dışında bir giriş ,seçilen elemanın komşusu olmaması veya daha önceden yok edilmiş bir elemanın girilmesi gibi hatalardan biriyle karşılaşırsa tekrar input alınması için continue ile döngüyü devam ettirir.

Eğer Kullanıcı inputu sorunsuz bir şekilde girdiyse verilen input komşu kontrol fonksiyonumuza gönderilerek gerekli komşular yok edilir ve ardından yok edilen eleman sayısına göre hesaplanan adım puanımız olan deger*fibonacci(n) metodumuzdaki implement edilmiş fonksiyon sayesinde hesaplanır ve ekrana yazılır.

```
# silinen elemanların yerine üstündeki
elemanların kaydırılması
        for i in range(satirlar-1, 0, -1):
            for j in range(sutunlar):
                if nptahta[i, j] == -1:
                    for k in range(i-1, -
1, -1):
                        if nptahta[k, j]
!= -1:
                            nptahta[i, j],
nptahta[k, j] = nptahta[k, j], nptahta[i,
j]
                            break

        # silinen elemanlar yerine boşluk
karakteri yazdırma
        nptahta_str = np.where(nptahta ==
-1, ' ', nptahta.astype(str))
        for row in nptahta_str:
            row_str = ' '.join(row)
            print(row_str)

        # oyunun son halinin cikti.txt
adlı dosyaya yazdırılması
        with open("cikti.txt", "w") as f:
            for row in nptahta_str:
                f.write(' '.join(map(str,
row)) + '\n')
```

Uygulamamızın son kısmında ise oyunun oynanması açısından önemli işlevlerden biri olan silinen elemanlar yerine diğer elemanların kaydırılması işlevi gerçekleştirilmektedir.

Karakterler silindikten sonra oyunun sonraki adımının ekrana yazıldığı kısım gelmektedir.Bu kısımda -1 olan yani silinen elemanlar boşlukla değiştirilip ekrana yeni adım olarak yazdırılmaktadır.

```
#parametre olarak matrisin yani oyun
tahtamızın boyutunu alıyor
#6 satır / 10 sutun verdim her değere göre
çalışıyor
sayi_yok_etme_oyunu(6,10)
```

Sayı Yok Etme oyununun konsol üzerinde çalışmış versiyonu aşağıdaki gibidir;

```
Matris:
[[1 4 2 7 2 8 6 9 4 3]
 [3 7 8 6 5 6 5 5 8 6]
 [0 1 7 3 0 4 2 9 8 5]
 [6 2 9 0 8 0 3 8 7 0]
 [2 2 0 7 1 5 8 8 8 8]
 [2 6 8 3 9 9 4 8 8 9]]
Puan:  0
Hangi satırı seçmek istersiniz (1-6): 5
Hangi sütunu seçmek istersiniz (1-10): 10
Puan:  104
1 4 2 7 2 8
3 7 8 6 5 6 6     3
0 1 7 3 0 4 5   4 6
6 2 9 0 8 0 2 9 8 5
2 2 0 7 1 5 3 5 8 0
2 6 8 3 9 9 4 9 7 9
Hangi satırı seçmek istersiniz (1-6): 6
Hangi sütunu seçmek istersiniz (1-10): 5
Puan:  113
```

```
1 4 2 7
3 7 8 6 2 8 6     3
0 1 7 3 5 6 5   4 6
6 2 9 0 0 4 2 9 8 5
2 2 0 7 8 0 3 5 8 0
2 6 8 3 1 5 4 9 7 9
Hangi satırı seçmek istersiniz (1-6): 6
Hangi sütunu seçmek istersiniz (1-10): 1
Puan:  119
    2 7
    8 6 2 8 6     3
1 4 7 3 5 6 5   4 6
3 7 9 0 0 4 2 9 8 5
0 1 0 7 8 0 3 5 8 0
6 6 8 3 1 5 4 9 7 9
Hangi satırı seçmek istersiniz (1-6):
```

Bu şekilde biz uygulamadan çıkana kadar devam etmektedir ve çıktığımızda tablonun son halini cikti.txt olarak yazdırmaktadır.

KUBİLAY BİRER

211307086

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

KOCAELİ ÜNİVERSİTESİ

PYTHON PROGRAMLAMA