RAPOR BAŞLIĞI

HAZIRLAYAN

ADI SOYADI : RABİA ŞENLİK–MUSTAFA YILDIRIM
ÖĞRENCİ NUMARASI : 190303027-190303033
TESLİM TARİHİ : 18/12/2022

DERS ADI : ALGORİTMA TASARIMI VE ANALİZİ
DERS YÜRÜTÜCÜSÜ : IŞIL KARABEY AKSAKALLI

Rapor/Ödev Başlığı :
Hazırlayanın Adı Soyadı :

a) Kodumuzu en kötü durum için analiz etme:

```python
def search_all(matrix, value):
    # Matrix satır ve sütun sayısı alınır
    rows = len(matrix)
    cols = len(matrix[0])

    # Satır ve sütun indeksleri 0'dan başlatılır
    row = 0
    col = cols - 1

    # Toplam karşılaştırma sayısı sıfır olarak başlatılır
    count = 0

    # Sonuç listesi oluşturulur
    result = []

    # Matrisin sol üst köşesinden başlanır ve arama yapılır
    while row < rows and col >= 0:
        # Karşılaştırma sayısı artırılır
        count += 1

        # Eğer aranan değer, mevcut indeksteki değerden küçükse, sütun indeksi azaltılır
        if matrix[row][col] > value:
            col -= 1
        # Eğer aranan değer, mevcut indeksteki değerden büyükse, satır indeksi artırılır
        elif matrix[row][col] < value:
            row += 1
        # Eğer aranan değer, mevcut indeksteki değere eşitse, sonuç listesine pozisyon ve karşılaştırma sayısı eklenir
        # ve satır ve sütun indeksleri azaltılarak arama devam edilir
        else:
            result.append((row, col, count))
            col -= 1
            row += 1

    # Sonuç listesi döndürülür
    return result
```

```python
matrix = [[10 ,30 ,45 ,50 ,58 ,71 ,79 ,86 ,89 ,93 ,99 ,107,112],
          [13 ,34 ,48 ,66 ,69 ,78 ,85 ,88 ,94 ,96 ,100,115,118],
          [15 ,44 ,51 ,67 ,72 ,83 ,87 ,91 ,97 ,103,105,117,121],
          [17 ,46 ,53 ,70 ,74 ,84 ,90 ,93 ,98 ,104,109,120,189],
          [23 ,55 ,64 ,77 ,81 ,93 ,101,111,116,122,130,147,201],
          [37 ,65 ,73 ,80 ,82 ,106,110,119,125,129,152,169,205],
          [39 ,68 ,76 ,102,108,113,114,124,131,137,161,178,210],
          [40 ,93 ,123,126,140,144,148,150,157,160,162,202,267],
          [43 ,128,133,135,149,164,166,171,177,183,192,204,301],
          [400,500,600,700,800,900,901,902,903,904,905,906,909]]
```

search_all() fonksiyonunun en kötü durum performansı, aranan değerin matris içerisinde en son sütunda bulunması durumunda olacaktır. Bu durumda, fonksiyon her bir sütun için en az bir kez karşılaştırma işlemi yaparak aramanın sonucunu bulacaktır. Bu nedenle, fonksiyonun en kötü durum performansı O(nm) olacaktır.

Bu performans, matrisin satır sayısının (n) ve sütun sayısının (m) çarpımına eşittir. Örneğin, matrisin satır sayısı 10 ve sütun sayısı 13 ise, en kötü durum performansı O(10 * 13) = O(130) olacaktır. Bu, fonksiyonun en fazla 130 adımda arama işlemini tamamlayabileceği anlamına gelir.

Bu performans, matrisin satır ve sütun sayıları arttıkça daha yavaş hale gelecektir. Örneğin, matrisin satır sayısı 100 ve sütun sayısı 1000 ise, en kötü durum performansı O(100 * 1000) = O(100000) olacaktır. Bu, fonksiyonun en fazla 100000 adımda arama işlemini tamamlayabileceği anlamına gelir.

b) Aşağıdaki anahtar değerlerimiz için algoritmamızdaki karşılaştırma sayısı:

Aşağıdaki kod verilen key değerleri için yukarıda verilen arama algoritmasını ve matrisi kullanarak arama yapar ve arama yapılan her key değeri için yapılan karşılaştırma sayısını ekrana yazdırır.

1. Matrisin satır ve sütun sayısı alınır.
2. Satır ve sütun indeksleri, matrisin sol üst köşesinde (0, cols-1) olarak başlatılır.
3. Karşılaştırma sayısı sıfır olarak başlatılır.
4. Matrisin sol üst köşesinden başlanır ve arama yapılır. Bu işlem, satır sayısının ve sütun sayısının sınırları aşılana kadar devam eder.
5. Eğer aranan değer, mevcut indeksteki değerden küçükse, sütun indeksi azaltılır. Eğer aranan değer, mevcut indeksteki değerden büyükse, satır indeksi artırılır. Eğer aranan değer, mevcut indeksteki değere eşitse, arama başarılı olur ve pozisyon ve karşılaştırma sayısı döndürülür.
6. Eğer aranan değer bulunamazsa, arama başarısız olur ve None döndürülür.

Bu fonksiyon, verilen matris içinde arama yaparken, satır ve sütun indekslerini azaltarak veya artırarak matrisin diğer elemanlarını da kontrol etmektedir. Bu nedenle, fonksiyonun performansı linear search algoritmasına benzerdir ve dizinin büyüklüğüne göre performansı düşük olabilir. Binary search algoritması gibi daha etkili bir arama algoritması kullanılması daha uygun olabilir

```python
keys = [30, 107, 51, 93, 162, 123, 111, 134, 300]

for key in keys:
    result = search_all(matrix, key)
    print(f"Key {key} için:")
    print(f"Pozisyonlar: {result}")
    print(f"Toplam yapılan karşılaştırma: {sum([t[2] for t in result])}")
    print()
```

Kodumuzu çalıştırdığımızda terminalde şöyle bir sonuç alırız:

```
Key 30 için:
Pozisyonlar: [(0, 1, 12)]
Toplam yapılan karşılaştırma: 12

Key 107 için:
Pozisyonlar: [(0, 11, 2)]
Toplam yapılan karşılaştırma: 2

Key 51 için:
Pozisyonlar: [(2, 2, 13)]
Toplam yapılan karşılaştırma: 13

Key 93 için:
Pozisyonlar: [(0, 9, 4), (3, 7, 8), (4, 5, 10), (7, 1, 16)]
Toplam yapılan karşılaştırma: 38

Key 162 için:
Pozisyonlar: [(7, 10, 10)]
Toplam yapılan karşılaştırma: 10

Key 123 için:
Pozisyonlar: [(7, 2, 18)]
Toplam yapılan karşılaştırma: 18

Key 111 için:
Pozisyonlar: [(4, 7, 10)]
Toplam yapılan karşılaştırma: 10

Key 134 için:
Pozisyonlar: []
Toplam yapılan karşılaştırma: 0

Key 300 için:
Pozisyonlar: []
Toplam yapılan karşılaştırma: 0
```

c) Yukarıda verdiğimiz matriste 1-130 arasında olan 50 tane değeri seçip her biri için programı işlettik.50 kez işletilme sonrası ortalama ve standart sapmasını bulma:

```
# 1 ve 130 arasında olan elemanların listesini oluşturun
elements = [elem for row in matrix for elem in row if 1 <= elem <= 130]

# 50 tane rastgele eleman seçin
selected_elements = random.sample(elements, 50)

print("\nrastgele oluşturulan dizi:\n", selected_elements)


# Karşılaştırma sayılarını tutan liste oluşturun
comparison_counts = []

# Her bir eleman için arama işlemini yapın
for elem in selected_elements:
    result = search_all(matrix, elem)
    # Eğer sonuç listesi boş değilse, karşılaştırma sayısını listeye ekleyin
    if result:
        comparison_counts.append(result[0][2])

# Karşılaştırma sayılarının ortalamasını ve standart sapmasını hesaplayın
mean = statistics.mean(comparison_counts)
stdev = statistics.stdev(comparison_counts)
print("\n\n\n")
print("Ortalama:", mean)
print("Standart Sapma:", stdev)
```

```
rastgele oluşturulan dizi:
 [80, 44, 113, 87, 117, 110, 45, 76, 90, 53, 98, 72, 84, 108, 39, 126, 68, 119, 73, 124, 105, 13, 58, 34, 109, 81, 130, 46, 70,
 93, 96, 125, 106, 116, 129, 91, 102, 112, 65, 114, 48, 122, 30, 120, 17, 123, 43, 50, 97, 64]



Ortalama: 11.64
Standart Sapma: 4.420545723074914
```

Rapor/Ödev Başlığı :
Hazırlayanın Adı Soyadı :

KAYNAKÇA

1-https://python-istihza.yazbel.com/donguler.html

2- https://medium.com/kodcular/nedir-bu-big-o-notation-b8b9f1416d30

3- https://www.youtube.com/watch?v=XQqjUSOi45o

4- https://www.serkanseker.com/tr/algoritama-karmasiklik-analizi/

5- https://web.ogu.edu.tr/Storage/egulbandilar/Uploads/AlgoritmaAnalizi.pdf

6-https://www.hurriyet.com.tr/egitim/aritmetik-ortalama-nedir-nasil-hesaplanir-aritmetik-ortalama-formulu-var-mi-41818986

7- https://standart-sapma.hesaplama.net/

8- https://www.miuul.com/not-defteri/matrislere-giris-ve-pythonda-matrisler

9- https://forum.yazbel.com/t/python-matris-carpimi/10359

10- https://www.pythontr.com/makale/python-listelerde-matrisler-47

11- https://github.com/saffetmurat/python/blob/master/MatrisIslemleri.py

12- https://techpy.net/tr/2021/03/29/listeler-vektorler-matrisler/