T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU:ALGORİTMA ANALİZİ

ÖĞRENCİ ADI:Sude Naz DEMİRTAŞ /İclal ÇENGEL
ÖĞRENCİ NUMARASI:210501012/21050013

DERS SORUMLUSU:
PROF. DR Hüseyin Tarık DURU

TARİH:
30.03.2024

1 GİRİŞ

1.1 Projenin amacı

Projenin amacı, verilen sıralı olmayan tam sayı dizisi üzerinde fark değerlerini bulmak için farklı algoritmaların tasarlanması ve analiz edilmesidir.

- Verilen tam sayı dizisinden fark değerlerini bulacak algoritmaların tasarlanması.
- Her bir algoritmanın karmaşıklık analizlerinin yapılması.
- Algoritmaların gerçeklenmesi ve doğrulanması.
- Algoritmaların performanslarının karşılaştırılması.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

- Visual Studio Code x64 1.87.2

2.2 Fonksiyonel gereksinimler

•Verilen tam sayı dizisi üzerinde fark değerlerini bulacak algoritmaların tasarlanması.
•Algoritmaların karmaşıklık analizlerinin yapılması.
•Algoritmaların gerçeklenmesi ve doğrulanması.
•Algoritmaların performanslarının karşılaştırılması.

3 TASARIM

3.1 Mimari tasarım

- Algoritmaların modüler bir şekilde tasarlanması ve ayrıntılı karmaşıklık analizlerinin yapılması.

3.2 Kullanılacak teknolojiler

- Python programlama dili kullanılacak.

3.3 Veri tabanı tasarımı

- Veri tabanı kullanılmadığı için bu başlık göz ardı edilmiştir.

3.4 Kullanıcı arayüzü tasarımı

- Kullanıcı arayüzü tasarımı gereksinimi olmadığı için bu başlık göz ardı edilmiştir.

4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları (Pseudo Kodları)

Seçenek 1:

Pseudo Kod 1:

```
MinimumFarkBul1(dizi):
    minimum_fark = sonsuz
    for i from 0 to n-1:
        for j from i+1 to n-1:
            fark = |dizi[j] - dizi[i]|
            if fark < minimum_fark:
                minimum_fark = fark
    return minimum_fark
```

Pseudo Kod 2:

```
MinimumFarkBul2(dizi):
    Sırala(dizi) // Diziyi sıralama algoritması kullanarak sırala
    minimum_fark = sonsuz
    for i from 0 to n-2:
        fark = |dizi[i+1] - dizi[i]|
        if fark < minimum_fark:
            minimum_fark = fark
    return minimum_fark
```

Seçenek 2:

Pseudo Kod 3:

```
MinimumNFarkBul1(dizi):
    minimum_farklar = []
    for i from 0 to n-1:
        for j from i+1 to n-1:
            fark = |dizi[j] - dizi[i]|
            minimum_farklar.ekle(fark)
    minimum_farklar.sırala() // Dizi elemanlarını küçükten büyüğe sırala
    return minimum_farklar[0:n/
```

Pseudo Kod 4:

```
MinimumNFarkBul2(dizi):
    Sırala(dizi) // Diziyi sırala
    minimum_farklar = []
    for i from 0 to n-2:
        fark = |dizi[i+1] - dizi[i]|
        minimum_farklar.ekle(fark)
    minimum_farklar.sırala() // Dizi elemanlarını küçükten büyüğe sırala
    return minimum_farklar[0:n/2]
```

4.2 Görev dağılımı

- Görevler eşit olarak dağıtılmıştır ve her bir ekip üyesi tarafından katkı sağlanmıştır.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Algoritma tasarımı ve karmaşıklık analizlerindeki zorluklar üzerinde çalışılmış ve çözümler bulunmuştur.

4.4 Proje isterlerine göre eksik yönler

- Proje gereksinimlerine uygun olarak kodlanması beklenen tüm görevler tamamlanmıştır, herhangi bir eksiklik bulunmamaktadır.

5 TEST VE DOĞRULAMA

5.1 Karmaşıklık Analizi

Algoritma 1 (MinimumFarkBul1):

En İyi Durum Analizi: En iyi durumda, algoritma her bir elemanı diğer elemanlarla tek tek karşılaştırdığında, minimum farkın ilk iki eleman arasında olduğu durumdur. Bu durumda, iç içe iki döngünün toplam çalışma zamanı O(n^2) olur.
En Kötü Durum Analizi: En kötü durumda, her bir elemanın diğer tüm elemanlarla farkının kontrol edilmesi gerektiğinde gerçekleşir. Bu durumda da iç içe iki döngünün toplam çalışma zamanı O(n^2) olur.

Algoritma 2 (MinimumFarkBul2):

En İyi Durum Analizi: En iyi durumda, dizinin zaten sıralı olduğu durumdur. Sıralama işlemi O(n log n) karmaşıklığına sahiptir. Ardından, ardışık elemanların farkı kontrol edildiğinde ise O(n) karmaşıklığı ile minimum fark bulunur. Dolayısıyla, toplam karmaşıklık O(n log n) + O(n) = O(n log n) olur.
En Kötü Durum Analizi: En kötü durumda, dizinin sırasız olduğu durumdur. Sıralama işlemi O(n log n) karmaşıklığına sahiptir. Ardından, ardışık elemanların farkı kontrol edildiğinde ise O(n) karmaşıklığı ile minimum fark bulunur. Dolayısıyla, toplam karmaşıklık yine O(n log n) + O(n) = O(n log n) olur.

Algoritma 3 (MinimumNFarkBul1):

En İyi Durum Analizi: En iyi durumda, tüm farkların hesaplandığı ve sıralandığı durumdur. Farkların hesaplanması O(n^2) karmaşıklığına

sahiptir. Ardından, sıralama işlemi O(n log n) karmaşıklığına sahiptir. Dolayısıyla, toplam karmaşıklık O(n^2) + O(n log n) = O(n^2) olur.
En Kötü Durum Analizi: En kötü durumda da, tüm farkların hesaplandığı ve sıralandığı durumdur. Karmaşıklık analizi yine O(n^2) + O(n log n) = O(n^2) olarak kalır.

Algoritma 4 (MinimumNFarkBul2):

En İyi Durum Analizi: En iyi durumda, dizinin zaten sıralı olduğu durumdur. Sıralama işlemi O(n log n) karmaşıklığına sahiptir. Ardından, ardışık elemanların farkı kontrol edildiğinde ise O(n) karmaşıklığı ile minimum farklar bulunur. Dolayısıyla, toplam karmaşıklık O(n log n) + O(n) = O(n log n) olur.

En Kötü Durum Analizi: En kötü durumda, dizinin sırasız olduğu durumdur. Sıralama işlemi O(n log n) karmaşıklığına sahiptir. Ardından, ardışık elemanların farkı kontrol edildiğinde ise O(n) karmaşıklığı ile minimum farklar bulunur. Dolayısıyla, toplam karmaşıklık yine O(n log n) + O(n) = O(n log n) olur.

5.2 Yazılımın test süreci

```
[Running] python -u "c:\Users\sudem\Masaüstü\deneme.py"
Algoritma 1 için geçen süre: 0.5236964225769043
Algoritma 2 için geçen süre: 0.0019979476928710938
Algoritma 3 için geçen süre: 1.7100169658660889
Algoritma 4 için geçen süre: 0.002998828887939453

[Done] exited with code=0 in 2.611 seconds

[Running] python -u "c:\Users\sudem\Masaüstü\deneme.py"
Algoritma 1 için geçen süre: 0.6861367225646973
Algoritma 2 için geçen süre: 0.0009996891021728516
Algoritma 3 için geçen süre: 1.9797980785369873
Algoritma 4 için geçen süre: 0.0009987354278564453
```

Algoritma 1 ve Algoritma 2 Karşılaştırması:

Algoritma 1 (MinimumFarkBul1), dizinin tüm elemanları arasındaki farkı kontrol	Algoritma 2 (MinimumFarkBul2), diziyi önce sıralayarak sonra ardışık elemanların farklarını

ederek minimum farkı bulmaktadır. Bu nedenle, algoritmanın karmaşıklığı O(n^2) olarak değerlendirilir.	kontrol ederek minimum farkı bulmaktadır. Bu nedenle, Algoritma 2'nin karmaşıklığı O(n log n) olur.

Test sonuçları, Algoritma 2'nin belirgin bir şekilde daha hızlı olduğunu göstermektedir. Örneğin, 3000 elemanlı bir dizide Algoritma 1'in geçen süresi 0.6861367225646973 saniye iken, Algoritma 2 için bu süre 0.0009996891021728516 saniyedir.

Algoritma 3 ve Algoritma 4 Karşılaştırması:

Algoritma 3 (MinimumNFarkBul1), n/2 en küçük farkı bulmaya çalışırken, sıralama işlemi yapmadan tüm farkları hesaplar. Karmaşıklığı O(n^2) olarak değerlendirilir.	Algoritma 4 (MinimumNFarkBul2), n/2 en küçük farkı bulmaya çalışırken, önce diziyi sıralayarak sonra ardışık elemanların farklarını kontrol eder. Karmaşıklığı O(n log n) olur.

Test sonuçlarına göre, Algoritma 4'ün Algoritma 3'e göre daha hızlı olduğu görülmektedir. Örneğin, 3000 elemanlı bir dizide Algoritma 3'ün geçen süresi 1.9797980785369873 saniye iken, Algoritma 4 için bu süre 0.0009987354278564453 saniyedir.

Genel olarak, sıralama işlemi gerektiren algoritmaların (Algoritma 2 ve Algoritma 4), sıralama işlemi yapılmayan algoritmalara (Algoritma 1 ve Algoritma 3) göre daha hızlı olduğu görülmektedir. Veri setinin büyüklüğüne bağlı olarak, en uygun algoritmanın seçimi yapılarak performansın artırılması sağlanabilir.

5.3 Performans Karşılaştırması

Algoritma 1 ve Algoritma 2 Karşılaştırması:

Algoritma 1 (MinimumFarkBul1): Bu algoritma, her eleman için diğer tüm elemanların farkını kontrol ederek minimum farkı bulur. Karmaşıklığı O(n^2) olarak değerlendirilir.	Algoritma 2 (MinimumFarkBul2): Bu algoritma, diziyi önce sıralayarak sonra ardışık elemanların farklarını kontrol ederek minimum farkı bulur. Karmaşıklığı O(n log n) olur.

3000 elemanlı bir dizide, Algoritma 1 için geçen süre 0.6861367225646973 saniye iken, Algoritma 2 için bu süre 0.0009996891021728516 saniyedir. Sonuç olarak, Algoritma 2'nin belirgin bir şekilde daha hızlı olduğu gözlemlenmiştir.

Algoritma 3 ve Algoritma 4 Karşılaştırması:

Algoritma 3 (MinimumNFarkBul1): Bu algoritma, n/2 en küçük farkı bulmaya çalışırken, tüm farkları hesaplar. Karmaşıklığı O(n^2) olarak değerlendirilir.	Algoritma 4 (MinimumNFarkBul2): Bu algoritma, n/2 en küçük farkı bulmaya çalışırken, önce diziyi sıralayarak sonra ardışık elemanların farklarını kontrol eder. Karmaşıklığı O(n log n) olur.

3000 elemanlı bir dizide, Algoritma 3 için geçen süre 1.9797980785369873 saniye iken, Algoritma 4 için bu süre 0.0009987354278564453 saniyedir. Sonuç olarak, Algoritma 4'ün Algoritma 3'e göre daha hızlı olduğu görülmüştür.