Algoritma Analizi

Ödev - 2

Böl ve Yönet Algoritmalar

Öğrenci Adı: Adem Alp ŞAHİN

Öğrenci Numarası: 22011090

Dersin Eğitmeni: Mehmet Amaç Güvensan

Video Linki: https://youtu.be/DsnFVysHSx4

1- Problemin Çözümü:

Kullanıcıdan ilk başta dizinin boyutu(N) alınmıştır, dizide ilgili boyuta göre yer açılmıştır. Diziye ilk başta 1'den N'e kadar değerler sırasıyla atanmıştır ve shuffle fonksiyonu ile diziyi karıştırma işlemi yapılmıştır.

K-th way Merge Sort işlemini yapmak için döngünün içinde "2nd Way", "3rd Way"…"10th Way" olacak şekilde özelleştirilmiş Merge Sort algoritması kullanılmıştır. Merge Sort algoritması alınan k değerine göre ilgili diziyi aldığı "left" ve "right" parametreleri sınırlarında k parçaya böler, bölme işlemi alt dizilerde en fazla k-1 eleman kalana kadar devam eder. En son k-1 eleman kaldığında ise bu alt diziler sıralı olmadığından ve Merge işlemi de sıralı dizileri birleştirmeye yönelik olduğundan ilgili en fazla k-1 elemana sahip olan diziler Selection Sort ile sıralanır. Ardından sıralı alt diziler parçadan bütüne doğru sıralı olma özellikleri bozulmayacak şekilde birleştirilir, en son ortaya çıkan dizi ise ilk baştaki karışık dizinin sıralanmış hali olur.

Sıralanmış diziler hangi k değeri için oldukları belirtilerek kaç saniye içinde gerçekleştikleri kullanıcıya bilgi vermek amaçlı yazdırılır.

2- Karşılaşılan Sorunlar:

İlk başta karışık diziyi oluşturmak için bir döngüyü N kez döndürüp her defasında rastgele bir sayı üretip bu sayının dizide olup olmadığını kontrol ediliyordu ancak belli bir eleman sayısından sonra çok fazla zaman harcanmaya başlamıştı, çözüm olarak diziyi ilk başta sıralı oluşturup sonrasında karıştırarak harcanan zaman 1/1000 oranında azaltıldı.

Birleştirme işleminin içinde sort gerçekleştiği için k=2 olan durumda algoritma normalde çalışması gereken hızdan daha yavaş çalışmaktadır.

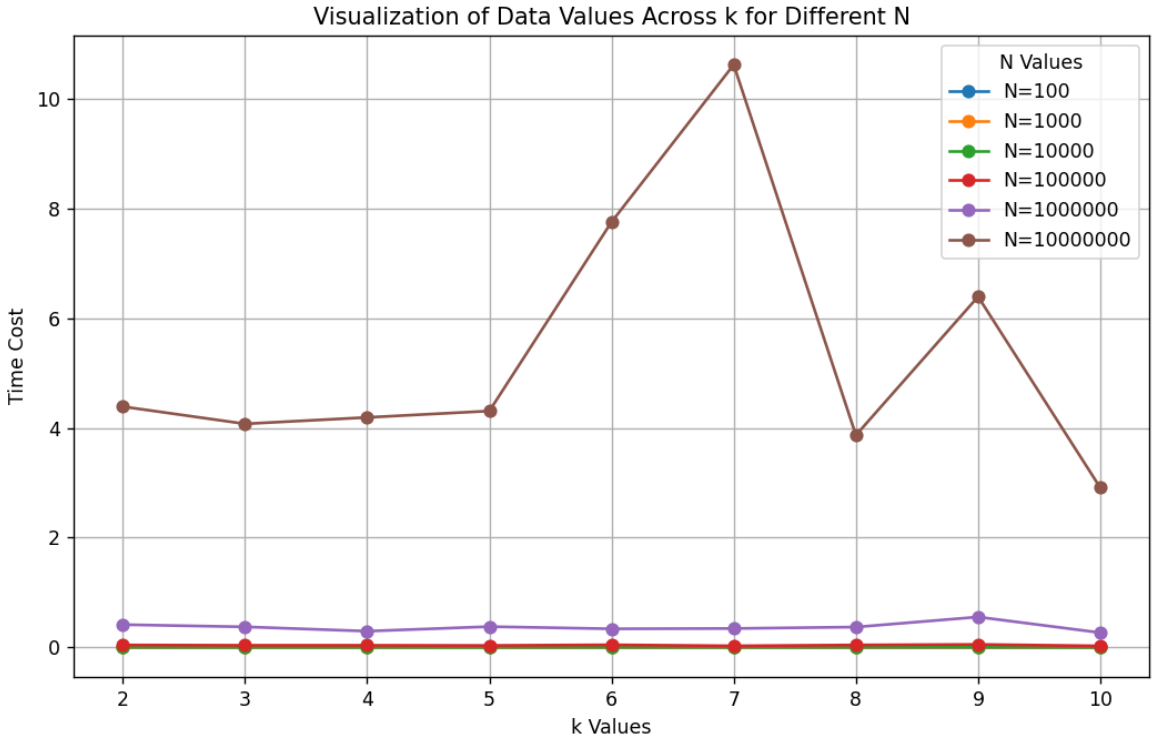
3- Karmaşıklık Analizi:

```
mergeSort:
Input: Arr: Int Array - Array containing the values to be sorted
    left: Int - Starting index of the subarray
    right: Int - Ending index of the subarray
    counter: Int - Current depth level of recursion
    k: Int - Number of partitions
    N: Int - Length of the array

IF right > left  THEN  // Check if the subarray has more than (k-1) elements
```

```
    DECLARE m: Int Array of size (k - 1)  // Array to store partition points

    // Calculate partition points to divide the array into k parts
    FOR i FROM 0 TO k - 2 DO
       m[i] := ((right - left) * (i + 1))
       m[i] := m[i] / k
       m[i] := left + m[i]
    ENDFOR


    CALL mergeSort with Arr, left, m[0], counter + 1, k, N
    FOR i FROM 0 TO k - 2 DO
       CALL mergeSort with Arr, m[i] + 1, m[i + 1], counter + 1, k, N
    ENDFOR
    CALL mergeSort with Arr, m[k - 2] + 1, right, counter + 1, k, N

    // Merge the k divided subarrays back together
    CALL merge with Arr, left, m, right, counter + 1, k, N
ENDIF
```

$T(n) = T(n/k) + n \rightarrow k = k^1$ then $T(n) \in O(n\log_k n)$ from Master Theorem

4-Ekran Çıktıları:




```
N: 100----
k=2 took 0.000 seconds to execute--Succesfully sorted.
k=3 took 0.000 seconds to execute--Succesfully sorted.
k=4 took 0.000 seconds to execute--Succesfully sorted.
k=5 took 0.000 seconds to execute--Succesfully sorted.
k=6 took 0.000 seconds to execute--Succesfully sorted.
k=7 took 0.000 seconds to execute--Succesfully sorted.
k=8 took 0.000 seconds to execute--Succesfully sorted.
k=9 took 0.000 seconds to execute--Succesfully sorted.
k=10 took 0.000 seconds to execute--Succesfully sorted.
N: 1000----
k=2 took 0.000 seconds to execute--Succesfully sorted.
k=3 took 0.000 seconds to execute--Succesfully sorted.
k=4 took 0.000 seconds to execute--Succesfully sorted.
k=5 took 0.000 seconds to execute--Succesfully sorted.
k=6 took 0.001 seconds to execute--Succesfully sorted.
k=7 took 0.000 seconds to execute--Succesfully sorted.
k=8 took 0.001 seconds to execute--Succesfully sorted.
k=9 took 0.000 seconds to execute--Succesfully sorted.
k=10 took 0.001 seconds to execute--Succesfully sorted.
```

```
N: 10000----
k=2 took 0.004 seconds to execute--Succesfully sorted.
k=3 took 0.003 seconds to execute--Succesfully sorted.
k=4 took 0.004 seconds to execute--Succesfully sorted.
k=5 took 0.003 seconds to execute--Succesfully sorted.
k=6 took 0.003 seconds to execute--Succesfully sorted.
k=7 took 0.003 seconds to execute--Succesfully sorted.
k=8 took 0.005 seconds to execute--Succesfully sorted.
k=9 took 0.005 seconds to execute--Succesfully sorted.
k=10 took 0.002 seconds to execute--Succesfully sorted.
N: 100000----
k=2 took 0.041 seconds to execute--Succesfully sorted.
k=3 took 0.043 seconds to execute--Succesfully sorted.
k=4 took 0.038 seconds to execute--Succesfully sorted.
k=5 took 0.035 seconds to execute--Succesfully sorted.
k=6 took 0.047 seconds to execute--Succesfully sorted.
k=7 took 0.028 seconds to execute--Succesfully sorted.
k=8 took 0.043 seconds to execute--Succesfully sorted.
k=9 took 0.055 seconds to execute--Succesfully sorted.
k=10 took 0.025 seconds to execute--Succesfully sorted.
```

```
N: 1000000----
k=2 took 0.429 seconds to execute--Succesfully sorted.
k=3 took 0.392 seconds to execute--Succesfully sorted.
k=4 took 0.306 seconds to execute--Succesfully sorted.
k=5 took 0.383 seconds to execute--Succesfully sorted.
k=6 took 0.343 seconds to execute--Succesfully sorted.
k=7 took 0.358 seconds to execute--Succesfully sorted.
k=8 took 0.408 seconds to execute--Succesfully sorted.
k=9 took 0.603 seconds to execute--Succesfully sorted.
k=10 took 0.271 seconds to execute--Succesfully sorted.
N: 10000000----
k=2 took 4.356 seconds to execute--Succesfully sorted.
k=3 took 4.012 seconds to execute--Succesfully sorted.
k=4 took 3.987 seconds to execute--Succesfully sorted.
k=5 took 4.317 seconds to execute--Succesfully sorted.
k=6 took 8.915 seconds to execute--Succesfully sorted.
k=7 took 9.941 seconds to execute--Succesfully sorted.
k=8 took 3.729 seconds to execute--Succesfully sorted.
k=9 took 6.270 seconds to execute--Succesfully sorted.
k=10 took 2.913 seconds to execute--Succesfully sorted.
```

```
N: 100----
k=2 took 0.000 seconds to execute--Succesfully sorted.
k=3 took 0.000 seconds to execute--Succesfully sorted.
k=4 took 0.000 seconds to execute--Succesfully sorted.
k=5 took 0.000 seconds to execute--Succesfully sorted.
k=6 took 0.000 seconds to execute--Succesfully sorted.
k=7 took 0.000 seconds to execute--Succesfully sorted.
k=8 took 0.000 seconds to execute--Succesfully sorted.
k=9 took 0.000 seconds to execute--Succesfully sorted.
k=10 took 0.000 seconds to execute--Succesfully sorted.
N: 1000----
k=2 took 0.001 seconds to execute--Succesfully sorted.
k=3 took 0.000 seconds to execute--Succesfully sorted.
k=4 took 0.001 seconds to execute--Succesfully sorted.
k=5 took 0.000 seconds to execute--Succesfully sorted.
k=6 took 0.000 seconds to execute--Succesfully sorted.
k=7 took 0.000 seconds to execute--Succesfully sorted.
k=8 took 0.001 seconds to execute--Succesfully sorted.
k=9 took 0.000 seconds to execute--Succesfully sorted.
k=10 took 0.001 seconds to execute--Succesfully sorted.
```

```
N: 10000----
k=2 took 0.004 seconds to execute--Succesfully sorted.
k=3 took 0.004 seconds to execute--Succesfully sorted.
k=4 took 0.003 seconds to execute--Succesfully sorted.
k=5 took 0.003 seconds to execute--Succesfully sorted.
k=6 took 0.003 seconds to execute--Succesfully sorted.
k=7 took 0.003 seconds to execute--Succesfully sorted.
k=8 took 0.005 seconds to execute--Succesfully sorted.
k=9 took 0.005 seconds to execute--Succesfully sorted.
k=10 took 0.003 seconds to execute--Succesfully sorted.
N: 100000----
k=2 took 0.041 seconds to execute--Succesfully sorted.
k=3 took 0.036 seconds to execute--Succesfully sorted.
k=4 took 0.038 seconds to execute--Succesfully sorted.
k=5 took 0.035 seconds to execute--Succesfully sorted.
k=6 took 0.045 seconds to execute--Succesfully sorted.
k=7 took 0.029 seconds to execute--Succesfully sorted.
k=8 took 0.042 seconds to execute--Succesfully sorted.
k=9 took 0.052 seconds to execute--Succesfully sorted.
k=10 took 0.024 seconds to execute--Succesfully sorted.
```

```
N: 1000000----
k=2 took 0.421 seconds to execute--Succesfully sorted.
k=3 took 0.391 seconds to execute--Succesfully sorted.
k=4 took 0.297 seconds to execute--Succesfully sorted.
k=5 took 0.384 seconds to execute--Succesfully sorted.
k=6 took 0.337 seconds to execute--Succesfully sorted.
k=7 took 0.351 seconds to execute--Succesfully sorted.
k=8 took 0.378 seconds to execute--Succesfully sorted.
k=9 took 0.564 seconds to execute--Succesfully sorted.
k=10 took 0.267 seconds to execute--Succesfully sorted.
N: 10000000----
k=2 took 4.360 seconds to execute--Succesfully sorted.
k=3 took 4.018 seconds to execute--Succesfully sorted.
k=4 took 4.014 seconds to execute--Succesfully sorted.
k=5 took 4.677 seconds to execute--Succesfully sorted.
k=6 took 5.485 seconds to execute--Succesfully sorted.
k=7 took 9.767 seconds to execute--Succesfully sorted.
k=8 took 3.727 seconds to execute--Succesfully sorted.
k=9 took 6.214 seconds to execute--Succesfully sorted.
k=10 took 2.980 seconds to execute--Succesfully sorted.
```