Ödev-2: Böl ve Yönet Algoritmaları

Öğrenci Adı: Tarık TAŞKIN

Öğrenci Numarası: 21011022

Dersin Eğitmeni: Doç.Dr. Mehmet Amaç GÜVENSAN

1- Problemin Çözümü:

Kullanıcıdan dizinin boyutu (N) ve sıralama işlemi yapılırken dizinin kaça bölüneceği (k) bilgileri alınır. Ardından dizi önce sıralı bir biçimde oluşturulur, daha sonra son indise gelecek rastgele bir indis seçilir ve değerler yer değiştirir. Bu işlem dizinin sonundan başlayarak başına kadar gerçekleştirilir ve dizi her bir elemanı bir kere içerecek şekilde karıştırılmış olunur.

Daha sonra diziyi sıralama işlemine geçilir. Her seferinde dizi k parçaya bölünür. Bölünen parçaların boyutu k'dan küçük olmadığı sürece bölünmeye devam eder. Bölünen parçalar k'dan küçük ise parçalar, insertion sort algoritması ile sıralanır. Her bir parça sıralandıktan sonra parçalar sıralı bir şekilde birleştirilir. Bu birleştirme işlemi min-heap yapısı kullanılarak gerçekleştirilir. Min-heap yapısı sıralı her bir parçanın ilk elemanıyla yani en küçük elemanıyla doldurulur. Ardından parçalarda eleman kalmayana kadar min-heap'den en küçük eleman çekilir, çekilen eleman sıralı diziye aktarılır ve çekilen elemanın ait olduğu parçada eğer başka eleman varsa sıralı diziye aktarılması için min-heap'e eklenir. Min-heap'de eleman kalmadığında parçalardaki bütün elemanlar sıralı bir şekilde diziye eklenmiş olur. Bu şekilde program, merge sort algoritmasını k-way merging kullanarak gerçekleştirmiş olur.

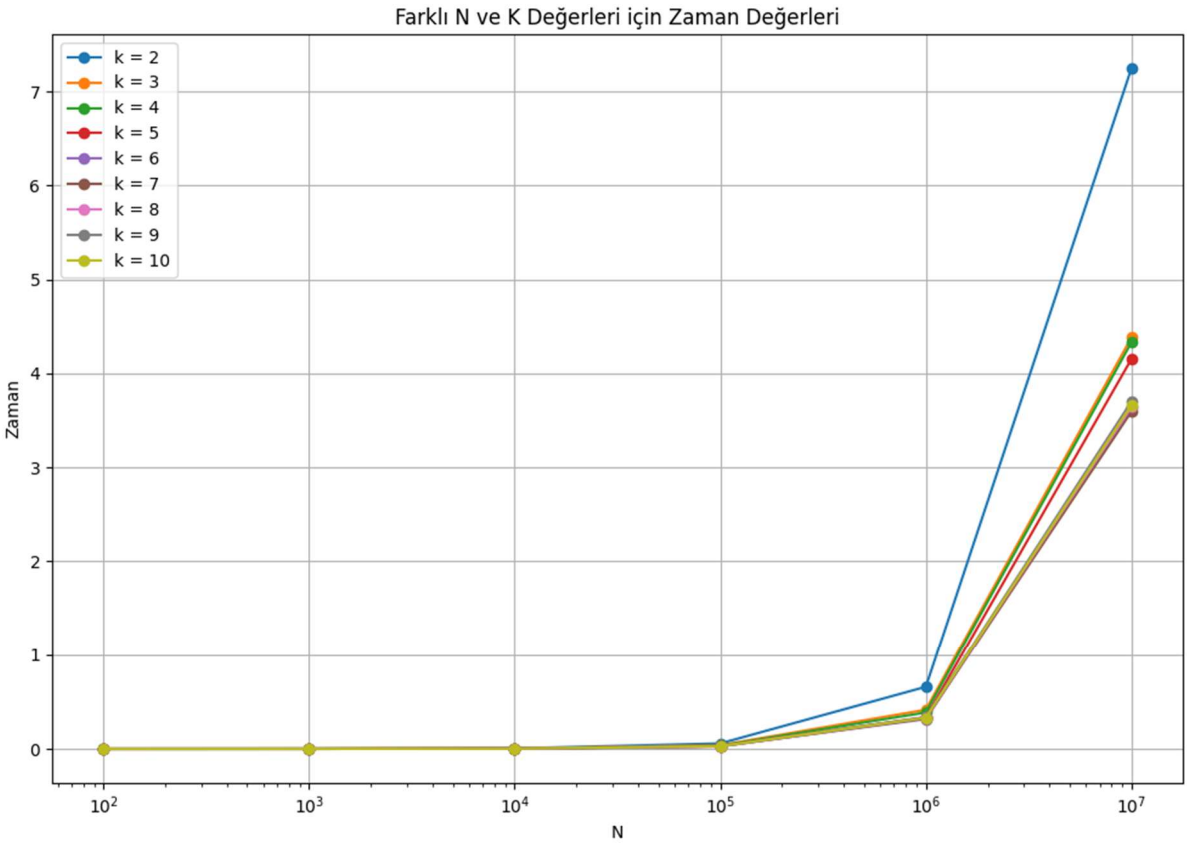
2- Karşılaşılan Sorunlar:

N elemanlı ve her bir değeri eşsiz dizi; rastgele bir sayı seçilip, bu sayının dizide olup olmadığı kontrol edildikten sonra diziye eklenmesi $O(N^2)$ karmaşıklığında gerçekleşiyordu. Bunun yerine Fisher-Yates shuffle algoritması kullanılarak rastgele, eşsiz N elemanlı dizi oluşturma işlemi $O(N)$ karmaşıklığında gerçekleştirilmiş oldu.

Karşılaşılan bir diğer sorun ise sıralanan dizilerin birleştirilmesinde her bir parçanın en küçük elemanlarını karşılaştırıp en küçüğünü sıralanan diziye eklenmesiydi. Bu şekilde gerçekleştirilen algoritmanın karmaşıklığı $O(N * k)$ olmaktaydı. Ama bunun yerine daha min-heap yapısı kullanılarak k dizinin birleştirilmesi $O(N * logk)$ karmaşıklığında gerçekleştirildi.

N'in büyük değerleri için diziyi ekrana bastırma zaman aldığından dolayı $N \leq 1000$ durumlarda dizi ekrana bastırılmıştır. Ardından dizi doğru sıralanıp sıralanmadığı kontrol edilip kullanıcı bilgilendirilmiştir.

3- Karmaşıklık Analizi:



	N = 100	N = 1.000	N = 10.000	N = 100.000	N = 1.000.000	N = 10.000.000
k = 2	0.000000	0.000500	0.005000	0.057100	0.658100	7.248900
k = 3	0.000001	0.000300	0.003200	0.038500	0.414600	4.384300
k = 4	0.000000	0.000200	0.002600	0.033600	0.385300	4.332600
k = 5	0.000000	0.000200	0.002400	0.030900	0.334700	4.154900
k = 6	0.000000	0.000200	0.002700	0.027900	0.315900	3.638200
k = 7	0.000000	0.000200	0.002100	0.024900	0.323900	3.595900
k = 8	0.000000	0.000200	0.002300	0.027400	0.333200	3.645600
k = 9	0.000000	0.000200	0.002200	0.028200	0.334900	3.702100
k = 10	0.000000	0.000200	0.002300	0.028400	0.329500	3.666300

Her N değeri için karışık 10 dizi oluşturulmuştur. Bu dizilerin her k değeri için sıralanma süreleri hesaplanıp ardından ortalaması alınmıştır.

N değeri arttıkça programın çalışma süresi artmış, N değeri sabitken k değerinin artmasıyla genelde çalışma süresinde bir azalma görülmüştür. N'in 10.000 değerine kadar farklı k değerleri için çalışma süreleri pek fark etmezken 10.000 değerinden sonraki değerlerde k'nın artmasıyla çalışma süreleri azalmıştır. Program, genel olarak $k = 7$ değerinde en düşük çalışma sürelerini üretmiştir. K'nın 10'dan büyük değerleri de bu dizilerde denendiğinde daha düşük çalışma süreleri elde edilmiştir.

$kWayMergeSort(array[left, right], k)$:
  $FOR\ i\ 0\ to\ k$:
    $subArrays[i] = array\big[(i * k), \big((i + 1) * k\big)\big]$
    $kWayMergeSort(subArrays[i], k)$
  $ENDFOR$
  $kWayMerge(subArrays, k)$

Algoritmadaki en karmaşık işlem k-way merge sort algoritmasıdır. Dizi her seferinde k parçaya bölünür ve bölünen parçalar $O(N * logk)$ karmaşıklığında birleştirilir.

$$T(N) = k * T\left(\frac{N}{k}\right) + N * \log k$$

$$T(N) = k * \left(k * T\left(\frac{N}{k^2}\right) + \frac{N}{k} * logk\right) + N * \log k$$

$$T(N) = k^i * T\left(\frac{N}{k^i}\right) + i * O(N * \log k)$$

$$k^i = N \rightarrow i = \log_k N$$

$$T(N) = N * T(1) + \log_k N * O(N * logk) \cong O(N * \log_k N * \log k) \equiv O(N * \log N)$$

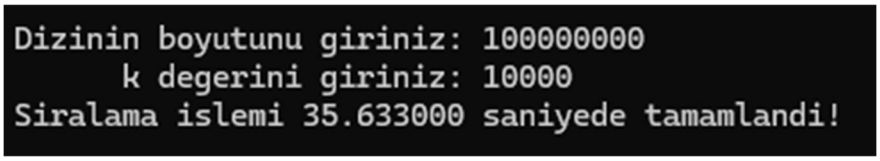
$$(N * \frac{\log N}{\log k} * \log k = N * \log N\ \ e\text{ş}itli\check{g}inden\ yararlanılır.)$$

4- Ekran Çıktıları:

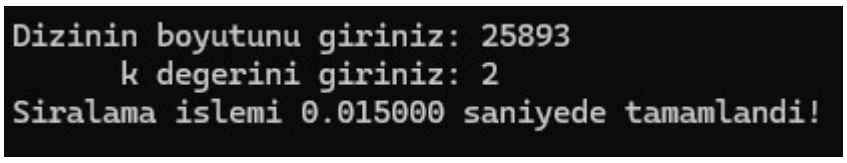
$$N \leq 0$$
$$k < 2$$

```
Dizinin boyutunu giriniz: 0
Dizinin boyutunu yanlis girdiniz! (N>0)

Dizinin boyutunu giriniz: 4664
      k degerini giriniz: 0
k degerini yanlis girdiniz! (k>1)

      k degerini giriniz: 1
k degerini yanlis girdiniz! (k>1)

      k degerini giriniz: 2
Siralama islemi basarili!
```

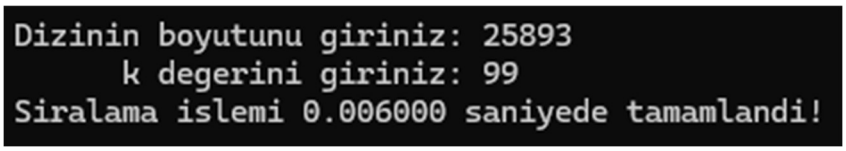
$N = 100.000.000$
$k = 10.000$



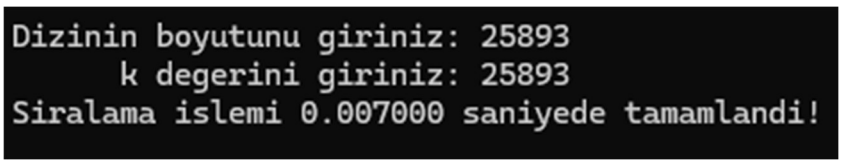
$N = 25.893$
$k = 2$



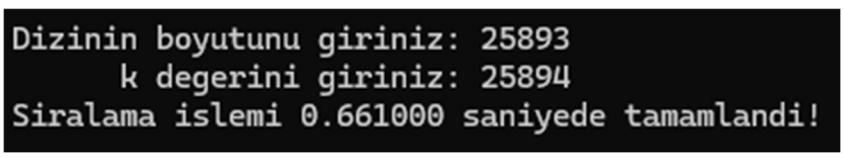
$N = 25.893$
$k = 99$



$N = 25.893$
$k = 25.893$



$N = 25.893$
$k = 25.894$



Bu durumda $k > N$ olduğu için dizi parçalara ayrılamaz, parçalara ayrılmayan dizi de insertion sort ile sıralanır. Bir üstteki örnekteki süre ile buradaki süre arasında neredeyse 100 katlık bir süre farkı vardır.

$N = 35$
$k = 13$

```
Dizinin boyutunu giriniz: 35
      k degerini giriniz: 13

Sirasiz Dizi: 8 12 33 2 14 15 3 13 32 27 24 18 22 1 19 7 5 31 23 6 17 35 25 10 34 30 28 11 26 21 4 29 9 16 20
 Sirali Dizi: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Siralama islemi 0.000000 saniyede tamamlandi!
```

$N = 2$
$k = 2$

```
Dizinin boyutunu giriniz: 2
      k degerini giriniz: 2

Sirasiz Dizi: 2 1
 Sirali Dizi: 1 2
Siralama islemi 0.000000 saniyede tamamlandi!
```

$N = 29$
$k = 6$

```
Dizinin boyutunu giriniz: 29
      k degerini giriniz: 6

Sirasiz Dizi: 22 6 10 20 19 3 16 4 29 23 11 13 28 9 18 17 12 26 15 7 27 25 24 8 14 5 21 2 1
 Sirali Dizi: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Siralama islemi 0.000000 saniyede tamamlandi!
```