K-Way Merge Sort

Öğrenci Adı: Muhammed İkbal AKGÜNDOĞDU

Öğrenci Numarası: 21011075

Dersin Eğitmeni: M. Elif Karslıgil

Video Linki:

https://youtu.be/hdYGqkY9qJc

Problemin Çözümü:

Kullanıcıdan alınan herhangi bir değer yoktur main içerisinde 3 adet for döngsü bulunmaktadır 1.si 10 farklı dizi ile çalışmak için kullanılan ilk for bu bize tek çalıştırmada 10 farklı diziden değer okumayı sağlayacaktır 2.si N değerleri için bu forda her girişte 10^i*100 formülü ile N= 100,1000,1000 … değerleri sağlanmıştır son dizi ise K = 1,2,3, 4…10 için kullanılmıştır

K'içerisinde uygun diziler fonksiyon ile önce oluşturulmuştur sıralı bir şekilde ve sornasında karıştırılmıştır böylece uniq olarak sırasız bir dizi elde edilmiş ve işleme hazır hale getirilmiştir. Oluşturulan dizi orijinal diziye atanmıştır işlemler ise copyarray denilen dizide olacaktır çünkü işlem bittiğinde farlı K ve N değerleri için aynı dizi kullanılacaktır ve orijinal dizide değişiklik olmaması bu şekilde sağlanır yani işlemlere başlanmadan copyarray dizisi orijinale eşitleniyor.

Dizi mergesort fonkisoynuna yollanıyor ve burada k kaç ise ona göre bölünme işlemi yapa yapa ilerleniyor mesela k=2 ise 2 ye bölüm yapıla yapıla dizi bölünüyor ya d k=6 ise 6ya bölünüyor. bundan sonra tüm bölümler bittikten sonra recursiveden çıkış şartı tetiklendi ise merge fonksiyonu çalıştırılıyor ve teker teker birleştirme yapılarak ve bu birleştirme esnasında sıralama yapılarak dizi tekrardan ilk boyutuna ama sıralanmış bir şekilde geliyor böylece önce diziyi bölmüş sonra fethetmiş oluyoruz.

Karşılaşılan Sorunlar:

Mergesort işlemleri sırasında eldeki dizi k değerinden küçük olursa bölüm olmuyordu bunu mergesort içerisine bir sıralama kodu koyarak çözdüm ve direkt return verdim böylece dizi sıralanmış olarak birleştirilmeyi bekliyor. 2. Olarak tek dizi üzerinde işlemler yaparken dizi değişikliğe uğruyordu bu durumda ise değişik k ve n değerleri için karşılaştırma yapamıyordum çünkü farklı diziler oluyordu bunu da üstte bahsettiğim üzere bir orijinal bir de işlemlerin yapılacağı kopya dizisi oluşturarak ve her işlemden önce kopya dizisini orijinale eşitleyerek daha doğru bir karşılaştırma ortamı sağlamış oldum. Tek çalıştırmada 10 farklı diziyi sıralamak yerine tek dizide işlem yapıyordum bunu ise dışarıya 10'a kadar dönen bir for döngüsü ekleyerek çözdüm dış döngüde orijinal diziyi her döngü başına tekrardan karıştırıyordum. Elimde bir sürü değer oluşuyordu bunları terminalden tek tek karşılaştırmak ise zor oluyordu bunu ise her değeri "output.txt" dosyasına kaydederek daha okunabilir ve elde kalıcı bir belge oluşturarak kolaylaştırdım ayrıca karşılaştırma esnasında Python üzerinden kolayca grafik kodu yazdım ve karşılaştırmayı farklı k ve ne değerleri için sürenin nasıl değiştiğini görselleştirmiş oldum.

Karmaşıklık Analizi:

Karmaşıklığı analiz etmek için merge sort algoritmasının genel yapılarına bakmamız gerekiyor:
k-way merge sort algoritması, her adımda diziyi k parçaya böler. Bu bölme işlemi rekürsif olarak yapılır ve her bölme işlemi, k'ya bağlı olarak tekrar eder.

- **Bölme işlemi:** Her bölme işlemi, diziyi k'ya böler ve her parça üzerinde merge sort çalıştırır. Bu işlem logaritmik bir şekilde tekrar eder, yani:

depth of recursion=logkn
Bölme işlemi her seferinde diziyi daha küçük parçalara ayırır.

- **Birleştirme işlemi:** Her birleşim adımında, tüm diziyi birleştirirsiniz. Bu birleştirme işlemi her seviyede dizinin tamamını işlediğinden, her bir seviyedeki birleşme işlemi **O(n)** zaman alır.

Toplam karmaşıklık, bölme işlemlerinin derinliği ile birleştirme işlemlerinin karmaşıklığının çarpımına eşittir:
O(nlogkn)
d. Insertion Sort:
Eğer bölümler yeterince küçükse, merge sort'un alt kısmında **insertion sort** kullanılır. Bu, küçük dizileri sıralamak için tercih edilen bir algoritmadır.
Insertion sort'un karmaşıklığı:
O(n^2)
Burada, **insertion sort** yalnızca küçük dizilerde çalıştığı için, büyük n için genellikle merge sort'un etkisi daha fazla olur.
e. Merge İşlemi:
merge_i fonksiyonu k-way merge algoritmasını uygular. Her seviyede, diziyi sıralamak ve birleştirmek için **O(n)** zaman harcar.
Sonuç:
Toplam karmaşıklığı şu şekilde hesaplayabiliriz:

- **Dış döngüler** (herhangi bir önemli etkisi yok): **O(1)**
- **Dizi oluşturma ve rastgele karıştırma: O(n)**
- **Merge Sort (k-way): O(n \log_k n)**
- **Insertion Sort (alt düzey): O(n^2)** (bu küçük parçalar için etkili)

Sonuç olarak, bu algoritmanın zaman karmaşıklığı:
O(n^2) daha büyüktür insertion sorttan kaynaklı ama küçük sayılar için büyük sayılar kullanıldığında ise mergesortun karmaşıklığı baskın gelecektir bundan kaynaklı olarak karmaşıklığımız

$O(nlogkn)$

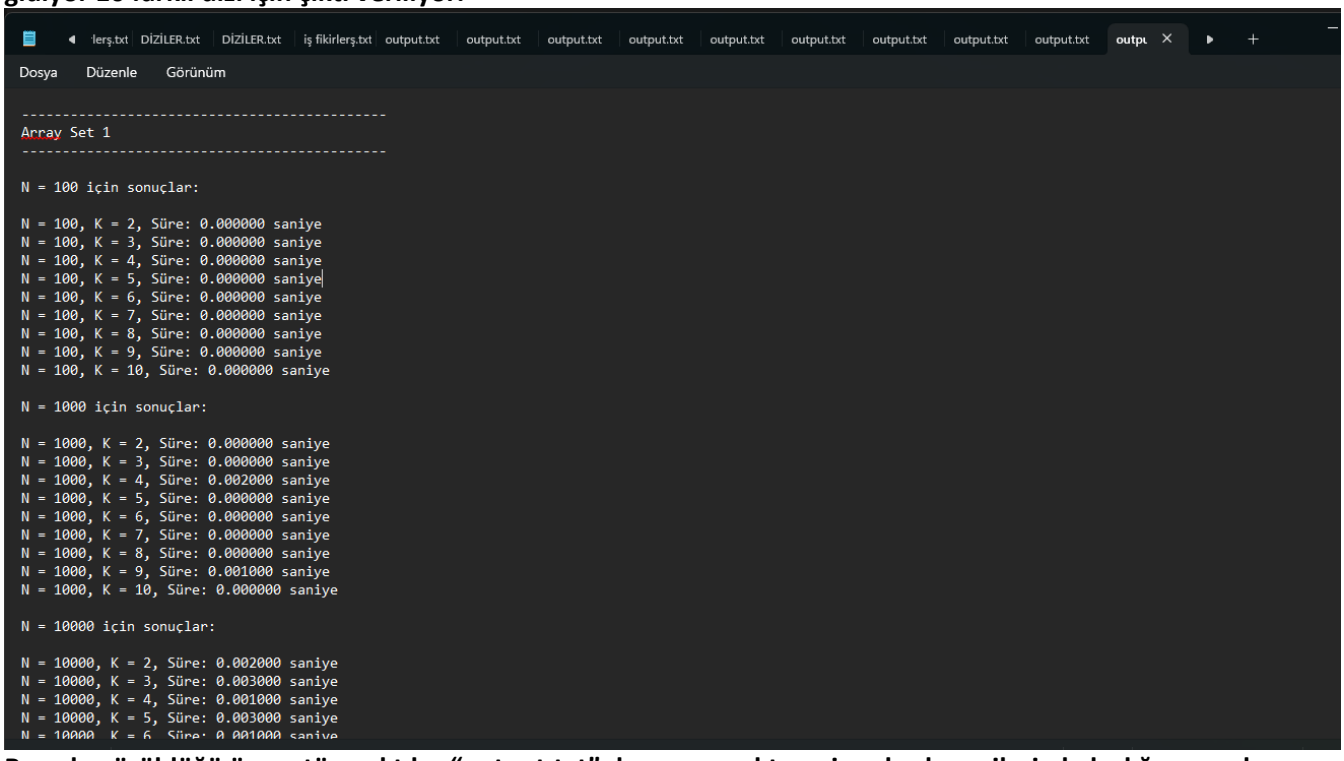
Ekran Çıktıları:

```
-----------------------------------------
Array : 1
-----------------------------------------
Array of size 100:
N = 100 K= 2 icin kullanilan zaman : 0.000000
N = 100 K= 3 icin kullanilan zaman : 0.000000
N = 100 K= 4 icin kullanilan zaman : 0.000000
N = 100 K= 5 icin kullanilan zaman : 0.000000
N = 100 K= 6 icin kullanilan zaman : 0.000000
N = 100 K= 7 icin kullanilan zaman : 0.000000
N = 100 K= 8 icin kullanilan zaman : 0.000000
N = 100 K= 9 icin kullanilan zaman : 0.000000
N = 100 K= 10 icin kullanilan zaman : 0.000000

Array of size 1000:
N = 1000 K= 2 icin kullanilan zaman : 0.000000
N = 1000 K= 3 icin kullanilan zaman : 0.000000
N = 1000 K= 4 icin kullanilan zaman : 0.002000
N = 1000 K= 5 icin kullanilan zaman : 0.000000
N = 1000 K= 6 icin kullanilan zaman : 0.000000
N = 1000 K= 7 icin kullanilan zaman : 0.000000
N = 1000 K= 8 icin kullanilan zaman : 0.000000
N = 1000 K= 9 icin kullanilan zaman : 0.001000
N = 1000 K= 10 icin kullanilan zaman : 0.000000
```

```
Array of size 1000000:
N = 1000000 K= 2 icin kullanilan zaman : 0.276000
N = 1000000 K= 3 icin kullanilan zaman : 0.219000
N = 1000000 K= 4 icin kullanilan zaman : 0.194000
N = 1000000 K= 5 icin kullanilan zaman : 0.183000
N = 1000000 K= 6 icin kullanilan zaman : 0.190000
N = 1000000 K= 7 icin kullanilan zaman : 0.232000
N = 1000000 K= 8 icin kullanilan zaman : 0.205000
N = 1000000 K= 9 icin kullanilan zaman : 0.223000
N = 1000000 K= 10 icin kullanilan zaman : 0.189000

Array of size 10000000:
N = 10000000 K= 2 icin kullanilan zaman : 2.940000
N = 10000000 K= 3 icin kullanilan zaman : 2.263000
N = 10000000 K= 4 icin kullanilan zaman : 2.100000
N = 10000000 K= 5 icin kullanilan zaman : 1.887000
N = 10000000 K= 6 icin kullanilan zaman : 1.956000
N = 10000000 K= 7 icin kullanilan zaman : 2.220000
N = 10000000 K= 8 icin kullanilan zaman : 2.212000
N = 10000000 K= 9 icin kullanilan zaman : 2.384000
N = 10000000 K= 10 icin kullanilan zaman : 2.109000
```

```
-----------------------------------------
Array : 2
-----------------------------------------
Array of size 100:
N = 100 K= 2 icin kullanilan zaman : 0.000000
N = 100 K= 3 icin kullanilan zaman : 0.000000
N = 100 K= 4 icin kullanilan zaman : 0.000000
N = 100 K= 5 icin kullanilan zaman : 0.000000
N = 100 K= 6 icin kullanilan zaman : 0.000000
N = 100 K= 7 icin kullanilan zaman : 0.000000
N = 100 K= 8 icin kullanilan zaman : 0.000000
N = 100 K= 9 icin kullanilan zaman : 0.000000
N = 100 K= 10 icin kullanilan zaman : 0.000000

Array of size 1000:
N = 1000 K= 2 icin kullanilan zaman : 0.000000
N = 1000 K= 3 icin kullanilan zaman : 0.000000
N = 1000 K= 4 icin kullanilan zaman : 0.001000
N = 1000 K= 5 icin kullanilan zaman : 0.000000
N = 1000 K= 6 icin kullanilan zaman : 0.000000
N = 1000 K= 7 icin kullanilan zaman : 0.000000
N = 1000 K= 8 icin kullanilan zaman : 0.000000
N = 1000 K= 9 icin kullanilan zaman : 0.000000
N = 1000 K= 10 icin kullanilan zaman : 0.001000
```

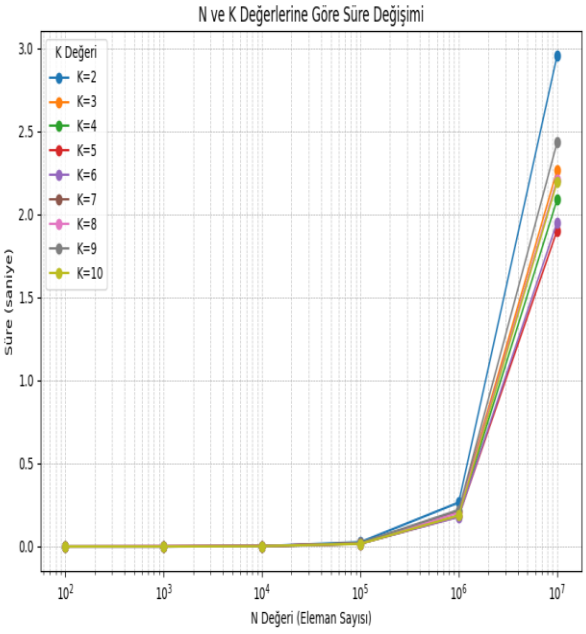
Burada görmüş olduğumuz üzere terminal çıktıları veriliyor bu çıktılar ise array1, array2, … array10 a kadar gidiyor 10 farklı dizi için çıktı veriliyor.

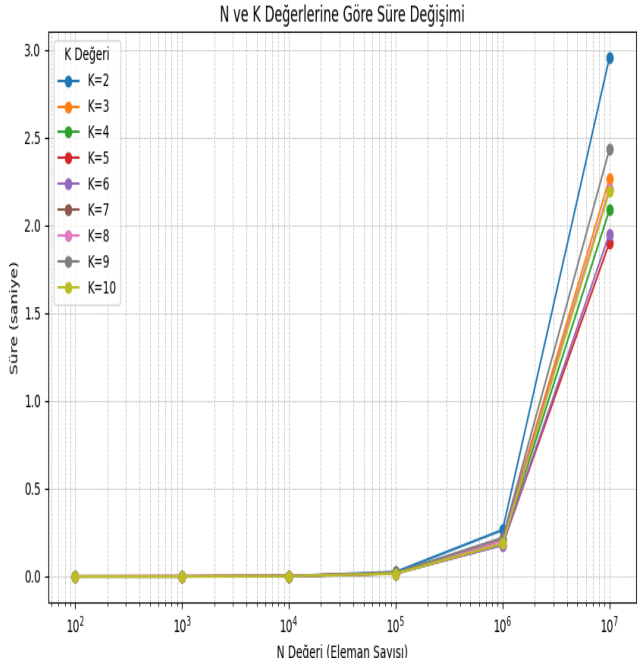


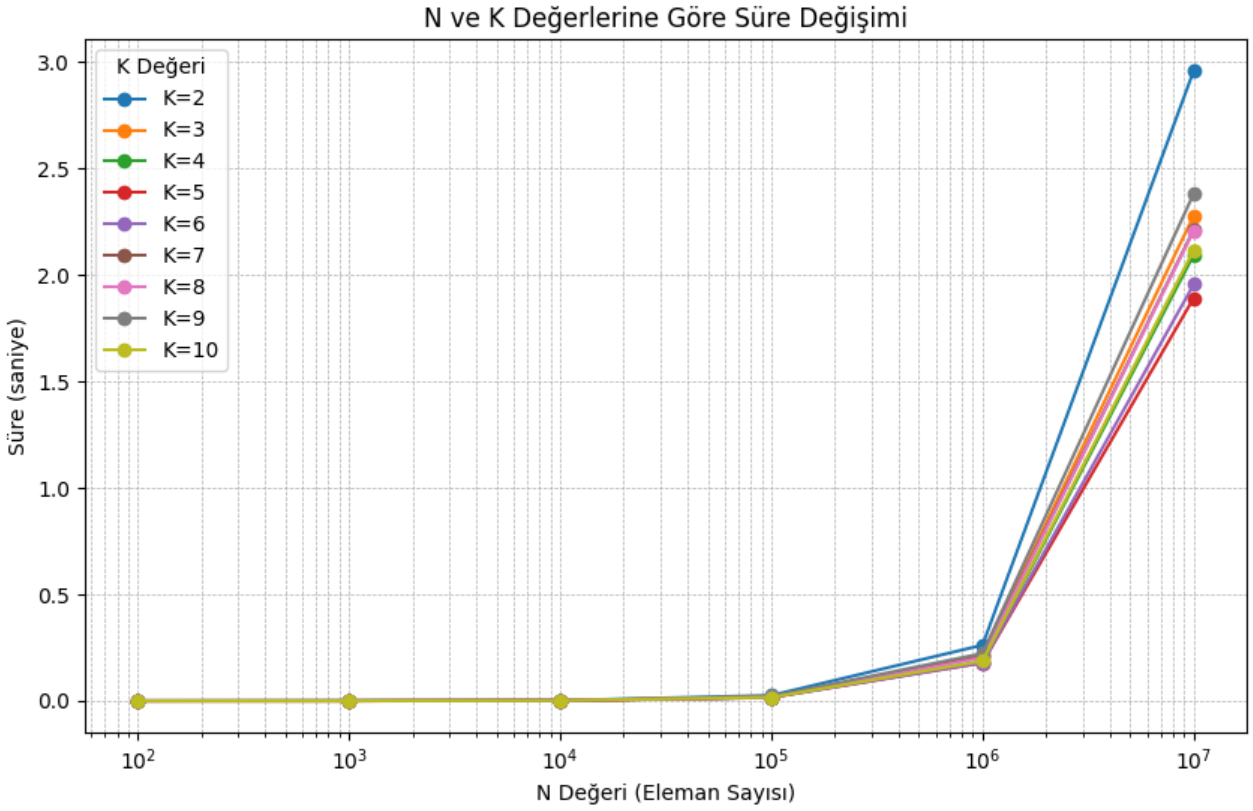
Burada görüldüğü üzere tüm çıktılar “output.txt” dosyasına çıktı veriyor buda verilerin kalıcılığı açısından atılmış bir adım.

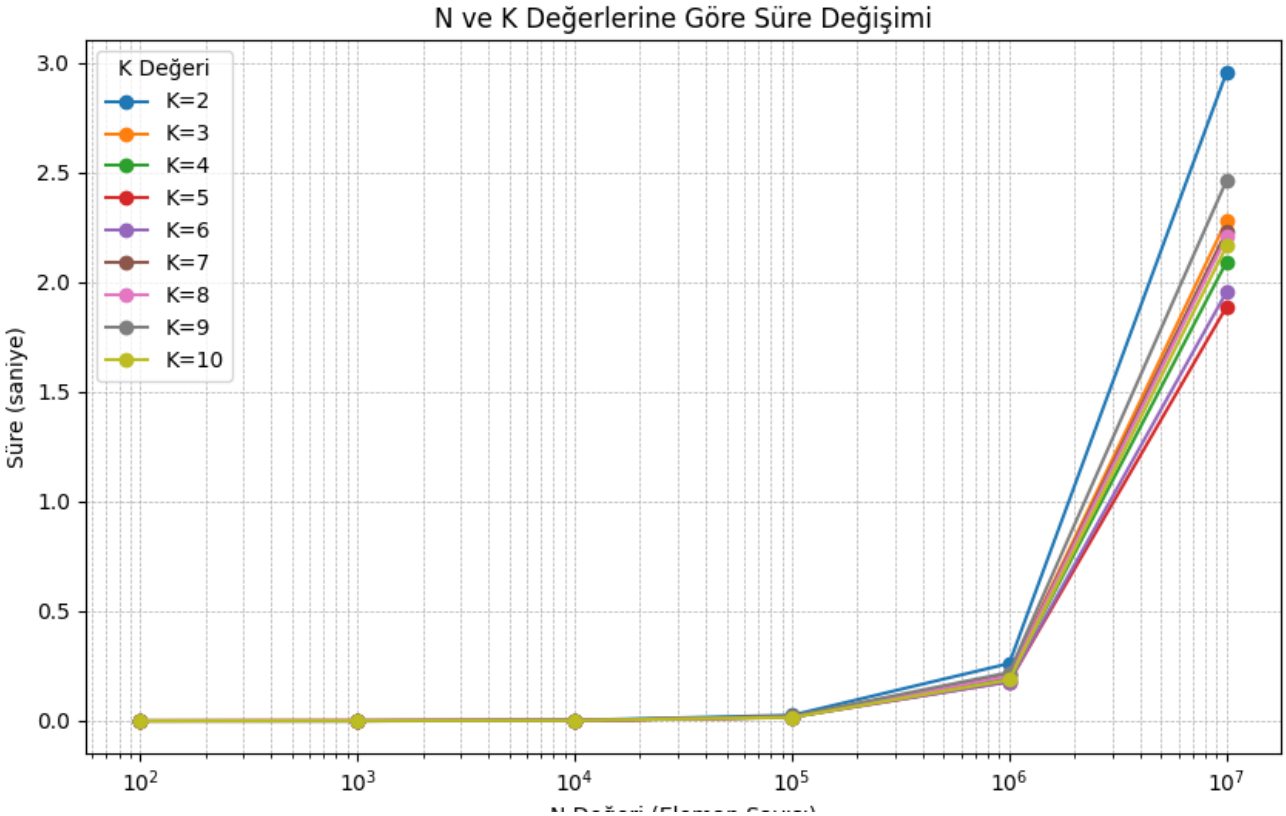
```python
data10 = {
    "N": [100, 1000, 10000, 100000, 1000000, 10000000],
    "K2": [0.0, 0.0, 0.002, 0.025, 0.261, 2.914],
    "K3": [0.0, 0.0, 0.002, 0.019, 0.206, 2.260],
    "K4": [0.0, 0.0, 0.002, 0.018, 0.183, 2.083],
    "K5": [0.0, 0.0, 0.001, 0.017, 0.179, 1.873],
    "K6": [0.0, 0.0, 0.002, 0.017, 0.187, 1.938],
    "K7": [0.0, 0.0, 0.001, 0.016, 0.208, 2.200],
    "K8": [0.0, 0.0, 0.002, 0.018, 0.219, 2.280],
    "K9": [0.0, 0.0, 0.002, 0.021, 0.222, 2.403],
    "K10": [0.0, 0.001, 0.001, 0.017, 0.187, 2.123]
}

# DataFrame oluşturma
df = pd.DataFrame(data)

# Çizim
plt.figure(figsize=(10, 6))
for k in range(2, 11):
    plt.plot(df["N"], df[f"K{k}"], marker="o", label=f"K={k}")

# Grafik ayarları
plt.xscale("log")  # N değerlerini logaritmik ölçekle göster
plt.xlabel("N Değeri (Eleman Sayısı)")
plt.ylabel("Süre (saniye)")
plt.title("N ve K Değerlerine Göre Süre Değişimi")
plt.legend(title="K Değeri")
plt.grid(True, which="both", linestyle="--", linewidth=0.5)

# Grafiği gösterme
plt.show()
```

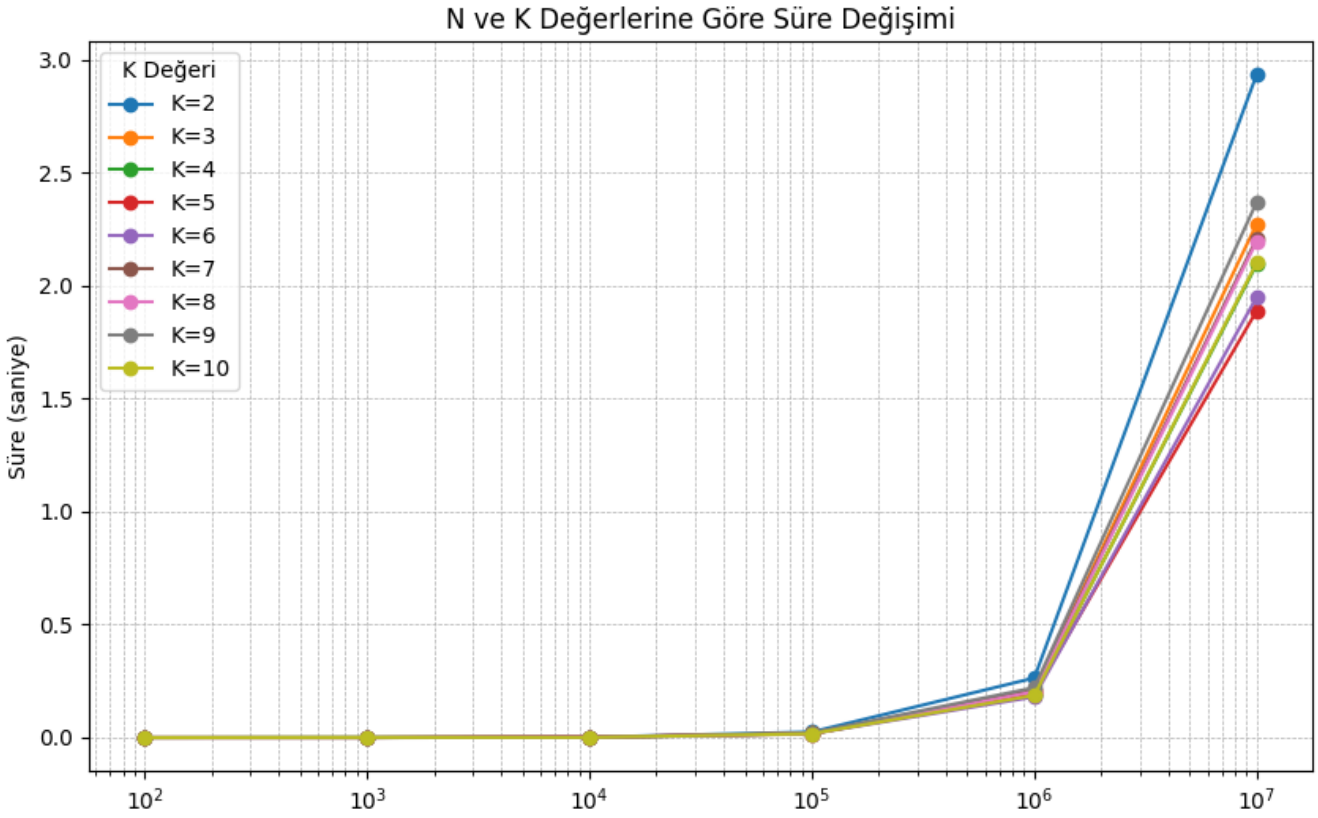
Burası Python kodları üstte çıktıların aynısı olan 10 adet datasetim mevcut aşağıda ise bu kodun bize sunduğu grafikler mevcut.

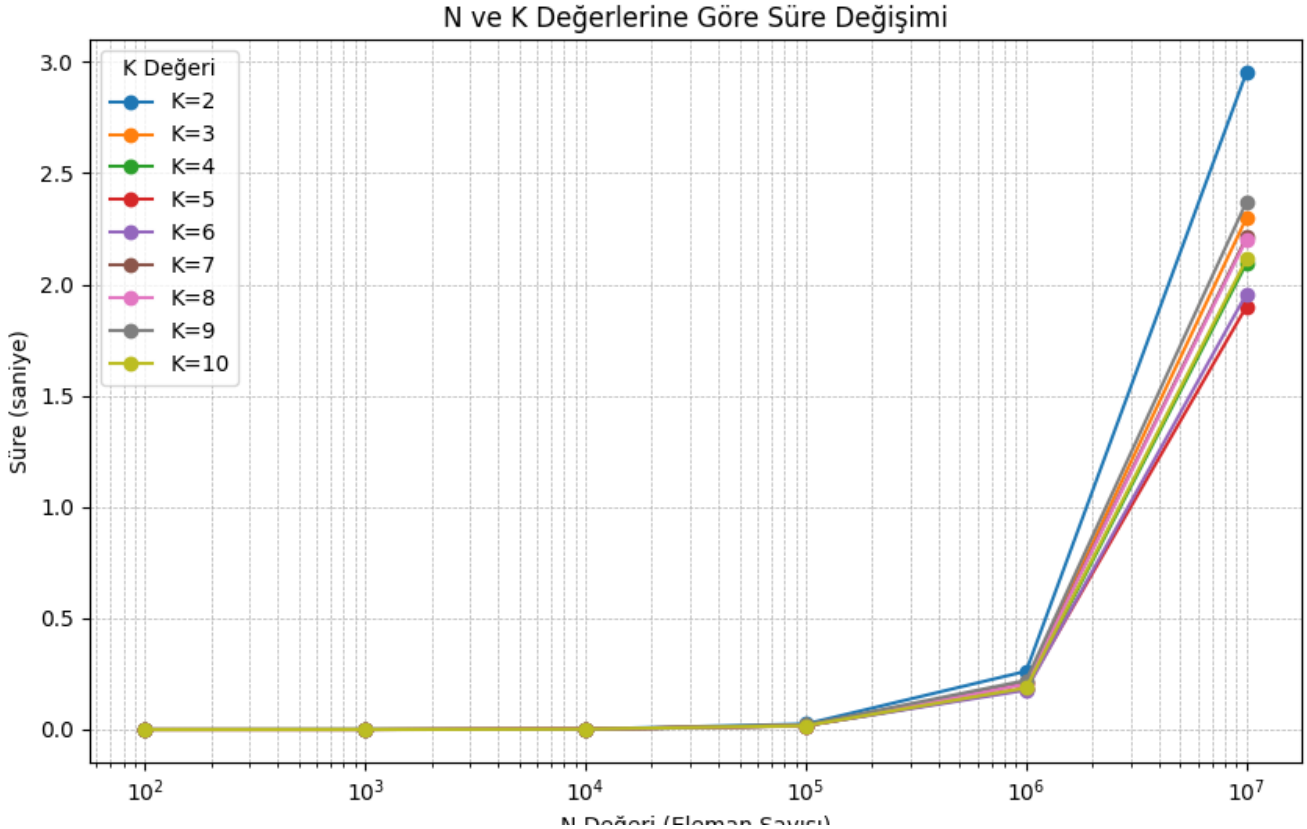


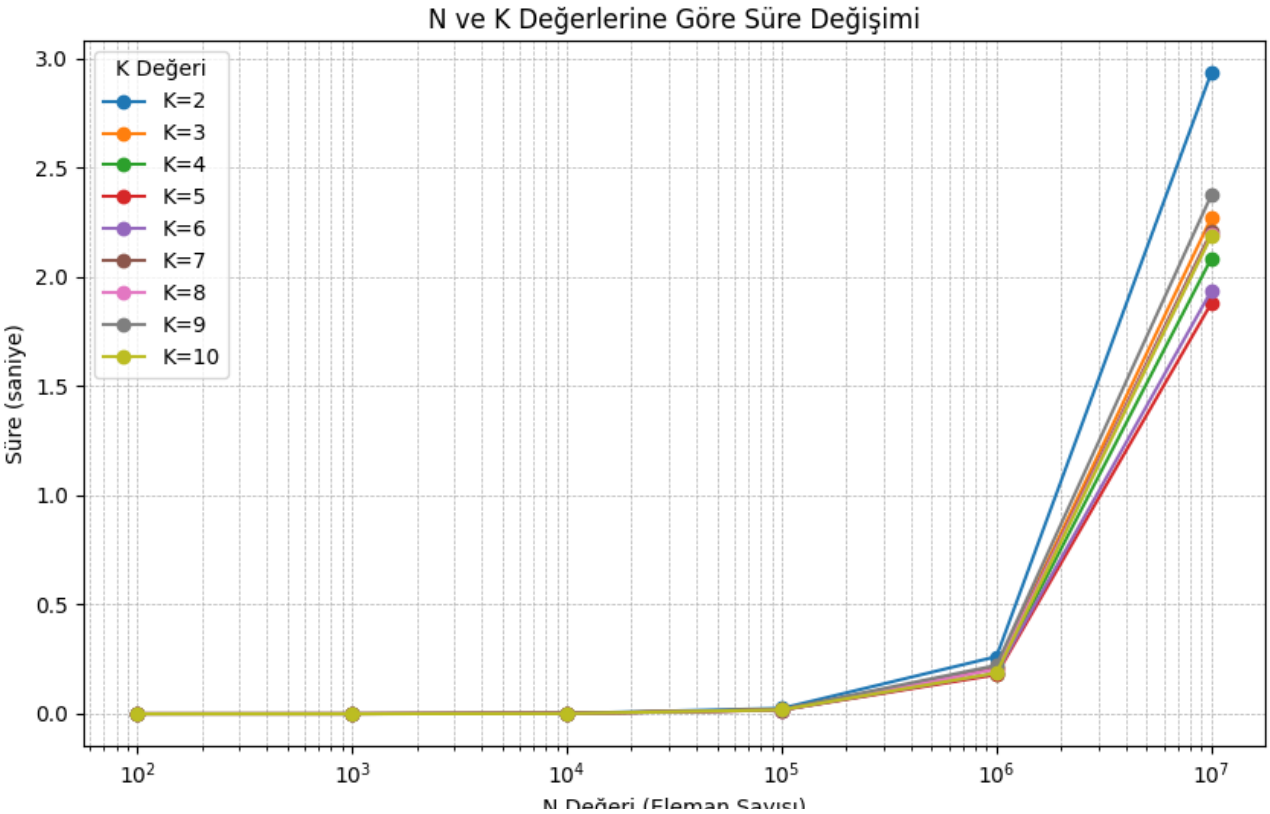


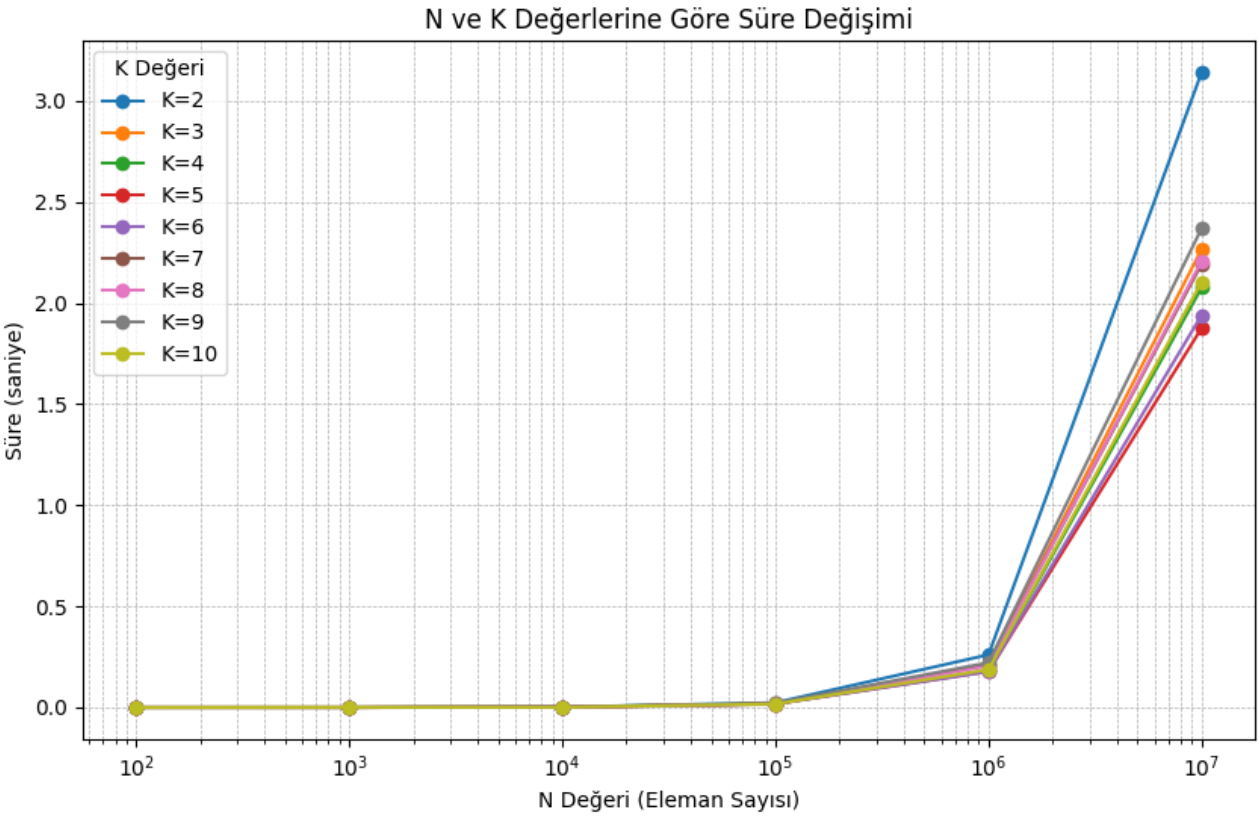

N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)
N Değeri (Eleman Sayısı)

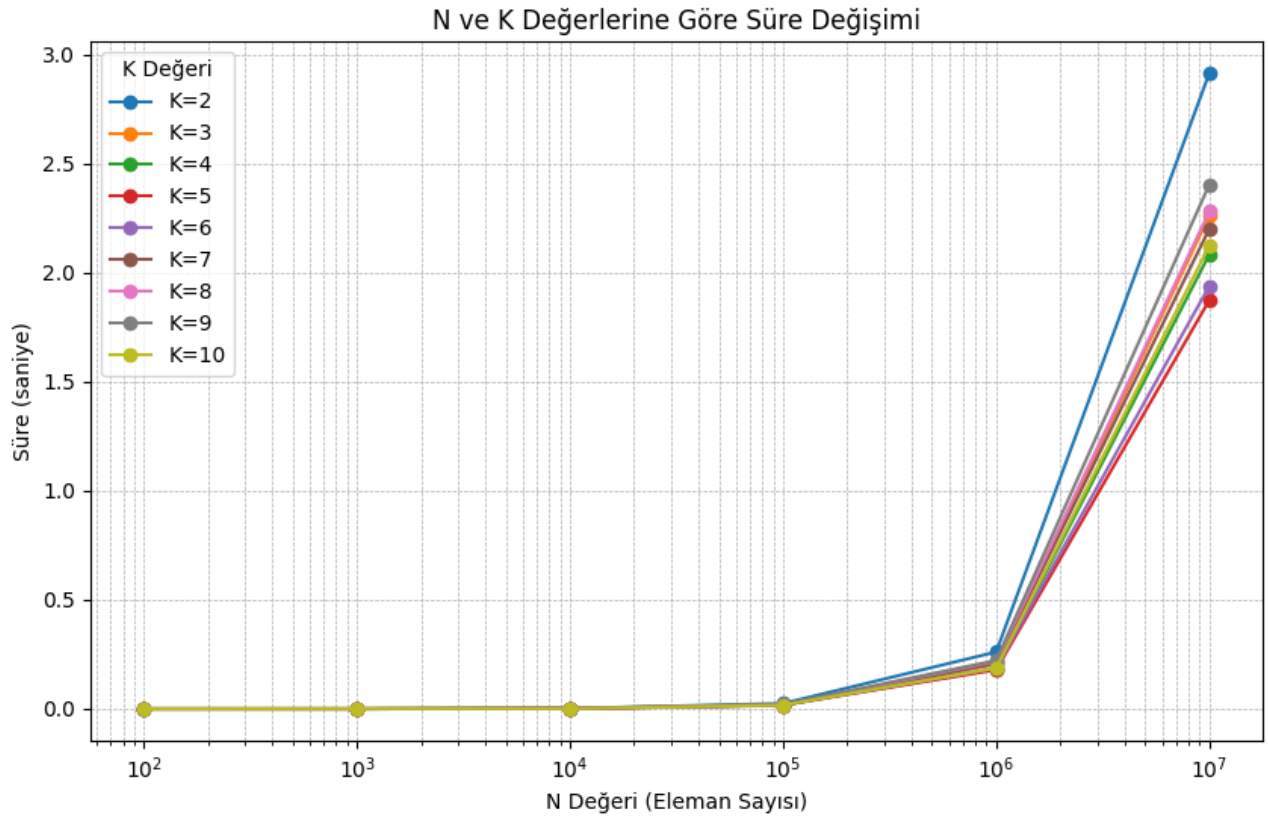

N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)

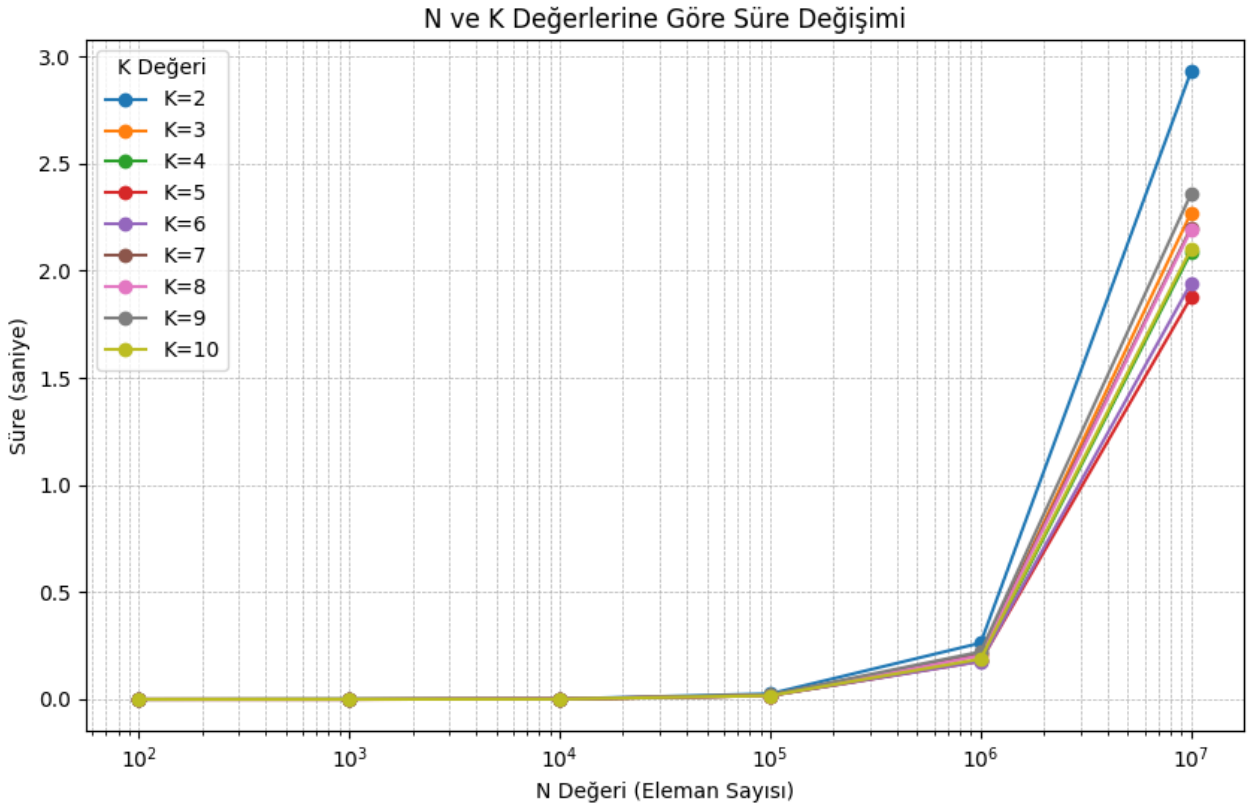

N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10^2
10^3
10^4
10^5
10^6
10^7


N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10^2
10^3
10^4
10^5
10^6
10^7

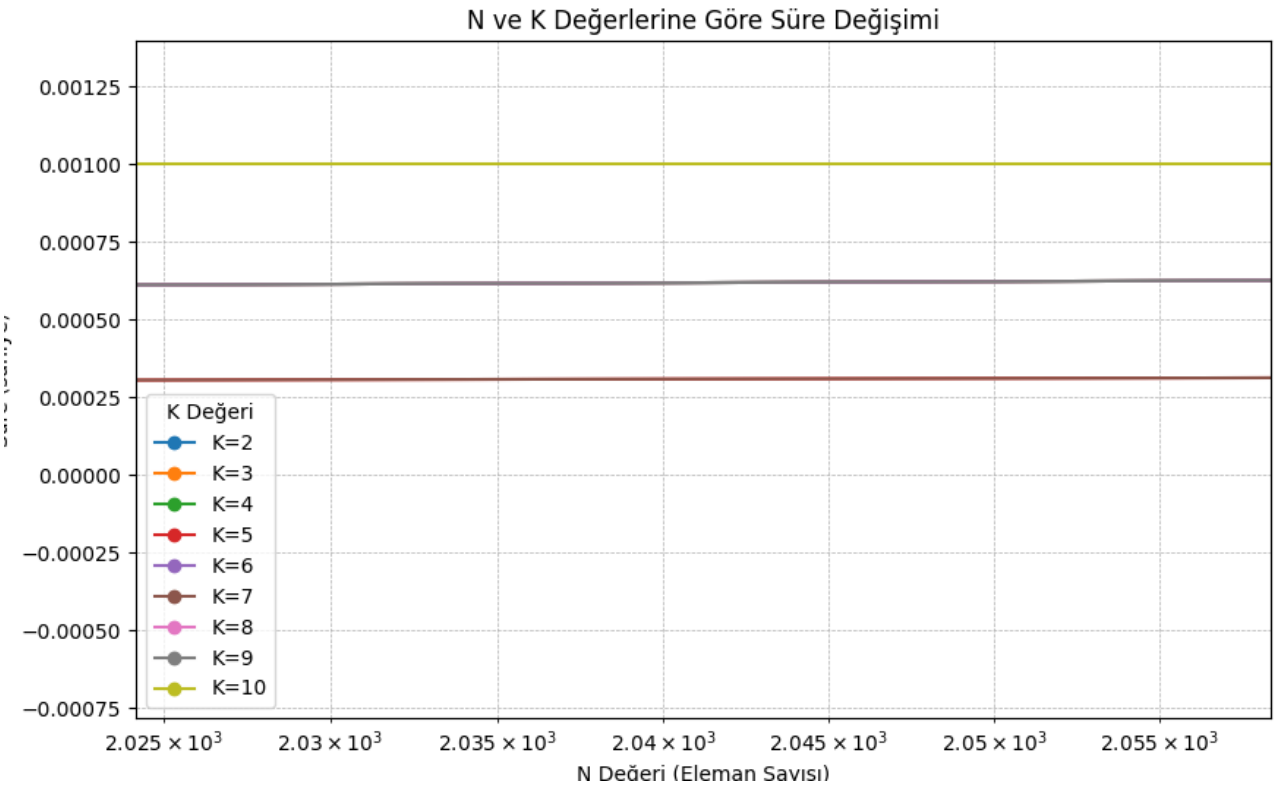

N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10^2
10^3
10^4
10^5
10^6
10^7
N Değeri (Eleman Sayısı)


N ve K Değerlerine Göre Süre Değişimi
K Değeri
K=2
K=3
K=4
K=5
K=6
K=7
K=8
K=9
K=10
Süre (saniye)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10^2
10^3
10^4
10^5
10^6
10^7
N Değeri (Eleman Sayısı)





Sırasıyla da 10 farklı dizinin çıktıları yukarıda verildi buradan küçük N değerleri için önemsenecek bir fark olmadığını ama N değerleri büyüdükçe K değerlerinin hız üzerinde ki etkisini görebilmekteyiz ama bu tamamen doğrusal bir şekilde değil yani K=10 en hızlı sıralama dır diyemeyiz karmaşıklık hesabına göre K değerleri hızı farklı şekillerde etkiliyor.



Çok fazla zoom yapıldığı vakit farklı dizilerin sıralanması aşamasında çok ufak derecede önemsenmeyecek farklar olduğunu görebiliriz küçük N değerlerinde farklı K değerleri için ama N değeri büyüdükçe K değerlerinin farkı fazlalaşmaya ve önemsenecek düzeye geliyor.