**Son Teslim tarihi: 11.11.2024**

# Ödev #2

OĞUZHAN TOPALOĞLU
Ç19052025 – Grup 1

*Bilgisayar Mühendisliği Bölümü,*
*Elektrik-Elektronik Fakültesi,*
*Yıldız Teknik Üniversitesi*


Istanbul, 2024

**Video Linki: https://youtu.be/-0aRKnWz5bM**

# İÇERİK

**1. Problemin Çözümü**

Çözümümde büyük bir diziyi daha küçük parçalara bölüp bu parçaları ayrı ayrı sıraladım ve ardından bu parçaları birleştirdim yani merge ettim. Bu çözüm, k-way merge sort olarak bilinen algoritmanın bir implementasyonudur.

Çözümümü psödö kod olarak aşağıya yazıyorum:

```
function mergeSort(arr, size, k, result):
    if size <= 1:
        result[0] = arr[0] if size == 1
        return

    mid = (size + k - 1) / k
    create arrays `subarrs` and `sizes` of length k

    for i = 0 to k - 1:
        start = i * mid
        end = min(start + mid, size)
        sizes[i] = end - start
        subarrs[i] = copy of arr[start to end]
        mergeSort(subarrs[i], sizes[i], k, subarrs[i])
    mergeKSortedArrays(subarrs, sizes, k, result, size)
```

## 2. Karşılaşılan Sorunlar

İmplementasyonum sırasında en çok runBenchmark() fonksiyonunu yazarken zorlandım. PDF'te özellikle aynı dizilerin k değerleri için kullanılması gerektiğini demesi biraz kafamı karıştırdı. Kaç tane döngü olacağını ve hangi döngünün içte hangisinin dışta olacağını pek anlamamıştım. Sonradan dış döngünün k=[2,10] ve iç döngünün HOW_MANY sefer olması gerektiğini anladım ve kodu yazdım.

Burada HOW_MANY sayısı 12 olarak seçtiğim 10'dan büyük bir sayıdır. Her (N,k) ikilisi için bir zaman metriği hesaplarken 12 kere test yapıp bunların ortalamasını alıyorum.

### 3. Karmaşıklık Analizi

mergeSort fonksiyonu verilen diziyi her seferinde k alt diziye bölerek rekürsif olarak kendini çağırıyor. size boyutundaki diziyi, her bölmede yaklaşık size / k uzunluğunda k alt parçaya ayırıyor. Her alt parça için mergeSort aynı işlemi uyguluyor, böylece her rekürsif adımda k alt dizi elde ediyoruz. Rekürsif adımların her birinin maliyeti O(N) ile ifade edilebilir, çünkü mergeKSortedArrays fonksiyonunu her bölme işleminin sonunda çağırıyoruz.

Her bir mergeSort çağrısında, k yeni alt diziye bölme işlemi yapıldığından, fonksiyonun zaman karmaşıklığı T(N) = k * T(N/k) + O(N) olarak yazılır. Master Teoremi ile bunu çözersek de T(N) = O(N log_k N) buluruz, yani çözüm nlogn büyüklüğündedir.

mergeKSortedArrays için analiz (gerekiyor mu bilmiyorum):

Bu döngü toplamda totalSize = N eleman kadar sürer, çünkü mergeKSortedArrays fonksiyonu her bir index değeri için bir eleman ekleyerek result dizisini sıralı hale getirir. Bu yüzden index < totalSize döngüsünün kendisi O(N) adımda tamamlanacaktır.

for döngüsü her index adımında k sıralı dizi arasında en küçük elemanı bulmak için çalışır. Bu for döngüsünün maliyeti O(k)'dir çünkü her index adımında k dizinin başındaki en küçük elemanı ararız. Bu O(k) maliyeti, index döngüsü boyunca N kez tekrarlandığı için toplam maliyet O(N * k) olur.

# 4. Ekran Görüntüleri