Algoritma Analizi Ödev 2

Öğrenci Adı: Mehmet Emin Aydin
Öğrenci No: 20011011
Dersin Eğitmeni: Mine Elif Karslıgil
Video Linki: https://youtu.be/D2G83FbItF4

Soru 1)

Herbir ikili eleman sorguları oluşturana kadar kişileri n/2 iterasyonla ayırırız. En sonda ikili sorgularda birbirlerini doğrulayan kişiler ya kesin doğru söylüyordur ya da kesin yalan söylüyordur. Bu ikili çiftleri birbirleriyle rekürans bağıntıyla combine ederken diğer ikili bağıntıların ilk elemanından itibaren soldaki ikilinin ilk elemanıyla sorgulatırız. Birbirlerini doğrularlarsa sağa atmaya devam ederiz. Tüm bağıntılardan geri dönüldüğünde dizinin en sağından itibaren doğru söyleyen kişiler yan yana dizilmiş olur.

Notasyonla göstererek anlatırsak;

ABCDEFGH kişileri olsun. X in Y için ifadesi Xy(ifade) olarak gösterilirse;

X:çaldı; Y:çalmadı.

Gerçek verilere hakim olduğumuzu varsayarak ABCEF nin doğru söylediğini biliyor olalım.

ABCD		EFGH	
AB	CD	EF	GH
Ab(Y),Ba(Y)	Cd(X),Dc(X)	Ef(Y),Fe(Y)	Gh(Y),Hg(Y)

A ve C için

Ac(Y),Ca(Y) olacağından

DCAB

E ve G için

Eg(X),Ge(X) olacağından

EF ve GH çiftlerinden biri hırsız diğeri doğru

Çoğunluk doğru olduğuna ve D ve CAB ayrı görüşte olduğuna göre CAB nin onayladığı ikili sağa kayacak ve tamamlanmış olacak.

C ve E için

Ce(Y),Ec(Y) olacağından

DGHCABEF olarak birleştirme işlemi tamamlanmış olur ve en sağdan H ye kadar olan kişilerin doğru söylediği bilinir.

0 dan n/2 ye ve n/2+1den n e kadar olan iki taraf da ayrı ayrı rekürsif ilerlediğinden 2 adet rekürans ve en sonda elemanları birleştirirken eleman sayısı kadar karşılaştırma yaptığımızdan

$T(n) = 2*T(n/2)+c*n$ olacağından $n*\log n$ olarak ifade edebiliriz.

Soru 2)
1 - Problem Tanımı

Bu problemde amaç kendi içinde birbirinden farklı boyutta elemanlar içeren ve diğer dizide bir eşi olan ‘kilit’ ve ‘anahtar’ dizilerindeki kilit ve anahtarları kendi içlerinde kıyaslamadan sıralamak ve eşleştirmektir. Bunu en efektif şekilde yapmak için Quick Sort yaklaşımı kullanılmalıdır.

2 - Problemin Çözümü

1. Rastgele Pivota Dayalı Eşleme (matchPairs Fonksiyonu):
   - Her adımda bir rastgele pivot eleman seçilir.
   - Seçilen pivot eleman kullanılarak "lock" dizisi ikiye bölünür.
   - "key" dizisi de pivot elemanı kullanılarak bölünür.
   - Bu işlem, her seçilen rastgele pivot eleman için tekrarlanır.

2. Partition Fonksiyonu:
   - Belirlenen pivot elemanına göre "lock" dizisinde, pivottan küçük elemanlar sol tarafta ve pivottan büyük olanlar pivotun sağ tarafından olacak şekilde sıralanır.
   - Aynı işlem "key" dizisi için de gerçekleştirilir.

3. Recursive Yaklaşım:
   - Pivotun solunda ve sağında olan kümeler için random pivot seçimiyle bu işlemler altkümelerde de tekrarlanır.

3 - Rekürans Bağıntısı ve Karmaşıklık Analizi:

Rastgele bir anahtar seçilir ve kilit dizisinde eşi aranmak üzere dolaşılmaya başlanır. Bu yapılırken bir yandan partitioning yani pivottan küçüklerin sola, büyüklerin sağa atılması işlemi yapılır. Eşi bulunduktan sonra ve partitioning tamamlandıktan sonra bu pivot değeri kullanılarak aynı işlem anahtar dizisine uygulanır. Bu iki işlem toplamda 2n-2

karşılaştırmada tamamlanır. Sonrasında aynı işlemi ortalama olarak n/2 eleman içeren pivotlarla ayrılmış anahtar ve kilit dizilerine tekrar uygularız. Sonuç olarak aşağıdaki rekürans bağıntısını elde ederiz.

```
QuickSort(Int Array Lock[], Int Array Key[], Int pivot)
        if low<high
                random      <— low + rand() % (high - low)
                pivot       <— Partition(Lock[low…high], Key[random])
                Partition(Key[low…high], Lock[Pivot])
                QuickSort(Lock[low…pivot-1], Key[low…pivot-1])
                QuickSort(Lock[pivot+1…high], Key[pivot+1…high])

Partition(Int Arrray Arr[], Int low, Int high, Int pivot)
        Int i <— low
        for j from low to high
                if Arr[j] < pivot
                        CALL SWAP with Arr[i], Arr[j]   // dizinin 2 elemanını değiştiren
                                                        //fonksiyon
                        i++
                else if Arr[j] == pivot
                        CALL SWAP with Arr[j], Arr[high]
                        j--
        ENDFOR
        CALL SWAP with Arr[i], Arr[high]
        return i
```

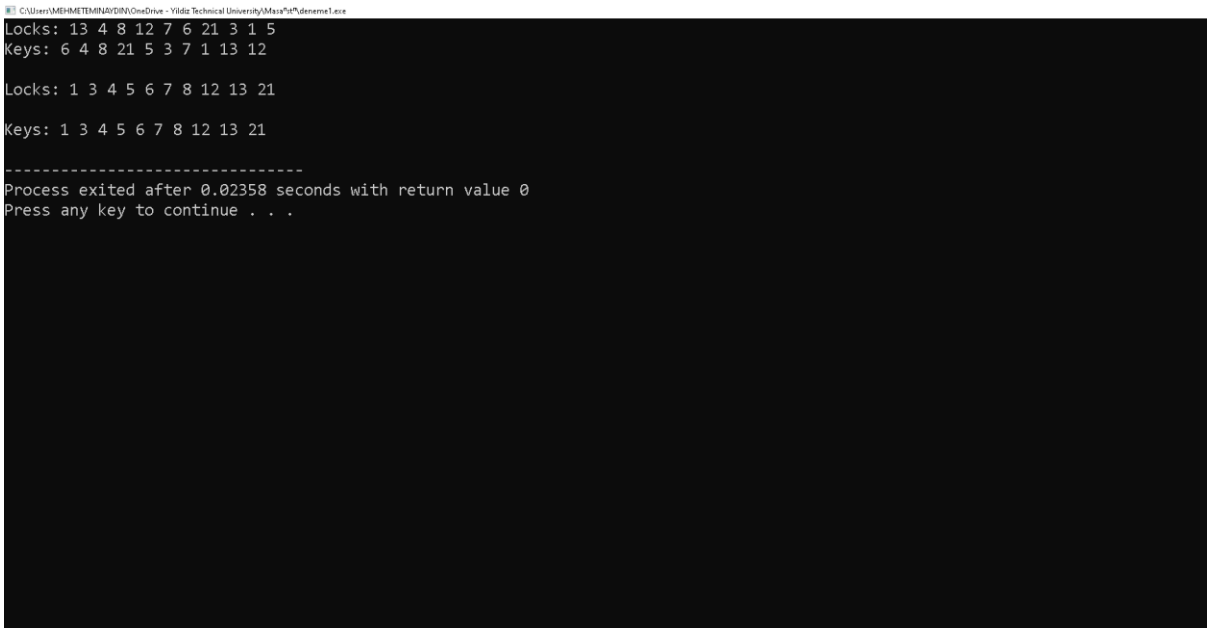
$$T(n) = 2T(\frac{n}{2}) + 2n$$

Master Teorem'den faydalanılarak bu işlemin zamansal karmaşıklığı T(n) = O(n logn) olarak elde edilir.

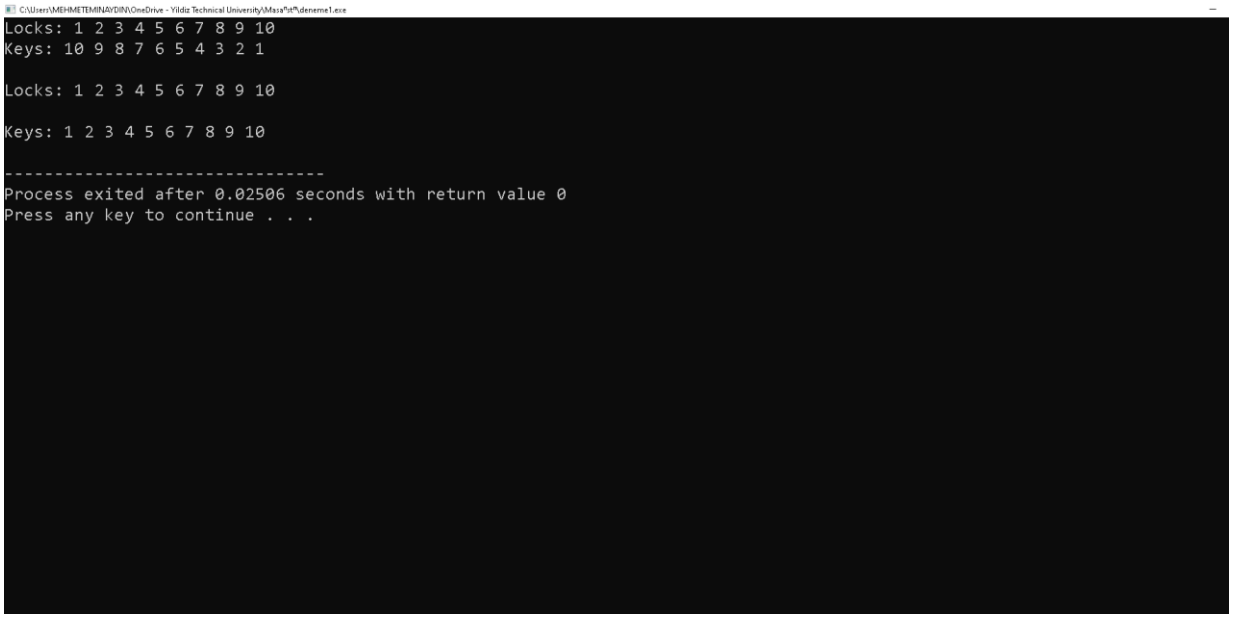
a=2 b=2 c=1 → a=b∧d → T(n) = O(n logn)

4 - Ekran Çıktıları:

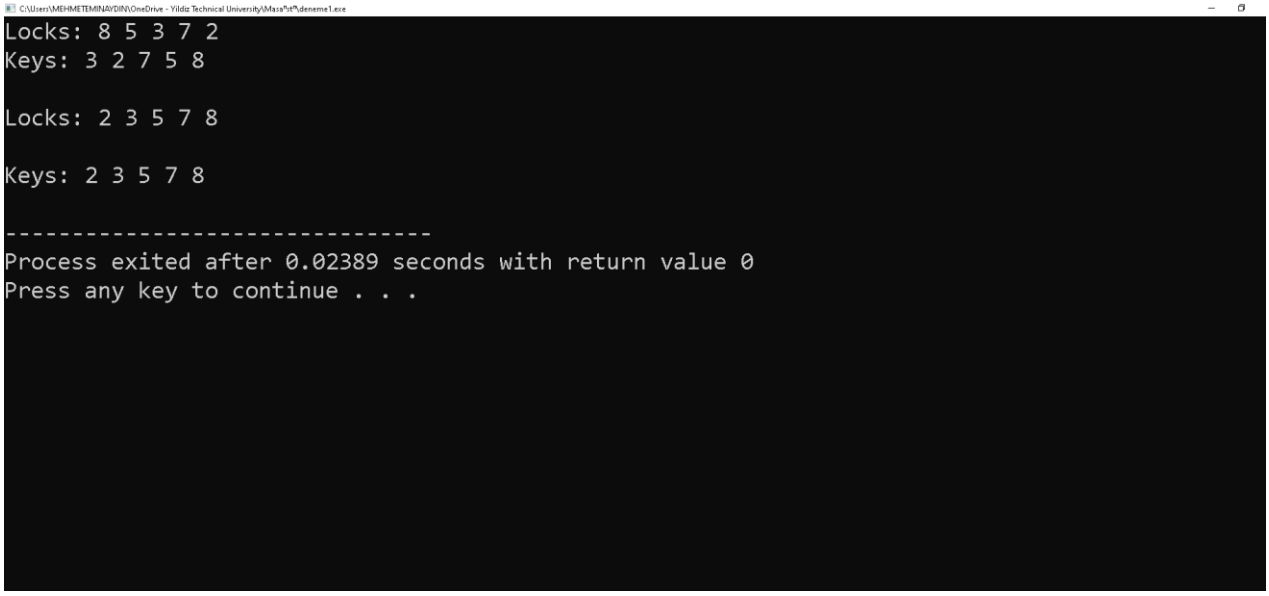
Senaryo 1-



Senaryo 2-



Senaryo 3-



Senaryo 4-