BLM 3021 Algoritma Analizi

Algoritma Analizi 2023-2024 Güz Dönemi

2. Ödevi

VIDEO LINKI

https://youtu.be/ZJHyj7Icjdc?si=5FsGYLh7tJsa9M-i

Ders Yürütücüsü: M. Elif KARSLIGIL

Ödevi Yapan: Berkay ATES

Ders Grubu: 1. Grup

No:21011609

berkay.ates1@std.yildiz.edu.tr

6.11.2023

SORU 1

Problemin Tanımı

Elimizde bulunan bir toplulukta doğru veya yanış beyanlarda bulunan insanlar var, bu insanların yarısından fazlası her durumda doğru olanı söylemekte . Bu grup içerisindeki insanların doğru söyleyenleri hem suçsuz hemde karşısındaki hakkında doğru cevaplar veriyor, yani karşıdaki hırsız ise karşısındakine hırsız diyor. Yalancı olanlar ise karşısındakini hem doğrulayabilir hem de yalanlayabilir ayrıca kendi hakkında da yalan söylüyor.

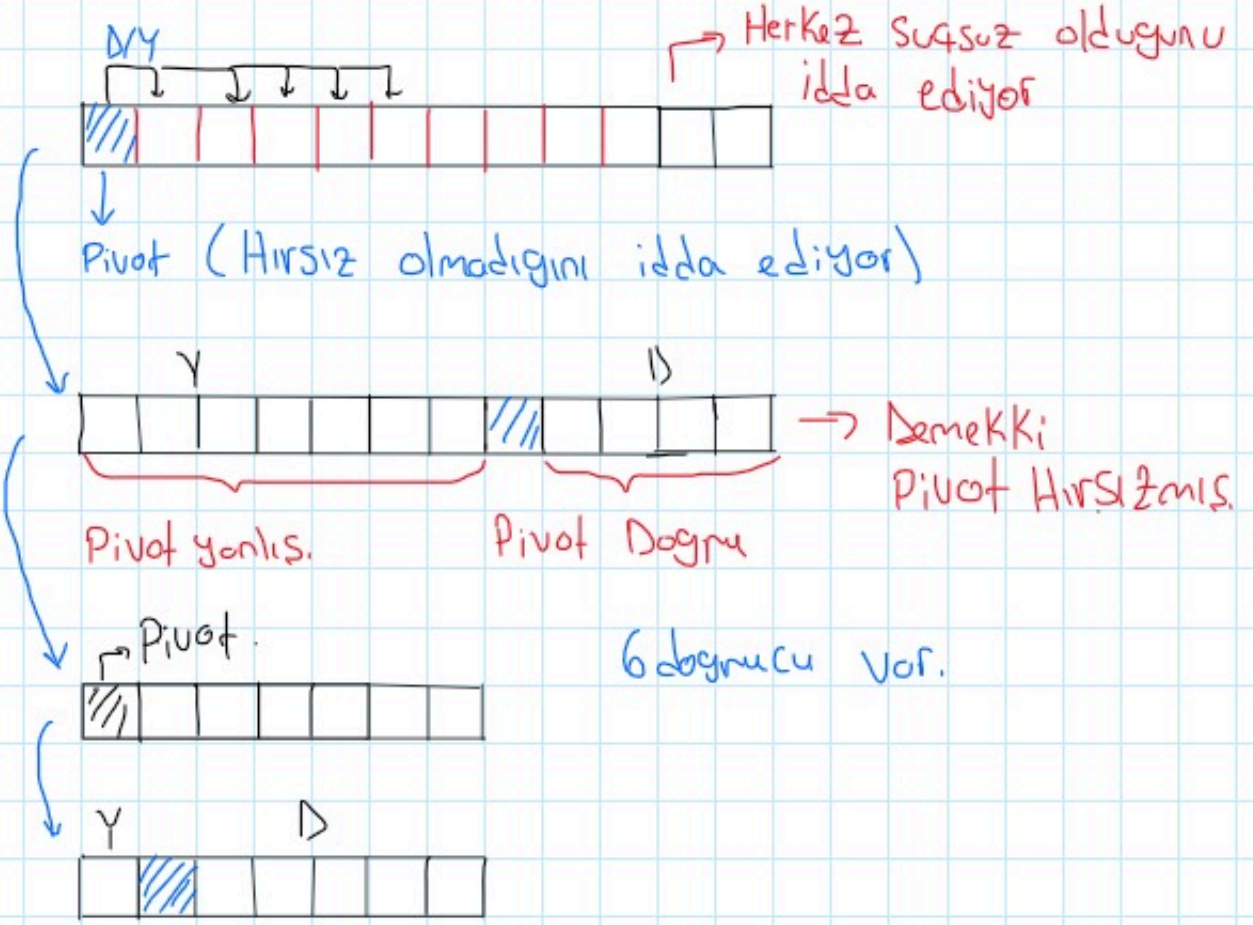
Problemin Çözümü

1 -Gruptan biri rastgele olarak pivot olarak seçilir

2- Gruptaki diğer kişilere pivot eleman “doğru mu söylüyor ?” Diye sorulur. Bu adım sonunda pivotun sağında pivot doğru söylüyor diyenler, solunda ise pivot yalan söylüyor diyenleri toplarız.

3 - Eğer pivot doğru söylüyor diyenlerin sayısı yarıdan fazla ise demek ki pivot doğru söylüyordur. Eğer çoğunluk pivot yalan söylüyor tarafında ise, bu defa pivot yalan söylüyor diyenlerin tarafından birini seçip aynı işlemi tekrarlarız.

✡ Eger grubun yarisi pivota yalanci diyorsa secilen pivot yalanci oldugu için, bu defa pivota yalanci diyenlerin arasından birini seçer ve aynı işlemi tekrarlarız.



✡ Pivotu doğrulayanların sayısı grup sayısının yarısına eşit veya fazla olduğu için pivotun kesin doğru söylüyor olduğundan emin olabiliriz.

```
dogruyuBul( A[L .... R])
  if(R-L >= n/2)
     P = Sorgula( A[L .... R] )
     iff (P > (L+R)\2)
       return dogruyuBul(A[L,.... P])
     else
       return P;
```

Recurance Bagintisi

$$\begin{cases} T(n) = T(n/2) + n \\ T(2^k) = T(2^{k-1}) + 2^k \end{cases}, \quad n = 2^k, \quad T(0) = 1$$

$$= T(2^{k-2}) + 2^k + 2^{k-1}$$

$$= T(2^{k-3}) + 2^k + 2^{k-1} + 2^{k-2}$$

$$\vdots$$

$$= T(2^{k-k}) + 2^k + 2^{k-1} + \cdots\cdots 2^1$$

$$= 2^0 + 2^1 + 2^2 + 2^3 + \cdots\cdots 2^k$$

$$2^{k+1} - 1$$

$$2\,2^k - 1$$

$$\boxed{2n-1}$$

Soru2

Problemin Tanımı

Elimizde birbirleri ile uyuşan fakat karşılıklı olarak farklı noktalarda bulunan anahtar ve kilitler var. Bizden bu anahtar ve kilitleri birbirleri işe eşleştirmemiz isteniyor. Uygulanan kısıt ise aynı dizi içerisindeki elemanları birbirleri ile karşılaştıramamız.

Problemin Çözümü

1 - Birinci gruptan herhangi bir eleman pivot olarak seçeriz ve bu pivotla diğer diziyi QUICK SORT mantıgını kullanarak sıralarız.

2- İkinci diziyi de en başta seçtiğimiz pivotu kullanarak yine QUICK SORT mantığı ile sıralarız.

3 - Ilk defa sıraladığımızda her iki dizi için de aynı pivot elemanını sectığimiz icin diziler birbirine yakın oranlarda sıralanmış olacak ve sonrasında da bu işlem tekrar edeceği icin iki dizi de birbirine göre aynı anda sıralanacak. Yani bir dizinin sıralama işlemi gittiğinde digeri ya çoktan sıralanmış yada daha yeni sıralanmış halde olacak.

Ekran Çıktıları

```
[Running] cd "c:\Users\atesb\Desktop\HW2_21011609\" && gcc
Keys and Locks before matching them :
Locks: 42 93 31 71 2 24 86 68 39 56
Keys: 93 24 31 56 42 86 2 39 71 68

Matched keys and locks are :
Locks: 2 24 31 39 42 56 68 71 86 93
Keys: 2 24 31 39 42 56 68 71 86 93

[Done] exited with code=0 in 2.379 seconds
```

```
[Running] cd "c:\Users\atesb\Desktop\HW2_21011609\" &&
Keys and Locks before matching them :
Locks: 42 93 31 71 2 24 86 68 39 56
Keys: 2 24 31 39 42 56 68 71 86 93

Matched keys and locks are :
Locks: 2 24 31 39 42 56 68 71 86 93
Keys: 2 24 31 39 42 56 68 71 86 93

[Done] exited with code=0 in 2.258 seconds
```

```
[Running] cd "c:\Users\atesb\Desktop\HW2_21011609\" && gcc 21011
Keys and Locks before matching them :
Locks: 93 86 71 68 56 42 39 31 24 2
Keys: 2 24 31 39 42 56 68 71 86 93

Matched keys and locks are :
Locks: 2 24 31 39 42 56 68 71 86 93
Keys: 2 24 31 39 42 56 68 71 86 93

[Done] exited with code=0 in 1.834 seconds
```

Recurrance Çözümü

$aT(n/b) + d \rightarrow f(n) = n^{d}$ (initial condition)

a = Çözülmesi gereken alt problem sayısı

b = Problemin kaç parçaya bölündüğü

d = initial condition

Master Theorme göre

$n^d \log n \leftarrow a = b^d \rightarrow 2 = 2^1$ olduğu için

$= n^1 \log n$

#KAZANIMLAR

1) Quick Sort ile alıştırma yaparak nasıl caşıldığını daha iyi anladım.

2) Recursive fonksiyonlar üzerine olan yetkinliğimi arttırdım

3) Recurrance bağıntısı hesaplama üzerine tekrar yaptım

4) Bir algoritmanın oluşturulmasında veya bir problemin çözümünde divide and conquer yaklaşımın işleri ne kadar kolaylaştırabileceğini farkettim.