ALGORİTMA ANALİZİ 2. ÖDEV
BÖL VE YÖNET ALGORİTMALAR

Adı Soyadı: Büşra Medine GÜRAL

Öğrenci Numarası: 20011038

Mail: medine.gural@std.yildiz.edu.tr

Dersin Eğitmeni: M. Elif KARSLIGİL

Video Linki: -

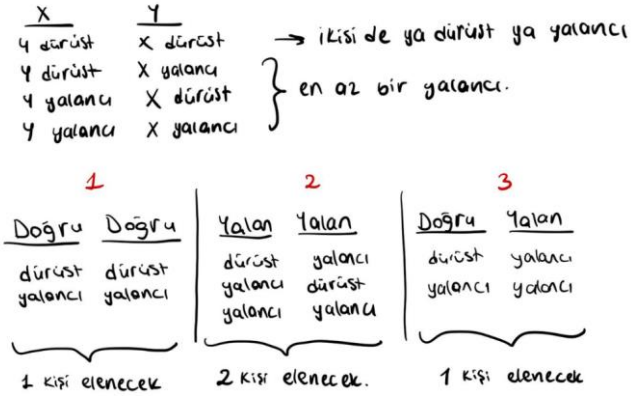
1. SORU

1.1 Problem Tanımı

Ödevin 1. sorusunda bir grup insan içerisindeki hırsız grubunu bulmak için ikişerli gruplara karşılıklı olarak bir soru sorulmaktadır. Bu soruya hırsız grubu hem doğru hem de yanlış cevap verebilir. '100 kişilik bir grupta doğru cevap veren bir kişi kaç adımda bulunabilir?' sorusuna cevap aranmaktadır.

1.2 Problem Çözümü

Problemin çözümünde doğru söyleyen kişilerin yalan söyleyenlerden fazla olduğu varsayılmakta ve aşağıdaki gibi birkaç senaryoyu değerlendirmek gerekmektedir.



İkili gruplandırmalarda bu şema rekürsif olarak gerçekleştirilerek her bir adımda kişi sayısı yarıya indirilmiş olur. Çözümün başında varsayılan 'dürüst sayısı> yalancı sayısı' ifadesi her bir adımda korunur. Nedeni şu şekildedir: 1. senaryoda ikisi de birbiri için 'doğru' ifadesini kullandığından bir kişi elenmelidir ve elenecek kişi ya dürüst ya da yalancıdır. 3. senaryoda elenecek olan kişi, kendisi için 'yalan' ifadesi kullanılmış olan kişi olmalıdır, haliyle sağ taraftaki kişi elenmektedir. 2. senaryoda iki kişi birbiri hakkında 'yalan' ifadesi verdiğinden ikisini de elemek mantıklıdır. Total duruma bakıldığında doğru söyleyenlerin sayısı hala yalancılardan fazladır. Bu yöntemle çözüldüğünde kişi sayısı 100 iken $log_2 100$ yaklaşık olarak 7'ye eşit olmaktadır. Haliyle 7 adımda doğru söyleyen bir kişiye ulaşılabilir. Ancak senaryolardaki kişilerin daha farklı dağılımları için adım sayısı 5'e kadar düşebilir.

1.3. Karşılaşılan Sorunlar

Soruda yalan söyleyen birinin hem yalan hem doğru söyleyebilme ihtimalinin olması zorluk çekilen en büyük kısımdı. Dürüst kişilerin sayısının her bir adımda yalancı sayısından fazla olma durumu keşfedilene kadar algoritma oluşturulamadı.

1.4. Karmaşıklık Analizi

Rekürans bağıntısı için T(n) = T(n/2)+ n/2 denilebilir. Her bir adımda dizi yarıya bölündüğünden ve bir tarafı ile ilgilenildiğinden T(n/2) kısmı, bu dizideki elemanlar karşılaştırıldığından n/2 kısmı oluşur. Master teorem ile çözülürse:

a=1, b=2 olur. $n^d = n$ ise d=1'dir. $b^d = 2$ ise $a < b^d$ olduğundan teoremin 1. şartı kabul edilir.

$\theta(n^d) = \theta(n)$

2. SORU

2.1 Problem Tanımı

Ödevin 2. kısmındaki problemde farklı büyüklükte N adet anahtarın, farklı büyüklükte N adet odayı açması gerekmektedir. Her bir kilidin sadece belli bir anahtara uygun olduğu bilinmektedir ve anahtarların boyutları kilitlerin boyutlarıyla karşılaştırılamamaktadır. Anahtarların hangi odalara uyduğunu bulmak için O(N*logN) karmaşıklığına sahip bir sıralama algoritması kullanılması gerekmektedir.

2.2 Problem Çözümü

Problemin çözümü için Quick Sort algoritması kullanılmaktadır. Bu algoritmada bir pivot eleman seçilir, daha sonra partition fonksiyonu ile pivot elemandan küçük olan elemanlar solunda, büyük olanlar ise sağında toplanır. Daha sonra pivot doğru yere yerleştirilir, ardından sol ve sağ taraftaki parçalar için aynı işlem tekrarlanır.

Soruda key ve lock eşleştirmesi yapılabilmesi için iki dizinin de quick sort ile sıralanması gerekmektedir. Bu sıralama için öncelikle ‘keys’ dizisi için ‘locks’ dizisinden rastgele bir pivot seçilmiş ve ‘keys’ dizisinin her bir elemanı pivot ile karşılaştırılmaya başlanmıştır. Dizinin başından sonuna ilerleyen ve dizinin sonundan başa doğru ilerleyen iki ayrı indis ile karşılaştırma sonucu oluşan ‘swap’ işlemi gerçekleştirilmiştir. Seçilen pivota eşit eleman ise dizinin en sağına atılarak karşılaştırma işlemine devam edilmiştir. Swap işlemleri tamamlandıktan sonra pivot doğru konuma alınarak kendisinden büyük elemanlar sağında, küçük elemanlar solunda kalacak şekilde dizi güncellenmiştir. Aynı işlem ‘keys’ dizisindeki pivotun konumundaki değer pivot alınarak ‘locks’ dizisi için yapılır. Sonrasında ise sıralama işlemi rekürsif olarak dizilerin sol ve sağ tarafta kalan parçaları üzerinde gerçekleşir.

Algoritma ve kodun genel mantığı için Algoritma Analizi Uygulama-1’deki son sorudan faydalanılmıştır.

2.3. Karşılaşılan Sorunlar

Tasarlanan ilk partition algoritmasında pivot olarak en sondaki değer seçilmekteydi. Pivotun rastgele seçilmesi koda sonradan eklendiğinden bazı case’ler için ‘sonsuz döngü’ problemi yaşandı. Bunu çözmek adına while döngülerine i ve j için eşitlik kontrolü eklendi.

2.4. Karmaşıklık Analizi

partition:
input: arr: the given array
low: start of the array
high: end of the array
pivot: element indicating the point at which the array will be divide
i = low
j = high
while i<=j do
while j>=0 and arr[j]>=pivot do
if arr[j]==pivot then
call swap(arr[j], arr[high])
end if
j= j-1
end while
while arr[i]<pivot do
i=i+1
end while
call swap(arr[i], arr[j])
end while

```
        call swap(arr[i], arr[j])
        call swap(arr[i], arr[high])
        return i
```

quickSort:

input: keys: array that contains keys

```
        locks: array that contains locks
        low: start of the array
        high: end of the array
        if low < high then
                randomPivot= random(low, high)
                pLoc = partition(keys, low, high, locks[randomPivot])
                partition(locks, low, high, keys[pLoc])
                quickSort(keys, locks, low, pLoc-1)
                quickSort(keys, locks, pLoc+1, high)
        end if
```

Rekürans bağıntısı eleman sayısı n olan bir dizi için T(n) = T(k) + T(n-k+1) + 2n olarak gösterilir. En iyi durum için pivot her zaman dizinin ortanca elemanı olur ve dizi sürekli n/2 şeklinde alt dizilere bölünür. Bu sebeple rekürans bağıntısı aşağıdaki gibi gösterilip backward substition yöntemi ile çözülebilir.

$T(n) = 2T(n/2) + 2n$

$T(1) = \emptyset$ Çünkü $n=1$ iken işlem yok.

$T(n/2) = 2T(n/4) + n$

$T(n/4) = 2T(n/8) + \frac{n}{2}$

$T(n) = 2\left[2T(n/4) + n\right] + 2n$

$= 4T(n/4) + 4n$

$= 4\left[2T(n/8) + \frac{n}{2}\right] + 4n$

$= 8T(n/8) + 6n$

$\vdots$

$T(n) = 2^i \cdot T(n/2^i) + 2n \cdot i$

$\frac{n}{2^i} = 1 \Rightarrow n = 2^i \Rightarrow \log_2 n = i$

$= \underbrace{2^{\log_2 n}}_{n} \cdot \underbrace{T(1)}_{0} + 2n \cdot \log_2 n$

$= 2n \cdot \log_2 n \in O(n \log n)$

Klasik bir Quick Sort algoritması için partition işlemi bir kere gerçekleştiğinden T(n) fonksiyonu 2T(n/2)+n şeklinde yazılır ve n, partition işleminin karmaşıklığını gösterir. Ancak sorunun çözümü için kullanılan algoritmada partition işlemi 2 kere yapılır, haliyle T(n) fonksiyonuna 2n eklenir. Karmaşıklık Big-Oh notasyonunda gösterildiğinden işlem sonucunda ‘2’ kat sayısının bir önemi kalmaz.

2.5. Ekran Görüntüleri