MinMax (Divide&Conquer) // Geçen haftanın konusu

BinarySearch (İkili Arama)
Diğerinden farkı dizinin sıralı olması

Algoritma

Aranan ifade ve Diziyi burada BSRC(k,n) belirtmeye gerek yok, değişmedikleri için

```
BSRC(k, n)   //
    if (k > n) then return false;
    else
        m = (k + n) / 2
        if ( A[m] == x ) then return true;
        else if x < A[m]                //Sol tarafı çağır
            return BSRC (k, m - 1)
        else
            return BSRC(m + 1, n)
        end if
    end if
```

Analiz
Özyineleme burada da var
$T(n)$ *karşılaştırma sayısı*

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n) = T(\frac{n}{2}) \underset{if\,(A[m]==x)}{+1} & n > 1 \end{cases}$$

$2^{k-1} \le n \le 2^k$ aralığında değerler alabilir.

$n = 2^k$

k=1 n=2 $T(n) = T(\frac{n}{2}) + 1$

k=2 n=4 $T(n) = T(\frac{n}{4}) + 1 + 1$

k=3 n=8 $T(n) = T(\frac{n}{8}) + 1 + 1 + 1$

$\vdots$

k-1 $T(n) = \underbrace{T(\frac{n}{2^{k-1}})}_{T(1)=1 \; için \; n=2^{k-1}} + k - 1$

$n = 2^{k-1}$
$k = \log n + 1$

$T(n) = 1 + \log n + 1 - 1$
$T(n) = 1 \log n + 1$

$T(n) = 1 \log n + 1$
O(logn)

MergeSort (Birleştirmeli Sıralama)

Algoritma

```
MergeSort (A, k, t)
    if (k < t)
        m = (k + t) / 2
        MergeSort (A, k, m)
        MergeSort (A, m+1, t)
        Merge (A, k, m, t)          // Birleştirme işlemi
    end if
end
```

Analiz

En iyi durum (min karşılaştırma)

$T(n)$ *karşılaştırma sayısı*

$$T(n) = \begin{cases} 0 \; karşılaştırma & n = 1 \\ 2T(\frac{n}{2}) + \frac{n}{2} & n > 1 \end{cases}$$

$n = 2^k$

k=1 n=2 $T(n) = 2\underbrace{T(\frac{n}{2})}_{T(1)=0} + \frac{n}{2}$

k=2 n=4 $T(n) = 2(2T(\frac{n}{4}) + \frac{n}{4}) + \frac{n}{2} \Rightarrow 4T(\frac{n}{4}) + \frac{n}{2} + \frac{n}{2}$

k=3 n=8 $T(n) = 4(2T(\frac{n}{8}) + \frac{n}{8}) + \frac{n}{2} + \frac{n}{2} \Rightarrow 8T(\frac{n}{8}) + \frac{n}{2} + \frac{n}{2} + \frac{n}{2}$

$\vdots$

k n=2^k $T(n) = 2^k(2T\underbrace{(\frac{n}{2^k})}_{\substack{1 \text{ için } n=2^k \\ T(1)=0}}) + k \cdot \frac{n}{2} \Rightarrow k \cdot \frac{n}{2}$

$$n = 2^k$$
$$k = \log n$$

$$T(n) = k \cdot \frac{n}{2}$$

$$T(n) = \frac{1}{2} n \log n$$

$$T(n) = \frac{1}{2} n \log n$$

O(nlogn)

MergeSort (Birleştirmeli Sıralama)

Analiz

En kötü durum (max karşılaştırma)

$T(n)$ *karşılaştırma sayısı*

$$T(n)=\begin{cases} 0 & n=1 \\ 2T(\frac{n}{2})-1 & n>1 \end{cases}$$ // T(1)=0

k=1 n=2 $T(n)=2\underset{T(1)=0}{T(\frac{n}{2})}+n-1$

k=2 n=4 $T(n)=2(2T(\frac{n}{4})+\frac{n}{2}-1)+n-1 \Rightarrow 4T(\frac{n}{4})+n-2+n-1$

k=3 n=8 $T(n)=2(2T(\frac{n}{8})+\frac{n}{4}-1)+2n-3 \Rightarrow 8T(\frac{n}{8})+n-4+n-2+n-1$

⋮

k $T(n)=\underbrace{2^k T(\frac{n}{2^k})}_{T(1)=0}+k\cdot n-\left(\sum_{k=0}^{k-1}2^j\right)$

$$n=2^k$$
$$k=\log n$$

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

$T(n)=n\log n-n+1$

$$\sum_{k=1}^{n} r^{k-1}=\frac{1-r^n}{1-r}=\frac{1-2^{k-1}}{1-2}=2^{k-1}-1$$

QuickSort (Hızlı Sıralama) // Böl Yönet

Çalışma zamanı; α pivotun bulunması için harcanan zaman

$$T(n) = \underbrace{T(k)}_{pivotun\ solu} + T(n-k) + \alpha n$$

k eleman	n-k eleman

n eleman

Analiz

En kötü durum (En küçük elemanı seçmesi / pivot her zaman en küçük eleman)

$T(1) = 1$

1.adım $\quad T(n) = T(1) + T(n-1) + \alpha n$

2.adım $\quad T(n) = T(1) + T(1) + T(n-2) + \alpha(n-1) + \alpha n$

3.adım $\quad T(n) = 3T(1) + T(n-3) + \alpha(n-2) + \alpha(n-1) + \alpha n$

⋮

i.adım $\quad T(n) = T(n-i) + i \cdot T(1) + \alpha \sum_{j=0}^{i-1} n - j$

$$n - i = 1$$
$$i = n - 1$$

$$\sum_{j=0}^{i-1} n - j = \sum_{j=0}^{n-2} n - j = \alpha \left[(n-2) \cdot n - \sum_{j=0}^{n-2} j \right]$$

$$\underbrace{\sum_{j=0}^{n-2} j}_{(n-0)+(n-1)\cdots+2} = \frac{(n-2)(n-1)}{2}$$

$$T(n) = \underbrace{n \cdot T(1)}_{Doğrusal\ zaman} + \alpha \underbrace{[\frac{1}{2} n^2 - \frac{1}{2} n - 1]}_{karesel}$$

Genel İfade

$T(n) = an^2 + bn + c$

$T(n) = an^2 + bn + c$

O(n^2)'dir diyebiliriz

QuickSort (Hızlı Sıralama)

Analiz

En iyi durum

$$k = \frac{n}{2}$$

$\frac{n}{2}$ eleman	$\frac{n}{2}$ eleman

n eleman

1.adım $T(n) = 2T(\frac{n}{2}) + \alpha n$

2.adım $T(n) = 2\left[2T(\frac{n}{4}) + \alpha\frac{n}{2}\right] = 4T(\frac{n}{4}) + \alpha n + \alpha n$

3.adım $T(n) = 4\left[2T(\frac{n}{8}) + \alpha\frac{n}{4}\right] + \alpha n + \alpha n = 8T(\frac{n}{8}) + \alpha n + \alpha n + \alpha n$

$\vdots$

k.adım $T(n) = 2^k T(\frac{n}{2^k}) + k\alpha n = \underbrace{n \cdot T(1)}_{Doğrusal} + \underbrace{\alpha n \log n}_{n\log aritmik}$

Genel İfade

$$T(n) = \underbrace{n \cdot T(1)}_{Doğrusal} + \underbrace{\alpha n \log n}_{n\log aritmik}$$

$T(n) = nT(1) + \alpha n \log n$
O(nlogn)'dir diyebiliriz

$T(n) = O(n \log n)$
$T(n) = \Omega(n \log n)$

DİNAMİK PROGRAMLAMA

Bir problem küçük alt problemlere bölünür, bu alt problemler için bütün parametreler önceden bellidir. Özyineleme içeren problemlerde karşımıza çıkıyor

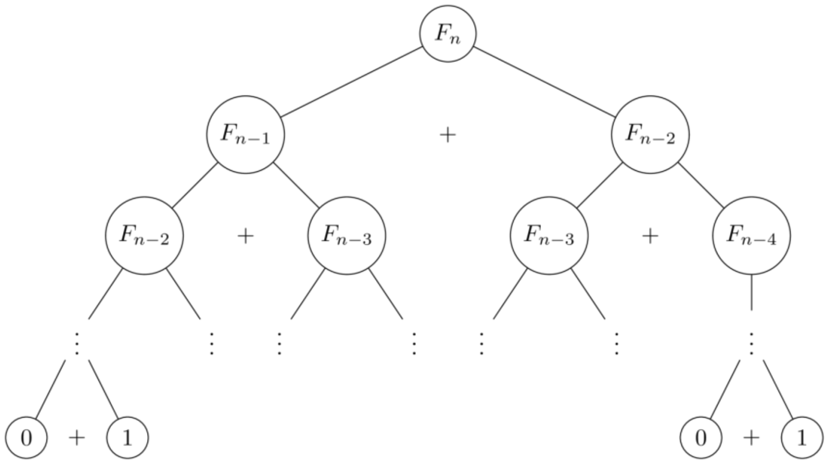
Fibonacci

Durma noktası 0 ve 1 // Onun dışında herşeyi topla

$F(n) = F(n-1) + F(n-2)$

$F(0) = 0$

$F(1) = 1$



0	1	1	……	F(n)

if x=0 return 0
if x=1 return 1
return $F(x-1) + F(x-2)$

$T(n) = T(n-1) + T(n-2) + 1$

$T(n) = O(2^n)$

Özyineleme varsa $O(2^n)$

Öz yineleme olmaz ise;

fib(x)

$\text{int } f[\ ] = new \text{ int } \underbrace{[x+1]}_{x'e\ ulaşabilmek\ için}$

$f[0] = 0$

$f[1] = 1$

⋮

for (int i=2;i<x+1;i++)
{
$f[i] = f[i-1] + f[i-2]$
}

return $f[x]$

Dizi üzerinde fibonacci O(n)