Böl-ve-fethet algoritması

4 adet $n/2 \times n/2$ boyutlarında alt matrislere böleriz.

Fikir

$n \times n$ matris = $(n/2)\times(n/2)$ altmatrisin 2×2 matrisi:

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \cdot \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]_{n\times n}$$

$$C \quad = \quad A \quad \cdot \quad B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}\right\}$$

8 tane matris çarpımı

4 tane " toplamı

Çalışma Zamanı

$$T(n) = 8T(n/2) + \Theta(n^2)$$

alt matrislerin toplama işi

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ & & \\ a_{n1} & & a_{nn} \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1n} \\ & & \\ b_{n1} & \dots & b_{nn} \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ & & & \\ & & & \end{bmatrix}$$

n adet top.
n adet top.
n
n
⋮
n
+ ______
n^2 adet toplama gerekir.

Master Metodu ile çözelim

n^2 ile $n^{\log_2 8} = n^3$ karşılaştır.

$n^2 = O(n^{3-\varepsilon})$ $\quad \varepsilon = 1$ için

Durum 1: $T(n) = \Theta(n^3)$

Standart algoritmadan daha iyi değil.

Strassen'in fikri

- 2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çöz.

$$
\begin{aligned}
P_1 &= a\cdot(f-h) \\
P_2 &= (a+b)\cdot h \\
P_3 &= (c+d)\cdot e \\
P_4 &= d\cdot(g-e) \\
P_5 &= (a+d)\cdot(e+h) \\
P_6 &= (b-d)\cdot(g+h) \\
P_7 &= (a-c)\cdot(e+f)
\end{aligned}
\qquad
\begin{aligned}
r &= P_5+P_4-P_2+P_6 \\
s &= P_1+P_2 \\
t &= P_3+P_4 \\
u &= P_5+P_1-P_3-P_7
\end{aligned}
$$

$$r = ae + \cancel{ah} + \cancel{de} + \cancel{dh} + \cancel{dg} - \cancel{de} - \cancel{ah} - \cancel{bh} + bg + \cancel{bh} - \cancel{dg} - \cancel{dh}$$

$$r = ae + bg$$

7 tane çarpma

18 tane toplama/çıkarma

Çalışma Zamanı

$$T(n) = 7T(n/2) + \Theta(n^2)$$

n^2 ile $n^{\log_2 7} \approx n^{2.81..}$ 'i karşılaştır.

$n^2 = O(n^{2.81-\varepsilon})$ $\qquad \varepsilon = 0.81$

Durum 1: $T(n) = \Theta(n^{\log_2 7})$

Sorular ve Çözümleri

1- Alg1 (A[1...n])

```
Alg1 ( A[1...n] )
    if n<=0
        return
    q=⌊n/2⌋
    if q is even
        Alg1(A[1... q])
    else
        Alg1(A[q+1... n])
```

Alg2 (A[1...n])

```
Alg2 ( A[1...n] )
    if n<=0
        return
    q=⌊n/4⌋
    if q is even
        Alg2(A[1... q])
        Alg2(A[q+1...2q])
    else
        Alg2(A[2q+1...3q])
        Alg2(A[3q+1... n])
```

Alg1 ve Alg2 nin çalışma zamanlarını bulunuz. Hangisi daha iyidir?

Çözüm:

Alg1 için $T(n) = 1 \cdot T(n/2) + \Theta(1)$

$= \Theta(\lg n)$

Alg2 için $T(n) = 2 \cdot T(n/4) + \Theta(1)$

1 ile $n^{\log_4 2} = n^{1/2}$ karşılaştır.

$1 = O(n^{1/2 - \varepsilon})$ $\varepsilon = 1/2$

durum 1, $T(n) = \Theta(\sqrt{n})$

Alg1 Alg2 den daha iyidir.

2. Tek Doruklu Arama

Bir A[1...n] dizisi eğer artan bir altdiziyi takip eden azalan bir altdiziye sahip ise tek doruklu bir dizidir.

Yani
$$A[i] < A[i+1] \quad \forall\ 1 \leq i < m$$
$$A[j] > A[j+1] \quad \forall\ m \leq j < n$$

$\underbrace{2\ 3\ 7\ 8\ 11}_{\text{artan}}$ $\overbrace{11\ 9\ 8\ 5\ 4\ 1\ 0}^{\text{azalan}}$ A[m] maksimumdur

Soru: Tek doruklu dizide maksimum elemanı $\Theta(\lg n)$ zamanda bulan bir algoritma geliştiriniz.

$$\begin{array}{lccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ A: & 2 & 3 & 7 & 8 & 11 & 9 & 8 & \cancel{5} & \cancel{4} & \cancel{1} & \cancel{-2} \end{array}$$

$\cancel{2}\ \cancel{3}\ \cancel{7}\ 8\ 11\ 9$

```
Tek_Doruklu_Arama (A[1..n])
    a=1
    b=n
    while a<b
        mid = ⌊(a+b)/2⌋
        if A[mid] > A[mid+1]
            b= mid
        else
            a= mid+1
    return A[a]
```

$$\begin{array}{ccccccccccc} a=1 & 2 & 3 & 4 & 5 & \overset{mid}{6} & 7 & 8 & 9 & 10 & 11=b \\ 2 & 3 & 7 & 8 & 11 & 9 & 8 & 5 & 4 & 1 & -2 \end{array}$$

$$\begin{array}{cccccc} a=1 & & mid & & & b=6 \\ 2 & 3 & 7 & 8 & 11 & 9 \end{array}$$

$$\begin{array}{ccc} a=4 & mid & b=6 \\ 8 & 11 & 9 \end{array}$$

$$\begin{array}{cc} a=4 & b=5 \\ 8 & 11 \\ mid & \end{array}$$

11
a=5
b=5

Çalışma Zamanı: $T(n) = 1 \cdot T(n/2) + \Theta(1) = \Theta(\lg n)$

Ödev: Tek_Doruklu_Arama algoritmasını özyinelemeli algoritmaya dönüştürünüz.

3. n elemanlı bir dizide tekrarlı eleman varsa True yok ise False döndüren verimli bir algoritma tasarlayın.

```
Tekrarlımı(A[1...n])
    for i=1 to n
        for J=i+1 to n
            if A[i]=A[J]
                return True
    return False
```

$T(n) = \Theta(n^2)$

```
Tekrarlımı_v2(A[1...n])
    Merge_Sort(A[1...n])
    for i=1 to n-1
        if A[i]=A[i+1]
            return True
    return False
```

$\Theta(n\lg n) \leftarrow$ Merge_Sort(A[1...n])

$\Theta(n) \leftarrow$ for döngüsü

$T(n) = \Theta(n\lg n) + \Theta(n) = \Theta(n\lg n)$