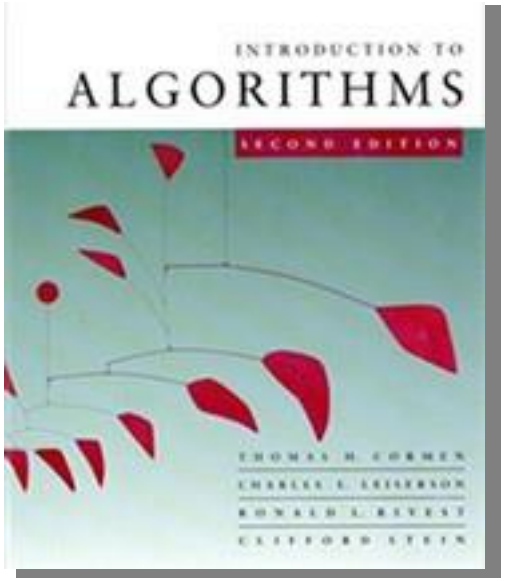
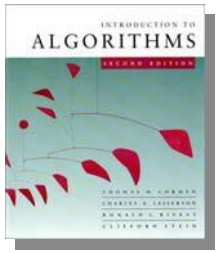
Algoritmalar



DERS 3

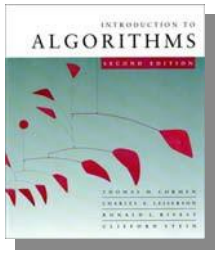
Böl ve Fethet(Divide and Conquer)

- İkili arama
- Sayı üstelleri
- Fibonacci sayıları
- Matriks çarpımı
- Strassen'in algoritması



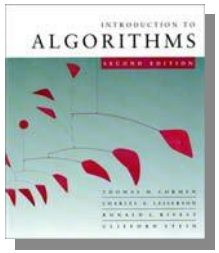
Böl-ve-hükmet tasarım paradigması

1. Problemi (anlık durumu) alt problemlere **böl**.
2. Altproblemleri özyinelemeli olarak çözüp, onları **fethet**.
3. Altproblem çözümlerini **birleştir**.



Birleştirme sıralaması

1. ***Bölmek:*** Kolay.
2. ***Hükmetmek:*** 2 altdiziyi özyinelemeli sıralama.
3. ***Birleştirmek:*** Doğrusal-zamanda birleştirme.



Birleştirme sıralaması

1. Bölmek: Kolay.

2. Hükmetmek: 2 altdiziyi özyinelemeli sıralama.

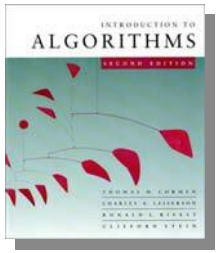
3. Birleştirmek: Doğrusal-zamanda birleştirme.

$$T(n) = 2\,T(n/2) + \Theta(n)$$

altproblem sayısı

altproblem boyutu

bölme ve birleştirme işi



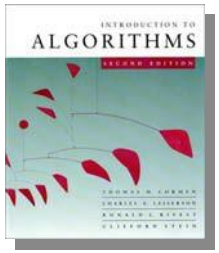
Master teoremi (hatırlatma)

$T(n) = a\,T(n/b) + f(n)$

DURUM 1: $f(n) = O(n^{\log_b a - \varepsilon})$, sabit $\varepsilon > 0$
$\Rightarrow T(n) = \cup(n^{\log_b a})$.

DURUM 2: $f(n) = \cup(n^{\log_b a})$
$\Rightarrow T(n) = \cup(n^{\log_b a})$.

DURUM 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$, sabit $\varepsilon > 0$,
ve düzenleyici koşul (regularity condition).
$\Rightarrow T(n) = \cup(f(n))$.



Master teoremi (hatırlatma)

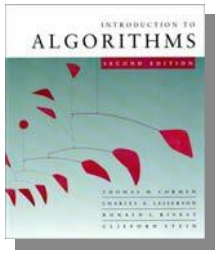
$T(n) = a\,T(n/b) + f(n)$

DURUM 1: $f(n) = O(n^{\log_b a - \varepsilon})$, sabit $\varepsilon > 0$
$\Rightarrow T(n) = \cup(n^{\log_b a})$.

DURUM 2: $f(n) = \cup(n^{\log_b a})$
$\Rightarrow T(n) = \cup(n^{\log_b a})$.

DURUM 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$, sabit $\varepsilon > 0$,
ve düzenleyici koşul (regularity condition).
$\Rightarrow T(n) = \cup(f(n))$.

Birleştirme sıralaması: $a = 2$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n^{\log_2 2} = n$
$\Rightarrow$ DURUM 2 ($k = 0$) $\Rightarrow T(n) = \Theta(n \lg n)$



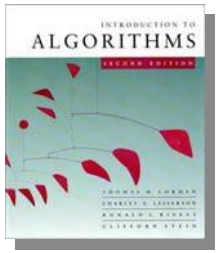
İkili arama

Sıralı dizide bir elemanını bulma:

1.Böl: Orta elemanı belirle.

2.Hükmet: 1 altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.



İkili arama

Sıralı dizide bir elemanını bulma:

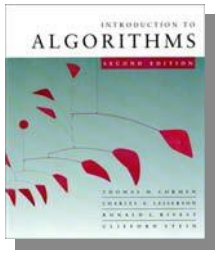
1.Böl: Orta elemanı belirle.

2.Hükmet: 1altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9' u bul.

3	5	7	8	9	12	15



İkili arama

Sıralı dizide bir elemanını bulma:

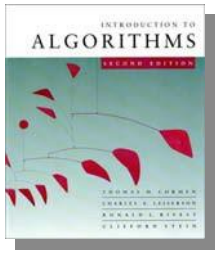
1.Böl: Orta elemanı belirle.

2.Hükmet: 1altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9'u bul.

3	5	7	8	9	12	15



İkili arama

Sıralı dizide bir elemanını bulma:

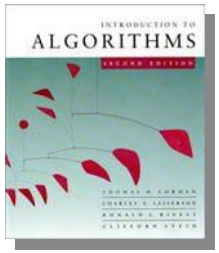
1.Böl: Orta elemanı belirle.

2.Hükmet: 1altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9'u bul.

3	5	7	8	9	12	15



İkili arama

Sıralı dizide bir elemanını bulma:

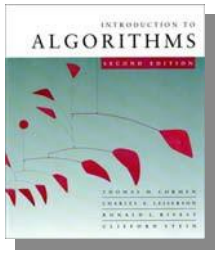
1.Böl: Orta elemanı belirle.

2.Hükmet: 1 altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9'u bul.

3 5 7 8 9 12 15



İkili arama

Sıralı dizide bir elemanını bulma:

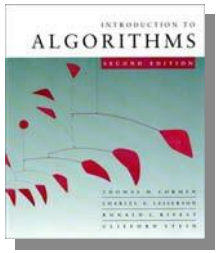
1.Böl: Orta elemanı belirle.

2.Hükmet: 1 altdizide özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9'u bul

3	5	7	8	9	12	15



İkili arama

Sıralı dizilimin bir elemanını bulma:

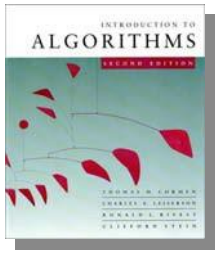
1.Böl: Orta elemanı belirle.

2.Hükmet: 1 altdizilimde özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9'u bul.

3 5 7 8 9 12 15



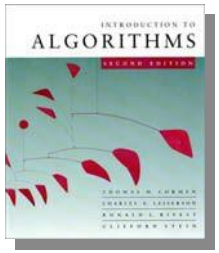
İkili arama için yineleme

$$T(n) = 1\,T(n/2) + \cup(1)$$

altproblem sayısı

altproblem boyutu

bölme ve birleştime işi



İkili arama için yineleme

$$T(n) = 1\,T(n/2) + \cup(1)$$

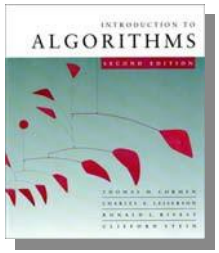
altproblem sayısı

altproblem boyutu

bölme ve birleştime işi

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow$ DURUM 2

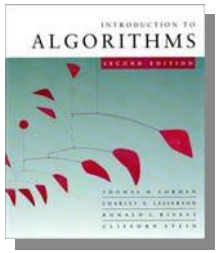
$\Rightarrow T(n) = \cup(\lg n)$.



Bir sayının üstellenmesi

Problem: a^n'yi, $n \in \mathbb{N}$ iken hesaplama.

Saf (Naive) algoritma: $\cup(n)$.



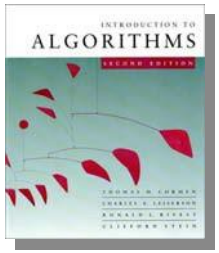
Bir sayının üstellenmesi

Problem: a^n'yi, $n \in \mathbb{N}$ iken hesaplama.

Saf (Naive) algoritma: $\Theta(n)$.

Böl-ve-fethet algoritması:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & n \text{ çift sayıysa;} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & n \text{ tek sayıysa.} \end{cases}$$



Bir sayının üstellenmesi

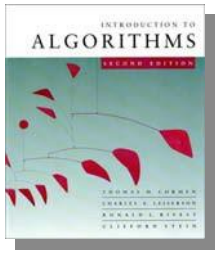
Problem: a^n 'yi $n \in \mathbb{N}$ iken hesaplama.

Saf (Naive) algorithm: $\cup(n)$.

Böl-ve-fethet algoritması:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & n \text{ çift sayıysa;} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & n \text{ tek sayıysa.} \end{cases}$$

$$T(n) = T(n/2) + \cup(1) \Rightarrow T(n) = \cup(\lg n) .$$

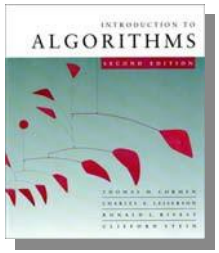


Fibonacci sayıları

Özyinelemeli tanım:

$$F_n = \begin{cases} 0 & \text{eğer } n = 0 \text{ ise;} \\ 1 & \text{eğer } n = 1 \text{ ise;} \\ F_{n-1} + F_{n-2} & \text{eğer } n \geq 2 \text{ ise.} \end{cases}$$

0 1 1 2 3 5 8 13 21 34 …



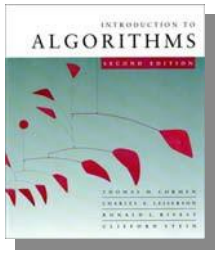
Fibonacci sayıları

Özyinelemeli tanım:

$$F_n = \begin{cases} 0 & \text{eğer } n = 0 \text{ ise;} \\ 1 & \text{eğer } n = 1 \text{ ise;} \\ F_{n-1} + F_{n-2} & \text{eğer } n \geq 2 \text{ ise.} \end{cases}$$

0 1 1 2 3 5 8 13 21 34 …

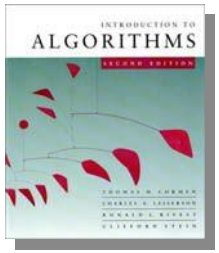
Saf özyinelemeli algoritma: $\Omega(\phi^n)$ (üstel zaman), buradaki $\phi = (1+\sqrt{5})/2$ **altın oran**'dır (***golden ratio***).



Fibonacci sayılarını hesaplama

Aşağıdan yukarıya:

- $F_0, F_1, F_2, \ldots, F_n$'i sırayla, her sayı iki öncekinin toplamı olacak şekilde hesaplayın.
- Yürütüm süresi: $\Theta(n)$.



Fibonacci sayılarını hesaplama

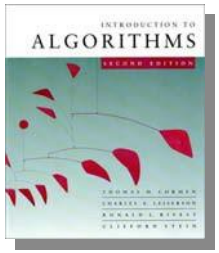
Aşağıdan yukarıya:

- $F_0, F_1, F_2, \ldots, F_n$'i sırayla, her sayı iki öncekinin toplamı olacak şekilde hesaplayın.
- Çalışma zamanı: $\Theta(n)$.

Saf özyinelemeli kare alma (Naive recursive squaring):

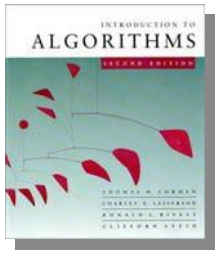
$F_n = \phi^n / \sqrt{5}$ yakın tamsayı yuvarlaması.

- Özyinelemeli kare alma: $\Theta(\lg n)$ zamanı.
- Bu yöntem güvenilir değildir, yuvarlama hatalarına gebedir.



Özyineleme ile kare alma (Recursive squaring)

Teorem: $$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n .$$

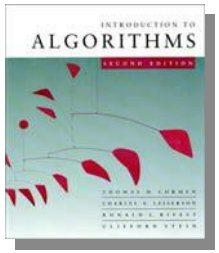


Özyineleme ile kare alma (Recursive squaring)

Teorem: $$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n .$$

Algoritma: Özyineleme ile kare alma.

Süre $= \Theta(\lg n)$.



Özyineleme ile kare alma (Recursive squaring)

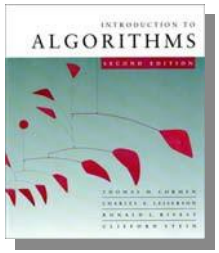
Teorem: $\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$.

Algoritma: Özyineleme ile kare alma.

Süre $= \Theta(\lg n)$.

Teoremin ispatı. (n 'de tümevarım)

Taban ($n = 1$): $\begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^1$



Özyineleme ile kare alma

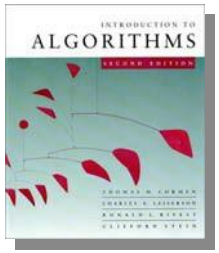
Tümevarım adımı ($n \geq 2$):

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n}$$

■



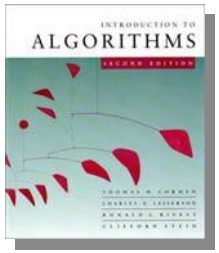
Matrislerde çarpma

Girdi: $A = [a_{ij}]$, $B = [b_{ij}]$.
Çıktı: $C = [c_{ij}] = A \cdot B$.

$\Big\}$ $i, j = 1, 2, \ldots, n$.

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$



Standart algoritma

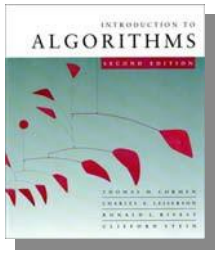
for $i \leftarrow 1$ **to** n *(i 1'den n'ye kadar)*

do for $j \leftarrow 1$ **to** n *(j 1'den n'ye kadar)*

do $c_{ij} \leftarrow 0$

for $k \leftarrow 1$ **to** n

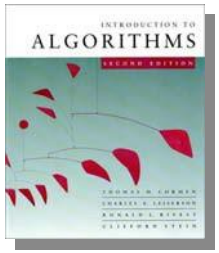
do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$



Standart algoritma

for $i \leftarrow 1$ **to** n *(i 1'den n'ye kadar)*
  do for $j \leftarrow 1$ **to** n *(j 1'den n'ye kadar)*
    do $c_{ij} \leftarrow 0$
      for $k \leftarrow 1$ **to** n
        do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Çalışma zamanı $= \Theta(n^3)$



Böl-ve-fethet algoritması

FİKİR

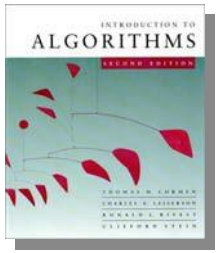
$n \times n$ matris = $(n/2)\times(n/2)$ altmatrisin 2×2 matrisi:

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$C = A \cdot B$$

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

8 çarpma $(n/2)\times(n/2)$ altmatriste
4 toplama $(n/2)\times(n/2)$ altmatriste



Böl-ve-fethet algoritması

FİKİR

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

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

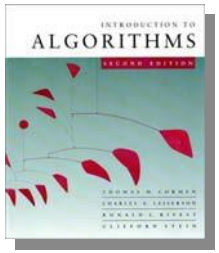
$$C = A \cdot B$$

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

recursive (özyinelemeli)

8 çarpma $(n/2)\times(n/2)$ altmatriste,

4 toplama $(n/2)\times(n/2)$ altmatriste.



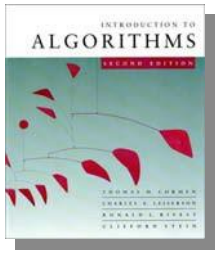
Böl-ve-Fethet algoritmasının çözümlemesi

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

altmatris sayısı

altmatris boyutu

altmatrisleri toplama işi



Böl-ve-Fethet algoritmasının çözümlemesi

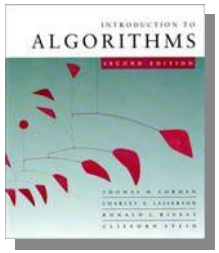
$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

altmatris sayısı

altmatris boyutu

altmatrisleri toplama işi

$n^{\log_b a} = n^{\log_2 8} = n^3 \quad \Rightarrow$ DURUM 1 $\Rightarrow T(n) = \Theta(n^3)$.



Böl-ve-Fethet algoritmasının çözümlemesi

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

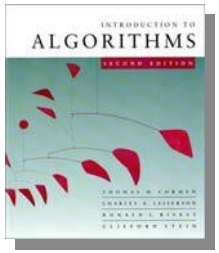
altmatris sayısı

altmatris boyutu

altmatrisleri toplama işi

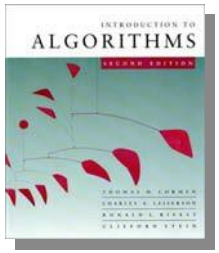
$n^{\log_b a} = n^{\log_2 8} = n^3 \quad \Rightarrow$ DURUM 1 $\Rightarrow T(n) = \Theta(n^3)$.

Sıradan algoritmadan daha iyi değil.



Strassen'in fikri

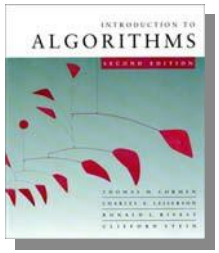
- 2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çöz.



Strassen’in fikri

- 2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çöz.

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

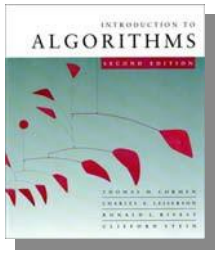


Strassen'in fikri

- 2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çöz.

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_5 + P_1 - P_3 - P_7$$



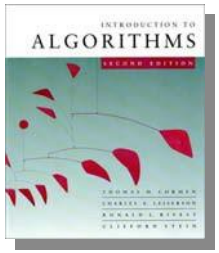
Strassen'in fikri

- 2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çöz.

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_5 + P_1 - P_3 - P_7$$

7 çarp., 18 topl. /çıkar.
Not: Çarpma işleminde **sırabağımsızlık yok!**

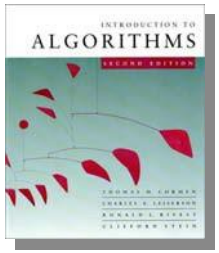


Strassen'in algoritması

1.Böl: A ve B'yi $(n/2)\times(n/2)$ altmatrislere böl. $+$ ve $-$ kullanarak çarpılabilecek terimler oluştur.

2.Fethet: $(n/2)\times(n/2)$ altmatrislerde özyinelemeli 7 çarpma yap.

3. Birleştir: $+$ ve $-$ kullanarak $(n/2)\times(n/2)$ altmatrislerde C 'yi oluştur.



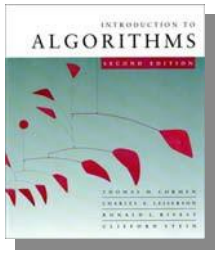
Strassen'in algoritması

1.Böl: A ve B'yi $(n/2)\times(n/2)$ altmatrislere böl. $+$ ve $-$ kullanarak çarpılabilecek terimler oluştur.

2.Fethet: $(n/2)\times(n/2)$ altmatrislerde özyinelemeli 7 çarpma yap.

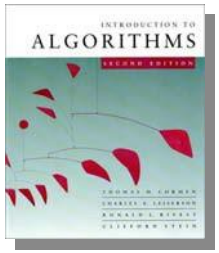
3. Birleştir: $+$ ve $-$ kullanarak $(n/2)\times(n/2)$ altmatrislerde C 'yi oluştur.

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



Strassen'in çözümlenmesi

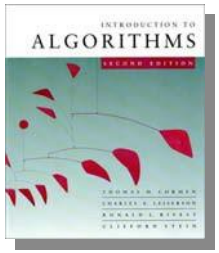
$$T(n) = 7\,T(n/2) + \Theta(n^2)$$



Strassen'in çözümlenmesi

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

$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow$ DURUM 1 $\Rightarrow T(n) = \Theta(n^{\lg 7})$.

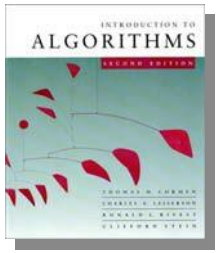


Strassen'in çözümlenmesi

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

$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow$ DURUM 1 $\Rightarrow T(n) = \Theta(n^{\lg 7})$.

2.81 değeri 3'den çok küçük görünmeyebilir ama, fark üstelde olduğu için, koşma süresine etkisi kayda değerdir. Aslında, $n \geq 32$ değerlerinde, Strassen'in algoritması günün makinelerinde normal algoritmadan daha hızlı çalışır.



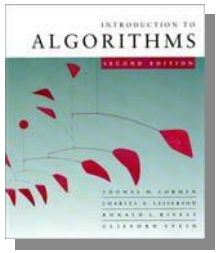
Strassen'in çözümlenmesi

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

$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow$ DURUM 1 $\Rightarrow T(n) = \Theta(n^{\lg 7})$.

2.81 değeri 3' den çok küçük görünmeyebilir ama, fark üstelde olduğu için, yürütüm süresine etkisi kayda değerdir. Aslında, $n \geq 32$ değerlerinde Strassen'in algoritması günün makinelerinde normal algoritmadan daha hızlı çalışır.

Bugünün en iyi değeri (teorik merak açısından, Coppersmith–Winograd algorithm): $\Theta(n^{2.376\ldots})$.



Sonuç

- Böl ve Fethet algoritma tasarımının güçlü tekniklerinden sadece biridir.
- Böl ve Fethet algoritmaları yinelemeler ve Ana (Master) metot kullanarak çözümlenebilir.

(bu nedenle bu matematiğin pratiğini yapın).

- Böl ve Fethet stratejisi genellikle verimli algoritmalara götürür.

