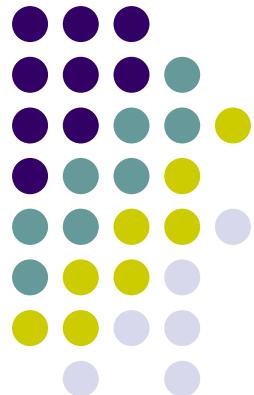


Introduction to Algorithm Design

Lecture Notes 5





ROAD MAP

- **Divide And Conquer**
 - Merge Sort
 - Quick Sort
 - Binary Tree and Its Properties
 - **Multiplication of Large Integers**
 - **Strassen's Matrix Multiplication**
 - Closest Pair of Points
 - Convex Hull



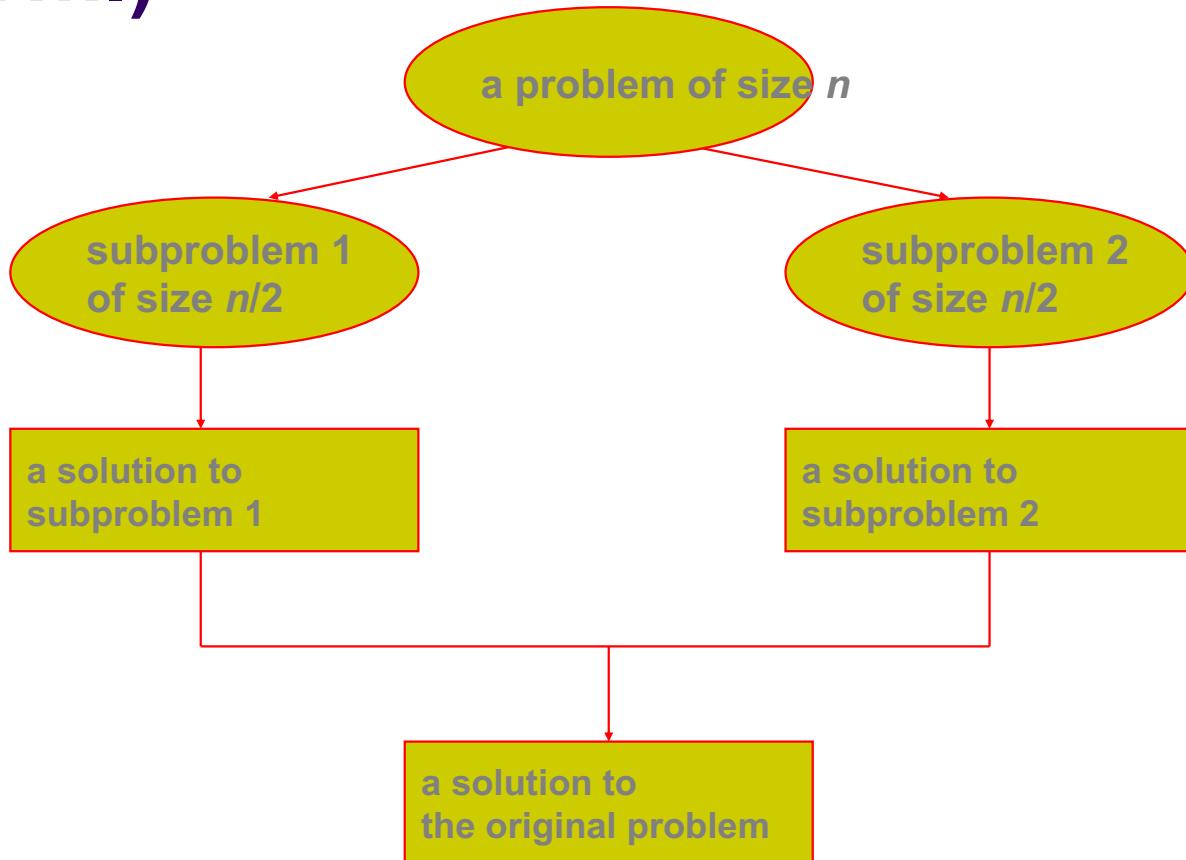
Divide-and-Conquer

En iyi bilinen algoritma tasarım stratejilerinden biridir.

YAKLAŞIM

1. Problemi iki ya da daha ufak parçalara böl.
2. Küçük parçaları özyineli bir şekilde çöz
3. Bu çözümleri birleştirerek orijinal probleme çözüm bul.

Divide-and-Conquer Tekniği (dvm.)





Divide-and-Conquer Örnekleri

- Sıralama: mergesort ve quicksort
- Binary tree traversals
- Multiplication of large integers
- Matris çarpımı: Strassen's algorithm
- Closest-pair ve convex-hull algoritmaları

General Divide-and-Conquer Recurrence



$$T(n) = aT(n/b) + f(n), \text{ öyle ki } f(n) \in \Theta(n^d), \quad d \geq 0$$

Master Teoremi:

If $a < b^d$, $T(n) \in \Theta(n^d)$

If $a = b^d$, $T(n) \in \Theta(n^d \log n)$

If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

$$a \quad b \quad d=1$$

Examples: $T(n) = 4T(n/2) + n \Rightarrow T(n) \in ?$

$T(n) = 4T(n/2) + n^2 \Rightarrow T(n) \in ? \Theta(n^{2(\log_2 4)})$

$T(n) = 4T(n/2) + n^3 \Rightarrow T(n) \in ?$

$$\begin{matrix} a & b & d \\ 4 & 2 & 3 \end{matrix}$$

$$\Theta(n^{\log_2 4}) = \Theta(n^2)$$

$$\Theta(n^{2(\log_2 4)})$$

$$\Theta(n^3)$$



Mergesort

- A[0..n-1] arrayini iki eşit parçaya böl, bu parçaları B ve C'ye kopyala
- B ve C array'lerini özyineli bir şekilde sırala
- B ve C arrayini A arrayinin içine merge et (sıralı bir şekilde birleştir).
 - Array'lerden birinde eleman kalmayıncaya kadar devam et:
 - Arraylerin kontrol edilmemiş ilk elemanlarını kontrol et.
 - Küçük olanı A'ya kopyala, eleman alınan arrayin index değişkenini arttır.
 - Diğerlerini kontrol etmeye devam et

Pseudocode of Mergesort



ALGORITHM

```
Mergesort(A[0..n - 1])
    //Sorts array  $A[0..n - 1]$  by recursive mergesort
    //Input: An array  $A[0..n - 1]$  of orderable elements
    //Output: Array  $A[0..n - 1]$  sorted in nondecreasing order
    if  $n > 1$ 
        copy  $A[0..\lfloor n/2 \rfloor - 1]$  to  $B[0..\lfloor n/2 \rfloor - 1]$ 
        copy  $A[\lfloor n/2 \rfloor ..n - 1]$  to  $C[0..\lceil n/2 \rceil - 1]$ 
        Mergesort(B[0..\lfloor n/2 \rfloor - 1])
        Mergesort(C[0..\lceil n/2 \rceil - 1])
        Merge(B, C, A)
```

Pseudocode of Merge



ALGORITHM *Merge($B[0..p - 1]$, $C[0..q - 1]$, $A[0..p + q - 1]$)*

//Merges two sorted arrays into one sorted array

$p+q$

//Input: Arrays $B[0..p - 1]$ and $C[0..q - 1]$ both sorted

//Output: Sorted array $A[0..p + q - 1]$ of the elements of B and C

$i \leftarrow 0; j \leftarrow 0; k \leftarrow 0$

while $i < p$ and $j < q$ **do**

if $B[i] \leq C[j]$

$A[k] \leftarrow B[i]; i \leftarrow i + 1$

else $A[k] \leftarrow C[j]; j \leftarrow j + 1$

$k \leftarrow k + 1$

if $i = p$

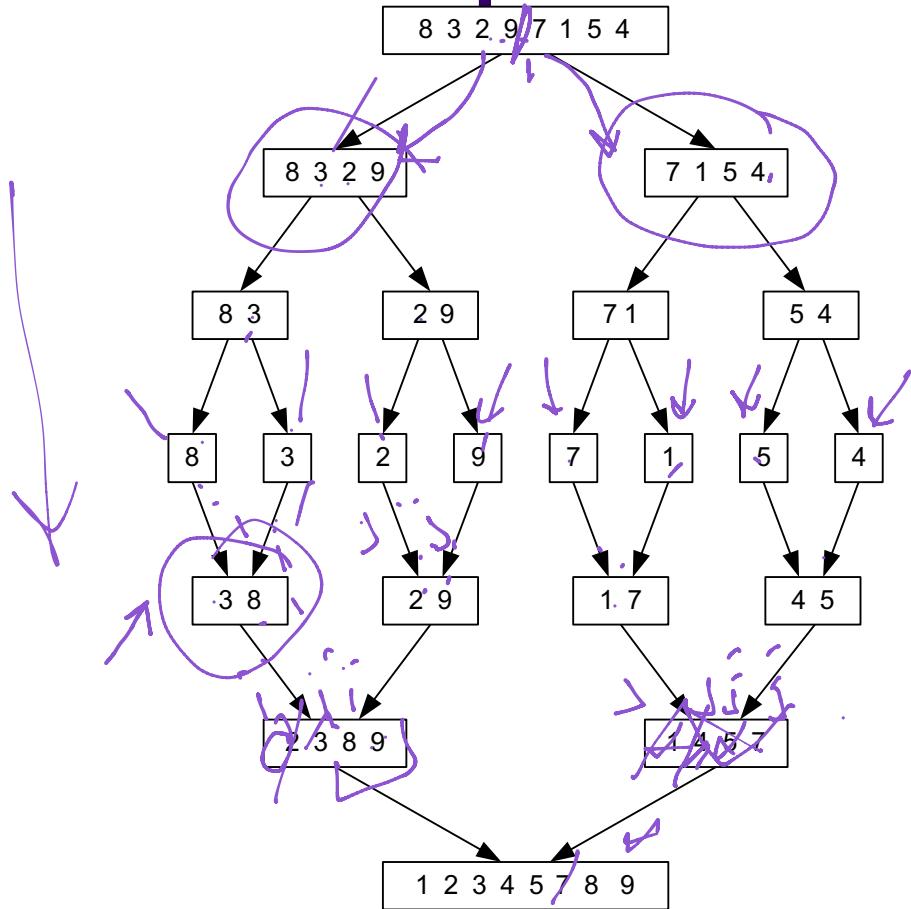
 copy $C[j..q - 1]$ to $A[k..p + q - 1]$

else copy $B[i..p - 1]$ to $A[k..p + q - 1]$

$\Theta(p+q)$

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

Mergesort Example





Mergesort Analizi

- Tüm durumlarda aynı sayıda işlem: $\Theta(n \log n)$

- Kıyaslama tabanlı sıralamalara göre minimum en kötü zaman karmaşıklığı:

$$\lceil \log_2 n! \rceil \approx n \log_2 n - 1.44n$$

- Alan isteği: $\Theta(n)$ (not in-place)
- Recursion olmadan da çalışır



Quicksort

- Quicksort, D & C stratejisine sahip önemli bir sıralama algoritmasıdır.
- Verilen bir $A[0 \dots n-1]$ arrayini sıralar .



Quicksort

Verilen $A[1..r]$ arrayini sıralamak için

Yaklaşım:

- Arrayin elemanlarını değerlerine bağlı olarak bölmelendirir.
- Bölmelendirmeden (Partitiondan) sonra $A[1]$ sıralı arraydeki son pozisyonundadır.
- Alt arrayleri sıralamaya devam et.



Quicksort

ALGORITHM Quicksort (A[1..r])

```
// Sorts a subarray by quicksort
// Input : A subarray A[l..r] of A[0..n-1],
defined by its left and right indices l and r
// Output : The subarray A[l..r] sorted in
nondecreasing order
```

If $l < r$

$s \leftarrow \text{Partition}(A[l..r])$

// s is a split position

$\text{Quicksort}(A[l..s-1])$

$\text{Quicksort}(A[s+1..r])$



Quicksort

- A[0..n-1] üzerinde nasıl partition yapılmalı?
 - Alt elemanlara bölmek için bir eleman seçmemiz lazım
 - Bu eleman *pivot* olarak adlandırılmaktadır.
- Pivot seçmek için çeşitli stratejiler vardır.
- Şimdi en basit stratejiyi kullanacağız
 - Pivot arrayin ilk elamanı olacak; $p=A[1]$



Quicksort

- Partitioning :

		$\rightarrow i$		$j \leftarrow$	
p	all are $\leq p$	$\geq p$	$\leq p$	all are $\geq p$

		$j \leftarrow$	$\rightarrow i$	
p	all are $\leq p$	$\leq p$	$\geq p$	all are $\geq p$

	$\rightarrow i = j \leftarrow$	
p	all are $\leq p$	$= p$

Quicksort



ALGORITHM HoarePartition (A[l..r])

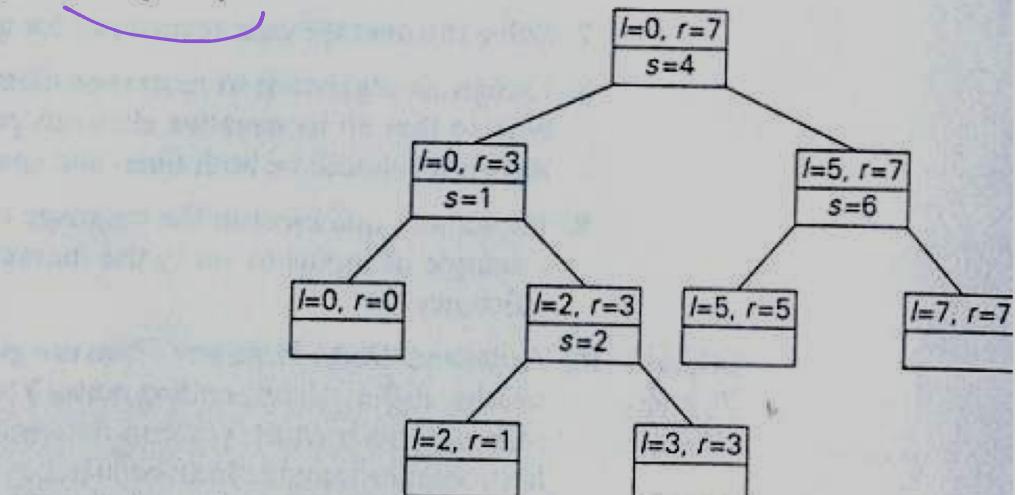
```
// Partitions a subarray by using its first element  
as a pivot  
// Input : A subarray A[l..r] of A[0..n-1], defined  
by its left and right indices l and r (l < r)  
// Output : A partition of A[l..r], with the split  
position returned as this function's value
```

```
p ← A[l];      i ← l ;      j ← r+1  
repeat  
    repeat i ← i+1 until A[i] ≥ p  
    repeat j ← j-1 until A[j] ≤ p  
    swap (A[i], A[j])  
until i ≥ j  
swap (A[i], A[j])      // undo last swap when i ≥ j  
swap (A[l], A[j])  
return j
```

$p = s = AC^2$
 $i = 1$



0	1	2	3	4	5	6	7
5	3	1	9	8	2	4	7
5	3	1	9	8	2	4	7
5	3	1	4	8	2	9	7
5	3	1	4	8	2	9	7
5	3	1	4	2	8	9	7
5	3	1	4	2	8	9	7
2	3	1	4	5	8	9	7
2	3	1	4				
2	3	1	4				
2	1	3	4				
2	1	3	4				
1	2	3	4				
1							
3	4						
3	4						
3	4						



3	9	7
8	7	9
8	7	9
7	8	9
7		



Quick Sort

- **Analiz:**

n : arraydeki eleman sayısı

$$T(\text{partition}) = O(n) \rightarrow n+1$$

- **Best case (En iyi durum)**

Tüm splitler (pivotlar hep median olarak seçilirse)
arrayin ortasından gerçekleştirilirse en iyi durum
olmaktadır.

$$T(n) = 2T(n/2) + n \quad \text{for} \quad n > 1$$

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



Quick Sort

- **Analiz:**

- En kötü durum (Worst-case)
 - Tüm splitler aşağıdaki gibi gerçekleşir
 - Alt arraylerden sonra bir tanesi boş, diğer ise arrayin uzunluğundan bir tane az elemana sahiptir.
 - Bu durum $A[0 .. n-1]$ artan fonksiyon ve $A[0]$ 'ı pivot olarak kullandığımızda oluşur.
 - Soldan sağa tarama $A[1]$ 'de biter
 - Sağdan sola tarama $A[0]$ 'a kadar gider.

$j \leftarrow i \rightarrow$

$A[0]$	$A[1]$	\dots	$A[n-1]$
--------	--------	---------	----------



Quick Sort

- Analiz :
 - En Kötü Durum (Worst-case)

14 | 2 7 5 100 110 95 1

$$T(n) = (n+1) + n + \dots + 3 = \frac{(n+1)(n+2)}{2} - 3 \in \Theta(n^2)$$



Quick Sort

Analiz :

- Ortalama Durum (Average Case)

Split pozisyonu olma olasılığı her index için aynıdır.

$$P = 1/n$$

$$T(n) = \frac{1}{n} \left(\sum_{k=1}^n (T(k-1) + T(n-k)) + n + 1 \right)$$

Quick Sort



$$T(n) = \frac{1}{n} \sum_{k=1}^n [T(k-1) + T(n-k)] + n + 1$$

$$nT(n) = \sum_{k=1}^n T(k-1) + T(n-k) + n(n+1)$$

$$nT(n) = 2(T(0) + T(1) + \dots + T(n-1)) + n(n+1)$$

$$- (n-1)T(n-1) = 2(T(0) + \dots + T(n-2)) + n(n-1)$$

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

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



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

$$\frac{T(n)}{n+1} = \frac{T(n-3)}{n-2} + \frac{2}{n+1} + \frac{2}{n} + \frac{2}{n-1}$$

⋮

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2 \sum_{k=3}^{n+1} \frac{1}{k}$$

$$\frac{T(n)}{n+1} = 2 \sum_{k=3}^{n+1} \frac{1}{k}$$

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

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

Büyük Tamsayıları Çarpma İşlemi (Multiplication of Large Integers)



- Bazı uygulamalar büyük ölçekli tamsayıların çarpılmasını gerektirebilir. (Örn: 100 basamaktan fazla iki sayıyı)
 - Criptoloji gibi
- Bu sayılar modern bilgisayarlar için çok uzundur.
 - Daha ayrı bir muamele görmelidirler.
 - Bunları işlemek birim zaman almaz.



Büyük Tamsayıları Çarpma İşlemi

- n basamaklı tamsayıları kağıt-kalem algoritması ile hesaplamak
 - Bir sayının her bir basamağı diğer sayının n basamağı ile çarpılmalı
 - Toplamda n^2 tane rakam çarpma işlemi gerçekleştirilir.
- n^2 rakam çarpımından daha az işlemle gerçekleştirilemez mi?



Büyük Tamsayıları Çarpma İşlemi

Örnek: 23 ve 14'ü çarpalım

$$23 = 2 \cdot 10^1 + 3 \cdot 10^0 \text{ and } 14 = 1 \cdot 10^1 + 4 \cdot 10^0.$$

$$\begin{aligned} 23 &= (2 \cdot 10^1 + 3 \cdot 10^0) * (1 \cdot 10^1 + 4 \cdot 10^0) \\ &= (2 * 1)10^2 + (3 * 1 + 2 * 4)10^1 + (3 * 4)10^0 \end{aligned}$$

- Toplamda 4 farklı çarpma işlemi var
- Ortanca terim aşağıdaki şekilde de hesaplanabilir

$$3 * 1 + 2 * 4 = (2 + 3) * (1 + 4) - (2 * 1) - (3 * 4)$$

- Yani çözüm toplamda üç farklı çarpma ile gerçekleştirilebilir.

$$10a = 2^3 a + 2a$$

Büyük Tamsayıları Çarpma İşlemi

Bir çift iki basamaklı sayı $a = a_1 a_0$ ve $b = b_1 b_0$, bu sayıların çarpımı c can aşağıdaki formül ile bulunabilir.

$$c = a * b = c_2 10^2 + c_1 10^1 + c_0$$

Burada

$$c_2 = a_1 * b_1 \rightarrow \text{ilk hanelerin çarpımı}$$

$$c_0 = a_0 * b_0 \rightarrow \text{son hanelerin çarpımı}$$

$$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0) \rightarrow \text{Ortanca hanenin sonucu}$$



Büyük Tamsayıları Çarpma İşlemi

- **Yaklaşım :**

$$a = a_1 a_0, \quad a = a_1 10^{n/2} + a_0 \text{ and}$$

$$b = b_1 b_0, \quad b = b_1 10^{n/2} + b_0$$



Büyük Tamsayıları Çarpma İşlemi

$$c = a * b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$$

$$c = (a_1 * b_1) 10^n + (a_1 * b_0 + a_0 * b_1) 10^{n/2} + (a_0 * b_0)$$

$$c = c_2 10^n + c_1 10^{n/2} + c_0$$

burada

$$c_2 = a_1 * b_1 \rightarrow \text{ilk yarısının çarpımı}$$

$$c_0 = a_0 * b_0 \rightarrow \text{İkinci yarısının çarpımı}$$

$$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0) \rightarrow \text{azaltılmış çarpma işlemi}$$



Büyük Tamsayıları Çarpma İşlemi

- Eğer $n/2$ çift ise, Aynı metodу kullanmaya devam edebiliris c_2 , c_1 and c_0 .
- *n-basamaklı sayıları çarpmak için recursive bir algoritmamız bulunmaktadır*
- Recursion nerede durur
 - $n, 1$ olduğunda
 - Sayıları direk çarptığımızdaki yere kadar



Büyük Tamsayıları Çarpma İşlemi

- **Analiz:**
 - Kaç tane çarpma işlemi gerçekleştirdik ?



Büyük Tamsayıları Çarpma İşlemi

- **Analiz :**

n-haneli sayıları çarpmak 3 tane n/2 hanelik çarpma işlemi gerektiriyor.

number

$$M(n) = 3M(n/2) \text{ for } n > 1, \quad M(1) = 1$$

$$\begin{aligned} M(2^k) &= 3M(2^{k-1}) = 3[3M(2^{k-2})] = 3^2M(2^{k-2}) \\ &= \dots = 3^iM(2^{k-i}) = \dots = 3^kM(2^{k-k}) = 3^k. \end{aligned}$$

$$M(n) = 3^{\log_2 n} = n^{\log_2 3} \approx n^{1.585}.$$



Büyük Tamsayıları Çarpma İşlemi

Tartışma:

- Günümüzde bir çok problemde kullanılmaktadır.
 - Criptoğrafi
 - Mobil aygıtların güvenlik birimlerinde
- 600 basamaktan sonra kağıt kalem metodundan iyi sonuç vermektedir.



Matris Çarpımı

- **Problem Tanımı:**

İki adet $n \times n$ matrisin çarpımını bulmak.

$$C = A \times B$$

- n^3 adet çarma işleminden daha az çarpma ile gerçekleştirilebilir.



Matris Çarpımı

Divide & conquer Stratejisi:

- A ve B iki tane $n \times n$ matris olsun, n de çift sayı olsun.
- A , B ve Çarpımları C 'yi dört adet $n/2 \times n/2$ alt matrise bölebiliriz.

$$\begin{array}{|c|c|} \hline ae + bg & af + bh \\ \hline ce + dg & cf + dh \\ \hline \end{array} = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} * \begin{array}{|c|c|} \hline e & f \\ \hline g & h \\ \hline \end{array}$$



Matris Çarpımı

8 Alt -Problem:

$$\begin{matrix} a \\ \hline \end{matrix} * \begin{matrix} e \\ \hline \end{matrix}$$

$$\begin{matrix} a \\ \hline \end{matrix} * \begin{matrix} f \\ \hline \end{matrix}$$

$$\begin{matrix} c \\ \hline \end{matrix} * \begin{matrix} e \\ \hline \end{matrix}$$

$$\begin{matrix} c \\ \hline \end{matrix} * \begin{matrix} f \\ \hline \end{matrix}$$

$$\begin{matrix} b \\ \hline \end{matrix} * \begin{matrix} g \\ \hline \end{matrix}$$

$$\begin{matrix} b \\ \hline \end{matrix} * \begin{matrix} h \\ \hline \end{matrix}$$

$$\begin{matrix} d \\ \hline \end{matrix} * \begin{matrix} g \\ \hline \end{matrix}$$

$$\begin{matrix} d \\ \hline \end{matrix} * \begin{matrix} h \\ \hline \end{matrix}$$

Analiz:

- 8 çarpma işlemi $\rightarrow (n/2) \times (n/2)$ matrix
 - 4 toplama işlemi $\rightarrow (n/2) \times (n/2)$ matrix
-
- $$\bullet T(n) = 8 * T(n/2) + \Theta(n^2) = \Theta(n^3)$$



Strassen'in Matris Çarpımı

- n^3 çarpmadan daha az sayıda çarpma işlemi gerçekleştirmek
- 2-by-2 matrix çarpımını düşünelim
 - 8 çarpma yerine 7 adet çarpma gerektiriyor.



Strassen'in Matris Çarpımı

Aşağıdaki formülleri kullanabiliriz

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ b_{10} & b_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$
$$= \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

burada

$$m_1 = (a_{00} + a_{11}) * (b_{00} + b_{11})$$

$$m_2 = (a_{10} + a_{11}) * b_{00}$$

$$m_3 = a_{00} * (b_{01} - b_{11})$$

$$m_4 = a_{11} * (b_{10} - b_{00})$$

$$m_5 = (a_{00} + a_{01}) * b_{11}$$

$$m_6 = (a_{10} - a_{00}) * (b_{00} + b_{01})$$

$$m_7 = (a_{01} - a_{11}) * (b_{10} + b_{11}).$$

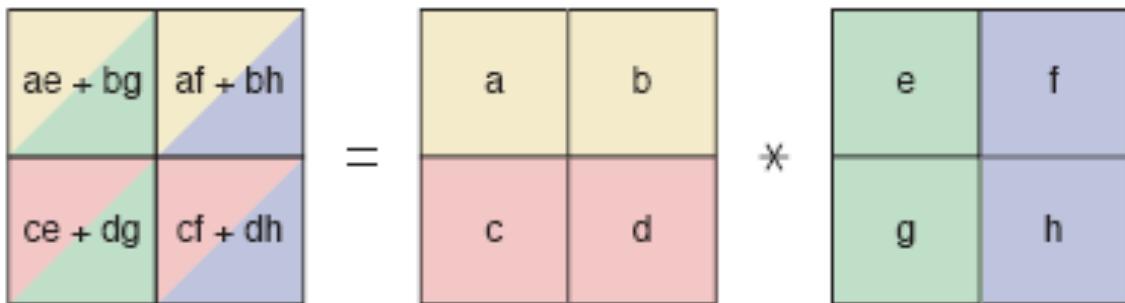


Strassen'in Matrix Çarpımı

- 7 tane çarpma işlemi var.
- Kaç tane toplama işlemi var?
- 2×2 'lik matrisler için bunu kullanmak uygun mudur?



Strassen'in Matrix Çarpımı



7 çarpma işlemi işlemi

$$P1 = a * (f-h)$$

$$P5 = (a+d) * (e+h)$$

$$P2 = (a+b) * h$$

$$P6 = (b-d) * (g+h)$$

$$P3 = (c+d) * e$$

$$P7 = (a-c) * (e+f)$$

$$P4 = d * (g-e)$$

Solution:

$$a * e + b * g = P5 + P4 - P2 + P6$$

$$a * f + b * h = P1 + P2$$

$$c * e + b * h = P3 + P4$$

$$c * f + d * h = P5 + P1 - P3 - P7$$



Strassen'in Matrix Çarpımı

Yaklaşım:

- Let A ve B $n \times n$ 'lik iki matris
 - n 2'nin bir kuvveti olsun
- A ve B 'yi $n/2 \times -n/2$ 'lik alt matrislere bölmek
- 7 alt matris çarpımı özyineli bir şekilde gerçekleştirilir.
- C matrisini elde etmek için gerekli toplama işlemleri yapılır.



Strassen'in Matris Çarpımı

- Analysis :

$$M(n) = 7M(n/2) \text{ for } n > 1, \quad M(1) = 1.$$

Since $n = 2^k$,

$$\begin{aligned} M(2^k) &= 7M(2^{k-1}) = 7[7M(2^{k-2})] = 7^2M(2^{k-2}) = \dots \\ &= 7^i M(2^{k-i}) \dots = 7^k M(2^{k-k}) = 7^k. \end{aligned}$$

Since $k = \log_2 n$,

$$M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807},$$



Strassen'in Matris Çarpımı

Tartışma:

- Toplama işlemlerini kontrol etmemiz lazım $A(n)$
- $A(n) \in \Theta(n^{\log_2 7})$
- Brute force'dan daha iyi
 - Brute force algoritması n^3
- Bellek kullanımı iyi mi?
- Matris çarpımı için en iyi algoritma değil
 - Coopersmith ve Winograd algorithm's verimi $O(n^{2.376})$



ROAD MAP

- **Divide And Conquer**
 - Binary Search
 - Maximum Subsequence Problem
 - Merge Sort
 - Quick Sort
 - Multiplication of Large Integers
 - Strassen's Matrix Multiplication
 - **Closest Pair of Points**
 - **Convex Hull**



Closest-Pair Problemi

- **Problem Tanımı:**

n adet düğümün içerisinde en yakın olanları bulma.

- Burada problemin iki boyutlu halini kullanacağız
- Noktakar kartezyen koordinatları ile (x,y) şeklinde verilmiş.
- İki nokta $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$ arasındaki uzaklığın standart öklit uzaklığını olduğunu varsayıyoruz.

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$



Closest-Pair Problemi

Yaklaşım:

1. Noktaları iki kümeye ayıriyoruz $n/2$ nokta içeren S_1 ve S_2 kümeleri olarak. Bölerken $x=c$ doğrusuna göre bölüyoruz.
 - C , x koordinatlarının ortancası
 - Dolayısıyla $n/2$ nokta doğrunun sağında ve $n/2$ nokta ve kendisi noktanın sağında bulunmaktadır.



Closest-Pair Problem

Approach :

2. S_1 ve S_2 'nin içindeki en yakın noktaları recursive bir şekilde bul

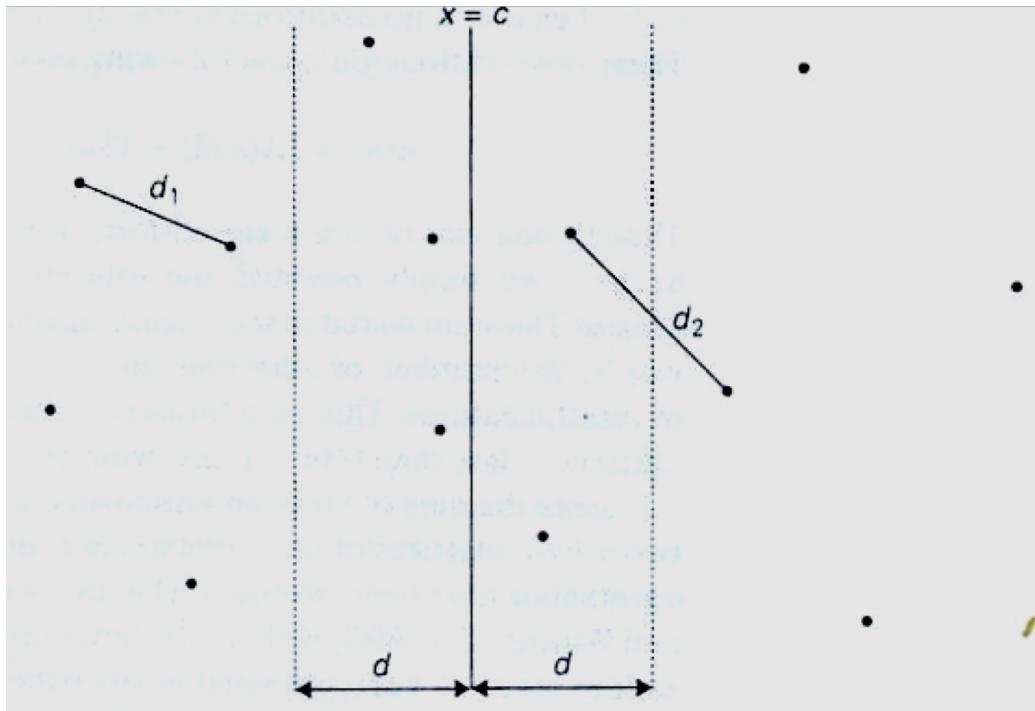
- d_1 ve d_2 sırasıyla S_1 ve S_2 'deki en yakın mesafeler olsun

let $d = \min\{d_1, d_2\}$

- d globaldeki minimum mesafe olmak zorunda değil
 - Daha yakın bir nokta çifti doğrunun iki karşılıklı tarafında bulunabilir.
 - Doğrunun d solu ve d sağı alındığında oluşan dikörtgendeki noktalar bir daha taranmalıdır.



Closest-Pair Problem



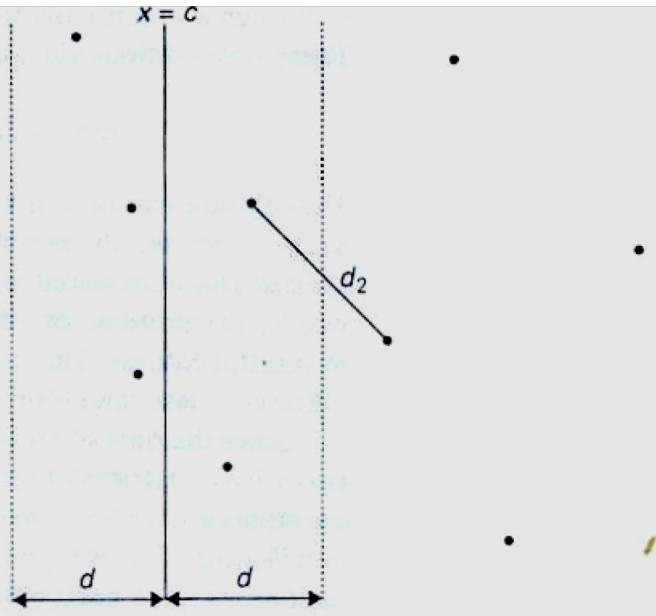
closest-pair problemı d&c yaklaşım



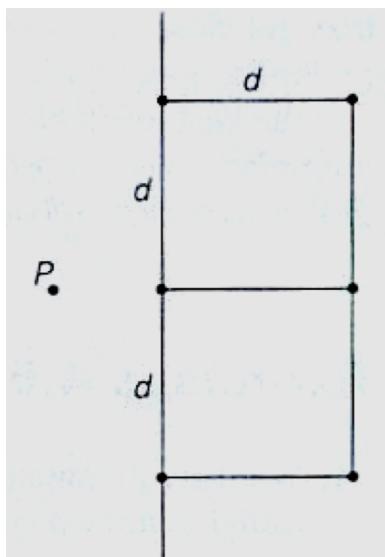
Closest-Pair Problemi

Approach :

3. C_1 ve C_2 doğrunun x eksenine göre d birim solundaki ve d birim sağindaki noktalar kümesi olsun.
4. C_1' deki $P(x,y)$ her nokta için, C_2' de bu noktaya d'den daha yakın bir nokta var mı diye bak.
5. C_2' deki noktaların y koordinatı da $[y-d, y+d]$ aralığında olmalıdır.
 - Bu şekilde en fazla 6 nokta vardır



- (a) Idea of the d&c algorithm for the closest-pair problem



- (b) The six points that may need to be examined for point P (worst case)



Closest-Pair Problem

Yaklaşım:

6. C_1 ve C_2 noktaları için kendi y koordinatlarına göre artan sırada bir liste oluşturulmalı
7. C_1' deki noktaların C_2 dekilere uzaklığını tara.



ALGORITHM *EfficientClosestPair(P, Q)*

```
//Solves the closest-pair problem by divide-and-conquer
//Input: An array P of  $n \geq 2$  points in the Cartesian plane sorted in
//       nondecreasing order of their x coordinates and an array Q of the
//       same points sorted in nondecreasing order of the y coordinates
//Output: Euclidean distance between the closest pair of points
if  $n \leq 3$ 
    return the minimal distance found by the brute-force algorithm
else
    copy the first  $\lceil n/2 \rceil$  points of P to array  $P_l$ 
    copy the same  $\lceil n/2 \rceil$  points from Q to array  $Q_l$ 
    copy the remaining  $\lfloor n/2 \rfloor$  points of P to array  $P_r$ 
    copy the same  $\lfloor n/2 \rfloor$  points from Q to array  $Q_r$ 
     $d_l \leftarrow EfficientClosestPair(P_l, Q_l)$ 
     $d_r \leftarrow EfficientClosestPair(P_r, Q_r)$ 
     $d \leftarrow \min\{d_l, d_r\}$ 
     $m \leftarrow P[\lceil n/2 \rceil - 1].x$ 
    copy all the points of Q for which  $|x - m| < d$  into array  $S[0..num - 1]$ 
     $dminsq \leftarrow d^2$ 
    for  $i \leftarrow 0$  to  $num - 2$  do
         $k \leftarrow i + 1$ 
        while  $k \leq num - 1$  and  $(S[k].y - S[i].y)^2 < dminsq$ 
             $dminsq \leftarrow \min((S[k].x - S[i].x)^2 + (S[k].y - S[i].y)^2, dminsq)$ 
             $k \leftarrow k + 1$ 
    return  $\sqrt{dminsq}$ 
```



Closest-Pair Problem

- Analiz :

$T(n)$, n adet önceden sıralanmış nokta için:

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

$$T(n) \in O(n \log n)$$