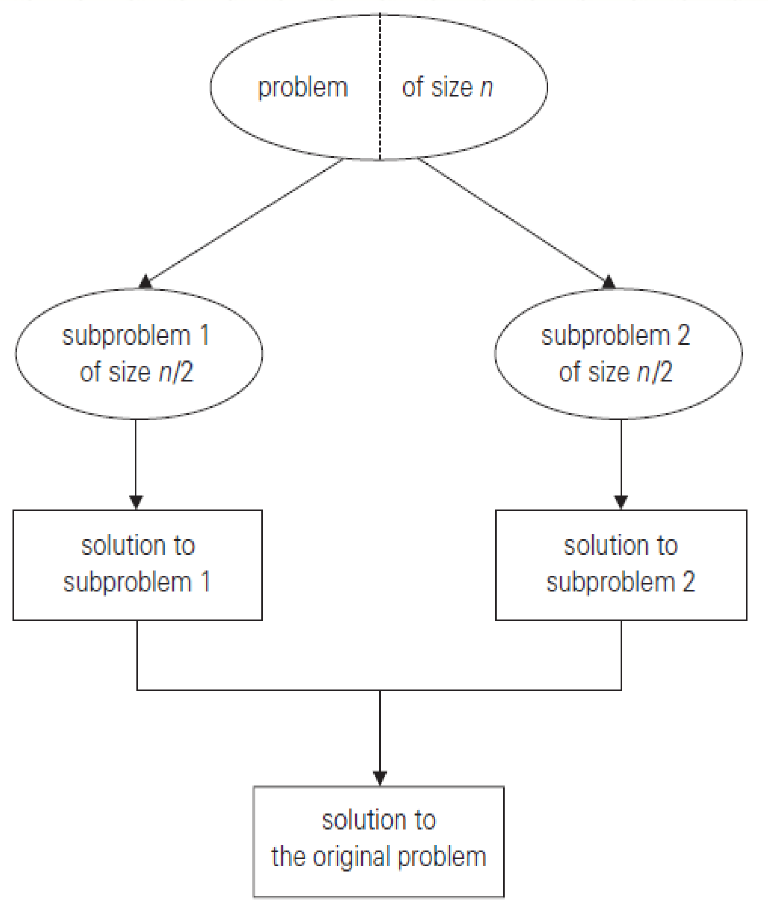
YZM 3207- ALGORİTMA ANALİZİ VE TASARIM

DERS#5: BÖL VE FETHET YÖNTEMİ

Böl ve Fethet Algoritmaları

- Böl ve fethet
  - En sık kullanılan algoritma tasarım tekniklerinden biri

1. Problem alt problemlere bölünür
   - Genellikle eşit büyüklükte
2. Alt problemler çözülür
3. Gerekliyse alt problem çözümleri birleştirilir



Master Teoremi

- Bir özyineleme ilişkisi için

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

Master Theorem If $f(n) \in \Theta(n^d)$ where $d \geq 0$ in recurrence

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d, \\ \Theta(n^d \log n) & \text{if } a = b^d, \\ \Theta(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

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

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

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

Birleştirmeli Sıralama

- Birleştirmeli Sıralama (Merge Sort)
  - Bir sırasız diziyi sıralı hale getirme
  - Diziyi özyinelemeli olarak ikiye ayırır
  - En küçük hale getirdikten sonra karşılaştırarak birleştirir

Birleştirmeli Sıralama

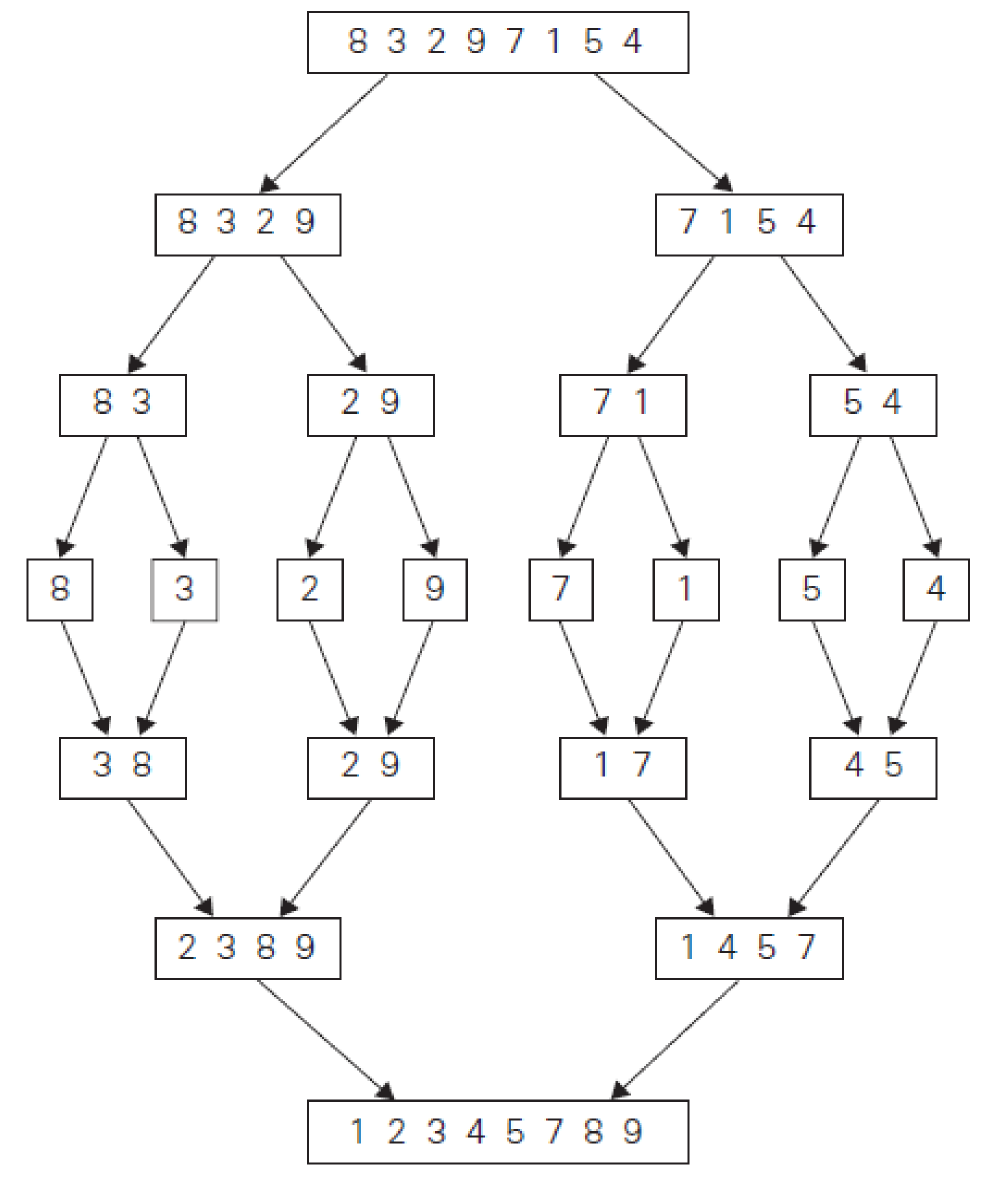
ALGORITHM *Mergesort*($A[0..n-1]$)

//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) //see below

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

//Merges two sorted arrays into one sorted array
//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 $B[i] \le C[j]$
    $A[k] \leftarrow B[i]$; $i \leftarrow i + 1$
  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 $A[k..p+q-1]$

Birleştirmeli Sıralama



Birleştirmeli Sıralama

- Özyineleme ilişkisi $C(n) = 2C(n/2) + C_{merge}(n) \quad \text{for } n > 1, \quad C(1) = 0.$
- $C_{merge}(n)$ incelenirse:
  - Her adımda bir karşılaştırma
    - Eğer dizi indisleri bir azaltılmaya devam ediyorsa
      - Yani dizilerden biri boşalmadıysa
    - En kötü durum:
      - Bir dizide tek eleman kaldığında diğerinin boşalmamış olması $C_{worst}(n) = 2C_{worst}(n/2) + n - 1 \quad \text{for } n > 1, \quad C_{worst}(1) = 0.$
      - $C_{merge}(n) = (n - 1)$

Master Theorem If $f(n) \in \Theta(n^d)$ where $d \geq 0$ in recurrence

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d, \\ \Theta(n^d \log n) & \text{if } a = b^d, \\ \Theta(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

- Master Teoremine göre
  - a=2, b=2, d=1

$C_{worst}(n) \in \Theta(n \log n)$

QuickSort

- Böl ve Fethet sıralama algoritması
- Merge sort'tan farkı
  - Diziyi alt dizilere konumlarına göre değil
  - Değerlerine göre böler

$$\underbrace{A[0] \ldots A[s-1]}_{\text{all are } \le A[s]} \quad A[s] \quad \underbrace{A[s+1] \ldots A[n-1]}_{\text{all are } \ge A[s]}$$

QuickSort

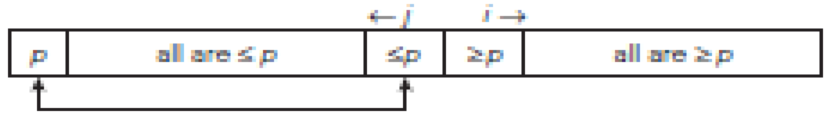
ALGORITHM *Quicksort*$(A[l..r])$

//Sorts a subarray by quicksort
//Input: Subarray of array $A[0..n-1]$, defined by its left and right
//   indices l and r
//Output: Subarray $A[l..r]$ sorted in nondecreasing order
if $l < r$
  $s \leftarrow Partition(A[l..r])$ //s is a split position
  $Quicksort(A[l..s-1])$
  $Quicksort(A[s+1..r])$

QuickSort

- Bir eleman pivot seçilir
  - Dizi biri pivot olmak üzere 3 alt diziye ayrılır
  - Diğer iki dizi arasında pivottan küçük elemanlar ile pivottan büyük elemanlar değiştirilir

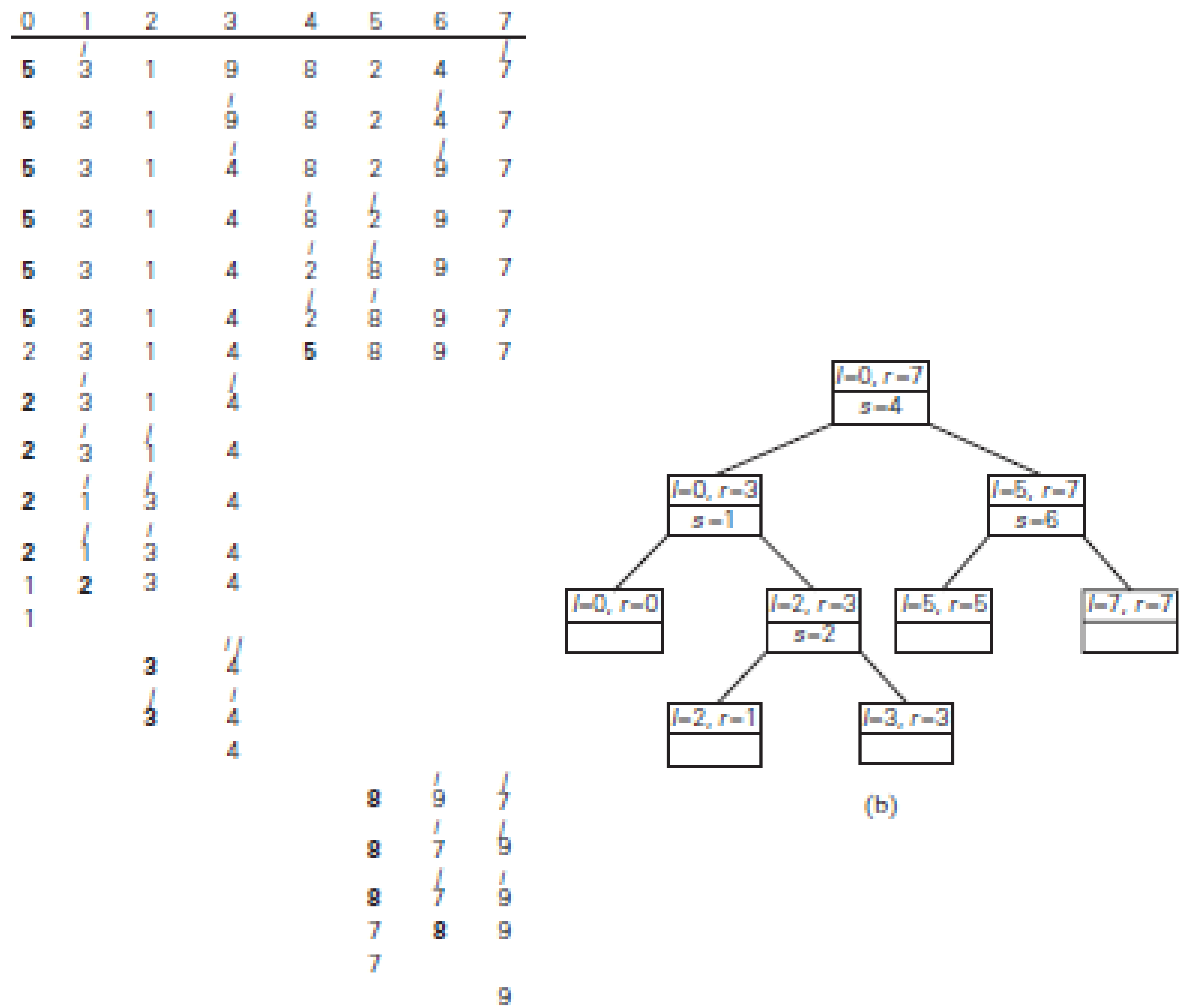
  - Pivotun solunda pivottan büyük elemanlar olacak şekilde yer değişimi yapılır



QuickSort

```
ALGORITHM  HoarePartition(A[l..r])
    //Partitions a subarray by Hoare's algorithm, using the first element
    //          as a pivot
    //Input: Subarray of array A[0..n − 1], defined by its left and right
    //          indices l and r (l < r)
    //Output: 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
```

QuickSort



(a) (b)

QuickSort

- En iyi durum:

$$C_{best}(n) = 2C_{best}(n/2) + n \quad \text{for } n > 1, \;\; C_{best}(1) = 0.$$

  – Master Teoremine göre: $C_{best}(n) \in \Theta(n \log_2 n)$

- En kötü Durum:

$$C_{worst}(n) = (n+1) + n + \cdots + 3 = \frac{(n+1)(n+2)}{2} - 3 \in \Theta(n^2).$$

- Ortalama Durum:

$$C_{avg}(n) = \frac{1}{n}\sum_{s=0}^{n-1}[(n+1) + C_{avg}(s) + C_{avg}(n-1-s)] \quad \text{for } n > 1,$$

Çok Basamaklı Sayıların Çarpılması

Çok basamaklı iki sayının çarpımı

A = 12345678901357986429   B = 87654321284820912836

Normal Çarpma Algoritması:

$$
\begin{array}{r}
a_1\ a_2 \ldots\ a_n \\
b_1\ b_2 \ldots\ b_n \\
(d_{10})\, d_{11} d_{12} \ldots d_{1n} \qquad\quad \\
(d_{20})\, d_{21} d_{22} \ldots d_{2n} \qquad\qquad\qquad\qquad \\
\ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots \qquad\qquad\qquad\quad \\
(d_{n0})\, d_{n1} d_{n2} \ldots d_{nn} \qquad\qquad\qquad\qquad\qquad
\end{array}
$$

Etkinliği: n^2

Böl ve Fethet ile Çarpım

Örnek: $A * B = ?$ $A = 2135$ and $B = 4014$

$A = (21 \cdot 10^2 + 35)$, $B = (40 \cdot 10^2 + 14)$

, $A * B = (21 \cdot 10^2 + 35) * (40 \cdot 10^2 + 14)$

$= 21 * 40 \cdot 10^4 + (21 * 14 + 35 * 40) \cdot 10^2 + 35 * 14$

Algoritma: $A = A_1 A_2$ ve $B = B_1 B_2$ ise (A ve B n-basamaklı, A_1, A_2, B_1, B_2 ise $n/2$-basamaklı sayılar),

$A * B = A_1 * B_1 \cdot 10^n + (A_1 * B_2 + A_2 * B_1) \cdot 10^{n/2} + A_2 * B_2$

Özyineleme ilişkisi: $M(n)$:

$$M(n) = 3M(n/2), \quad n>1, M(1) = 1$$

Strassen’in Matris Çarpım Algoritması

- Strassen’in matris çarpımı için geliştirdiği algoritma

$$\begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} * \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$

$$= \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 + M_3 - M_2 + M_6 \end{pmatrix}$$

$$M_1 = (A_{00} + A_{11}) * (B_{00} + B_{11})$$

$$M_2 = (A_{10} + A_{11}) * B_{00}$$

$$M_3 = A_{00} * (B_{01} - B_{11})$$

$$M_4 = A_{11} * (B_{10} - B_{00})$$

$$M_5 = (A_{00} + A_{01}) * B_{11}$$

$$M_6 = (A_{10} - A_{00}) * (B_{00} + B_{01})$$

$$M_7 = (A_{01} - A_{11}) * (B_{10} + B_{11})$$

Strassen'in Matris Çarpım Algoritması

- Matrisler ikinin katı değilse sıfır ile tamamlanır

Çarpım Sayısı: $M(n) = 7M(n/2), \quad M(1) = 1$

Çözüm:

$M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807}$ vs. n^3 (Kaba kuvvet)