Algorithms

Lecture 8

Quick Sort

Quick Sort

- **Quick Sort:**

  - Quick Sort description:

    1. Divide:

       ✓ Partition (rearrange) the array A[p .. r] into two (possibly empty) subarrays $A[p..q-1]$ and $A[q+1..r]$ such that:

       ⇒ each element of $A[p..q-1]$ is less than or equal to $A[q]$

       ⇒ $A[q]$ is less than or equal to each element of $A[q+1..r]$

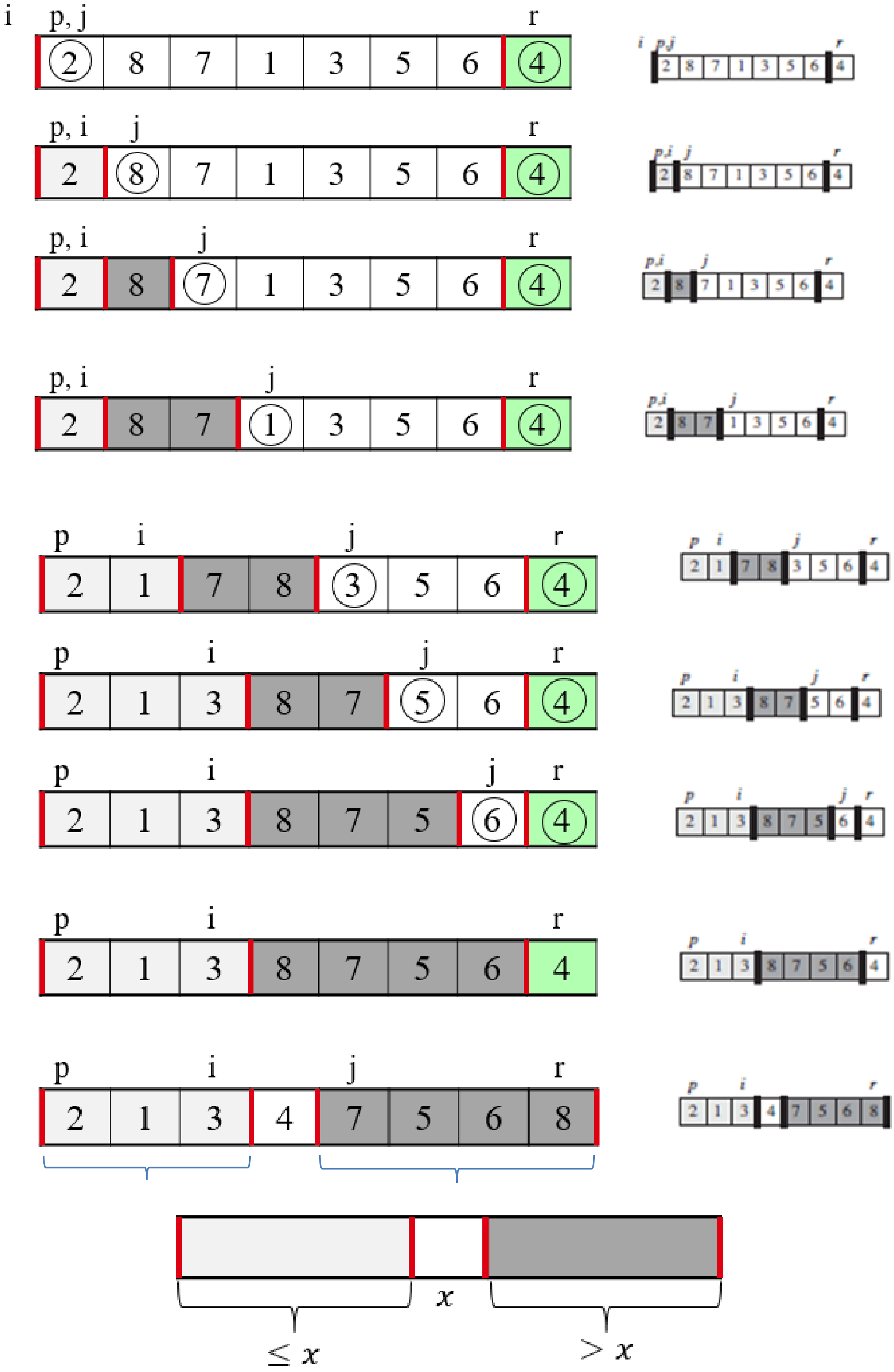
    2. Conquer:

       ✓ Sort the two subarrays $A[p..q-1]$ and $A[q+1..r]$ by recursive calls to quicksort.

    3. Combine:

       ✓ Combine the two subarrays so the entire array $A[p..r]$ is now sorted Because the subarrays are already sorted

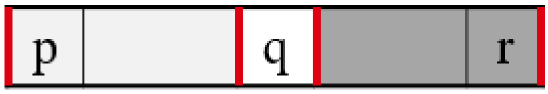
  - Quick Sort Example:



i p, j r
2 8 7 1 3 5 6 4
p, i j r
2 8 7 1 3 5 6 4
p, i j r
2 8 7 1 3 5 6 4
p, i j r
2 8 7 1 3 5 6 4
p i j r
2 1 7 8 3 5 6 4
p i j r
2 1 3 8 7 5 6 4
p i j r
2 1 3 8 7 5 6 4
p i r
2 1 3 8 7 5 6 4
p i j r
2 1 3 4 7 5 6 8
x
≤ x
> x

- <u>Quick sort algorithm:</u>

QUICKSORT (A, p, r)
    if $p < r$
        q = PARTITION (A, p, r)
        QUICKSORT (A, p, $q-1$)
        QUICKSORT (A, $q+1$, r)



- <u>Partition algorithm:</u>

PARTITION (A, p, r)
    $x = A[r]$
    $i = p - 1$
    for $j = p$ to $r - 1$
        if $A[j] \leq x$
            $i = i + 1$
            exchange $A[i]$ with $A[j]$
    exchange $A[i+1]$ with $A[r]$
    return $i + 1$

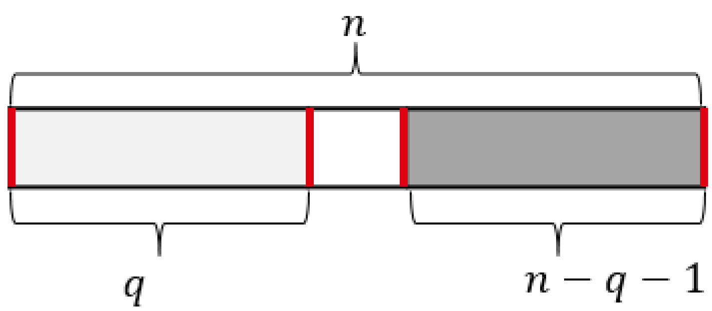
- <u>Quick sort algorithm calling:</u>

QUICKSORT (A, 1, A.length)

- <u>Quick Sort Analysis:</u>

PARTITION (A, p, r)	
$x = A[r]$	1
$i = p - 1$	1
for $j = p$ to $r - 1$	n
if $A[j] \leq x$	$n - 1$
$i = i + 1$	$n - 1$
exchange $A[i]$ with $A[j]$	$n - 1$
exchange $A[i + 1]$ with $A[r]$	1
return $i + 1$	1

QUICKSORT (A, p, r)	
if $p < r$	1
q = PARTITION (A, p, r)	$\Theta(n)$
QUICKSORT (A, p, $q - 1$)	T(q)
QUICKSORT (A, $q + 1$, r)	$T(n - q - 1)$



<u>Recurrence Equation:</u>

$$T(n) = T(q) + T(n - q - 1) + \Theta(n)$$

Worst case:

- one sub-problem with $n - 1$ elements and one with 0 elements

$$T(n) = T(n-1) + T(0) + \Theta(n)$$

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

- Using Tree Method:

$$\begin{array}{c|cc}
 & cn & cn \\
 & | & \\
h = n & c(n-1) & c(n-1) \\
 & | & \\
 & c(n-2) & c(n-2) \\
 & \vdots & \vdots \\
 & \vdots & \vdots \\
 & 1 & c
\end{array}$$

$$T(n) = cn + c(n-1) + c(n-2) + \cdots + c$$

$$= c\sum_{i=1}^{n} i = c\,\frac{n\,(n+1)}{2} = \frac{cn^2}{2} + \frac{cn}{2}$$

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

Best case:

- Two sub-problems, each of size no more than $n/2$

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

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

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

- Using <u>Master Method:</u>

$$a = 2,\ \ b = 2,\ \ f(n) = n$$

$$n^{\log_b a} = n^{\log_2 2} = \text{n}$$

$$n^{\log_b a} = f(n) \quad \rightarrow \quad \text{case 2}$$

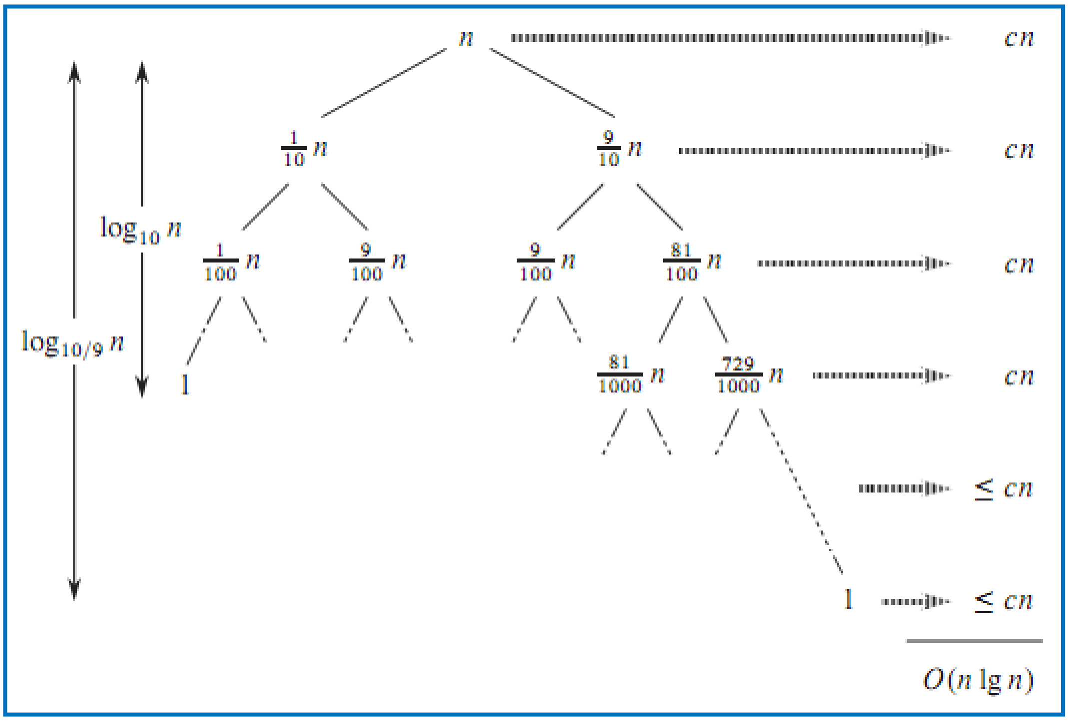
$$\therefore T(n) = \Theta(f(n) \lg n) = \Theta(n \lg n)$$

Average case:

- <u>Partitioning algorithm</u> always produces a 9-to-1 proportional split

$$T(n) = T(9n/10) + T(n/10) + \Theta(n)$$

- Using <u>Tree Method:</u>



$$T(n) = \sum_{i=1}^{h} c\,n = c\,n \sum_{i=1}^{\log_{10/9} n} 1 = c\,n \log_{10/9} n$$

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

Summary:

- Best Case:
  - ✓ Two sub-problems, each of size no more than $n/2$

$$\boxed{T(n) = \Theta(n \lg n)}$$

- Worst Case:
  - ✓ one sub-problem with $n - 1$ elements and one with 0 elements

$$\boxed{T(n) = \Theta(n^2)}$$

- Average Case:
  - ✓ Partitioning algorithm always produces a 9-to-1 proportional split

$$\boxed{T(n) = \Theta(n \lg n)}$$

- Randomized quicksort
  - Running time is independent of the input order.
  - No assumptions need to be made about the input distribution.
  - No specific input elicits the worst-case behavior.
  - The worst case is determined only by the output of a random-number generator.
- Quicksort in practice
  - Quicksort is typically over twice as fast as merge sort.