PES UNIVERSITY ONLINE

# DESIGN AND ANALYSIS OF ALGORITHMS
## UE19CS251

**Shylaja S S**
Department of Computer Science
& Engineering

# DESIGN AND ANALYSIS OF ALGORITHMS

## Quick Sort

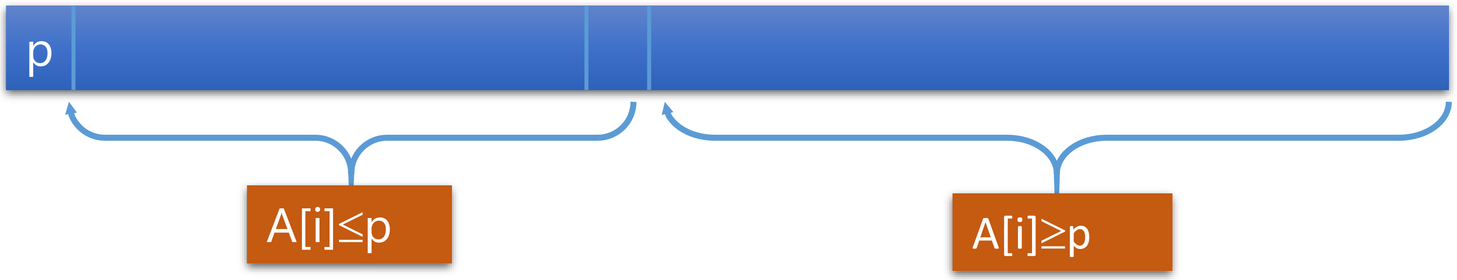
Major Slides Content: Anany Levitin

**Shylaja S S**

Department of Computer Science & Engineering

## Quick Sort

- Select a pivot (partitioning element) – here, the first element
- Rearrange the list so that all the elements in the first s positions are smaller than or equal to the pivot and all the elements in the remaining n-s positions are larger than or equal to the pivot




- Exchange the pivot with the last element in the first (i.e., ≤) subarray — the pivot is now in its final position
- Sort the two subarrays recursively

## Quick Sort - Algorithm

```
ALGORITHM  Quicksort(A[l .. 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: Subarray A[l .. r]  sorted in non decreasing order
if l  <  r
   s ←Partition(A[l .. r])          //s is a split position
   Quicksort(A[l  .. s - 1])
   Quicksort(A[s + 1 .. r])
```

## Quick Sort - Algorithm

```
ALGORITHM  Partition(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[i],  A[j])
return  j
```

## Quick Sort - Example

5 3 1 9 8 2 4 7

2 3 1 4 5 8 9 7

1 2 3 4 5 7 8 9

1 2 3 4 5 7 8 9

1 2 3 4 5 7 8 9

1 2 3 4 5 7 8 9

## Quick Sort - Analysis: Best Case

- The number of comparisons in the best case satisfies the recurrence:
- $C_{best}(n) = 2C_{best}(n/2) + n$ for $n > 1$, $C_{best}(1) = 0$

- According to Master Theorem

$$C_{best}(n) \in \Theta(n \log_2 n)$$

## Quick Sort - Analysis: Worst Case

- The number of comparisons in the worst case satisfies the recurrence

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

## Quick Sort - Analysis: Average Case

Let $C_{avg}(n)$ be the number of key comparisons made by Quick Sort on a randomly ordered array of size n

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

The solution for the above recurrence is:

$$C_{avg}(n) \approx 2n\ln n \approx 1.38n\log_2 n$$

# THANK YOU

**Shylaja S S**

Department of Computer Science & Engineering

**shylaja.sharath@pes.edu**