



# DESIGN AND ANALYSIS OF ALGORITHMS

## UE19CS251

---

**Shylaja S S**

Department of Computer Science  
& Engineering

# DESIGN AND ANALYSIS OF ALGORITHMS

---

## Merge Sort

Major Slides Content: Anany Levitin

**Shylaja S S**

Department of Computer Science & Engineering

## Merge Sort - Idea

---

- Split array  $A[0..n-1]$  into about equal halves and make copies of each half in arrays B and C
- Sort arrays B and C recursively
- Merge sorted arrays B and C into array A as follows:
  - Repeat the following until no elements remain in one of the arrays:
    - compare the first elements in the remaining unprocessed portions of the arrays
    - copy the smaller of the two into A, while incrementing the index indicating the unprocessed portion of that array
  - Once all elements in one of the arrays are processed, copy the remaining unprocessed elements from the other array into A

# DESIGN AND ANALYSIS OF ALGORITHMS

## Merge Sort - Algorithm

---

```
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 non decreasing 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.. $\lfloor n/2 \rfloor - 1$ ]
    Mergesort(B[0 ..  $\lfloor n/2 \rfloor - 1$ ])
    Mergesort(C[0 ..  $\lfloor n/2 \rfloor - 1$ ])
    Merge(B, C, A)
```

# DESIGN AND ANALYSIS OF ALGORITHMS

## Merge Sort - Algorithm

---



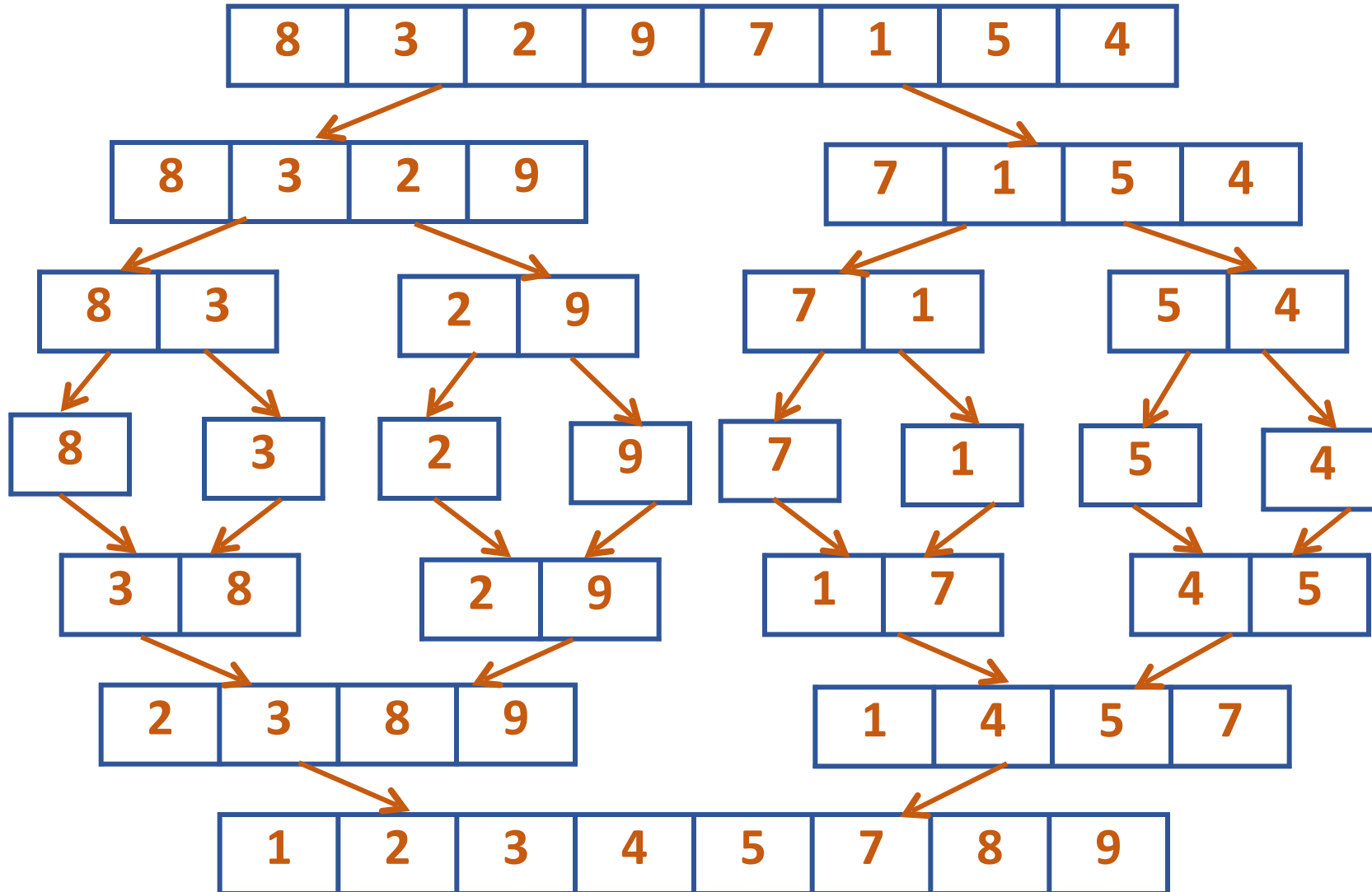
```
ALGORITHM Merge(B[0 .. p- 1], C[0 .. q -1], A[0 .. p + q -1])
//Merges two sorted arrays into one sorted array
//Input: Arrays B[0 .. p -1] and C[0 .. q -1] both sorted
//Output: Sorted array A[0 .. p + q -1] of the elements of B and
        C
i←0; j←0; k←0
while i < p and j < q do
    if B[i] ≤ C[j]
        A[k] ← B[i]; i ← i + 1
    else A[k] ←C[j]; j← j + 1
    k←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]
```

# DESIGN AND ANALYSIS OF ALGORITHMS

## Merge Sort - Example



**PES**  
UNIVERSITY  
ONLINE



- Assuming for simplicity that  $n$  is a power of 2, the recurrence relation for the number of key comparisons  $C(n)$  is:

$$C(n) = 2C(n/2) + C_{\text{merge}}(n) \text{ [for } n > 1], C(1) = 0$$

- The number of key comparisons performed during the merging stage in the worst case is:

$$C_{\text{merge}}(n) = n - 1$$

- Using the above equation:

$$C_{\text{worst}}(n) = 2C_{\text{worst}}(n/2) + n - 1 \text{ [for } n > 1], C_{\text{worst}}(1) = 0$$

- Applying Master Theorem to the above equation:

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



# THANK YOU

---

**Shylaja S S**

Department of Computer Science  
& Engineering

**[shylaja.sharath@pes.edu](mailto:shylaja.sharath@pes.edu)**