Pimpri Chinchwad Education Trust's

Pimpri Chinchwad College Of Engineering,

Nigdi,Pune-411044

Data structure Algorithm

Guided by Dr. Ashwini Matange

Case study base on merge sort

By:

Name: Harshwardhan Karanje

PRN:124B1B312

Dept. Comp, Div : B

By:

Name: Snehankit Zore

PRN:124B1B133

Dept. Comp, Div : B

By:

Name: Harshal Magar

PRN: 124B1B135

Dept. Comp, Div : B

By:

Name: Uday Barapatre

PRN:124B1B132

Dept. Comp, Div : B

1. Introduction

Sorting is one of the most important operations in computer science. It is used in searching, databases, data analysis, and many optimization tasks. Among different sorting methods, **Merge Sort** is special because it always runs in

O(n log n) time—whether the input is best, average, or worst case.

This case study explains the basics of Merge Sort: where it comes from, how it works, its step-by-step logic (pseudo-code), performance, and real-world uses. To make things clear, simple diagrams are also included.

2. Background

- **Inventor:** John von Neumann (1945)
- **Method Type:** Divide and Conquer
- **Main Features:**
  - **Stable:** Keeps equal elements in the same order.
  - **Memory Use:** Needs extra space of **O(n)**.
  - **Scalable:** Works well for large datasets.

3. Problem Statement

Imagine a company has unsorted employee performance scores:

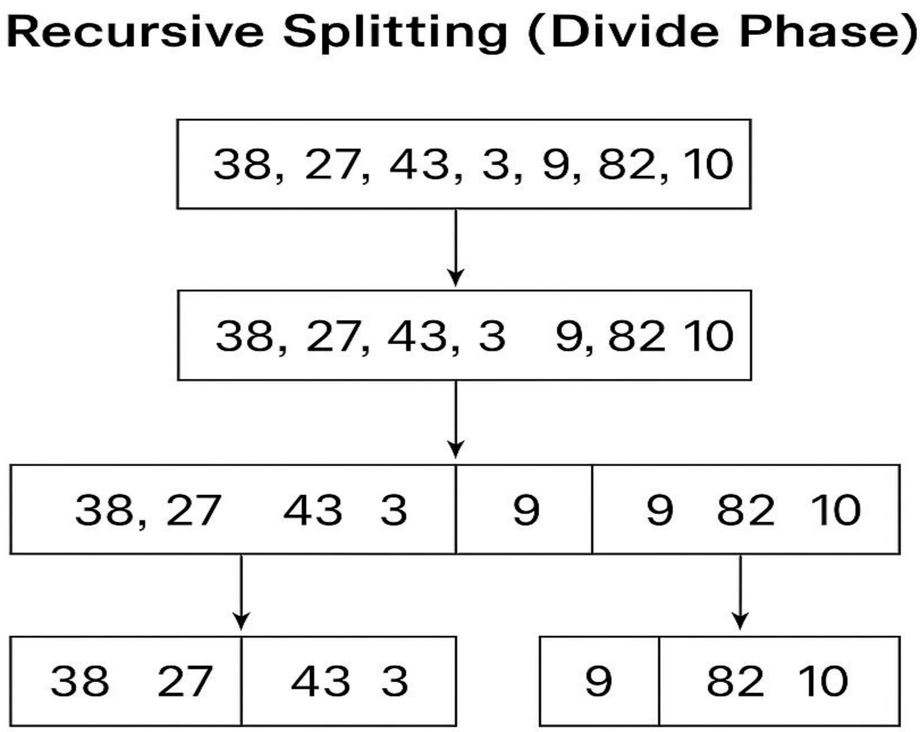
[38, 27, 43, 3, 9, 82, 10]

To generate accurate reports, sorting is essential. Quadratic-time methods like Bubble Sort become impractical on large data. Thus, Merge Sort is chosen for its efficiency and reliability.

4. Working of Merge Sort

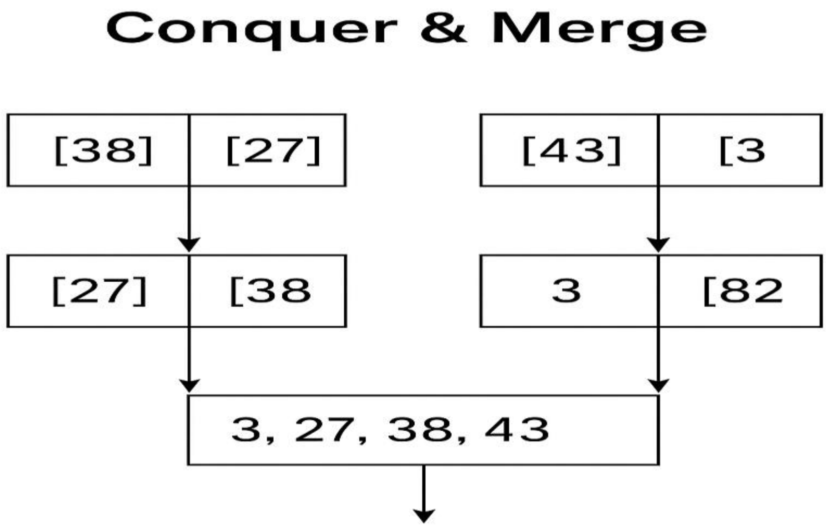
a) Divide

Breaks the array into halves recursively until each sub-array has only one element.



b) Conquer & Merge Pairs of single-element arrays are merged in ascending order. Examples:

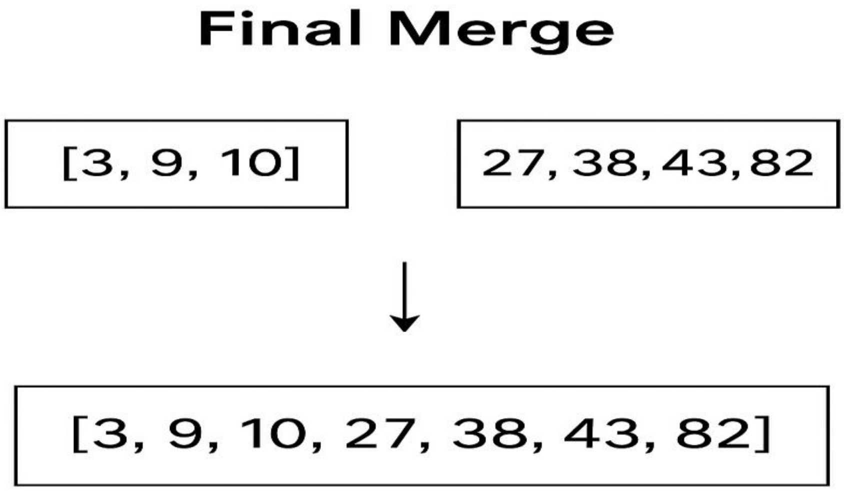
- [38] + [27] → [27, 38]
- [43] + [3] → [3, 43]



c) Combine

Finally, all sub-arrays merge to form the sorted array:

[3, 9, 10, 27, 38, 43, 82]



5. Algorithm Design & Pseudocode

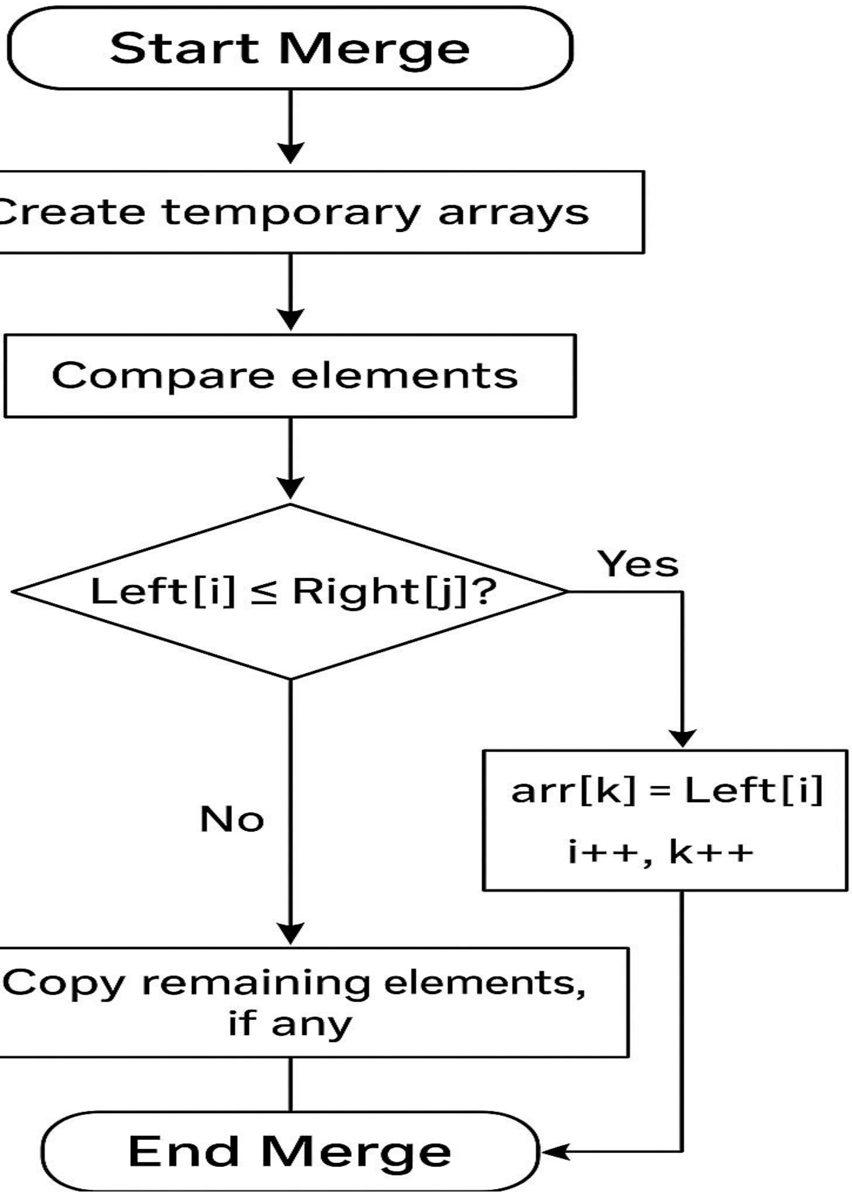
Pseudocode:

```
function mergeSort(arr, left, right):
    if left < right:
        mid = (left + right) / 2

        // Sort first half
        mergeSort(arr, left, mid)

        // Sort second half
        mergeSort(arr, mid + 1, right)

        // Merge the two halves
        merge(arr, left, mid, right)
function merge(arr, left, mid, right):

    // Create temporary arrays
    n1 = mid - left + 1
    n2 = right - mid
LeftArray[n1], RightArray[n2]
    for i = 0 to n1-1:
        LeftArray[i] = arr[left + i]
    for j = 0 to n2-1:
        RightArray[j] = arr[mid + 1 + j]
    // Merge the temporary arrays back
    i = 0, j = 0, k = left
    while i < n1 and j < n2:
        if LeftArray[i] <= RightArray[j]:
            arr[k] = LeftArray[i]
 i++
```

```
  else:
    arr[k] = RightArray[j]
    j++
  k++
```

// Copy remaining elements (if any)

```
while i < n1:
  arr[k] = LeftArray[i]
  i++, k++
while j < n2:
  arr[k] = RightArray[j]
  j++, k++
```



6. Runtime & Space Analysis

Time Complexity

- **Divide**: Creates log(n) levels due to halving each time.
- **Merge**: Each level performs O(n)work.
- **Total**: O(nlog(n)) in all cases

Space Complexity

Requires O(n) extra memory for temporary arrays during merging

Stability

It preserves the relative order of equal elements—important when sorting multi-attribute data

7. Comparison with Other Algorithms

Algorithm Comparison

Characteristic	Bubble Sort	Insertion Sort	Quick Sort	Merge Sort
Best Case	O(n)	O(n)	O(nlogttn)	O(nlogttn)
Worst Case	O(n2)	O(n2)	O(n2)	O(nlogttn)
Stable	Yes	Yes	No	Yes
Space	O(1)	O(1)	O(logttn)	O(n)

- **Merge Sort offers consistent performance and stability, though at the cost of extra memory.**

8. Real-World Applications

- External Sorting / Big Data: Handles datasets larger than memory via merge-based external sorting.
- Linked Lists: Efficient since random access is not needed.
- Parallel Processing: Easily parallelizable via independent subproblem decomposition.
- Databases & OS Scheduling: Used where stability is critical (e.g., multi-criteria sorting).

9. Conclusion

Merge Sort is both simple and powerful. It always runs in **O(n log n)** time, making it very useful for sorting large amounts of data. It also keeps equal elements in order (stable) and works well when scalability is needed. Although it uses extra memory, its reliability and efficiency make it a strong choice in real-world applications.

10. References

Wikipedia , https://www.geeksforgeeks.org/ and etc.