Sorting – Merge Sort

1. Introduction

Merge Sort is an efficient, comparison-based sorting algorithm that follows the **Divide and Conquer** strategy.

Unlike simple sorting algorithms, Merge Sort divides the array into smaller subarrays, sorts them, and then **merges them back** into a single sorted array.

Merge Sort is widely used because of its **guaranteed performance**, even for large datasets.

---

2. What is Merge Sort?

Merge Sort works in two main phases:

1. **Divide:**

   Split the array into two halves repeatedly until each subarray contains only one element.

2. **Merge:**

   Combine the subarrays back together in sorted order.

This process continues until the entire array is merged and sorted.

---

3. Why Do We Learn Merge Sort?

Merge Sort is important because:

- It has predictable performance
- Efficient for large datasets
- Introduces divide-and-conquer technique
- Forms the foundation for advanced algorithms
- Frequently asked in exams and interviews

---

4. Basic Idea Behind Merge Sort

The core idea is:

- Break a big problem into smaller problems
- Solve the smaller problems
- Combine the solutions to solve the original problem

Each merge operation combines two sorted arrays into one sorted array.

---

5. Logic for Merge Sort (Plain English)

1. Divide the array into two halves
2. Recursively sort the left half
3. Recursively sort the right half
4. Merge the two sorted halves
5. Repeat until the full array is sorted

---

6. Visualization of Merge Sort

Consider the array:

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

Divide:

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

Divide further:

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

Merge:

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

Final merge:

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

---

7. Merge Process Explanation

During merging:

- Compare the first elements of both subarrays
- Pick the smaller element
- Move the pointer forward
- Repeat until all elements are merged

This ensures the final array remains sorted.

---

8. Time and Space Complexity

Case	Time Complexity
Best Case	O(n log n)
Average Case	O(n log n)
Worst Case	O(n log n)

- **Space Complexity:** O(n)

  (Extra memory required for merging)

---

9. Advantages of Merge Sort

- Very efficient for large datasets
- Stable sorting algorithm
- Predictable performance

- Works well with linked lists
- Suitable for external sorting

10. Limitations of Merge Sort

- Requires extra memory
- Slower than Quick Sort for small arrays
- Not an in-place sorting algorithm
- More complex to implement

11. Real-World Applications

- Sorting large datasets
- Database sorting operations
- External sorting (files)
- Used in standard libraries
- Data analytics and processing

12. Comparison with Other Sorting Algorithms

Algorithm	**Time Complexity**	**Space**	**Stability**
Bubble Sort	$O(n^2)$	O(1)	Stable
Selection Sort	$O(n^2)$	O(1)	Not Stable
Insertion Sort	$O(n^2)$	O(1)	Stable
Merge Sort	O(n log n)	O(n)	Stable

13. Summary

- Merge Sort uses divide and conquer
- Divides array into smaller parts

- Merges sorted subarrays
- Time complexity is O(n log n)
- Efficient for large datasets
