Sorting in DSA

- **Sorting** is the process of arranging data in a particular order, usually **ascending (small → large)** or **descending (large → small).**
- Data can be sorted based on **numbers, characters, strings, or custom rules.**

🔹 Why Sorting

Sorting is one of the most fundamental operations in computer science because:

1. **Searching becomes efficient** (e.g., Binary Search requires sorted data).
2. **Organized data** is easier to analyze and process.
3. Used in **optimization problems**, like shortest path algorithms.
4. Helps in **data representation** (ranking systems, leaderboards, logs).
5. Many algorithms (like merge, binary search tree, heaps) internally rely on sorted data.

Example: Sorting student marks in ascending order to assign ranks.

🔹 Types of Sorting Techniques

A) Internal Sorting

- Sorting done entirely in **main memory (RAM).**
- Examples: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort, Heap Sort.

B) External Sorting

- Used when data is **too large to fit in main memory** and requires **secondary storage (disk/SSD).**
- Example: External Merge Sort.

C) Stable vs. Unstable Sorting

- **Stable Sort**: Preserves the relative order of duplicate elements.
- **Unstable Sort**: May not preserve duplicate order.

D) Comparison-based Sorting

- Sorting is done by comparing elements.
- Examples: Bubble Sort, Quick Sort, Merge Sort, Heap Sort.

E) Non-comparison-based Sorting

- Sorting is not based on direct comparison but on techniques like counting or hashing.
- Examples: Counting Sort, Radix Sort, Bucket Sort.

◆ Sorting Terminologies

1. **In-place Sorting**
    - Sorting that uses only a constant amount of extra memory.
    - Example: Bubble Sort, Quick Sort.
2. **Out-of-place Sorting**
    - Requires extra space for sorting.

- Example: Merge Sort.

3. **Stable Sorting**
    - If two equal elements appear in the same order before and after sorting.
    - Example: Merge Sort, Insertion Sort.
4. **Unstable Sorting**
    - Equal elements may not appear in the same order after sorting.
    - Example: Quick Sort, Heap Sort.
5. **Time Complexity**
    - Efficiency of sorting in terms of number of operations.
    - Best Case, Average Case, Worst Case analysis is important.
6. **Space Complexity**
    - Extra memory used apart from input data.
7. **Adaptive Sorting**
    - Sorting algorithm that becomes faster if the input is already partially sorted.
    - Example: Insertion Sort.