Quick Sort

Quick Sort is a **Divide and Conquer** algorithm that works by selecting a **pivot element** and partitioning the array around the pivot, such that:

- Elements **smaller than pivot** go to the left subarray.
- Elements **greater than pivot** go to the right subarray.
  Then, the same process is applied recursively to the left and right subarrays until the entire array is sorted.

🔹 Key Idea

1. **Choose a pivot** (can be first, last, middle, or random element).
2. **Partition the array** → rearrange elements so that all elements smaller than pivot come before it, and larger elements come after it.
3. **Recursively apply Quick Sort** to the left and right subarrays.

🔹 Example

Sort the array: `[10, 7, 8, 9, 1, 5]`

1. Choose pivot = `5`
   - Partition → `[1] [5] [10, 7, 8, 9]`
2. Recursively sort left part `[1]` (already sorted).
3. Sort right part `[10, 7, 8, 9]` with pivot `9`:
   - Partition → `[7, 8] [9] [10]`

4. Recursively sort `[7, 8]` with pivot `8`:
    - Partition → `[7] [8] []`

Final Sorted Array = `[1, 5, 7, 8, 9, 10]`

Quick Sort Algorithm (Stepwise)

1. If the array has 1 or 0 elements → already sorted.
2. Select a pivot element.
3. Partition the array around pivot.
4. Recursively apply Quick Sort to left and right subarrays.

Properties

- **Unstable**: Does not preserve the relative order of equal elements (unless carefully implemented).
- **Recursive algorithm.**
- Can be made **in-place** (uses only O(log n) extra space for recursion).

Time Complexity

- **Best Case**: O(n log n) → when pivot divides array into two equal halves.
- **Average Case**: O(n log n).
- **Worst Case**: O(n²) → when pivot is always smallest or largest element (unbalanced partition).

Worst case can be avoided by choosing **random pivots** or **median-of-three** method.

Space Complexity

- O(log n) (for recursion stack).

Advantages

Faster in practice compared to Merge Sort and Heap Sort.
Works in-place → no extra array needed.
Average time complexity O(n log n).

Disadvantages

Worst-case $O(n^2)$ if pivot selection is poor.
Not stable by default.
Recursive, so stack overflow may occur for very large arrays without optimization.

Applications

- Used in **system libraries** (like C, C++, Java sort functions often use a variation of Quick Sort).
- Suitable for sorting arrays stored in **main memory**.
- Efficient for **large datasets** where in-place sorting is required.