**Bubble Sort**:

* Simple
* Compares each pair of adjacent elements and swaps them if they are in the wrong order.
* Time Complexity: O(n²).

**Insertion Sort**:

* Builds the sorted array one item at a time.
* Efficient for small or partially sorted arrays.
* Time Complexity: O(n²).

**Quick Sort**:

* Efficient, widely used.
* Divides the array into partitions and sorts each partition.
* Time Complexity: O(n log n) on average, O(n²) worst-case.

**Merge Sort**:

* Divides the array into halves, sorts, and then merges.
* Stable and efficient for large data sets.
* Time Complexity: O(n log n).

**Time Complexity**:

* **Bubble Sort**: O(n²)
* **Quick Sort**: Average case O(n log n), worst case O(n²)
* Quick Sort is faster in practice for large datasets because of its average time complexity of O(n log n). It uses divide-and-conquer, making it more efficient in terms of time compared to Bubble Sort