**Exercise 3 – Sorting Algorithms**

**Different Sorting Algorithms**

#### **Bubble Sort:**

#### Repeatedly compare adjacent elements and swap them if they are in the wrong order.

#### **Insertion Sort:**

#### Builds the sorted array one element at a time by inserting elements into their correct position.

#### **Quick Sort:**

#### Divide-and-conquer. Select a pivot, partition the array into elements less than and greater than the pivot, and recursively sort the partitions.

#### **Merge Sort:**

#### Divide the array in half, recursively sort each half, then merge them together in sorted order.

**Comparing Quick Sort VS Bubble Sort**

| Feature | Bubble Sort | Quick Sort |
| --- | --- | --- |
| Best Case | O(n) | O(n log n) |
| Average Case | O(n²) | O(n log n) |
| Worst Case | O(n²) | O(n²) |
| Space Complexity | O(1) | O(log n) |
| Stable | Yes | No |
| Usage | Rare | Widely used |

**Why Quick Sort Is Preferred Over Bubble Sort?**

**Faster average performance** on large datasets.

**Efficient use of memory** with only O(log n) additional space (due to recursion).

**Scalable** for real-world use cases, unlike Bubble Sort which becomes impractical for more than a few dozen elements.

**Optimized versions** of Quick Sort are used in many standard libraries (e.g., Java, C++ STL).