**EX 3**

**Understanding Sorting Algorithms**

**1. Bubble Sort**

* **Description**: A simple comparison-based algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.
* **Time Complexity**:
  + Best-case: O(n) (when the list is already sorted)
  + Average-case: O(n^2)
  + Worst-case: O(n^2)
* **Space Complexity**: O(1) (in-place sorting)

**2. Insertion Sort**

* **Description**: Builds the sorted array one item at a time by repeatedly taking the next item and inserting it into the correct position among the already-sorted items.
* **Time Complexity**:
  + Best-case: O(n) (when the list is already sorted)
  + Average-case: O(n^2)
  + Worst-case: O(n^2)
* **Space Complexity**: O(1) (in-place sorting)

**3. Quick Sort**

* **Description**: A divide-and-conquer algorithm that picks a pivot element, partitions the array around the pivot (elements less than pivot to the left, greater to the right), and then recursively applies the same process to the subarrays.
* **Time Complexity**:
  + Best-case: O(nlogn)
  + Average-case: O(nlogn)
  + Worst-case: O(n^2) (when the pivot selection is poor)
* **Space Complexity**: O(log n) (due to recursion)

**4. Merge Sort**

* **Description**: Another divide-and-conquer algorithm that divides the list into halves, recursively sorts each half, and then merges the sorted halves to produce the sorted list.
* **Time Complexity**:
  + Best-case: O(nlogn)
  + Average-case: O(nlogn)
  + Worst-case: O(nlogn)
* **Space Complexity**: O(n) (requires additional space for merging)

**Analysis**

**Time Complexity Comparison**

* **Bubble Sort**:
  + Best-case: O(n)
  + Average-case: O(n^2)
  + Worst-case: O(n^2)
* **Quick Sort**:
  + Best-case: O(nlogn)
  + Average-case: O(nlogn)
  + Worst-case: O(n^2)

**Why Quick Sort is Generally Preferred Over Bubble Sort**

1. **Efficiency**: Quick Sort has an average time complexity of O(nlogn), making it much faster than Bubble Sort, especially for large datasets where Bubble Sort’s O(n^2) complexity becomes impractical.
2. **Divide and Conquer**: Quick Sort uses the divide-and-conquer strategy, which efficiently narrows down the sorting problem into smaller subproblems, leading to faster overall sorting.
3. **In-Place Sorting**: Like Bubble Sort, Quick Sort is an in-place sorting algorithm that leverages pivot-based partitioning to achieve better performance.
4. **Real-world Performance**: Despite its worst-case complexity being O(n^2), in practice, Quick Sort’s performance is significantly better due to good pivot selection strategies (like median-of-three) that help avoid the worst-case scenario.