**Exercise 3: Sorting Customer Orders**

1. **Understand Sorting Algorithms**

* **Bubble Sort**: It is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. It is not suitable for large data sets as it is average and worst-case time complexity are quite high.  
  Best Time Complexity: O(n)  
  Average/ Worst Time Complexity: O(n2)  
  **Insertion Sort**: It is another simple sorting algorithm that works by iteratively inserting each element of an unsorted list into its correct position in a sorted position of the list. It builds the sorted array one element at a time and for each element, it places it in its correct position relative to the sorted portion of the array.  
  Best Time Complexity: O(n)  
  Average/ Worst Time Complexity: O(n2)  
  **Quick Sort**: It is a sorting algorithm based on the Divide and Conquer that picks an element as pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.  
  Best/Average Time Complexity: O(n log n)  
  Worst Time Complexity: O(n2)  
  **Merge Sort**: It is a popular sorting algorithm which also follows Divide and Conquer. It works by recursively dividing the input array into two halves, recursively sorting the two halves and finally merging them back together to obtain the sorted array.  
  Best/Average/Worst Time Complexity: O(n log n)

2&3**. Setup and Implementation:** Shown in code.

4. **Analysis**:

* Comparison between Bubble Sort and Quick Sort.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Bubble** **Sort** | **Quick Sort** |
| Best Case | O(n) [already sorted] | O(n log n) |
| Average Case | O(n2) | O(n log n) |
| Worst Case | O(n2) [reverse sorted] | O(n2) [poor pivot choice] |
| Space | O(1) | O(log n) [due to recursion] |
| Stability | Yes | Not stable but can be made stable. |
| Speed | Slow | Very Fast |
| Usability | On Small Data | Real-world and large data. |

* Reasons why Quick Sort is preferred over Bubble Sort:  
  1. Faster Average Performance: Unlike Bubble Sort’s average time complexity of O(n2), Quick Sort’s average case time complexity is O(n log n), which is faster.  
  2. Efficient Use of Memory: Quick Sort uses in-place partitioning which has low memory overhead, even for larger arrays.  
  3. More Scalable: Quick Sort is better suited for sorting thousands or millions of elements.  
  4. More Widely Used: Quick Sort is more widely used as compared to Bubble Sort.