**Exercise 3: Sorting Customer Orders**

**Understanding Sorting Algorithms**

Sorting algorithms help you arrange items in a specific order, like putting books on a shelf or organizing a list of numbers. Here’s a simple guide to some common sorting methods:

**Bubble Sort**

**Bubble Sort** is like repeatedly checking pairs of books on a shelf and swapping them if they’re in the wrong order. You keep doing this until no more swaps are needed.

* **Best Case**: O(n) (If the books are already in order, just one pass through the shelf is needed)
* **Average Case**: O(n^2) (Usually requires many passes and swaps)
* **Worst Case**: O(n^2) (If the books are in reverse order, it will take the longest)

**Example**: Imagine you’re sorting a line of students by height, swapping them whenever they’re in the wrong order, until everyone is in the right place.

**Insertion Sort**

**Insertion Sort** is like sorting a hand of playing cards. You take each card and place it into its correct position in your hand, one card at a time.

* **Best Case**: O(n) (If the cards are already sorted, only one pass is needed)
* **Average Case**: O(n^2) (Usually takes more comparisons and shifts)
* **Worst Case**: O(n^2) (If the cards are in reverse order, it takes the most work)

**Example**: Imagine you’re organizing your cards by value. You pick each card and insert it into its correct spot in the sorted hand.

**Quick Sort**

**Quick Sort** picks a “pivot” card and then separates the rest into two groups: those smaller and those larger than the pivot. It does this repeatedly for each group.

* **Best Case**: O(n log n) (When the pivot splits the list into two equal parts)
* **Average Case**: O(n log n) (Generally fast)
* **Worst Case**: O(n^2) (If the pivot is the smallest or largest card each time)

**Example**: Picture sorting a list of friends by height. You pick one friend as a reference (pivot), then split everyone into shorter and taller friends, and repeat until sorted.

**Merge Sort**

**Merge Sort** divides the list into smaller and smaller parts until each part has one item, then merges them back together in order.

* **Best Case**: O(n log n) (Even if the list is already sorted, merging is needed)
* **Average Case**: O(n log n) (Consistently efficient)
* **Worst Case**: O(n log n) (Time remains the same)

**Example**: Imagine you’re sorting a stack of papers. You divide the stack into smaller piles, sort each pile, and then merge them back together in order.

In summary, each sorting method works differently and has its strengths and weaknesses. Bubble Sort and Insertion Sort are simpler but less efficient for large lists. Quick Sort and Merge Sort are more complex but handle bigger lists more efficiently.