**Understanding Sorting Algorithms**

**Bubble Sort**:

* **Description**: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
* **Time Complexity**:
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)

**Insertion Sort**:

* **Description**: Builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort.
* **Time Complexity**:
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)

**Quick Sort**:

* **Description**: An efficient sorting algorithm that uses a divide-and-conquer approach. It picks an element as a pivot and partitions the array around the pivot.
* **Time Complexity**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n^2) (but this is rare with good pivot selection)

**Merge Sort**:

* **Description**: A divide-and-conquer algorithm that divides the array into halves, sorts each half, and then merges the sorted halves.
* **Time Complexity**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n log n)

**Analysis**

**Performance Comparison**

* **Bubble Sort**: O(n^2), where n is the number of orders. Bubble sort has a high time complexity, making it inefficient for large datasets.
* **Quick Sort**: O(n log n) on average, where n is the number of orders. Quick sort has a much faster time complexity than bubble sort, making it suitable for large datasets.

**Why Quick Sort is Preferred**

Quick sort is generally preferred over bubble sort for several reasons:

* **Faster Time Complexity**: Quick sort has a much faster time complexity than bubble sort, making it more efficient for large datasets.
* **Scalability**: Quick sort can handle large datasets with ease, while bubble sort becomes impractically slow for large datasets.