**Algorithm Name:** Merge Sort

**Purpose:** Merge Sort is a divide-and-conquer algorithm used for sorting an array or list of elements. It divides the array into subarrays, sorts them, and then merges them back together.

**Overview:** Merge Sort works by recursively splitting the array into two halves until the base case is reached (a single element). It then merges the halves back together in sorted order.

**Pseudocode:**

MergeSort(array):

if length(array) > 1:

middle = length(array) / 2

left\_half = array[0:middle]

right\_half = array[middle:end]

MergeSort(left\_half)

MergeSort(right\_half)

merge(left\_half, right\_half, array)

**Time Complexity:**

* **Best Case:** O(n log n)
* **Worst Case:** O(n log n)
* **Average Case:** O(n log n)

**Explanation:** The time complexity is O(n log n) because the algorithm divides the array into halves (log n levels) and merges the sorted halves in linear time (n).

**Space Complexity:**

* **Overall:** O(n)

**Explanation:** Merge Sort requires additional space proportional to the size of the array for the temporary arrays used in the merging process.

**Comparison with Quick Sort:**

* **Strengths:** Merge Sort has a guaranteed time complexity of O(n log n), making it more reliable for large datasets.
* **Weaknesses:** Merge Sort requires additional space (O(n)), whereas Quick Sort is in-place with O(log n) space complexity.

**Applications:** Merge Sort is used in various applications like sorting data in databases, external sorting algorithms for large datasets, and in systems where stable sorting is required.

**Edge Cases:** Merge Sort handles edge cases such as empty arrays or arrays with a single element efficiently.

**Limitations:** Merge Sort can be less efficient for small datasets compared to simpler algorithms like Insertion Sort.