Merge Sort

Introduction

Merge Sort is a popular sorting algorithm known for its efficiency and stability. It follows the divide-and-conquer approach to sort a list by dividing it into smaller sublists, sorting those sublists, and then merging them back into a single sorted list.

Logic

1. Divide the unsorted list into two halves.
2. Recursively sort both halves.
3. Merge the sorted halves to create a single sorted list.

C++ implementation of Merge sort: Merge Sort.
Python implementation of Merge sort: Merge Sort.
Java implementation of Merge sort: Merge Sort.
Javascript implementation of Merge sort: Merge Sort.

Complexity

- **Time Complexity**:
  - Best Case: O(n log n)
  - Average Case: O(n log n)
  - Worst Case: O(n log n)
- **Space Complexity**:
  - O(n) (due to the space required for merging)

Advantages

- Efficient for larger datasets due to its consistent time complexity.
- Stable sorting algorithm, meaning the relative order of equal elements remains unchanged.
- Well-suited for sorting linked lists and external storage where random access is costly.

Considerations

- Requires additional memory space for merging the sublists.
- Slightly more complex to implement compared to simple algorithms like Bubble Sort.

Disadvantages/Limitations

- Can be less efficient than other sorting algorithms for very small datasets.

- Not an in-place sorting algorithm, as it requires additional memory for merging.

Edge Cases

- Merge Sort's performance remains relatively consistent regardless of the initial order of the list.
- The divide-and-conquer approach makes Merge Sort well-suited for parallelization.

External Links

For a deeper understanding of Merge Sort, you can explore these resources: - Merge Sort - Wikipedia - Sorting Algorithms: Merge Sort - GeeksforGeeks - Merge Sort Visualization - VisuAlgo