The recurrence relation of merge sort is:

T(n)={Θ(1)if n=12T(n2)+Θ(n)if n>1

$T(n)=\{$

$\Theta(1)$

$2T($

2

n

$)+\Theta(n)$

if $n=1$

if $n>1$

- T(n) Represents the total time time taken by the algorithm to sort an array of size n.
- 2T(n/2) represents time taken by the algorithm to recursively sort the two halves of the array. Since each half has n/2 elements, we have two recursive calls with input size as (n/2).
- O(n) represents the time taken to merge the two sorted halves

Complexity Analysis of Merge Sort

- **Time Complexity:**
  - **Best Case:** O(n log n), When the array is already sorted or nearly sorted.
  - **Average Case:** O(n log n), When the array is randomly ordered.

    - **Worst Case:** O(n log n), When the array is sorted in reverse order.
- **Auxiliary Space:** O(n), Additional space is required for the temporary array used during merging.

Applications of Merge Sort:

- Sorting large datasets
- External sorting (when the dataset is too large to fit in memory)
- Inversion counting
- Merge Sort and its variations are used in library methods of programming languages.
    - Its variation TimSort is used in Python, Java Android and Swift. The main reason why it is preferred to sort non-primitive types is stability which is not there in QuickSort.
    - Arrays.sort in Java uses QuickSort while Collections.sort uses MergeSort.
- It is a preferred algorithm for sorting Linked lists.
- It can be easily parallelized as we can independently sort subarrays and then merge.
- The merge function of merge sort to efficiently solve the problems like union and intersection of two sorted arrays.

Advantages and Disadvantages of Merge Sort

Advantages

- **Stability** : Merge sort is a stable sorting algorithm, which means it maintains the relative order of equal elements in the input array.
- **Guaranteed worst-case performance:** Merge sort has a worst-case time complexity of **O(N logN)** , which means it performs well even on large datasets.
- **Simple to implement:** The divide-and-conquer approach is straightforward.
- **Naturally Parallel** : We independently merge subarrays that makes it suitable for parallel processing.

Disadvantages

- **Space complexity:** Merge sort requires additional memory to store the merged sub-arrays during the sorting process.
- **Not in-place:** Merge sort is not an in-place sorting algorithm, which means it requires additional memory to store the sorted data. This can be a disadvantage in applications where memory usage is a concern.
- Merge Sort is **Slower than QuickSort in general as** QuickSort is more cache friendly because it works in-place.

Quick Links:

- Merge Sort Based Coding Questions
- Bottom up (or Iterative) Merge Sort
- Recent Articles on Merge Sort
- Top Sorting Interview Questions and Problems
- Practice problems on Sorting algorithm
- Quiz on Merge Sort

Merge Sort Introduction

Visit Course