

# Algorithms: Merge sort -Part-1

**Relevel**  
by Unacademy



# Introduction

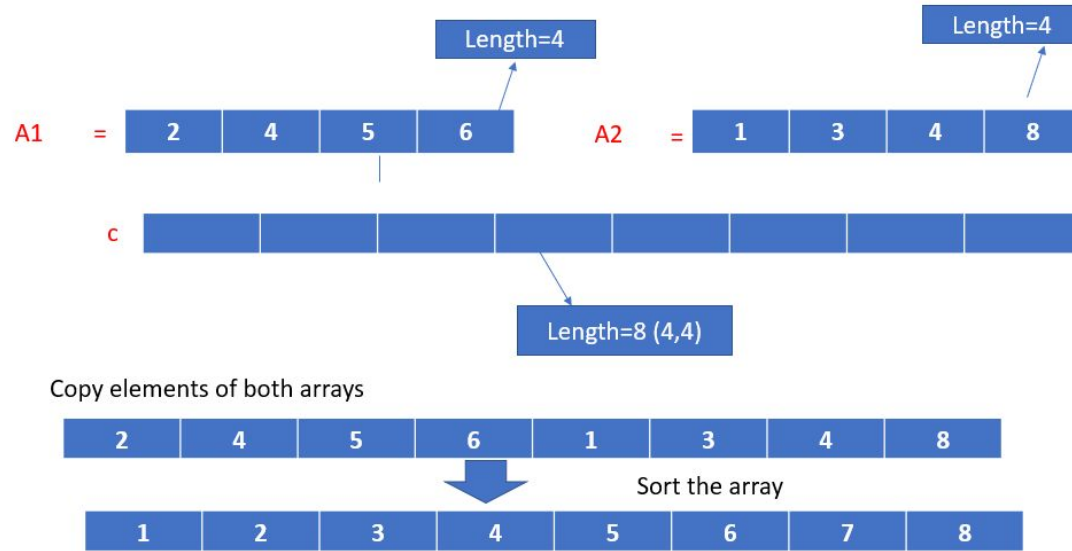
- This document contains a detailed course on Merge Sort and its application. After reading this document, candidates would be versed with the merge sort algorithm and how it can be used to solve various interview problems.
- Here in this course, we would be going through the following table of contents.  
Secondly, we would dive deep into the crux of the algorithm.

## Table of Contents

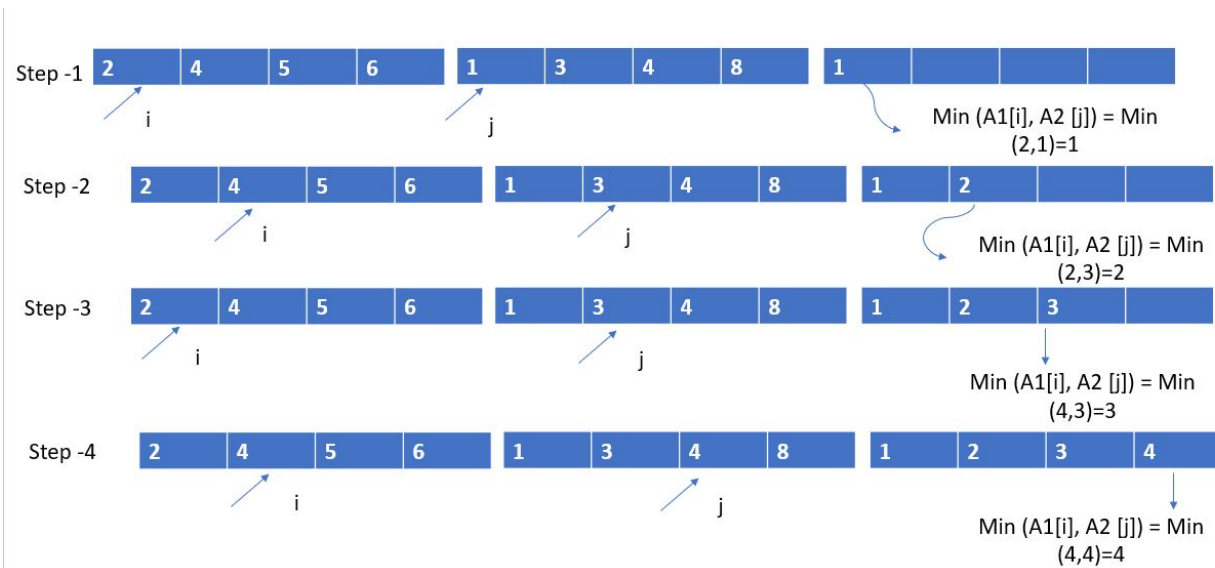
Why do we need another sorting algorithm if we already have bubble, selection and insertion sort ?

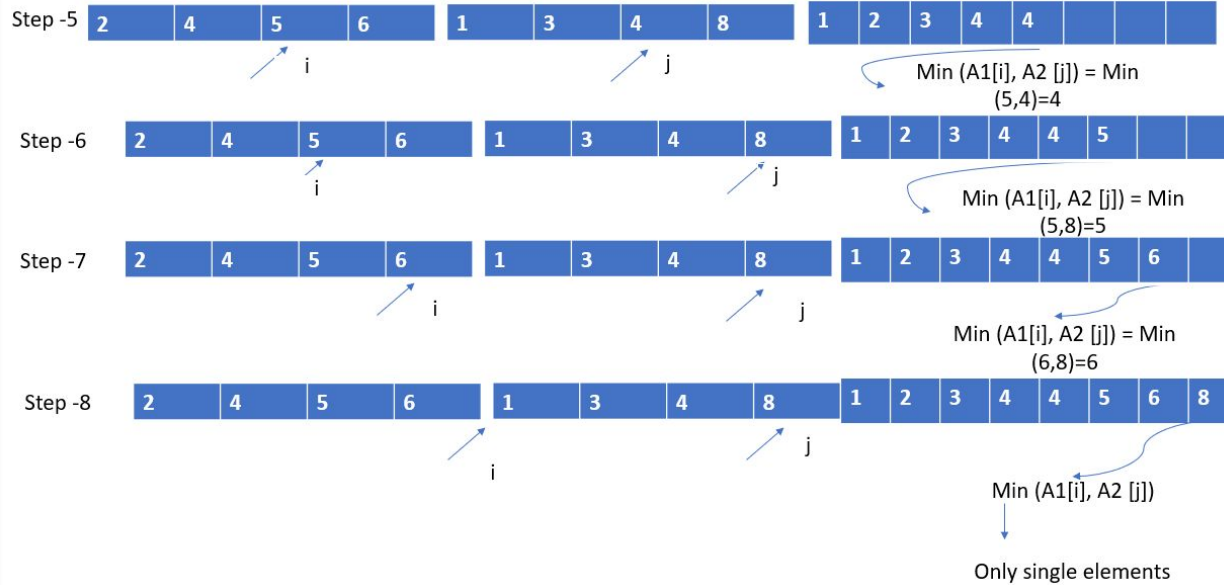
- Merge Algorithm
- Merge Sort Algorithm
- Stability and Inplace property of merge sort
- Time complexity analysis of Merge Sort
- Space Complexity analysis Merge Sort
- What is Divide And Conquer and how Merge Sort is following it ?
- Some important questions on the way merge sort works
- Applications Of Merge Sort

# Merge Algorithm



## Can we optimise the solution ?





## Coding Merge Algorithm

```
function mergeTwo(arr1, arr2) {  
  let merged = [];  
  let index1 = 0;  
  let index2 = 0;  
  let current = 0;  
  
  while (current < (arr1.length + arr2.length)) {  
  
    let isArr1Depleted = index1 >= arr1.length;  
    let isArr2Depleted = index2 >= arr2.length;  
  
    if (!isArr1Depleted && (isArr2Depleted || (arr1[index1] < arr2[index2]))) {  
      merged[current] = arr1[index1];  
      index1++;  
    } else {  
      merged[current] = arr2[index2];  
      index2++;  
    }  
  
    current++;  
  }  
  
  return merged;  
}  
  
mergeTwo(arr1, arr2);
```

What is Divide And Conquer and how Merge Sort is following it ?

Consider the following given array –



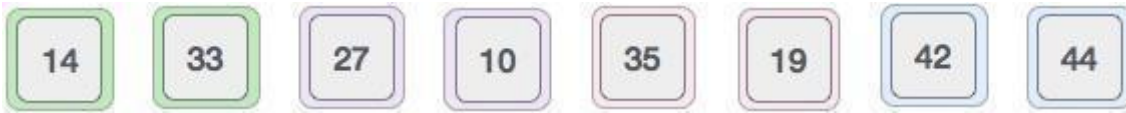
Merge sort first will divide the whole array into equal halves, We see here that an array of 8 values is divided into two arrays of size 4 values each



Now we divide these two arrays into halves.

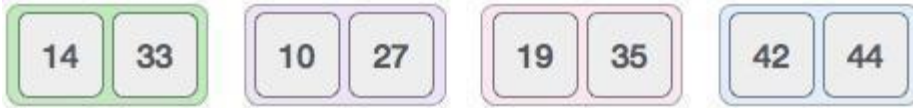


We further divide these arrays and we achieve single length array which can no longer be divided.



Now, we combine them in using merge algorithm.

We first compare the values from each array and then merge them into another array in a sorted fashion. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.



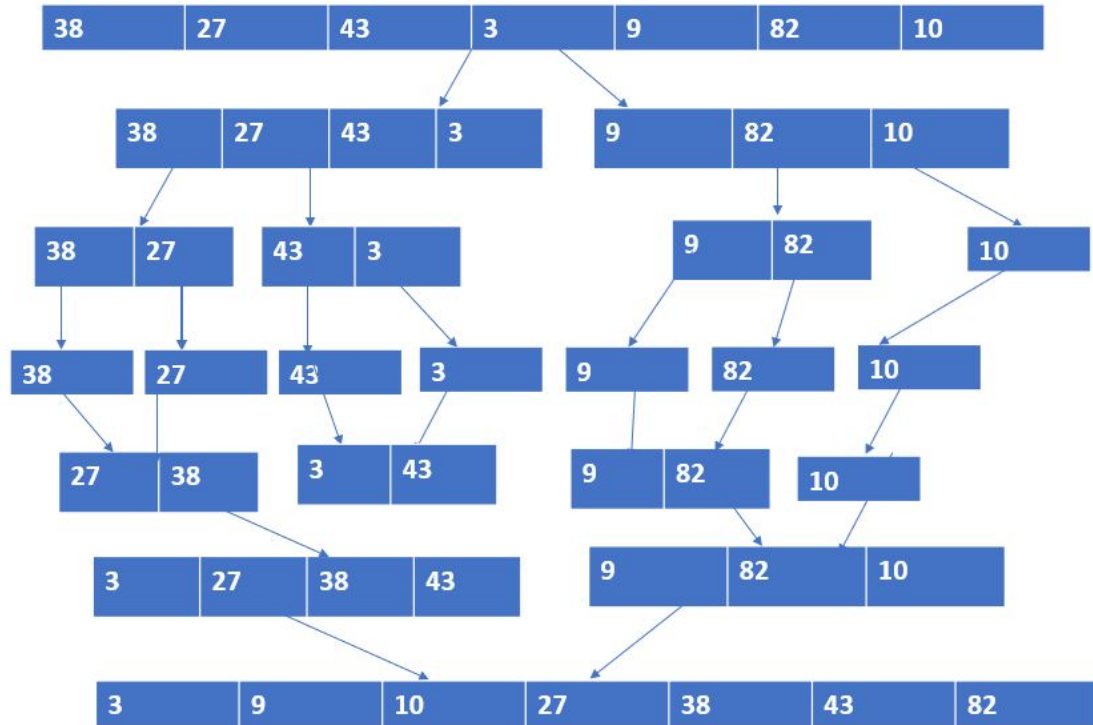
In the next iteration, we compare array values and merge them into the new array, in sorted order.



After the final merging, the final array should be-





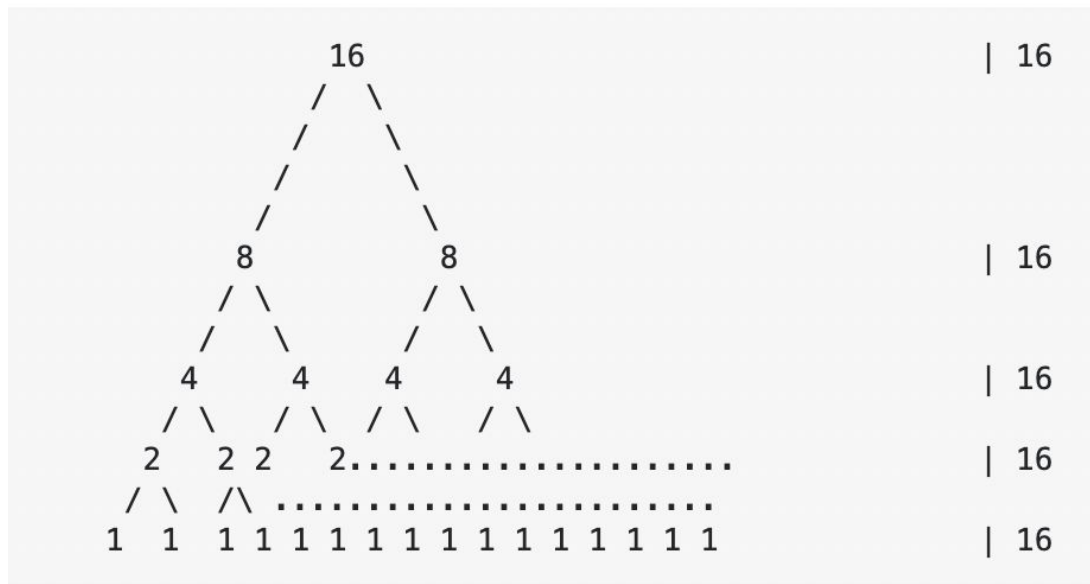


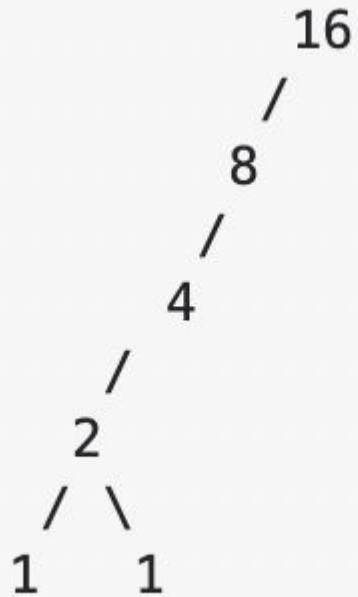
## Time Complexity

$$\begin{aligned} & \sum_{i=0}^{\log_2 n} 2^i O\left(\frac{n}{2^i}\right) \\ &= \sum_{i=0}^{\log_2 n} O(n) \\ &= (1 + \log_2 n) O(n) \\ &= O(n \log n) \end{aligned}$$

# Space Complexity

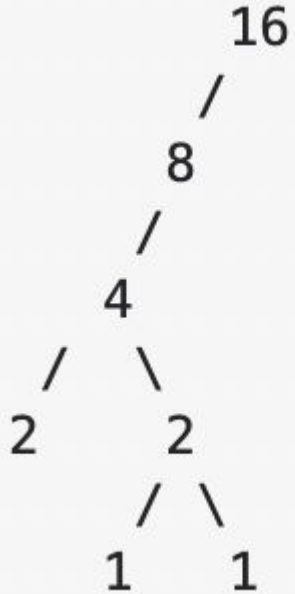
If you draw the merge sort space tree it gives a hint of  $O(n \log n)$





how number of space used is 32 ( $16 + 8 + 4 + 2 + 1 + 1$ ) =  $2n = 2 \cdot 16 < 3n$

Then it merge upwards



which is  $34 (16 + 8 + 4 + 2 + 2 + 1 + 1) < 3n$ .  
Then it merge upwards

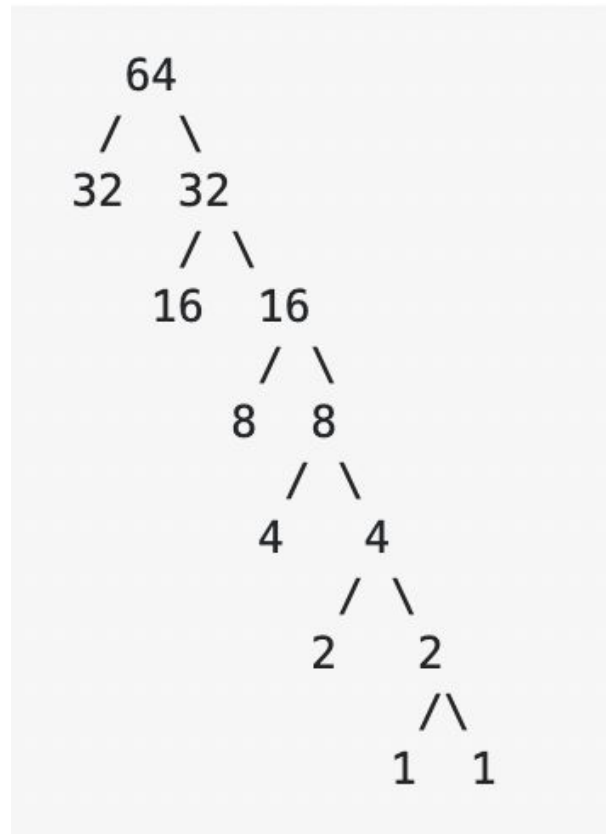


$36 (16 + 8 + 4 + 4 + 2 + 1 + 1) < 16 * 3 = 48$   
then it merge upwards



$$46 (16 + 8 + 8 + 4 + 4 + 2 + 2 + 1 + 1) < 3 \cdot n = 48$$

in a larger case,  $n = 64$





# Applications of Merge Sort

- Merge Sort is useful for sorting linked lists in  $O(n \log n)$  time
- Merge sort can be implemented without extra space for linked lists
- Merge sort is used for counting inversions in a list
- Library Sort functions of many languages use merge sort

## Practice Problem:

Q: Can we devise a modification to merge sort where we can make it inplace.

Q. Convert an array into another array where each element will be replaced with the number of elements that are smaller on the right side of the array to the element

## Upcoming Teaser

Quick Sort

Analysis

Applications

Quick select algorithm

Solving order statistics problem

median of medians vs quick select

Merge sort vs quick sort

**Thank you**