*Merge Sort*

Merge Sort is a *recursive* ***Divide and Conquer*** algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves and repeats this for every array.

1. Find the middle point to divide the array into two halves

*m = length / 2*

1. Call merge sort for first half

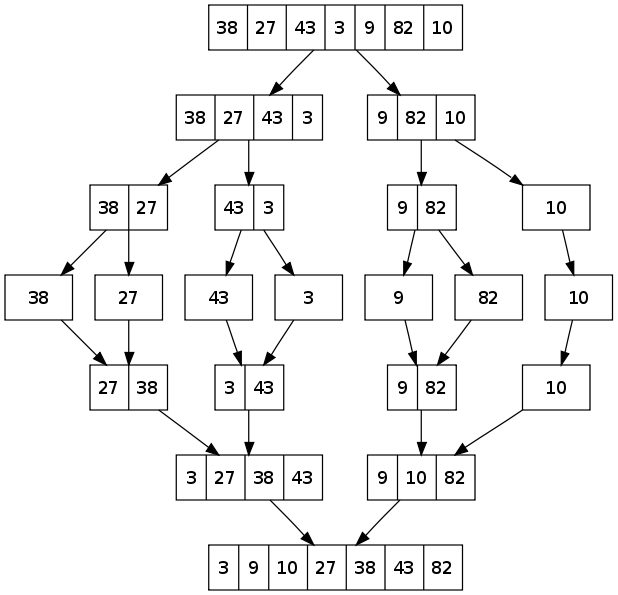
*mergeSort(arr, l, m)*

1. Call mergeSort for second half

*mergeSort(arr, m+1, r)*

1. Merge the two halves sorted

*merge(arr, l, m, r)*

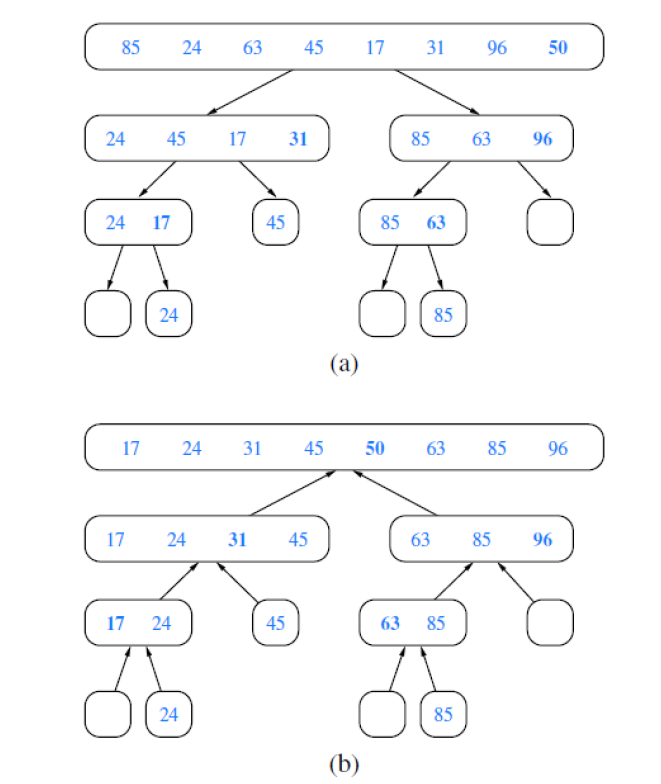


**Time Complexity***: T(n) = 2T(n/2) + O(n) = O(n logn)*. The recurrence can be solved either using the Recurrence Tree method or the Master method and it is equal to O(n logn) in all 3 cases (worst, average and best) as merge sort always divides the array into two halves and takes linear time to merge two halves.

*Quick Sort*

Quick Sort is a ***Divide and Conquer*** algorithm. It picks an element as *pivot* and *partitions the given array into two arrays*: one with elements smaller than the pivot and one with greater elements. The process repeats for all the subarray and then the result is combined into a sorted array. There are many different versions of quickSort that *pick pivot* in different ways.

1. Always pick *first* element as pivot.
2. Always pick *last* element as pivot.
3. Pick a *random* element as pivot.
4. Pick *median* as pivot.



**Time complexity**: *T(n) = T(n/a) + T((a-1)n/a) + O(n) = O(n log n)*

where a = elements smaller than the pivot

and *worst case* time complexity: *O(n²)*