Insertion Sort Algorithm

To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Working of Insertion Sort algorithm

Consider an example: arr[]: {12, 11, 13, 5, 6}

12	11	13	5	6

First Pass:

- Initially, the first two elements of the array are compared in insertion sort.

12	11	13	5	6

- Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position. Thus, swap 11 and 12.
- So, for now 11 is stored in a sorted sub-array.

11	12	13	5	6

Second Pass:

- Now, move to the next two elements and compare them

11	12	13	5	6

- Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11

Third Pass:

- Now, two elements are present in the sorted sub-array which are 11 and 12
- Moving forward to the next two elements which are 13 and 5

11	12	13	5	6

- *Both 5 and 13 are not present at their correct place so swap them*

11	12	5	13	6

- *After swapping, elements 12 and 5 are not sorted, thus swap again*

11	5	12	13	6

- *Here, again 11 and 5 are not sorted, hence swap again*

5	11	12	13	6

- *Here, 5 is at its correct position*

Fourth Pass:

- *Now, the elements which are present in the sorted sub-array are 5, 11 and 12*
- *Moving to the next two elements 13 and 6*

5	11	12	13	6

- *Clearly, they are not sorted, thus perform swap between both*

5	11	12	6	13

- *Now, 6 is smaller than 12, hence, swap again*

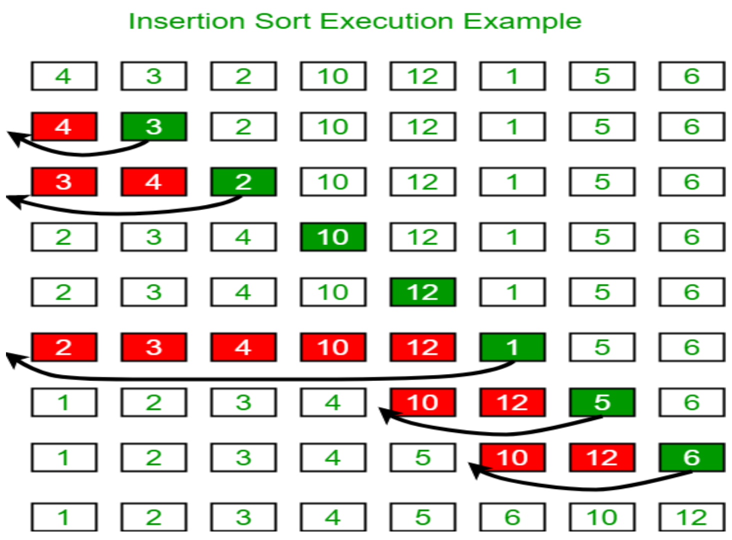
5	11	6	12	13

- *Here, also swapping makes 11 and 6 unsorted hence, swap again*

5	6	11	12	13

- *Finally, the array is completely sorted*

Illustrations:



Merge sort

Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array.

Merge sort is a recursive algorithm that continuously splits the array in half until it cannot be further divided i.e., the array has only one element left (an array with one element is always sorted). Then the sorted subarrays are merged into one sorted array.

Complexity Analysis of Merge Sort:

Time Complexity: O(N log(N)), Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation.

$$T(n) = 2T(n/2) + \theta(n)$$

The above recurrence can be solved either using the Recurrence Tree method or the Master method. It falls in case II of the Master Method and the solution of the recurrence is θ(Nlog(N)). The time complexity of Merge Sort isθ(Nlog(N)) 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.

Auxiliary Space: O(N), In merge sort all elements are copied into an auxiliary array. So N auxiliary space is required for merge sort.

The following figure illustrates the dividing (splitting) procedure.

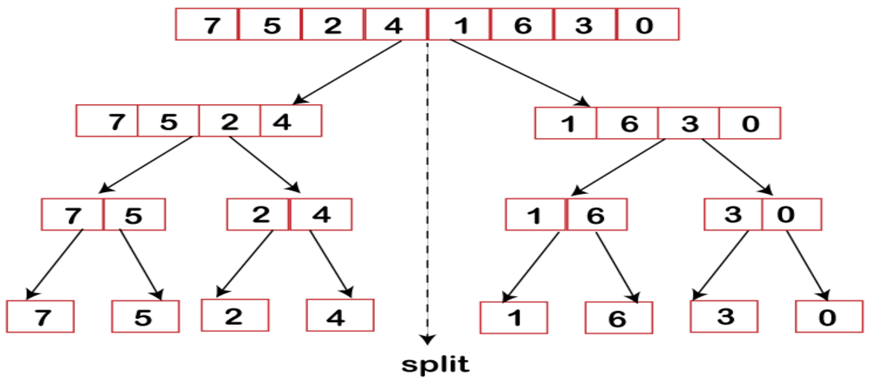


Figure 1: Merge Sort Divide Phase

FUNCTIONS: MERGE (A, p, q, r)

```
1. n 1 = q-p+1
2. n 2= r-q
3. create arrays [1.....n 1 + 1] and R [ 1.....n 2 +1 ]
4. for i ← 1 to n 1
5. do [i] ← A [ p+ i-1]
6. for j ← 1 to n2
7. do R[j] ← A[ q + j]
8. L [n 1+ 1] ← ∞
9. R[n 2+ 1] ← ∞
10. I ← 1
11. J ← 1
12. For k ← p to r
13. Do if L [i] ≤ R[j]
14. then A[k] ← L[ i]
15. i ← i +1
16. else A[k] ← R[j]
17. j ← j+1
```

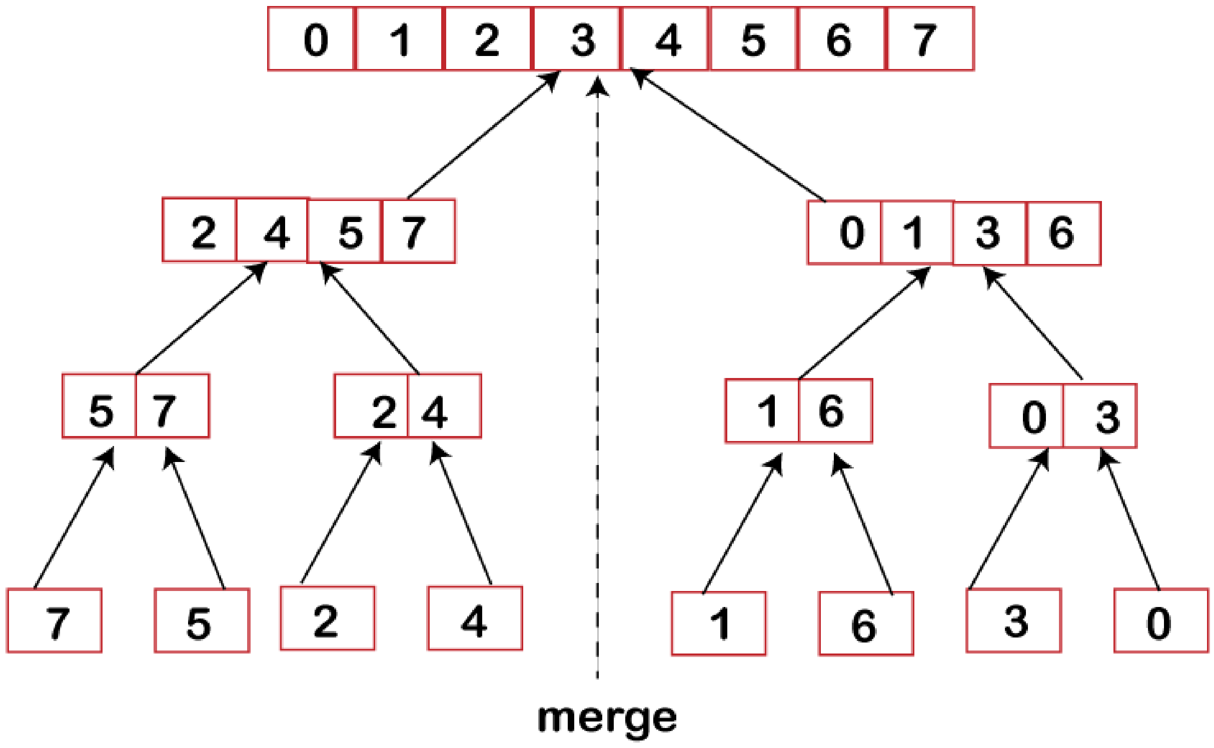


Figure 2: Merge Sort Combine Phase

The merge step of Merge Sort

Mainly the recursive algorithm depends on a base case as well as its ability to merge back the results derived from the base cases. Merge sort is no different algorithm, just the fact here the merge step possesses more importance.

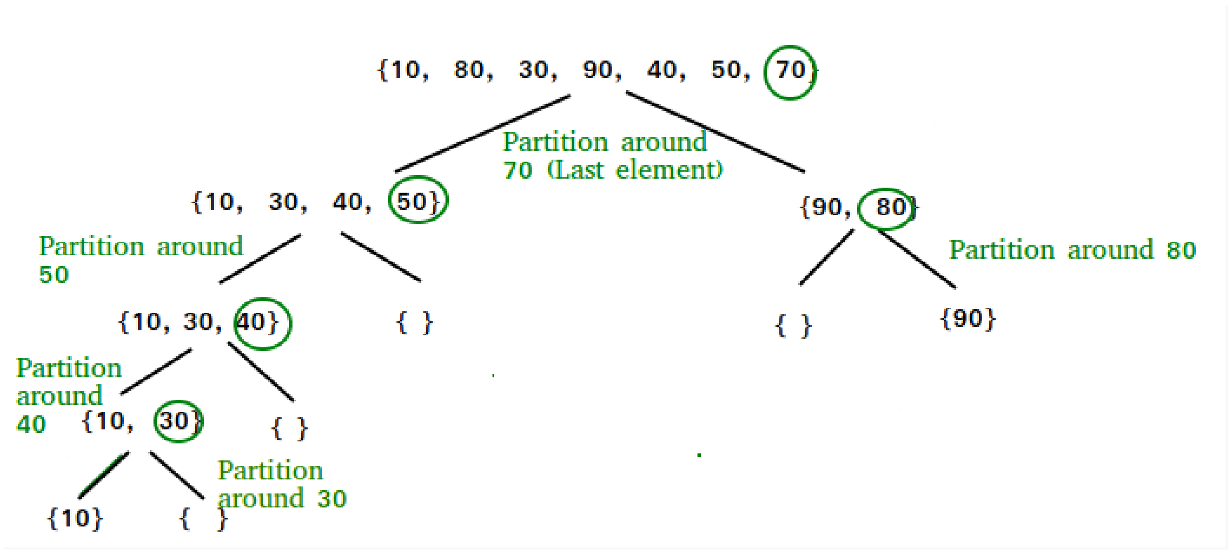
o any given problem, the merge step is one such solution that combines the two individually sorted lists(arrays) to build one large sorted list(array).

The merge sort algorithm upholds three pointers, i.e., one for both of the two arrays and the other one to preserve the final sorted array's current index.

Quick sort

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

QuickSort is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.



Choice of Pivot:

There are many different choices for picking pivots.

- Always pick the first element as a pivot.
- Always pick the last element as a pivot (implemented below)
- Pick a random element as a pivot.
- Pick the middle as the pivot.

Time and Space Complexity Analysis of Binary Search Algorithm