INSERTION SORT

- Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands.
- The array is virtually split into a sorted and an unsorted part.
- Values from the unsorted part are picked and placed at the correct position in the sorted part.

Characteristics of Insertion Sort:

- This algorithm is one of the simplest algorithm with simple implementation
- Basically, Insertion sort is efficient for small data values
- Insertion sort is adaptive in nature, i.e. it is appropriate for data sets which are already partially sorted.

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

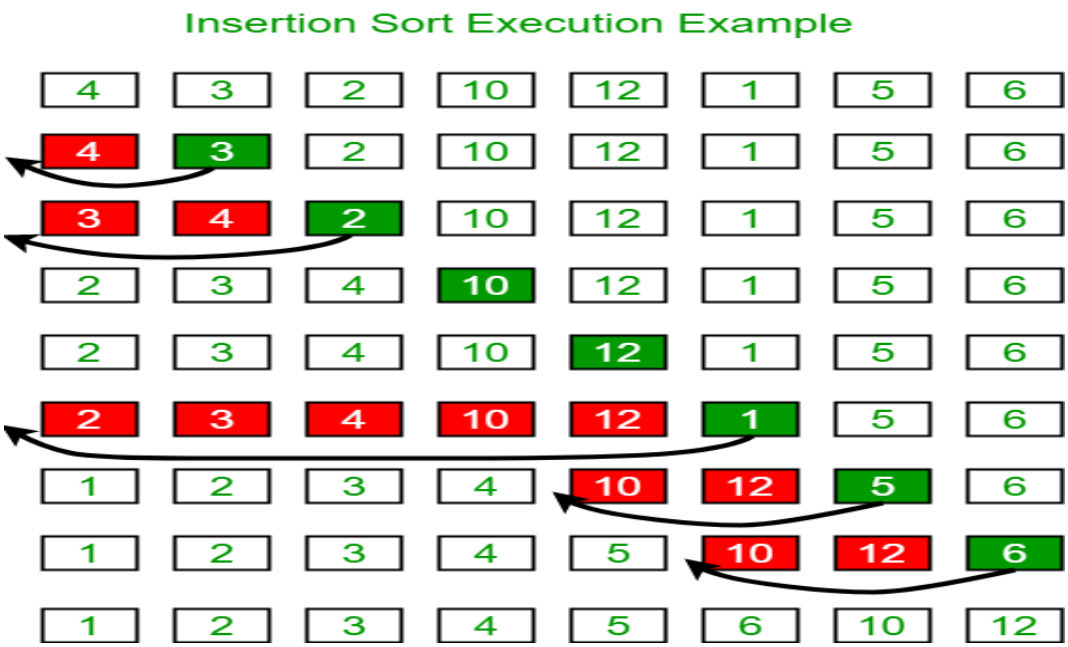
- Now, 6 is smaller than 12, hence, swap again

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 completeled.



Merge Sort Algorithm

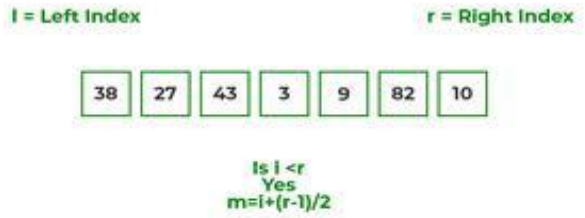
- Merge sort is defined as the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together.
- This process is repeated until the entire array is sorted.

Merge Sort Working Process:

- Think of it as a recursive algorithm continuously splits the array in half until it cannot be further divided.
- This means that if the array becomes empty or has only one element left, the dividing will stop, i.e. it is the base case to stop the recursion. If the array has multiple elements, split the array into halves and recursively invoke the merge sort on each of the halves.
- Finally, when both halves are sorted, the merge operation is applied. Merge operation is the process of taking two smaller sorted arrays and combining them to eventually make a larger one.

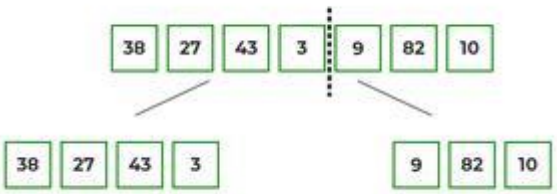
Lets consider an array arr[] = {38, 27, 43, 3, 9, 82, 10}

- At first, check if the left index of array is less than the right index, if yes then calculate its mid point

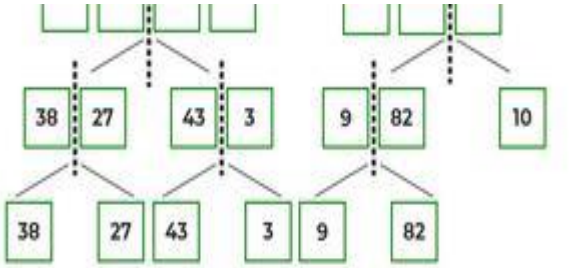


- **Now, as we already know that merge sort first divides the whole array iteratively into equal halves, unless the atomic values are achieved.**
- **Here, we see that an array of 7 items is divided into two arrays of size 4 and 3 respectively.**

- **Now, again find that is left index is less than the right index for both arrays, if found yes, then again calculate mid points for both the arrays.**

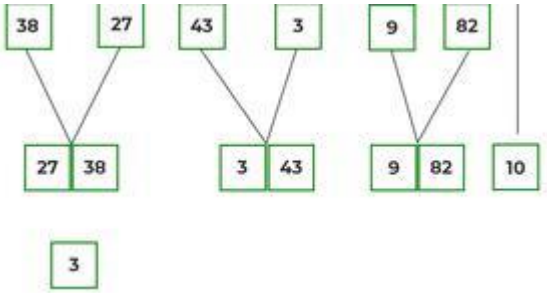


- **Now, further divide these two arrays into further halves, until the atomic units of the array is reached and further division is not possible.**

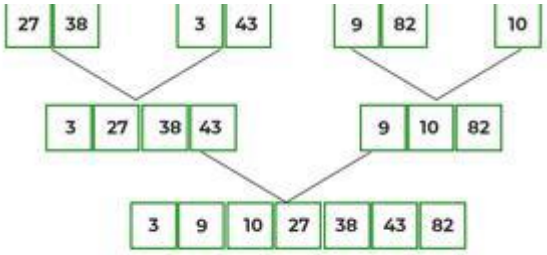


After dividing the array into smallest units merging starts, based on comparison of elements.

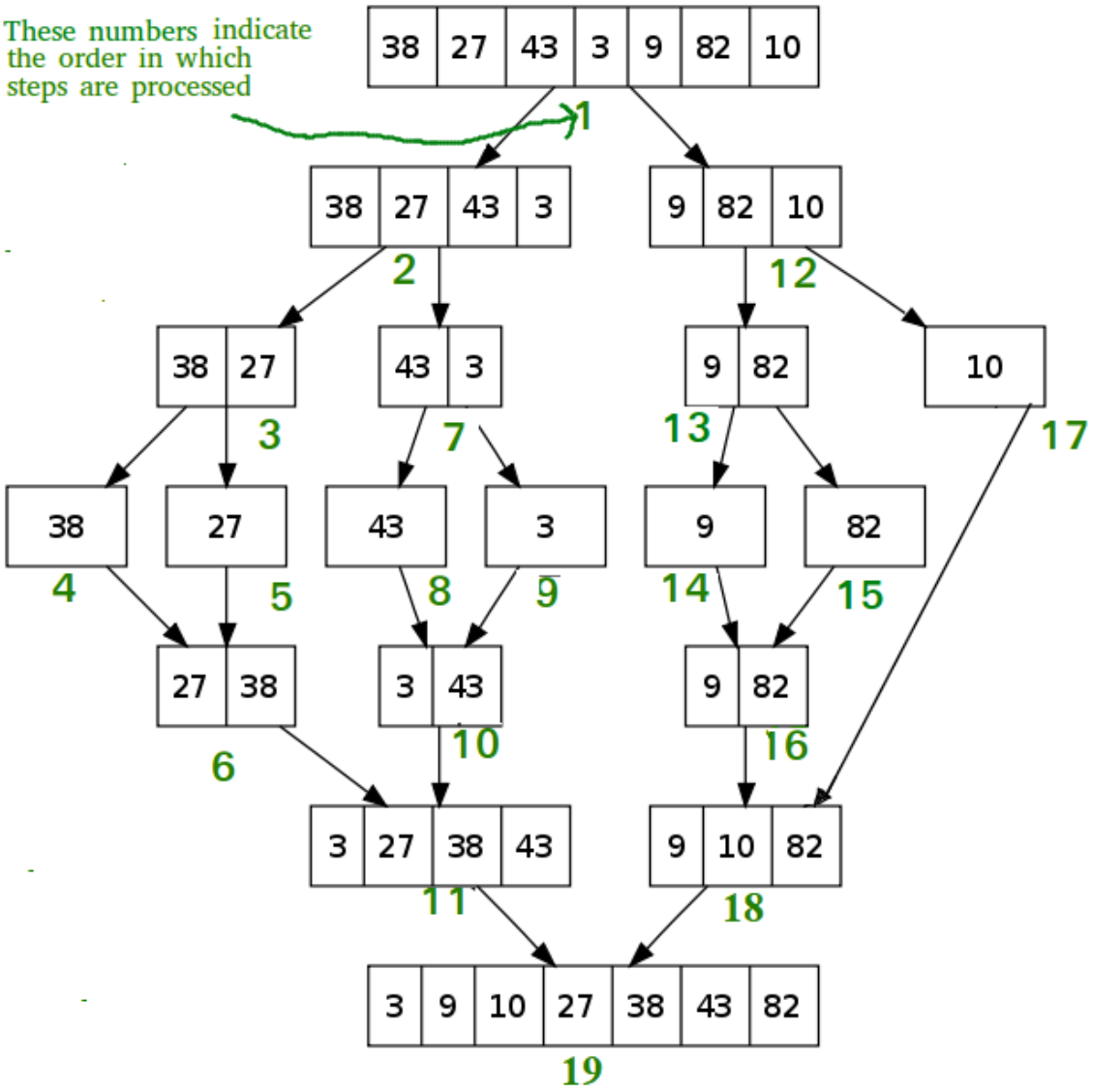
- **After dividing the array into smallest units, start merging the elements again based on comparison of size of elements**
- **Firstly, compare the element for each list and then combine them into another list in a sorted manner.**



- **After the final merging, the list looks like this:**



The following diagram shows the complete merge sort process for an example array {38, 27, 43, 3, 9, 82, 10}.


These numbers indicate the order in which steps are processed
38 27 43 3 9 82 10
1
38 27 43 3
9 82 10
2
12
38 27
43 3
9 82
10
3
7
13
17
38
27
43
3
9
82
4
5
8
9
14
15
27 38
3 43
9 82
10
16
6
3 27 38 43
9 10 82
11
18
3 9 10 27 38 43 82
19