**Merge Sort**

The **Merge Sort** algorithm is a sorting algorithm that is based on the **Divide and Conquer** paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.

**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.

* *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, 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.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 63 | 15 | 6 | 8 | 10 | 91 | 38 |

**Quick Sort**is a[Divide and Conquer algorithm](https://www.geeksforgeeks.org/divide-and-conquer-algorithm-introduction/). It picks an element as a pivot and partitions the given array around the picked pivot. There are many different versions of quick Sort that pick pivot in different ways.

* 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 median as the pivot.

The key process in **quick Sort**is a partition (). The target of partitions is, given an array and an element x of an array as the pivot, put x at its correct position in a sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

|  |  |
| --- | --- |
| 6 | 8 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 8 | 10 | 15 | 36 | 38 | 91 |

|  |  |  |
| --- | --- | --- |
| 10 | 38 | 91 |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 8 | 15 | 36 |

38

|  |  |
| --- | --- |
| 10 | 91 |

|  |  |
| --- | --- |
| 15 | 63 |

|  |  |
| --- | --- |
| 63 | 15 |

10

15

91

6

8

63

|  |
| --- |
| 38 |

|  |  |
| --- | --- |
| 10 | 91 |

|  |  |
| --- | --- |
| 6 | 8 |

|  |  |  |
| --- | --- | --- |
| 10 | 91 | 38 |

|  |  |  |  |
| --- | --- | --- | --- |
| 63 | 15 | 6 | 8 |