**Bubble Sort**

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets

**Algorithm**

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

begin BubbleSort(list)

 for all elements of list

 if list[i] > list[i+1]

 swap(list[i], list[i+1])

 end if

 end for

 return list

end BubbleSort

**Insertion Sort**

An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, **insertion sort**.

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets

### Algorithm

Now we have a bigger picture of how this sorting technique works, so we can derive simple steps by which we can achieve insertion sort.

**Step 1** − If it is the first element, it is already sorted. return 1;

**Step 2** − Pick next element

**Step 3** − Compare with all elements in the sorted sub-list

**Step 4** − Shift all the elements in the sorted sub-list that is greater than the

 value to be sorted

**Step 5** − Insert the value

**Step 6** − Repeat until list is sorted

**Selection Sort**

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

### Algorithm

**Step 1** − Set MIN to location 0

**Step 2** − Search the minimum element in the list

**Step 3** − Swap with value at location MIN

**Step 4** − Increment MIN to point to next element

**Step 5** − Repeat until list is sorted

**Merge Sort**

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being Ο(n log n), it is one of the most respected algorithms.

Merge sort first divides the array into equal halves and then combines them in a sorted manner.

### Algorithm

Merge sort keeps on dividing the list into equal halves until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

**Step 1** − if it is only one element in the list it is already sorted, return.

**Step 2** − divide the list recursively into two halves until it can no more be divided.

**Step 3** − merge the smaller lists into new list in sorted order.