Definition

Sorting means arranging an array or list of data according to their values in some specified order. Order can be ascending or descending. Sorting can be done on any type of data such as integers, floats, characters, strings. Suppose, we have an array of 5 integers 4, 8,-5, 0, 48. After sorting, the array will be -5, 0, 4, 8, 48 (ascending order) or 48, 8, 4, 0, -5 (descending order). Characters are generally ordered according to their ASCII values. Strings or words are generally sorted according to lexicographical Order or dictionary Order. For example, the following list of words {dog, cat, apple, bat, ball, door} will be sorted as {apple, ball, bat, cat, dog, door}.

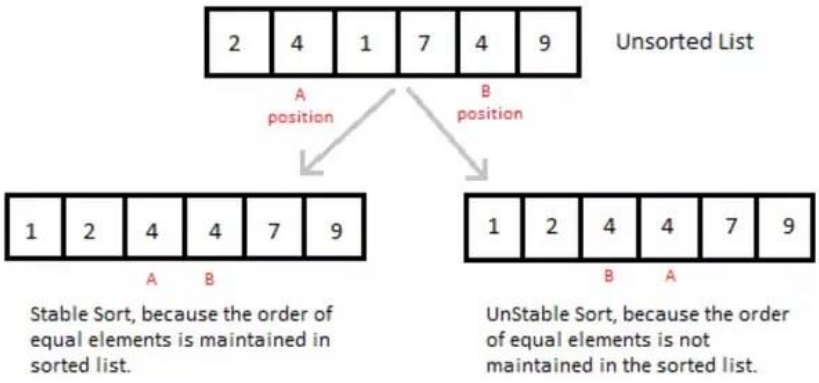
Types of sorting

There are two types of sorting – internal sorting and external sorting. An internal sort is any data sorting process that takes place entirely within the main memory of a computer. This is possible whenever the data to be sorted is small enough to all be held in the main memory. Some common internal sorting algorithms include: Bubble sort, Insertion sort, Quick sort, Heap sort, Radix sort and Selection sort.

External sorting is the type of sorting that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory, usually a hard disk drive. One example of external sorting is the external merge sort algorithm, which sorts chunks that each fit in RAM, then merges the sorted chunks together. In internal sorting, all the data is in main memory so access of data is easy but it is not so easy in external sorting.

Sort stability

A sorting algorithm is said to be stable if two objects with equal values (or keys) appear in the same order in the sorted output as they appear in the unsorted input. Whereas a sorting algorithm is said to be unstable if there are two or more objects with equal values which don't appear in same order before and after sorting.



Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

In the unsorted list, 4 appear twice at position 2 and 5. In stable sorting, their order is preserved in unsorted and sorted array (before and after sorting). In the case of unstable sort, this order of appearance before and after sorting is not necessarily preserved.

In place sort

An in-place sorting algorithm uses no extra storage space or very little amount of extra storage space (negligible) for producing the sorted output. It sorts the list only by modifying the order of the elements within the list. A sorting algorithm which is not in-place is sometimes called not-in-place or out-of-place.

Algorithm 1: Bubble Sort

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the array, compares adjacent elements and swaps them if they are in the wrong order. The pass through the array is repeated until the array is sorted. It is a comparison based sorting algorithm.

Pseudocode:

```
//A is an array of N values; Bubble sort is used to sort A in ascending order

For I = 0 to N - 2 // Outer loop

        For J = 0 to N - I - 2 // Inner loop

                If (A[J] > A[J + 1]) //Swap if required

                        Temp = A(J]

                        A[J] = A[J + 1]

                        A[J + 1] = Temp

                EndIf

        EndFor

    EndFor
```

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Example:

The original unsorted array is: 88 44 68 29 14

Outer loop 1(Pass 1)

Inner loop 1: 88 44 68 29 14

Inner loop 2: 44 88 68 29 14

Inner loop 3: 44 68 88 29 14

Inner loop 4: 44 68 29 88 14

The original array after 1st outer loop: 44 68 29 14 88

Outer loop 2(Pass 2)

Inner loop 1: 44 68 29 14 88

Inner loop 2: 44 68 29 14 88

Inner loop 3: 44 29 68 14 88

The original array after 2nd outer loop: 44 29 14 68 88

Outer loop 3(Pass 3)

Inner loop 1: 44 29 14 68 88

Inner loop 2: 29 44 14 68 88

The original array after 3rd outer loop: 29 14 44 68 88

Outer loop 4(Pass 4)

Inner loop 1: 29 14 44 68 88

The original array after 4th outer loop: 14 29 44 68 88

This is the final sorted array.

Green color denotes sorted part of the array and yellow color denotes which elements are being compared.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Time complexity of original classical Bubble sort is $O(n^2)$ for best (elements are in sorted order), worst (elements are in reverse sorted order) and average cases (elements are in any random order). But we can modify this algorithm slightly so that best case can be reduced to O(n) [for n elements in the array].

```
Flag = False //It is boolean type variable

For I = 0 to N - 2 // Outer loop

    For J = 0 to N - I - 2 // Inner loop

        If (A[J] > A[J + 1]) //Swap if required

            Temp = A[J]

            A[J] = A[J + 1]

            A[J + 1] = Temp

            Flag = True

        EndIf

    EndFor

    If (Flag = False) // No swap done at 1st outer loop i.e. array is already sorted

        Exit from algorithm

    EndIf

EndFor
```

Hence for modified Bubble sort best case - O(n) , worst case - $O(n^2)$, average case - $O(n^2)$. Bubble sort is stable. It is also an in place sort, space complexity is O(1).

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Algorithm 2: Selection Sort

The selection sort is a comparison based sorting algorithm that sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two sub-arrays in a given array.

1) The sub-array which is already sorted.

2) Remaining sub-array which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub-array is picked and moved to the sorted sub-array.

Pseudocode:

```
//A[] is an array of N values; Selection sort is used to sort A[] in ascending order

For I = 0 to N-1

        Min_Index = I

        For J = I + 1 to N-1

                If (A[J] < A[Min_Index])

                        Min_Index = J

                End-If

        EndFor

        If (I ≠ Min_Index) // Avoid swapping an element with itself

                Temp = A[I]

                A[I] = A[Min_Index]

                A[Min_Index] = Temp

        EndIf

    EndFor
```

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Example:

The original unsorted array is: 44 -6 98 2 -8

The original array after 1st outer loop: -8 -6 98 2 44

The original array after 2nd outer loop: -8 -6 98 2 44

The original array after 3rd outer loop: -8 -6 2 98 44

The original array after 4th outer loop: -8 -6 2 44 98

The original array after 5th outer loop: 8 -6 2 44 98

This is the final sorted array.

Green color denotes sorted part of the array and yellow color denotes unsorted part of the array.

Time complexity of Selection sort is $O(n^2)$ for all the cases (when there are n elements in the array). Number of comparisons of selection sort does not depend on the arrangement or order of the data. Selection sort is not stable. It is an in place sort, space complexity is O(1).

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Algorithm 3: Insertion Sort

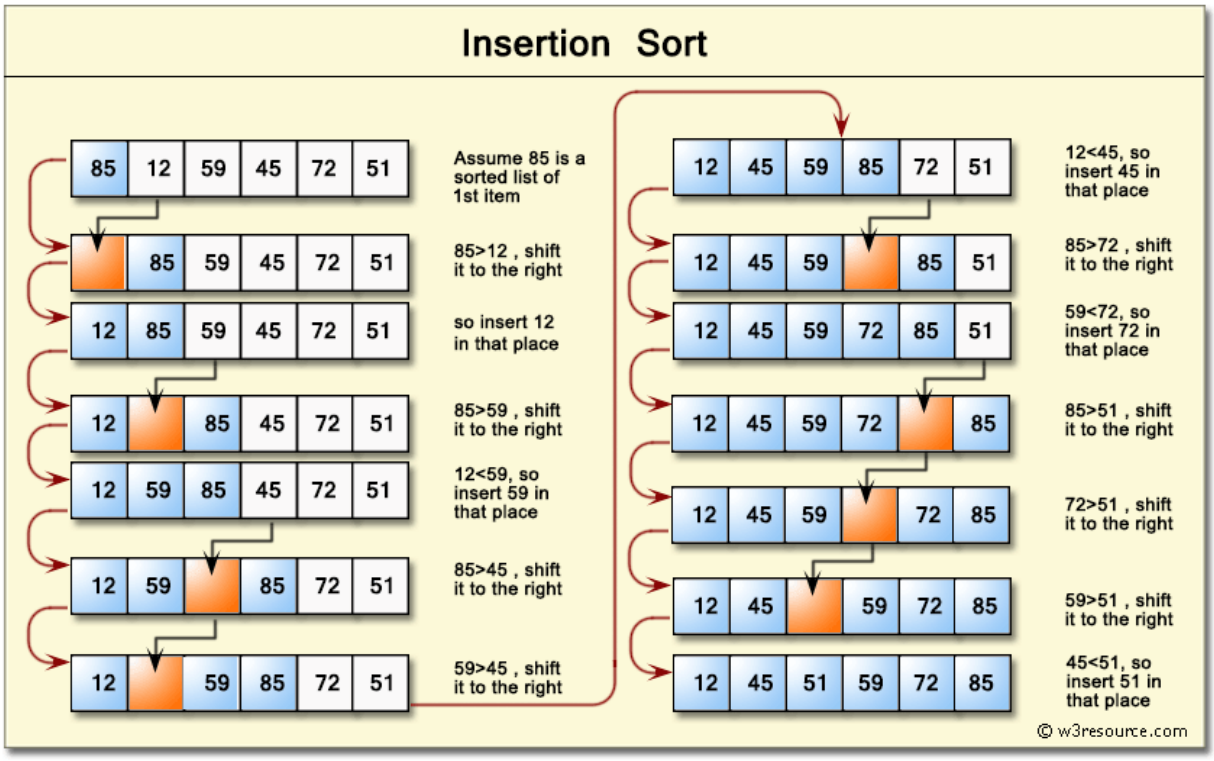
Insertion sort is a comparison based sorting algorithm which is based on the idea that one element from the input elements is consumed in each iteration to find its correct position i.e., the position to which it belongs in a sorted array. It iterates the input elements by growing the sorted array at each iteration. It compares the current element with the largest value in the sorted array. If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position. This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead.

Pseudocode:

```
//A[] is an array of N values; Selection sort is used to sort A[] in ascending order

For I = 1 to N-1

    Temp= A[I] // Current element to be inserted at its proper position in the sorted part

    J = I-1

    While ((J ≥0) and A[J] > Temp)

        A[J+1] = A[J ] // Shifting of elements larger than current element

        J = J - 1

    EndWhile

    A[J+1] = Temp // Insert current element at its proper position

EndFor
```

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Example:



Time complexity of Insertion sort is O(n) for best case (elements are in sorted order), $O(n^2)$ for both of the worst case (elements are in reverse sorted order) and average cases (elements are in any random order). Insertion sort is stable. It is also an in place sort, space complexity is O(1).

Reference books:

1. Data Structures Through C In Depth by S.K.Srivastava and Deepali Srivastava
2. Data Structures Using C by Reema Thareja

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata