Sorting is the act of arranging data in ascending or descending order based on linear relationships among data items.

It is possible to sort by name, number, and record. Sorting reduces the difficulty of finding information: for example, looking up a friend's phone number from a telephone dictionary is relatively straightforward because the names in the phone book are alphabetical.

What is Sorting?

Organizing data into a specific format is called sorting. Sorting algorithms specify the order in which data are arranged. Usually, data are ordered numerically or alphabetically.

In data structures, sorting refers to the arrangement of data in a preferred order. Sorting data allows a user to find information faster and easier. A dictionary is a simple example of sorting.

For instance, imagine we have a database of numbers in random orders. We want them to get it arranged in ascending order it is possible using sorting algorithm. You can have a look at the image below to understand it breifly.

What is Sorting

Why Sorting Is Important

Sorting is becoming more and more important as data search can be optimized to an extremely high degree if data is stored in a sorted manner. Data can also be arranged in a more readable manner when sorted.

Cheatsheet for Different Types of Sorting Techniques

Sorting techniques are determined by the situation. There are two factors to consider they are:

Program execution time, which is the time it takes to run the general program.

Program space, which is how much space the program occupies.

Each sorting technique differs by its efficiency and the amount of space it requires.

The time complexities of all the sorting algorithms are mentioned in the conclusion.

Sorting can be accomplished in several ways, as follows:

1 2 3 4 99 66 77 33 2 6 8 86 56

**1. Quick sort**

A quick sort algorithm uses a method that divides a large array of data into smaller arrays in order to sort it efficiently. An element is picked as a pivot and the given array is partitioned around it. An array of large values is divided into two arrays, one of which holds values smaller than the pivot value, the value on which the partition is made, and the other array holding values greater than the pivot value. Quicksort partitions around a pivot and then recursively sort the two subarrays based on the first partition.

1 2 3 4 99 66 77 33 2 6 8 86 56

1 2 3 4 66 99 77 33 2 6 8 86 56

1 2 3 4 66 77 99 33 2 6 8 86 56

1 2 3 4 66 77 33 99

**2. Bubble Sort**

This is the simplest and easiest sorting algorithm. In this method, two adjacent elements are compared and swapped until they are no longer in the intended order (ie) in ascending or descending order.

In other words, if a bubble sort is being performed to sort the input in ascending order, it will first compare the first two elements of the array. When the second element is smaller than the first, the algorithm will swap them, and then move on to the next.

Merge example : 1 2 3 4 99 66 77 33 2 6 8 86 56

1 2 3 4 99 66 77 33 2 6 8 86 56

1 2 3 4 99 66 77 33 2 6 8 86 56

1 2 3 4 66 77 99 2 33 6 8 56 86

1 2 3 4 66 77 99 2 6 33 56 86

1 2 2 3 4 6 33 56 66 77 86 99

**3. Merge sort**

Merge sort works similarly works based on the divide and conquers algorithm and it is similar to quick sort. The merge sort algorithm is very efficient and popular. In this function, the list is divided into two equal halves, a call to itself is made for each half, and the two sorted halves are merged. To merge the data, we must define the merge() function.

There are multiple sub-lists that are divided into halves until the sub-lists cannot be further divided. We combine the pair of lists of one element into two-element lists, sorting them along the way. In the same way, the sorted pair of two elements gets combined with the sorted list of four elements, and so on.

33 2 6

**4. Insertion Sort**

A simple sorting algorithm, the Insertion Sort works the same as when you are sorting cards in your hand. The first card in the card game is assumed to be sorted, and then we select an unsorted card. Unsorted cards that are greater than the first will be placed on the right; otherwise, they will be placed on the left. In the same way, all cards that are unsorted are taken out and placed exactly where they belong.

**5. Selection Sort**

The Selection Sort is an in-place comparison-based algorithm that divides the list into two portions: the sorted portion on the left and the unsorted portion on the right. The leftmost element of the unsorted array is swapped with the smallest element from the unsorted array, and the smallest element is part of the sorted array. The unsorted array boundary is moved one element to the right throughout this process.

6. Heap Sort

Heap Sort method uses the Binary Heap data structure for comparison-based sorting. In heap sort, each element in the list is eliminated from the heap part of the list, then it is inserted into the sorted part of the list. The remaining elements are sorted in the same manner.

7. Radix Sort

Radix Sort is an integer sorting method that sort data by significant position and value (place value) while using integer keys as keys. In a Radix sort, a subroutine called counting sort is used to sort an array of numbers. Radix sort isn't just for integers since integers can be used to represent strings (by hashing the strings into integers)

8. Bucket Sort

In Bucket Sort , elements are divided into multiple groups, also known as buckets. A bucket sort divides elements into uniform groups called buckets of a specific order of numbers. After each bucket is categorized, either by applying a recursive sorting algorithm or by using a different sorting algorithm, it is sorted individually. Once the elements have been sorted, they are gathered into groups.