# Counting Sort:

Counting Sort is a non-comparison-based sorting algorithm that works well when there is limited range of input values. It is particularly efficient when the range of input values is small compared to the number of elements to be sorted. The basic idea behind Counting Sort is to count the frequency of each distinct element in the input array and use that information to place the elements in their correct sorted positions.

Counting Sort Algorithm:

1. Declare an auxiliary array countArray[] of size max(inputArray[])+1 and initialize it with 0.
2. Traverse array inputArray[] and map each element of inputArray[] as an index of countArray[] array, i.e., execute countArray[inputArray[i]]++ for 0 <= i < N.
3. Calculate the prefix sum at every index of array inputArray[].
4. Create an array outputArray[] of size N.
5. Traverse array inputArray[] from end and update outputArray[ countArray[ inputArray[i] ] – 1] = inputArray[i]. Also, update countArray[ inputArray[i] ] = countArray[ inputArray[i] ]- – .

**Complexity Analysis of Counting Sort:**

- **Time Complexity**: O(N+M), where **N** and **M** are the size of **inputArray[]** and **countArray[]** respectively.
  - Worst-case: O(N+M).
  - Average-case: O(N+M).
  - Best-case: O(N+M).
- **Auxiliary Space:** O(N+M), where **N** and **M** are the space taken by **outputArray[]** and **countArray[]** respectively.

**Advantage of Counting Sort:**

- Counting sort generally performs faster than all comparison-based sorting algorithms, such as merge sort and quicksort, if the range of input is of the order of the number of input.
- Counting sort is easy to code
- Counting sort is a **stable algorithm**.

**Disadvantage of Counting Sort:**

- Counting sort doesn't work on decimal values.
- Counting sort is inefficient if the range of values to be sorted is very large.
- Counting sort is not an **In-place sorting** algorithm, It uses extra space for sorting the array elements.

```
def count_sort(input_array):

    M = max(input_array)

    count_array = [0] * (M + 1)


    for num in input_array:
        count_array[num] += 1


    for i in range(1, M + 1):
        count_array[i] += count_array[i - 1]


    output_array = [0] * len(input_array)

    for i in range(len(input_array) - 1, -1, -1):
        output_array[count_array[input_array[i]] - 1] = input_array[i]
        count_array[input_array[i]] -= 1

    return output_array

if __name__ == "__main__":

    input_array = [4, 3, 12, 1, 5, 5, 3, 9]


    output_array = count_sort(input_array)

    for num in output_array:
        print(num, end=" ")
```