Counting Sort

Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then doing some arithmetic to calculate the position of each object in the output sequence.
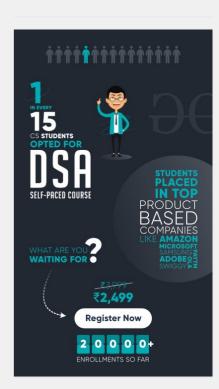
Let us understand it with the help of an example.

```
For simplicity, consider the data in the range 0 to 9.
Input data: 1, 4, 1, 2, 7, 5, 2
  1) Take a count array to store the count of each unique object.
  Index:     0  1  2  3  4  5  6  7  8  9
  Count:     0  2  2  0  1  1  0  1  0  0

  2) Modify the count array such that each element at each index
  stores the sum of previous counts.
  Index:     0  1  2  3  4  5  6  7  8  9
  Count:     0  2  4  4  5  6  6  7  7  7

The modified count array indicates the position of each object in
the output sequence.

  3) Output each object from the input sequence followed by
  decreasing its count by 1.
  Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 2.
  Put data 1 at index 2 in output. Decrease count by 1 to place
  next data 1 at an index 1 smaller than this index.
```

Recommended: Please solve it on "_PRACTICE_" first, before moving on to the solution.

Following is implementation of counting sort.

C++   C   Java   Python   C#   PHP

```java
// Java implementation of Counting Sort
class CountingSort
{
    void sort(char arr[])
    {
        int n = arr.length;

        // The output character array that will have sorted arr
        char output[] = new char[n];

        // Create a count array to store count of inidividul
        // characters and initialize count array as 0
        int count[] = new int[256];
        for (int i=0; i<256; ++i)
            count[i] = 0;

        // store count of each character
        for (int i=0; i<n; ++i)
            ++count[arr[i]];

        // Change count[i] so that count[i] now contains actual
        // position of this character in output array
        for (int i=1; i<=255; ++i)
            count[i] += count[i-1];

        // Build the output character array
        // To make it stable we are operating in reverse order.
        for (int i = n-1; i>=0; i--)
        {
            output[count[arr[i]]-1] = arr[i];
            --count[arr[i]];
        }
```

Most popular in Mathematical

```
        // Copy the output array to arr, so that arr now
        // contains sorted characters
        for (int i = 0; i<n; ++i)
            arr[i] = output[i];
    }

    // Driver method
    public static void main(String args[])
    {
        CountingSort ob = new CountingSort();
        char arr[] = {'g', 'e', 'e', 'k', 's', 'f', 'o',
                      'r', 'g', 'e', 'e', 'k', 's'
                     };

        ob.sort(arr);

        System.out.print("Sorted character array is ");
        for (int i=0; i<arr.length; ++i)
            System.out.print(arr[i]);
    }
}
/*This code is contributed by Rajat Mishra */
```

Output:

```
Sorted character array is eeeefggkkorss
```

Time Complexity: O(n+k) where n is the number of elements in input array and k is the range of input.

Auxiliary Space: O(n+k)

The problem with the previous counting sort was that we could not sort the elements if we have negative numbers in it. Because there are no negative array indices. So what we do is, we find the minimum element and we will store count of that minimum element at zero index.

C++   Java

```
// Counting sort which takes negative numbers as well
import java.util.*;

class GFG
{

    static void countSort(int[] arr)
    {
        int max = Arrays.stream(arr).max().getAsInt();
        int min = Arrays.stream(arr).min().getAsInt();
        int range = max - min + 1;
        int count[] = new int[range];
        int output[] = new int[arr.length];
        for (int i = 0; i < arr.length; i++)
        {
            count[arr[i] - min]++;
        }

        for (int i = 1; i < count.length; i++)
        {
            count[i] += count[i - 1];
        }

        for (int i = arr.length - 1; i >= 0; i--)
        {
            output[count[arr[i] - min] - 1] = arr[i];
            count[arr[i] - min]--;
        }

        for (int i = 0; i < arr.length; i++)
        {
            arr[i] = output[i];
        }
    }

    static void printArray(int[] arr)
    {
        for (int i = 0; i < arr.length; i++)
        {
            System.out.print(arr[i] + " ");
        }
        System.out.println("");
    }
```

```
        // Driver code
        public static void main(String[] args)
        {
            int[] arr = {-5, -10, 0, -3, 8, 5, -1, 10};
            countSort(arr);
            printArray(arr);
        }
}

// This code is contributed by princiRaj1992
```

Output:

```
-10 -5 -3 -1 0 5 8 10
```

Points to be noted:

1. Counting sort is efficient if the range of input data is not significantly greater than the number of objects to be sorted. Consider the situation where the input sequence is between range 1 to 10K and the data is 10, 5, 10K, 5K.

2. It is not a comparison based sorting. It running time complexity is O(n) with space proportional to the range of data.

3. It is often used as a sub-routine to another sorting algorithm like radix sort.

4. Counting sort uses a partial hashing to count the occurrence of the data object in O(1).

5. Counting sort can be extended to work for negative inputs also.

Exercise:

1. Modify above code to sort the input data in the range from M to N.

2. Is counting sort stable and online?

3. Thoughts on parallelizing the counting sort algorithm.

Snapshots:

	1	4	1	2	7	5	2

Index :	0	1	2	3	4	5	6	7	8	9
	0	2	2	0	1	1	0	1	0	0

Modify the count array by adding the previous counts.

	1	4	1	2	7	5	2

Index :	0	1	2	3	4	5	6	7	8	9
	0	2	4	0	1	1	0	1	0	0

2 + 2

	1	4	1	2	7	5	2

Index :	0	1	2	3	4	5	6	7	8	9
	0	2	4	4	1	1	0	1	0	0

4 + 0

	1	4	(1)	2	7	5	2

Index :	0	1	2	3	4	5	6	7	8	9
	0	0	4	4	4	6	6	7	7	7

Places :	1	2	3	4	5	6	7
	1	1			4		

	1	4	1	2	7	5	(2)

Index :	0	1	2	3	4	5	6	7	8	9
	0	0	2	4	4	5	6	6	7	7

Places :	1	2	3	4	5	6	7
	1	1	2	2	4	5	7

Quiz on Counting Sort

Coding Practice for Sorting

Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz

Selection Sort, Bubble Sort, Insertion Sort, Merge Sort, Heap Sort, QuickSort, Radix Sort, Counting Sort, Bucket Sort, ShellSort, Comb Sort, PegionHole Sorting
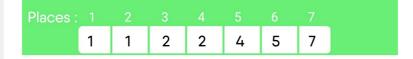
This article is compiled by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

C Program for Counting Sort

Java Program for Counting Sort

Median and Mode using Counting Sort

Sort an array of 0s, 1s and 2s (Simple Counting)

Comparison among Bubble Sort, Selection Sort and Insertion Sort

Why Quick Sort preferred for Arrays and Merge Sort for Linked Lists?

Counting k-mers via Suffix Array

Counting cross lines in an array

Counting numbers of n digits that are monotone

Bucket Sort To Sort an Array with Negative Numbers

Program to sort an array of strings using Selection Sort

Insertion sort to sort even and odd positioned elements in different orders

Rencontres Number (Counting partial derangements)

Counting numbers whose difference from reverse is a product of k

Counting even decimal value substrings in a binary string

Improved By : Mithun Kumar, spattk, krikti, sagarudasi2, princiraj1992, more

Article Tags : Mathematical   Sorting   Strings   counting-sort   Samsung

Practice Tags : Samsung   Strings   Mathematical   Sorting

👍

20

☐ To-do ☐ Done

2.3

Based on **152** vote(s)

Feedback/ Suggest Improvement   Add Notes   Improve Article

Previous
◄ HeapSort

Next
Comparator function of qsort() in C ►

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A computer science portal for geeks

COMPANY
About Us

LEARN
Algorithms

PRACTICE
Courses

CONTRIBUTE
Write an Article

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

Careers
Privacy Policy
Contact Us

Data Structures
Languages
CS Subjects
Video Tutorials

Company-wise
Topic-wise
How to begin?

Write Interview Experience
Internships
Videos