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 C implementation of counting sort.

C/C++

```
// C Program for counting sort
#include <stdio.h>
#include <string.h>
#define RANGE 255

// The main function that sort the given string arr[] in
// alphabatical order
void countSort(char arr[])
{
    // The output character array that will have sorted arr
    char output[strlen(arr)];

    // Create a count array to store count of inidividul
    // characters and initialize count array as 0
    int count[RANGE + 1], i;
    memset(count, 0, sizeof(count));

    // Store count of each character
    for(i = 0; arr[i]; ++i)
        ++count[arr[i]];

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

    // Build the output character array
    for (i = 0; arr[i]; ++i)
    {
        output[count[arr[i]]-1] = arr[i];
        --count[arr[i]];
    }

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

// Driver program to test above function
int main()
{
    char arr[] = "geeksforgeeks";//"applepp";

    countSort(arr);

    printf("Sorted character array is %sn", arr);
    return 0;
}
```

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
        for (int i = 0; i<n; ++i)
        {
            output[count[arr[i]]-1] = arr[i];
            --count[arr[i]];
        }

        // 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 */
```

Python

```
# Python program for counting sort

# The main function that sort the given string arr[] in
# alphabetical order
def countSort(arr):

    # The output character array that will have sorted arr
    output = [0 for i in range(256)]

    # Create a count array to store count of inidividul
    # characters and initialize count array as 0
    count = [0 for i in range(256)]

    # For storing the resulting answer since the
    # string is immutable
    ans = ["" for _ in arr]

    # Store count of each character
    for i in arr:
        count[ord(i)] += 1

    # Change count[i] so that count[i] now contains actual
    # position of this character in output array
    for i in range(256):
        count[i] += count[i-1]

    # Build the output character array
    for i in range(len(arr)):
        output[count[ord(arr[i])]-1] = arr[i]
        count[ord(arr[i])] -= 1

    # Copy the output array to arr, so that arr now
    # contains sorted characters
    for i in range(len(arr)):
        ans[i] = output[i]
    return ans

# Driver program to test above function
arr = "geeksforgeeks"
ans = countSort(arr)
print "Sorted character array is %s"  %("".join(ans))

# This code is contributed by Nikhil Kumar Singh
```

C#

```
// C# implementation of Counting Sort
using System;

class GFG {

    static void countsort(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
        for (int i = 0; i<n; ++i)
        {
            output[count[arr[i]]-1] = arr[i];
            --count[arr[i]];
        }

        // 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()
    {

        char []arr = {'g', 'e', 'e', 'k', 's', 'f', 'o',
                      'r', 'g', 'e', 'e', 'k', 's'
                      };
```

```
        countsort(arr);

        Console.Write("Sorted character array is ");
        for (int i=0; i<arr.Length; ++i)
            Console.Write(arr[i]);
    }
}

// This code is contributed by Sam007.
```

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)

Points to be noted:

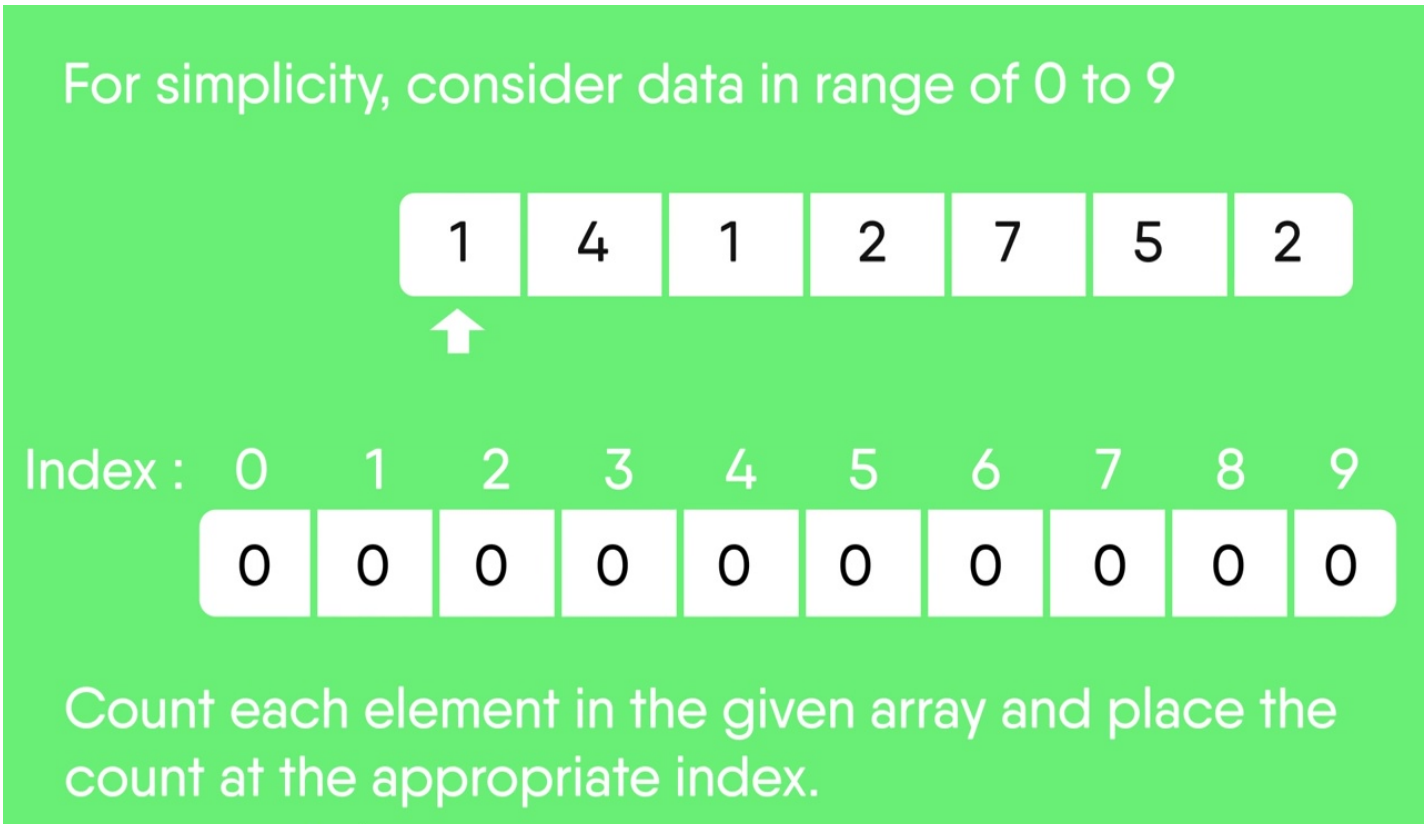
1. Counting sort is efficient if the range of input data is not significantly greater than the number of objects to be sorted. Consider the situation where the input sequence is between range 1 to 10K and the data is 10, 5, 10K, 5K.
2. It is not a comparison based sorting. It running time complexity 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.

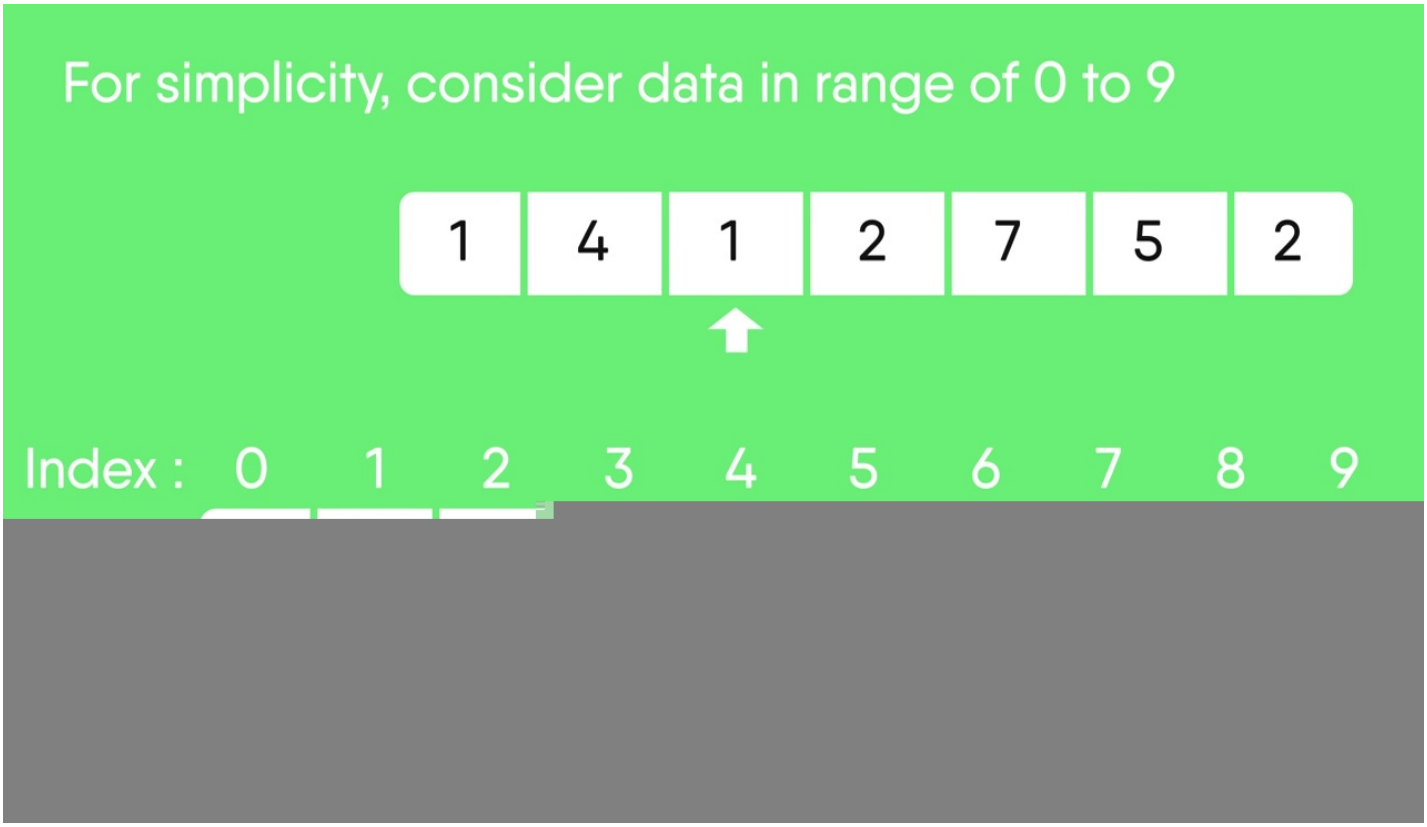
Exercise:

1. Modify above code to sort the input data in the range from M to N.
2. Modify above code to sort negative input data.
3. Is counting sort stable and online?
4. Thoughts on parallelizing the counting sort algorithm.

Counting Sort | GeeksforGeeks

Snapshots:





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.

Practice Tags : Strings Mathematical Sorting

Article Tags : Mathematical Sorting Strings

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Radix Sort
Bucket Sort
QuickSort
Heap Sort
Merge Sort
Recursive program to check if number is palindrome or not
Number of squares of maximum area in a rectangle
Number of ones in the smallest repunit
Stormer Numbers
Multiply the given number by 2 such that it is divisible by 10

(Login to Rate)

2.3 Average Difficulty : **2.3/5.0**
Based on **119** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

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

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org