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 (a kind of hashing). Then do some arithmetic operations to calculate the position of each object in the output sequence.

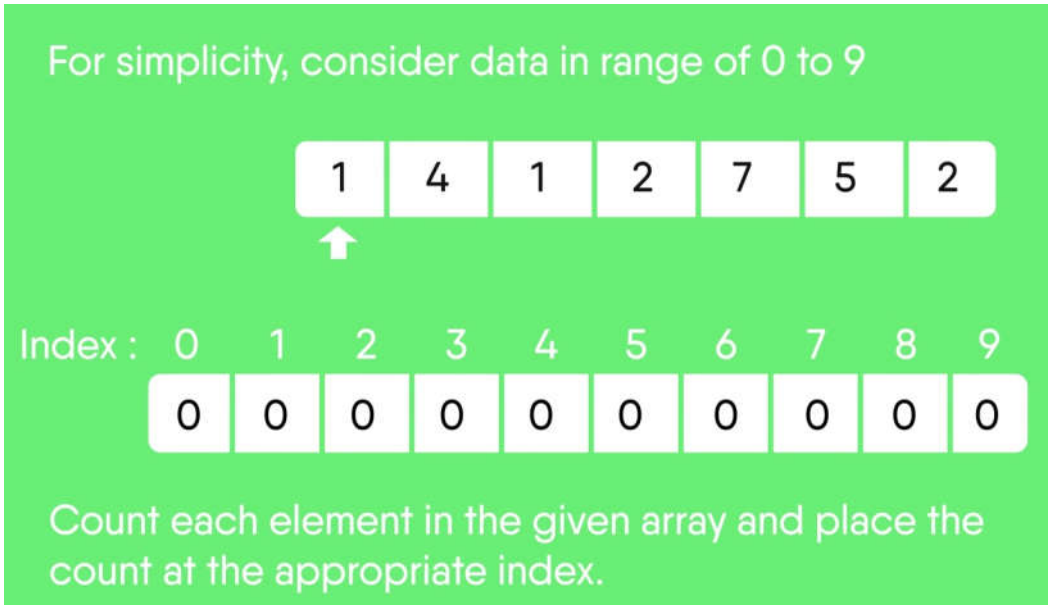
Characteristics of counting sort:

- Counting sort makes assumptions about the data, for example, it assumes that values are going to be in the range of 0 to 10 or 10 – 99, etc, Some other assumption counting sort makes is input data will be positive integers.
- Like other algorithms this sorting algorithm is not a comparison-based algorithm, it hashes the value in a temporary count array and uses them for sorting.
- It uses a temporary array making it a non-In Place algorithm.

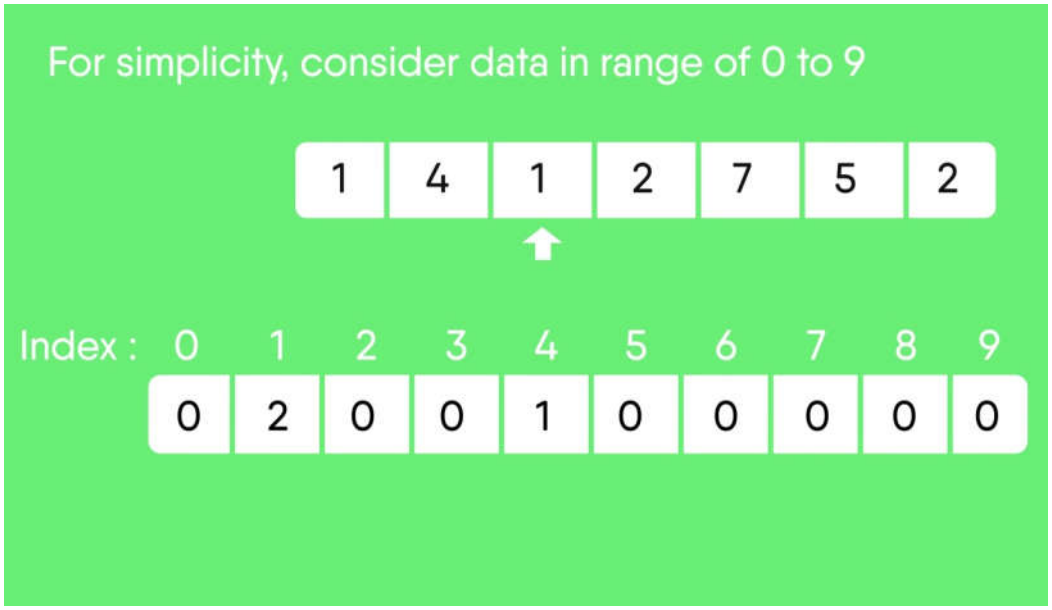
Example:

- For simplicity, consider the data in the range of 0 to 9.
- **Input data:** {1, 4, 1, 2, 7, 5, 2}
- Take a count array to store the count of each unique object.

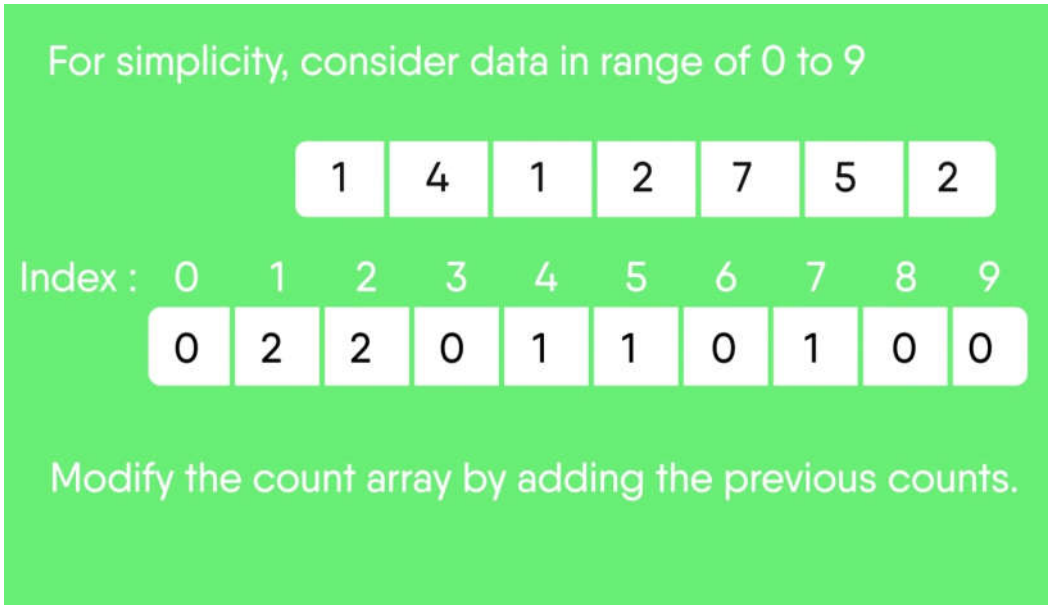
Follow the below illustration for a better understanding of the counting sort algorithm



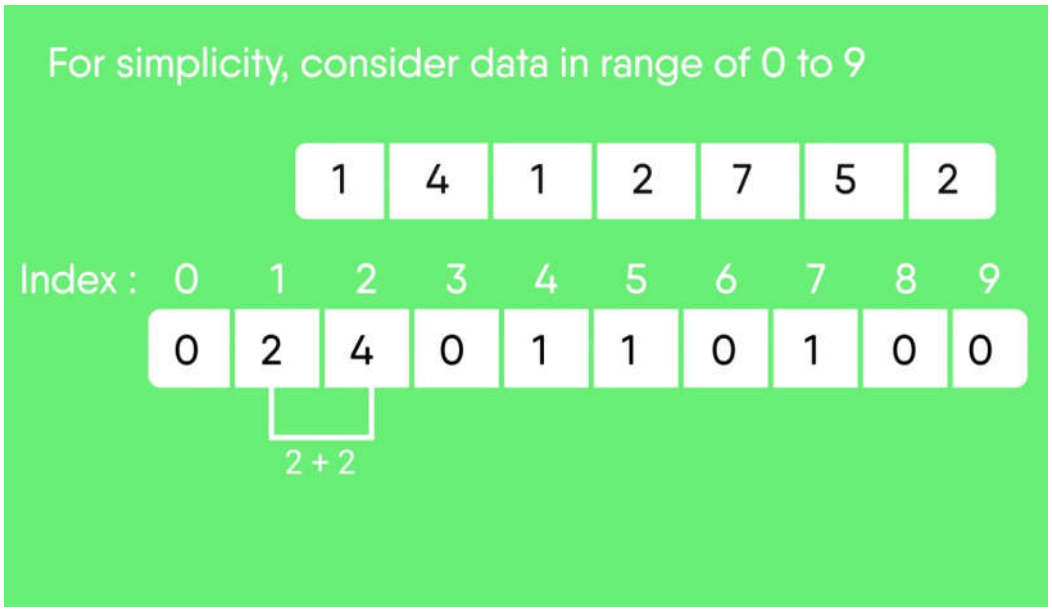
- Now, store the count of each unique element in the count array
- If any element repeats itself, simply increase its count.

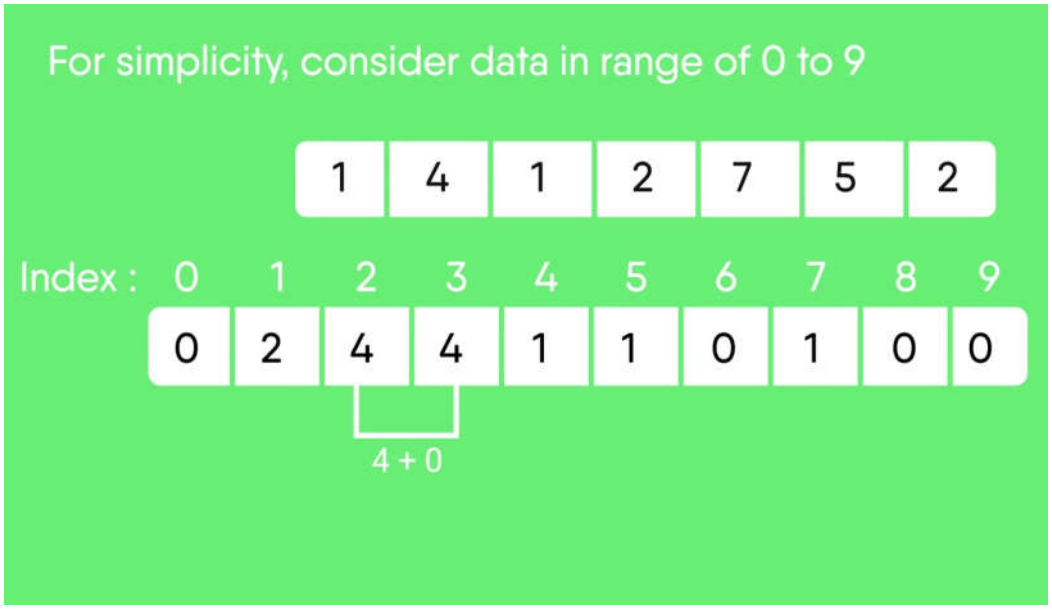


- Here, the count of each unique element in the count array is as shown below:
  - **Index:** 0 1 2 3 4 5 6 7 8 9
  - **Count:** 0 2 2 0 1 1 0 1 0 0



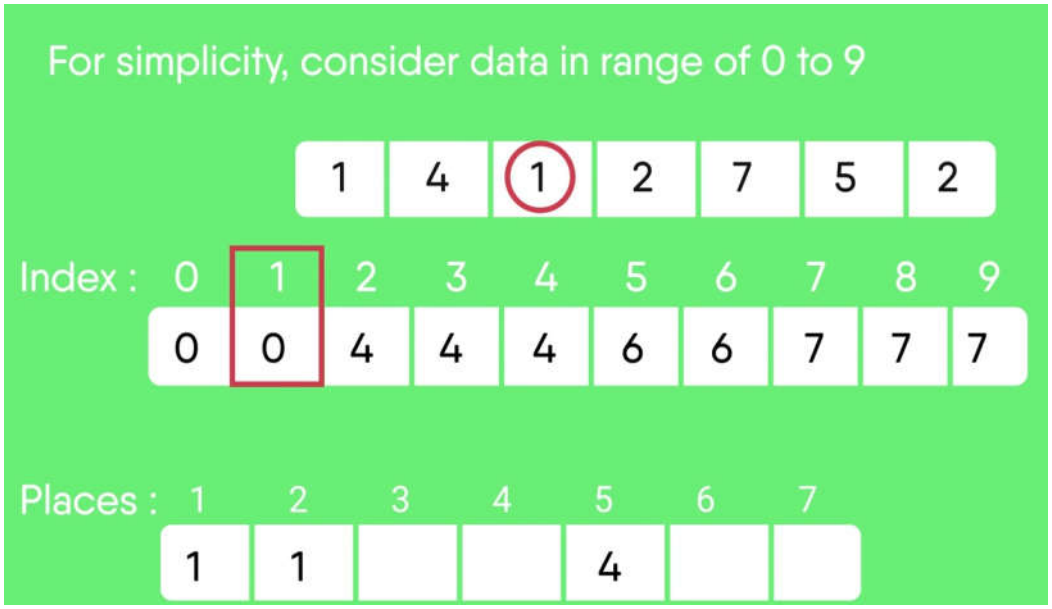
- Modify the count array such that each element at each index stores the sum of previous counts.
  - **Index:** 0 1 2 3 4 5 6 7 8 9
  - **Count:** 0 2 4 4 5 6 6 7 7 7
- The modified count array indicates the position of each object in the output sequence.
- Find the index of each element of the original array in the count array. This gives the cumulative count.





- Rotate the array clockwise for one time.
  - **Index:** 0 1 2 3 4 5 6 7 8 9
  - **Count:** 0 0 2 4 4 5 6 6 7 7

- Output each object from the input sequence followed by increasing its count by 1.
- Process the input data: {1, 4, 1, 2, 7, 5, 2}. The position of 1 is 0.
- Put data 1 at index 0 in output. Increase count by 1 to place next data 1 at an index 1 greater than this index.



- After placing each element in its correct position, decrease its count by one.

Below is the implementation of the above algorithm:

C++

```
// C++ Program for counting sort
#include <bits/stdc++.h>
#include <string.h>
using namespace std;
#define RANGE 255

// The main function that sort
// the given string arr[] in
// alphabetical 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 individual
    // 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]];
```

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

    /*
    For Stable algorithm
    for (i = sizeof(arr)-1; i>=0; --i)
    {
        output[count[arr[i]]-1] = arr[i];
        --count[arr[i]];
    }

    For Logic : See implementation
    */

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

// Driver code
int main()
{
    char arr[] = "geeksforgeeks";

    // Function call
    countSort(arr);

    cout << "Sorted character array is " << arr;
    return 0;
}

// This code is contributed by rathbhupendra
```

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
// alphabetical order
```

```
    // Create a count array to store count of individual
    // 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]];
    }

    /*
     For Stable algorithm
     for (i = sizeof(arr)-1; i>=0; --i)
    {
        output[count[arr[i]]-1] = arr[i];
        --count[arr[i]];
    }

    For Logic : See implementation
    */

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

// Driver code
int main()
{
    char arr[] = "geeksforgeeks"; //"applepp";

    // Function call
    countSort(arr);

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

```java
// Java implementation of Counting Sort

import java.io.*;

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 individual
        // 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]];
        }

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

```
        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 */
```

Python3

```
# Python3 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(len(arr))]

    # Create a count array to store count of individual
    # 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
```

```
    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 individual 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]];
        }
```

```
            arr[i] = output[i];
    }

    // Driver code
    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.
```

PHP

```
<?php
// PHP Program for counting sort

$RANGE = 255;

// The main function that sort
// the given string arr[] in
// alphabetical order
function countSort($arr)
{
    global $RANGE;

    // The output character array
    // that will have sorted arr
    $output = array(strlen($arr));
    $len = strlen($arr);

    // Create a count array to
    // store count of individual
    // characters and initialize
    // count array as 0
    $count = array_fill(0, $RANGE + 1, 0);

    // Store count of
    // each character
```

```
    // 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
    // To make it stable we are operating
    // in reverse order.
    for ($i = $len-1; $i >= 0 ; $i--)
    {
        $output[$count[ord($arr[$i])] - 1] = $arr[$i];
        --$count[ord($arr[$i])];
    }

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

// Driver Code
$arr = "geeksforgeeks"; //"applepp";

$arr = countSort($arr);

echo "Sorted character array is " . $arr;

// This code is contributed by mits
?>
```

Javascript

```
// Javascript implementation of Counting Sort
function sort(arr)
{
    var n = arr.length;

    // The output character array that will have sorted arr
    var output = Array.from({length: n}, (_, i) => 0);

    // Create a count array to store count of individual
    // characters and initialize count array as 0
```

```
    for (var i = 0; i < n; ++i)
        ++count[arr[i].charCodeAt(0)];
    // Change count[i] so that count[i] now contains actual
    // position of this character in output array
    for (var 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 (var i = n - 1; i >= 0; i--) {
        output[count[arr[i].charCodeAt(0)] - 1] = arr[i];
        --count[arr[i].charCodeAt(0)];
    }

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

// Driver method
    var arr = [ 'g', 'e', 'e', 'k', 's', 'f', 'o',
                'r', 'g', 'e', 'e', 'k', 's' ];

    arr = sort(arr);
    document.write("Sorted character array is ");
    for (var i = 0; i < arr.length; ++i)
        document.write(arr[i]);

// This code is contributed by shikhasingrajput
```

Output

```
Sorted character array is eeeefggkkorss
```

Time Complexity: O(N + K) where N is the number of elements in the input array and K is the range of input.
Auxiliary Space: O(N + K)

Counting Sort for an Array with negative elements:

To solve the problem follow the below idea:

The problem with the previous counting sort was that we could not sort the

So what we do is, find the minimum element and we will store the count of that minimum element at the zero index

Below is the implementation of the above approach:

C++

```
// C++ program for the above approach
#include <bits/stdc++.h>
using namespace std;

void countSort(vector<int>& arr)
{
    int max = *max_element(arr.begin(), arr.end());
    int min = *min_element(arr.begin(), arr.end());
    int range = max - min + 1;

    vector<int> count(range), output(arr.size());
    for (int i = 0; i < arr.size(); i++)
        count[arr[i] - min]++;

    for (int i = 1; i < count.size(); i++)
        count[i] += count[i - 1];

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

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

void printArray(vector<int>& arr)
{
    for (int i = 0; i < arr.size(); i++)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver code
int main()
{
```

```
    printArray(arr);
    return 0;
}
```

Java

```
// Java program for the above approach
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)
    {
```

```
            printArray(arr);
        }
    }

    // This code is contributed by princiRaj1992
```

Python3

```
# Python3 program for counting sort
# which takes negative numbers as well

# The function that sorts the given arr[]


def count_sort(arr):
    max_element = int(max(arr))
    min_element = int(min(arr))
    range_of_elements = max_element - min_element + 1
    # Create a count array to store count of individual
    # elements and initialize count array as 0
    count_arr = [0 for _ in range(range_of_elements)]
    output_arr = [0 for _ in range(len(arr))]

    # Store count of each character
    for i in range(0, len(arr)):
        count_arr[arr[i]-min_element] += 1

    # Change count_arr[i] so that count_arr[i] now contains actual
    # position of this element in output array
    for i in range(1, len(count_arr)):
        count_arr[i] += count_arr[i-1]

    # Build the output character array
    for i in range(len(arr)-1, -1, -1):
        output_arr[count_arr[arr[i] - min_element] - 1] = arr[i]
        count_arr[arr[i] - min_element] -= 1

    # Copy the output array to arr, so that arr now
    # contains sorted characters
    for i in range(0, len(arr)):
        arr[i] = output_arr[i]

    return arr


# Driver code
if __name__ == '__main__':
```

C#

```
// C# program for the above approach
using System;
using System.Collections.Generic;
using System.Linq;
class GFG {
    static void countSort(int[] arr)
    {
        int max = arr.Max();
        int min = arr.Min();
        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++) {
            Console.Write(arr[i] + " ");
        }
        Console.WriteLine("");
    }

    // 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 rutvik_56.
```

```
// Counting sort which takes negative numbers as well

    function countSort(arr)
    {
    var max = Math.max.apply(Math, arr);
    var min = Math.min.apply(Math, arr);

    var range = max - min + 1;
    var count = Array.from({length: range}, (_, i) => 0);
    var output = Array.from({length: arr.length}, (_, i) => 0);
    for (i = 0; i < arr.length; i++) {
        count[arr[i] - min]++;
    }

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

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

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

function printArray(arr)
{
    for (i = 0; i < arr.length; i++)
    {
        document.write(arr[i] + " ");
    }
document.write('<br>');
}

// Driver code
var arr = [ -5, -10, 0, -3, 8, 5, -1, 10 ];
countSort(arr);
printArray(arr);

// This code is contributed by Amit Katiyar
```

Output

Time complexity: O(N), where N is the total number of elements

Auxiliary Space: O(N)

Another Approach for counting sort for positive numbers

In this approach, we are going to do the same thing as explained above but we will be implementing using the map data structure of C++.

C++

```
#include <bits/stdc++.h>
using namespace std;

vector<int> countingSort(vector<int> vec, int n)
{
    for (int i = 0; i<n; cin>> vec[i], i++)
        ;
    map<int, int> count;
    // Here we are initializing every element of count to 0
    // from 1 to n
    for (int i = 0; i < n; count[i] = 0, i++)
        ;
    // Here we are storing count of every element
    for (int i = 0; i < n; count[vec[i]]++, i++)
        ;
    vector<int> sortedArr;
    int i = 0;
    while (n > 0) {
        // Here we are checking if the count[element] = 0
        // then incrementing for the next Element
        if (count[i] == 0) {
            i++;
        }
        // Here we are inserting the element into the
        // sortedArr decrementing count[element] and n by 1
        else {
            sortedArr.push_back(i);
            count[i]--;
            --n;
        }
    }
    return sortedArr;
}
```

```
        ;
    cout << endl;
}
signed main()
{
    vector<int> vec1 = { 6, 0, 7, 8, 7, 2, 0 };
    vector<int> sortedArr1
        = countingSort(vec1, vec1.size());
    printArr(sortedArr1, sortedArr1.size());

    vector<int> vec2 = { 4, 8, 1, 0, 1, 1, 0, 0 };
    vector<int> sortedArr2
        = countingSort(vec2, vec2.size());
    printArr(sortedArr2, sortedArr2.size());

    return 0;
}
```

Java

```
// Java code to implement the approach
import java.io.*;
import java.util.*;
import java.util.HashMap;

class GFG {

static ArrayList<Integer> countingSort(ArrayList<Integer> vec, Integer n)
{
    HashMap<Integer, Integer> count = new HashMap<>();

    // Here we are initializing every element of count to 0
    // from 1 to n
    Integer i;
    for (i = 0; i < n; i++)
    count.put(i,0);

    // Here we are storing count of every element
    for (i = 0; i < n; i++)
    {
    if(count.containsKey(vec.get(i)))
        count.put(vec.get(i),count.get(vec.get(i))+1);
    else
        count.put(vec.get(i),1);
    }
    ArrayList<Integer> sortedArr=new ArrayList<Integer>();
    i = 0;
```

```
        // then incrementing for the next Element
        if (count.get(i) == 0) {
            i++;
        }

        // Here we are inserting the element Integer the
        // sortedArr decrementing count[element] and n by 1
        else {
            sortedArr.add(i);
            count.put(i,count.get(i)-1);
            n--;
        }
        }
        return sortedArr;
    }

    static void printArr(ArrayList<Integer> vec, Integer n)
    {
        System.out.print("Sorted Array: ");
        for (Integer i = 0; i < n; i++)
        System.out.print(vec.get(i) + " ");
        System.out.print("\n");
    }
    public static void main (String[] args)
    {
        ArrayList<Integer> vec1 = new ArrayList<Integer>(Arrays.asList( 6, 0, 7, 8
        ArrayList<Integer> sortedArr1 = countingSort(vec1, vec1.size());
        printArr(sortedArr1, sortedArr1.size());

        ArrayList<Integer> vec2 = new ArrayList<Integer>(Arrays.asList( 4, 8, 1, 0
        ArrayList<Integer> sortedArr2 = countingSort(vec2, vec2.size());
        printArr(sortedArr2, sortedArr2.size());
    }
    }

    // This code was contributed by Pushpesh Raj.
```

Python3

```
def countingSort(vec, n):
    #for (int i = 0; i<n; cin>> vec[i], i++)
    count=dict();

    # Here we are initializing every element of count to 0
    # from 1 to n
    for i in range(0,n):
```

```
        if vec[i] in count.keys():
            count[vec[i]] += 1;
        else:
            count[vec[i]] = 1;


    sortedArr = [];
    i = 0;
    while (n > 0):
        # Here we are checking if the count[element] = 0
        # then incrementing for the next Element
        if (count[i] == 0) :
            i += 1;

        # Here we are inserting the element into the
        # sortedArr decrementing count[element] and n by 1
        else:
            sortedArr.append(i);
            count[i] -= 1;
            n = n - 1;

    return sortedArr;


def printArr(vec, n):
    print("Sorted Array: ");
    for i in range(0,n):
        print(vec[i], " ");

vec1 = [ 6, 0, 7, 8, 7, 2, 0 ];
sortedArr1 = countingSort(vec1, len(vec1));
printArr(sortedArr1, len(sortedArr1));

vec2 = [ 4, 8, 1, 0, 1, 1, 0, 0 ];
sortedArr2 = countingSort(vec2, len(vec2));
printArr(sortedArr2, len(sortedArr2));

# This code is contributed by ritaagarwal.
```

C#

```
// C# code to implement the approach
using System;
using System.Collections.Generic;

class GFG {
```

```
    // Here we are initializing every element of count to 0
    // from 1 to n
    int i;
    for (i = 0; i < n; i++)
      count.Add(i,0);

    // Here we are storing count of every element
    for (i = 0; i < n; i++)
    {
      if(count.ContainsKey(vec[i]))
        count[vec[i]]++;
      else
        count[vec[i]]=1;
    }
    List<int> sortedArr=new List<int>();
    i = 0;
    while (n > 0)
    {

      // Here we are checking if the count[element] = 0
      // then incrementing for the next Element
      if (count[i] == 0) {
        i++;
      }

      // Here we are inserting the element into the
      // sortedArr decrementing count[element] and n by 1
      else {
        sortedArr.Add(i);
        count[i]--;
        n--;
      }
    }
    return sortedArr;
  }

  static void printArr(List<int> vec, int n)
  {
    Console.Write("Sorted Array: ");
    for (int i = 0; i < n; i++)
      Console.Write(vec[i] + " ");
    Console.Write("\n");
  }
  public static void Main()
  {
    List<int> vec1 = new List<int>{ 6, 0, 7, 8, 7, 2, 0 };
    List<int> sortedArr1 = countingSort(vec1, vec1.Count);
    printArr(sortedArr1, sortedArr1.Count);
```

```
}

// This code was contributed by poojaagrawal2.
```

Javascript

```
const countingSort = (vec, n) => {

  // map to store the count of each element
  const count = {};

  // initialize every element of count to 0 from 0 to n-1
  for (let i = 0; i < n; count[i] = 0, i++);

  // store count of every element
  for (let i = 0; i < n; count[vec[i]]++, i++);
  const sortedArr = [];
  let i = 0;
  while (n > 0)
  {

    // if the count of the element is 0, then increment i
    // and move to the next element
    if (count[i] === 0) {
      i++;
    }

    // else, insert the element into the sorted array,
    // decrement count[element] and n by 1
    else {
      sortedArr.push(i);
      count[i]--;
      n--;
    }
  }
  return sortedArr;
};

const printArr = (vec, n) => {
  console.log("Sorted Array: ");
  for (let i = 0; i < n; console.log(vec[i] + " "), i++);
  console.log("\n");
};

const vec1 = [6, 0, 7, 8, 7, 2, 0];
const sortedArr1 = countingSort(vec1, vec1.length);
printArr(sortedArr1, sortedArr1.length);
```

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

Output

```
Sorted Array: 0 0 2 6 7 7 8
Sorted Array: 0 0 0 1 1 1 4 8
```

Time complexity: O(N), where N is the total number of elements
Auxiliary Space: O(N)

Important points:

- 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 the range 1 to 10K and the data is 10, 5, 10K, 5K.
- It is not a comparison-based sorting. Its running time complexity is O(n) with space proportional to the range of data.
- Counting sorting is able to achieve this because we are making assumptions about the data we are sorting.
- It is often used as a sub-routine to another sorting algorithm like the radix sort.
- Counting sort uses partial hashing to count the occurrence of the data object in O(1).
- The counting sort can be extended to work for negative inputs also.
- Counting sort is a stable algorithm. But it can be made stable with some code changes.

Exercise:

- Modify the above code to sort the input data in the range from M to N.
- Modify the code to make the counting sort stable.
- Thoughts on parallelizing the counting sort algorithm.

Related Articles:

- Quiz on Counting Sort
- Coding Practice for Sorting

Radix Sort, Counting Sort, Bucket Sort, ShellSort, Comb Sort, PigeonHole Sorting

This article is contributed by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 20 Apr, 2023

182

Recommended Problem

Counting Sort

Sorting Algorithms Microsoft Goldman Sachs +1 more

Solve Problem

Submission count: 32.7K

Similar Reads

1. C Program for Counting Sort
2. Java Program for Counting Sort
3. Sort an array of 0s, 1s and 2s (Simple Counting)
4. Implementing Counting Sort using map in C++
5. Counting Sort Visualization using JavaScript
6. Kth smallest or largest element in unsorted Array using Counting Sort
7. Median and Mode using Counting Sort
8. Find duplicates in an Array with values 1 to N using counting sort
9. Comparison among Bubble Sort, Selection Sort and Insertion Sort
10. Counting Inversions using Ordered Set and GNU C++ PBDS

2. Introduction to Max-Heap – Data Structure and Algorithm Tutorials
3. Introduction to Set – Data Structure and Algorithm Tutorials
4. Introduction to Map – Data Structure and Algorithm Tutorials
5. What is Dijkstra's Algorithm? | Introduction to Dijkstra's Shortest Path Algorithm

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : spattk, Mithun Kumar, krikti, princiraj1992, sagarudasi2, rathbhupendra, pushpeshrajdx01, abhijitjadhav1998, rutvik_56, samrat230599, amit143katiyar, soumyadip saha 1, sam_2200, shikhasingrajput, me190003045, poojaagrawal2, surinderdawra388, surindertarika1234, sagartomar9927, ratiagrawal, ishankhandelwals, kashishkumar2, mitalibhola94, noviced3vq6, janardansthox, princekumaras, suvarnasanapathi2001, chris_r

Article Tags : counting-sort, Samsung, Snapdeal, Algorithms, DSA

Practice Tags : Samsung, Snapdeal, Algorithms

Improve Article | Report Issue

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Interview Corner

Company Preparation

Preparation for SDE

Company Interview Corner

Experienced Interview

Internship Interview

Competitive Programming

Aptitude

Python

Python Tutorial

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC/SSC/BANKING

SSC CGL Syllabus

SBI PO Syllabus

IBPS PO Syllabus

UPSC Ethics Notes

UPSC Economics Notes

UPSC History Notes