KOTEBE METROPOLITAN UNIVERSITY

DATA STRUCTURE AND ALGORITHM

Group Members

Name ***ID No:***

1. Miliyon Birhanu……..CNCS/UR15320/12
2. Motuma Tefera……..CNCS/UR15337/12
3. Werkina Megarsa…….CNCS/UR15516/12
4. Oliyad Zelalem…….CNCS/UR12385 /12
5. Melaku H/Mariam…….CNCS/UR15293/12

Submission Date:… December 6 , 2021

Contents

- *Basic Idea*
- *Algorithm*
- *Implementation*
- *Analysis*

COUNTING SORT

Basic Idea:

- Counting sort is sorting an algorithm that sorts the elements of an array by counting the number of occurrences of each unique element in the array.
- The count is stored in an auxiliary array and the sorting is done by mapping the count as an index of the auxiliary array.

- This mapping is done by performing arithmetic calculations on those counts to determine the position of each key value (unique element) in the output sequences.
- It is often used as subroutine in an other sorting algorithm, radix sort, that can handle larger keys more efficiently.
- It is not a comparison sort.

Algorithm: counting sort

Step1: take input array and range(number of unique integer values involved).

Step2: create the output array of size same as input array. Create count array with size equal to the range and initialize values to zero.

Step3: count each element in the input array and place the count at the appropriate index of the count array.

Step4: modify the count array by adding the previous counts (cumulative). The modified count array indicates the position of each object/element in the output array.

Step5: output each object from the input array in to the sorted output array followed by decreasing its count by one.

Step6: print the sorted output array.

Implementation: counting sort

```
Countingsort()
{
Input_array[size]
Out_put array[size]
Range(or no of unique elements)
For int(int i = 0 to i < range)          //create count_array[range]&
        Count array[i]  = 0              //initialize all values to 0.
For int(int i = 0 to i < size)           //count each elements &
          ++ count_array[input_array[i]]  // place it in count_array
```

```
For int(int i = 1 to i < range)    //modify count_array[] to store
                                   //previous counts (cummulutive)
      Count_array[i] = cout_array[i] + cout_array[i-1]
For int(int i = 0 to i < size)  // place elements from input_array[] to
                                //output_array[]using this cont array[]
                                //that has the actual position of elements,
      output_array[--count_array[input_array[i]]] =     input_array[i]
For int(int i = 0 to i < size)   //transfer sorted values from output_array[]
                                 //to input_array[]
Input_array[i] = output_array[i]
```

Examples

```
#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]];
      // 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";

    countSort(arr);

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

Analysis: counting sort

- ✓ Time complexity: $O(n + k)$
- ✓ space complexity: $O(n + k)$

*Where **n** is the number of elements in input array and*

***k** is the range of input(e.g. Max - Min).*

THANK YOU !

THE END