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.
```


Following is C implementation of counting sort.

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

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

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

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

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

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

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

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

    countSort(str);
```


Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    printf("Sorted string is %s\n", str);
    return 0;
}
```

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.

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.

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

18 Comments GeeksforGeeks

Sort by Newest

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 11 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 15 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 40 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 41 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

Mayank • a month ago

http://www8.cs.umu.se/kurser/5... for parallel counting sort..

Reply • Share ›

Ayaan Ali • a month ago

what will b its output ?

Reply • Share ›

Nikhil Sreekumar • 2 months ago

For integers, add absolute value of largest negative number and then use cou

Reply • Share ›

Jon Smith • 2 months ago

a good

Referance: http://cybarlab.com/program-fo...

Reply • Share ›

Jon Smith • 2 months ago

A nice example about Program for Counting Sort:
http://cybarlab.com/program-fo...

Reply • Share ›

Code_Addict • 5 months ago

About thoughts on parallelizing the counting sort algorithm-

http://www.drdobbs.com/archite...

1 Reply • Share ›

jay • 8 months ago

its complexity will be increased as u say .....
for(i=0;i<range;i++) {="" for(j="0;j<RANGE[i];j++)" printf("%d",i);="" }="" ....="
...="" suppose="" array="" is="" 1="" 5="" 5="" 5="" 5="" 5="" 5="" 5="" 5="" 5="
is="" much="" greater="" as="" compared="" to="" o(10+11)="" i.e.="" o(range

Reply • Share ›

jay • 8 months ago

its complexity will be increased as u say .....
for(i=0;i<range;i++) {="" for(j="0;j<RANGE[i];j++)" printf("%d",i);="" }="" ....="
...="" suppose="" array="" is="" 1="" 5="" 5="" 5="" 5="" 5="" 5="" 5="" 5="" 5="
is="" much="" greater="" as="" cmpared="" to="" o(10+11)="" i.e.="" o(range+

Reply • Share ›

rakitic • 10 months ago

@geeks...i dont think there is any use of last step , in step 2 only...we can trav
the numbers based on the count . what do u say ??

Reply • Share ›

jay ➚ rakitic • 8 months ago

its complexity will be increased as u say .....

complexity O(RANGE* RANGE[i])

2 Reply • Share ›

ankur jain • 10 months ago

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>


void countSort(char str[])
```

```
{
        int count[256]={0},len,i;
        int pos=0;
        len=strlen(str);
        for (i = 0; i < len ; ++i)
        {
                count[str[i]]++;
        }

        for (i = 0; i < 256 ; ++i)
        {
```

see more

• Reply • Share ›

rk_roy ankur jain • 10 months ago

@ankur jain..... i got it after seeing ur code...thnx
and GFG u guys are worth 'Hats off'...

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

AMIT • 11 months ago

Someone please give some thoughts on parallelizing the counting sort

• Reply • Share ›

Dnyaneshwar • a year ago

#include
main()
{
int n,nn,i,j;

printf("\nEnter the max val of array");
scanf("%d",&n);

printf("Enter the how many number u want");
scanf("%d",&nn);

int a[nn],b[nn];
printf("\nEnter the number smaller than %d\n",n);
for(i=1;i<=nn;i++)
{
scanf("%d",&a[i]);
}
for(i=1;i<=n;i++)
{
b[i]=0;

see more

• Reply • Share ›

gh05t • a year ago

Here is a code which works fro neagtive range,the code is very easy to paralle thread for each chunk of data in each of the three steps.

[sourcecode language="ruby"]
a=gets.to_i #range is [a,b]
b=gets.to_i
A=Hash.new(0)
n=gets.to_i #no. of elements
i=0
B=Array.new
C=Array.new(n)
while i<n
k=gets.to_i

A[k]+=1
i+=1
end
i=a

see more

Reply • Share ›

nikhil • a year ago
What does online means here????
Reply • Share ›

as • a year ago
test
Reply • Share ›

Venki • a year ago
Nice exercise questions Aashish. Keep it up.

3. Counting sort can be made stable if we fill from backwards. If you fill in forw
elements will change. I won't commit it is an online algorithm. There will be so
elements to their final position. However, at any point in time, the partial arrays
too.

4. Nice thought on parallel programming on counting sort. I will reply later.

4 Reply • Share ›

Subscribe Add Disqus to your site