Sort elements by frequency | Set 2

Given an array of integers, sort the array according to frequency of elements. For example, if the input array is {2, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12}, then modify the array to {3, 3, 3, 3, 2, 2, 2, 12, 12, 4, 5}.
In the previous post, we have discussed all methods for sorting according to frequency. In this post, method 2 is discussed in detail and C++ implementation for the same is provided.

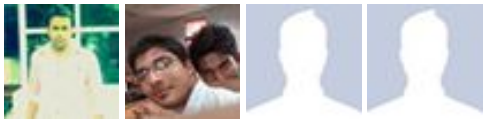
Following is detailed algorithm.
1) Create a BST and while creating BST maintain the count i,e frequency of each coming element in same BST. This step may take O(nLogn) time if a self balancing BST is used.
2) Do Inorder traversal of BST and store every element and count of each element in an auxiliary array. Let us call the auxiliary array as 'count[]'. Note that every element of this array is element and frequency pair. This step takes O(n) time.
3) Sort 'count[]' according to frequency of the elements. This step takes O(nLohn) time if a O(nLogn) sorting algorithm is used.
4) Traverse through the sorted array 'count[]'. For each element x, print it 'freq' times where 'freq' is frequency of x. This step takes O(n) time.

Overall time complexity of the algorithm can be minimum O(nLogn) if we use a O(nLogn) sorting algorithm and use a self balancing BST with O(Logn) insert operation.

Following is C++ implementation of the above algorithm.

```
// Implementation of above algorithm in C++.
#include <iostream>
#include <stdlib.h>
using namespace std;

/* A BST node has data, freq, left and right pointers */
```

```
struct BSTNode
{
    struct BSTNode *left;
    int data;
    int freq;
    struct BSTNode *right;
};

// A structure to store data and its frequency
struct dataFreq
{
    int data;
    int freq;
};

/* Function for qsort() implementation. Compare frequencies to
   sort the array according to decreasing order of frequency */
int compare(const void *a, const void *b)
{
    return ( (*(const dataFreq*)b).freq - (*(const dataFreq*)a).freq )
}

/* Helper function that allocates a new node with the given data,
   frequency as 1 and NULL left and right  pointers.*/
BSTNode* newNode(int data)
{
    struct BSTNode* node = new BSTNode;
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    node->freq = 1;
    return (node);
}

// A utility function to insert a given key to BST. If element
// is already present, then increases frequency
BSTNode *insert(BSTNode *root, int data)
{
    if (root == NULL)
        return newNode(data);
    if (data == root->data) // If already present
        root->freq += 1;
    else if (data < root->data)
        root->left = insert(root->left, data);
    else
        root->right = insert(root->right, data);
    return root;
```


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

```
}

// Function to copy elements and their frequencies to count[].
void store(BSTNode *root, dataFreq count[], int *index)
{
    // Base Case
    if (root == NULL) return;

    // Recur for left substree
    store(root->left, count, index);

    // Store item from root and increment index
    count[(*index)].freq = root->freq;
    count[(*index)].data = root->data;
    (*index)++;

    // Recur for right subtree
    store(root->right, count, index);
}
```

```
// The main function that takes an input array as an argument
// and sorts the array items according to frequency
void sortByFrequency(int arr[], int n)
{
    // Create an empty BST and insert all array items in BST
    struct BSTNode *root = NULL;
    for (int i = 0; i < n; ++i)
        root = insert(root, arr[i]);

    // Create an auxiliary array 'count[]' to store data and
    // frequency pairs. The maximum size of this array would
    // be n when all elements are different
    dataFreq count[n];
    int index = 0;
    store(root, count, &index);

    // Sort the count[] array according to frequency (or count)
    qsort(count, index, sizeof(count[0]), compare);

    // Finally, traverse the sorted count[] array and copy the
    // i'th item 'freq' times to original array 'arr[]'
    int j = 0;
    for (int i = 0; i < index; i++)
    {
        for (int freq = count[i].freq; freq > 0; freq--)
            arr[j++] = count[i].data;
    }
```

```
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

/* Driver program to test above functions */
int main()
{
    int arr[] = {2, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12};
    int n = sizeof(arr)/sizeof(arr[0]);
    sortByFrequency(arr, n);
    printArray(arr, n);
    return 0;
}
```

Output:

```
3 3 3 3 2 2 2 12 12 5 4
```

Exercise:
The above implementation doesn't guarantee original order of elements with same frequency (for example, 4 comes before 5 in input, but 4 comes after 5 in output). Extend the implementation to maintain original order. For example, if two elements have same frequency then print the one which came 1st in input array.

This article is compiled by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recent Comments

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.

40 Comments GeeksforGeeks

Sort by Newest

zzer • 21 days ago

why not simply use map in STL？

 • Reply • Share ›

Suryabhan Singh • 8 months ago

another solution in cpp without creating BST

```
bool compare(pair<int,int> a,pair<int,int> b)
{
    return(a.second > b.second);
}

void fun(int a[],int size)
{
    sort(a,a+size);
    pair<int ,int=""> p;
    int i=0;
    vector<pair<int,int> > v;
    while(i<size) {="" p.first="a[i];" p.second="1;" while(a[i]="a[i-
```

 • Reply • Share ›

Kai Luo • 8 months ago

we can use counting sort to reduce the time complexity down to O(n + len(arr

 • Reply • Share ›

itengineer ➦ Kai Luo • 3 months ago

Space complexity would increase to max(a[n])

 • Reply • Share ›

Jones ➚ Kai Luo • 7 months ago

number's range is not fixed so count sort can't be used.

^ | v • Reply • Share ›

Sarath Chandra Prasad • 9 months ago

ignore ..static int a=35;

^ | v • Reply • Share ›

Sarath Chandra Prasad • 9 months ago

ignore ..static int a=35;

^ | v • Reply • Share ›

Sarath Chandra Prasad • 9 months ago

#include<stdio.h>
static int a=35;.

int main()
{

int n=0, i,j, temp, a[]= {12, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12};.

sort(a, 0,10);.

for(i=0;i<11;i++).

{.

printf("n%d", a[i]);.

}.
}

void sort(int a[], int i, int n).

see more

• Reply • Share ›

Sarath Chandra Prasad • 9 months ago

#include<stdio.h>
static int a=35;.

int main()
{

int n=0, i,j, temp, a[]= {12, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12};.

sort(a, 0,10);.

for(i=0;i<11;i++).

{.

printf("n%d", a[i]);.

}.
}

void sort(int a[], int i, int n).

see more

• Reply • Share ›

himanshu • 10 months ago

```
/*Here is an approach using sorting and without indexes of elements.I

#include<stdio.h>
#include<stdio.h>
#include<malloc.h>
```

```
#define inf 100000000

struct b
{
    int val;
    int count;
};
void merge(int a[],int p,int q,int r)
{
    int i,j,k;

    int n1=q-p+1;
    int n2=r-q;
```

see more

Reply • Share ›

Himanshu • 10 months ago

Can any one tell me what's the problem with this approach

i)can out one by one element from input and check in the other array (a struct initially empty) if the element is present there than just increase the frequency frequency one.

2)sort that array by freq and print.

i think its a simple one having extra space (same as above one)and time com

Reply • Share ›

itengineer21 Himanshu • 3 months ago

how come it would be O(nlogn). You would need nlogn just to sort your complexity for checking each n every element from input array and incr the struct array.

• Reply • Share ›

sambhavsharma Himanshu • 3 months ago

Size of the structure array? say you have an input array of size 50 and the size of your structure array?

• Reply • Share ›

Pankaj Goyal • 10 months ago

how is it a counting sort??

• Reply • Share ›

Niranjan Sharma • 11 months ago

this is only useful when given numbers are in specific range..
what if array is something like this arr[] = {150007 , 1 , 300005 , 150007 , 4999

• Reply • Share ›

Arulmozhi • a year ago

we dont need a bst when we are going to store the count in auxiliary array. @M
post.

2 • Reply • Share ›

alexchao • a year ago

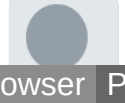
I think we can use a map,to solve this problem;As map structure using red-bla
search tree !

• Reply • Share ›

Tapas Mahanta • a year ago

plus ..by creating..bst..the code will be pretty lengthy

• Reply • Share ›



Tapas Mahanta • a year ago

you can just create the count[ ] array..by traversing the original array and while original array..do count[n]++...this way...freq of 2 is stored at count[2]..and so

2 ^ | v • Reply • Share ›

sambhavsharma ➦ Tapas Mahanta • 3 months ago

There is no limit on an element's value. So creating a freq array in not

^ | v • Reply • Share ›

hxgxs1 • a year ago

Please excuse the time complexity of this code, the logic is quit similar to the many times a particular element occurs in the array.

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

// find maximum from the counter array
int find_max(int *c,int n)
{
int i,j,max=0,index=-1;

for(i=0;i<n;i++) {="" if(max="" <="" c[i])="" {="" index="i;" max="c[i];" }="" }="" re
int="" n,i,j,k,*a,*c,max="-9999,index;" printf("\nenter="" the="" number="" of="
a="(int" *)malloc(sizeof(int)*n);="" printf("\nenter="" the="" integers\n");="" for(i=
scanf("%d",&amp;a[i])="" if(max="" <="" a[i])="" max="a[i];" }="" c="(int" *)mall
for(i="0;i&lt;=max;i++)" c[i]="0;" for(i="0;i&lt;n;i++)" c[a[i]]++;="" for(k="0;k&lt;n;
index="find_max(c,max+1);" }="" if(index="=-1)" break;="" for(j="0;j&lt;c[index];
}="" c[index]="0;" }="" }="">
```

1 ^ | v • Reply • Share ›

naive • a year ago

We can use hashing if range of numbers is given.

1 ^ | v • Reply • Share ›

geminisdb • a year ago

Why can't we just use QuickSort or MergeSort, with the only change being in t
We can build a HashMap which keeps counts for each unique array element.
Instead of comparing a[i] with a[j],
we compare Count_HashMap.getValue(a[i]) with Count_HashMap.getValue(a[

What is the advantage of using a BST over Quicksort or Mergesort?

^ | v • Reply • Share ›

RAMBOAMIT • a year ago

```
#include <cstdio>
#include<iostream>
#include<cstdlib>
#include <limits.h>
#include<queue>
#include<map>
#include<vector>
#include<set>
#include<algorithm>
#include<cmath>
#include<string>
#include<cstring>
using namespace std;
//bool myfunction (int i,int j) { return (i<j); }="" bool="" comp(pair<int,="" int="">
if(a.first%100 == b.first%100)
{
return a.first > b.first;
}
```

see more

atul • a year ago

Given method can be modified , so that it can work for cases where element h

add other parameter(int index) in bstnode.

```
struct BSTNode
{
struct BSTNode *left;
int data;
int freq;
int index;
struct BSTNode *right;
};
```

add another parameter in dataFreq

```
struct dataFreq
{
int data;
int index;
int freq;
```

see more

Reply • Share ›

Nileshwar Shukla • a year ago

I think it is counting sort! you can search for it in wikepedia.

Reply • Share ›

saimadhu.cse@gmail.com • a year ago

here is the simple solution in c using arrays

```
// Problem: Sorting the number according to the frequency
```

```
// Coded by saimadhu.polamuri on 10may2013
#include<stdio.h>
main(){
int number,i,j;
int frequence_count=0;
printf("Enter how many number you want: ");
scanf("%d",&number) // Number of number to be sort for freqency order
int number_array[number],reference_array[number],frequency_array[number]
for(i =0;i<number;i++){ printf("enter="" the="" number:="" ");="" scanf("%d",&a
for(i="0;i&lt;number;i++){" reference_array[i]="number_array[i];" }="" for="" cal
for(i="0;i&lt;number;i++){" for(j="0;j&lt;number;j++){" if(number_array[i]="refe
frequence_count++;="" }="" }="" frequency_array[i]="frequence_count;" freque
for(i="0;i&lt;number;i++){" printf("\n="" %d="" frequency="" is="" %d",number_
frequency="" sorting="" int="" frequency_referency_array[number];="" for(i="0;
frequency_referency_array[i]="frequency_array[i];" }="" int="" temp;="" for(i="0
```

see more

Reply • Share ›

gr81 • a year ago

```
#include <iostream>
#include <stdlib.h>

using namespace std;

void swap(int &a, int &b)
{
        int tmp = a;
        a = b;
        b = tmp;
```

```
#ifdef DEBUG
void print(int *a, int num)
{
        for(int i = 0; i < num; ++i)
                cout << a[i] <<" ";
```

see more

Reply • Share ›

Scott Shipp • a year ago

One possible solution in Java.

Nicely formatted version at http://pastebin.com/1t4EpDCK.

Visit my site, http://code.scottshipp.com.

```
import java.util.LinkedHashMap;

/*
 * Given an array of integers, sort the array according to frequency (
 * For example, if the input array is {2, 3, 2, 4, 5, 12, 2, 3, 3, 3,
 * modify the array to {3, 3, 3, 3, 2, 2, 2, 12, 12, 4, 5}.
 */
public class frequencySort1 {

        public static void main(String[] args) {
                int[] unsortedArray = { 2,12,3,2,4,2,3,3,3,12,5 };
```

see more

Reply • Share ›

just for fun • a year ago

```
package test.test;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map.Entry;

public class ArrayByFrequecny {

public static void main(String[] args) {
int[] array={2, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12};
HashMap store = new HashMap();

for (int i = 0; i < array.length; i++) {

if(store.containsKey(array[i])){
int temp= store.get(array[i]);
temp++;
store.put(array[i],temp);
}else{
```

see more

• Reply • Share ›

aghtyui • a year ago

Arrays
Bit Magic
C/C++ Puzzles
Articles
GFacts
Linked Lists
MCQ

Misc
Output
Strings
Trees

Sort elements by frequency | Set 2
May 5, 2013

Given an array of integers, sort the array according to frequency of elements. I
2, 4, 5, 12, 2, 3, 3, 3, 12}, then modify the array to {3, 3, 3, 3, 2, 2, 2, 12, 12, 4,
In the previous post, we have discussed all methods for sorting according to fi
discussed in detail and C++ implementation for the same is provided

see more

• Reply • Share ›

Ahmed I. Khalil • a year ago

Here is the same algorithm in C++ without the need to have inorder tree traver

```
#include <iostream>
#include <vector>
#include <algorithm>

struct Node {
    int data;
    Node *left;
    Node *right;
    int freq;
    Node(int data) {
        freq = 1;
        this->data = data;
        left = right = NULL;
```

```
};
```

see more

Reply • Share ›

Dheeraj Sharma • a year ago

Like it...Good Concept..

Reply • Share ›

Manish Ranjan • a year ago

welldone bro....

Reply • Share ›

Subhajit • a year ago

@GeeksForGeeks
This code is not having a complexity of O(nlog(n)) as this is not a balances BS
complexity for this algorithm is O(n^2). AVL insertion logic has to to applied to

Reply • Share ›

cpbcrec ➦ Subhajit • a year ago

Overall time complexity of the algorithm can be minimum O(nLogn) if v
and use a self balancing BST with O(Logn) insert operation.
It has been mentioned.Please read carefully.

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

Reply • Share ›

Sujeet Jaiswal • a year ago

well bro..good to see your article here...try to optimize it..if possible!