Count the number of occurrences in a sorted array

Given a sorted array arr[] and a number x, write a function that counts the occurrences of x in arr[]. Expected time complexity is O(Logn)

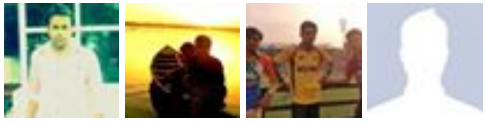
Examples:

```
Input: arr[] = {1, 1, 2, 2, 2, 2, 3,},   x = 2
Output: 4 // x (or 2) occurs 4 times in arr[]

Input: arr[] = {1, 1, 2, 2, 2, 2, 3,},   x = 3
Output: 1

Input: arr[] = {1, 1, 2, 2, 2, 2, 3,},   x = 1
Output: 2

Input: arr[] = {1, 1, 2, 2, 2, 2, 3,},   x = 4
Output: -1 // 4 doesn't occur in arr[]
```

Method 1 (Linear Search)
Linearly search for x, count the occurrences of x and return the count.

Time Complexity: O(n)

Method 2 (Use Binary Search)
1) Use Binary search to get index of the first occurrence of x in arr[]. Let the index of the first occurrence be i.
2) Use Binary search to get index of the last occurrence of x in arr[]. Let the index of the last occurrence be j.



3) Return (j – i + 1);

```
/* if x is present in arr[] then returns the count of occurrences of x
   otherwise returns -1. */
int count(int arr[], int x, int n)
{
  int i; // index of first occurrence of x in arr[0..n-1]
  int j; // index of last occurrence of x in arr[0..n-1]

  /* get the index of first occurrence of x */
  i = first(arr, 0, n-1, x, n);

  /* If x doesn't exist in arr[] then return -1 */
  if(i == -1)
    return i;

  /* Else get the index of last occurrence of x. Note that we
      are only looking in the subarray after first occurrence */
  j = last(arr, i, n-1, x, n);

  /* return count */
  return j-i+1;
}

/* if x is present in arr[] then returns the index of FIRST occurrence
   of x in arr[0..n-1], otherwise returns -1 */
int first(int arr[], int low, int high, int x, int n)
{
  if(high >= low)
  {
    int mid = (low + high)/2;  /*low + (high - low)/2;*/
    if( ( mid == 0 || x > arr[mid-1]) && arr[mid] == x)
      return mid;
    else if(x > arr[mid])
      return first(arr, (mid + 1), high, x, n);
    else
      return first(arr, low, (mid -1), x, n);
  }
  return -1;
}


/* if x is present in arr[] then returns the index of LAST occurrence
   of x in arr[0..n-1], otherwise returns -1 */
int last(int arr[], int low, int high, int x, int n)
{
  if(high >= low)
```


Popular Posts

```
  {
    int mid = (low + high)/2;  /*low + (high - low)/2;*/
    if( ( mid == n-1 || x < arr[mid+1]) && arr[mid] == x )
      return mid;
    else if(x < arr[mid])
      return last(arr, low, (mid -1), x, n);
    else
      return last(arr, (mid + 1), high, x, n);
  }
  return -1;
}

/* driver program to test above functions */
int main()
{
  int arr[] = {1, 2, 2, 3, 3, 3, 3};
  int x =  3;  // Element to be counted in arr[]
  int n = sizeof(arr)/sizeof(arr[0]);
  int c = count(arr, x, n);
  printf(" %d occurs %d times ", x, c);
  getchar();
  return 0;
}
```

Time Complexity: O(Logn)
Programming Paradigm: Divide & Conquer

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

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.

33 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...

Bhuvan • 3 months ago

```
Find_Key(int arr[], int size, int key){

int begin = 0;

int end = size - 1;

int mid = end / 2;

int res = INT_MIN;

while (begin != mid)

{

if (arr[mid] < key)

begin = mid;

else

{
```

see more

Reply • Share ›

Sat • 4 months ago

mid = (low + high)/2 will cause overflow if low and high are high. Java binary s written this way -> mid = low + (high - low)/2; But this change alone isn't suffic way, the loop invariant low <= high should be changed to low < high. Once's th other changes when low < high. Overall this post has many hidden issues.

1 Reply • Share ›

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

prastut • 4 months ago

```
#include<stdio.h>
int count(int a[],int start,int end,int num)
{
int mid=(start+end)/2,lcount=0,rcount=0;
while(start<=end)
{
lcount=count(a,start,mid-1,num);
rcount=count(a,mid+1,end,num);

if(a[mid]==num)
{
if(lcount==-1)
lcount=0;
if(rcount==-1)
rcount=0;
return 1+lcount+rcount;
}
else
```

see more

Reply • Share ›

din • 8 months ago

code in java validate me...

```
int count(){
        int c = 0;
        int[] a = {1, 1, 2, 2, 2, 2, 3};
        int  l = 0;
        int key = 1;
        int r = a.length-1;
```

```
while(l<=r){
    int mid = (l+r)/2;
    if(a[mid]==key){
        c++;
        int n=1;
        int k =-1;
            while(mid+n<=a.length-1 &&a[mid+n]==key){
                n++;
                c++;
            }
```

see more

• Reply • Share ›

vinay • 8 months ago

#include<conio.h>

#include<stdio.h>

void main()

{

int arr[]={1,1,2,2,2,2,3};

int num,i,count=0;

clrscr();

printf("no please");

scanf("%d",&num) ;

for(i=0;i<7;i++)

{

see more

1 ^ | v • Reply • Share ›

Abhishek • 8 months ago

In above solution, there is some duplicacy in the logic. We can modify the bina
number of count of the specific element in one go. Working code is as follows

```
//this function returns the number of occurrences of element x in array a.
//the main function call will be - binary(0,a.length-1,a,x);
public static int binary(int low, int high, int[] a, int x)
{
int mid = -1;
if ( high < low)
return 0;

// this is the case when the whole sub array is filled with element x
if ( a[low] == x && a[high] ==x )
return high-low+1;

mid = (high+low)/2 ;
int count = 0 ;

if ( a[mid] == x)
```

see more

^ | v • Reply • Share ›

Ankit Malhotra • a year ago

Recursion leads to stack calls. Here is code across 2 functions with each func
2logn. Also returns reference position of first occurrence or position to be inse

```
#include <iostream>
```

```
#define MAXCOUNT 99999
using namespace::std;
typedef unsigned counter;
typedef long element;

// first insert position in sorted order
// returns false with left = count for item > last
// Check boundaries before use
bool firstsortloc (element terms[], counter n, element item, counter &
{
  counter right = n, mid;
  left = 0;
  while (left != right && terms[left] != item) {
    mid = left + (right - left)/2;
```

see more

• Reply • Share ›

Ankit Malhotra Ankit Malhotra • a year ago

Optimize firstsortloc by adding a left increment as the first statement to

```
while (left != right && terms[left] != item) {
 ++left; // We know it's not item
```

• Reply • Share ›

Ankit Malhotra Ankit Malhotra • a year ago

Ignore the update for ++left. Original code is correct.

• Reply • Share ›

Ankit Malhotra Ankit Malhotra • a year ago

Ignore this optimization as it is a bug. The original code is corre

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

rahul sundar • a year ago

The fourth argument n is never used in function first and last. We can get rid c

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

Reply • Share ›

bunty • 2 years ago

Optimization: Method2:

Function last() can be optimized by removing the recursive call with paramete

-As the original call of last() was with low=i, which is the first occurrence of X,
- In last(), if X does not lie between i and new mid, then it is 100% sure that X v
high (which is higher than first range)
So just recursively call last(a, lo, mid-1, x) will either find the last occurrence o

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

Reply • Share ›

bunty ➚ bunty • 2 years ago

In other words,
- As we have found the first index of X, which is i,
- then the last index will be grater than or equal to i without any gap/hol
solve the purpose.
- If in some case, where left part doesn't have X, then we have already

Anuj Bansal • 2 years ago

Here is a code that also works in O(lgN).

```
 #include<stdio.h>
#include<math.h>
#define MAX 12

int number(int a[MAX], int low, int high, int x) {
        if(low <= high) {
                int mid,i,j;
                mid = (low+high)/2;
                if(a[mid] == x) {
                        i = number(a,low,mid-1,x);
                        j = number(a,mid+1,high,x);
                        if(i != -1 && j != -1)
                                return i+j+1;
                        else if(i == -1 && j != -1)
                                return j+1;
                        else if(j== -1 && i != -1)
```

see more

Ankur • 3 years ago

If mid==0 or mid==n-1 then the function returns mid

Now if the element is not present then the result will be wrong

Here i think we should put a check before return mid statement that
if(mid==0){

if(x!=a[mid]) return -1
}
similarly for Last func

1 ^ | v • Reply • Share ›

LoneShadow • 3 years ago

The function first doesnt seem to work for the following input. For example, arr first occurrence of 1.

Start with low=0, high= 4, x=1.
1. Mid = 2, and since arr[mid-1] is not less than x(arr[mid-1] is arr[1], which is again with parameters 0, 1

2. With 0,1 as low and high mid will be (0+1)/2, which is 0. The first conditiona not true, a[mid] is a[0], which is NOT equal to 1). So the function will be called

3. This will eventually return -1, which is wrong.

Also, not really sure why n is being passed into first/last. Not being used as far

^ | v • Reply • Share ›

LoneShadow ➚ LoneShadow • 3 years ago

One fix I can think off the top of my head is to call the function again wi you want to include the value of mid itself (its equal, not less or more to

^ | v • Reply • Share ›

LoneShadow ➚ LoneShadow • 3 years ago

Sorry, my bad. Worked it out.

^ | v • Reply • Share ›

skulldude • 3 years ago

Well, can someone tell me why in the second method is the condition (m==0) element is the required element?

```
if( ( mid == 0 || x > arr[mid-1]) && arr[mid] == x)
```

Reply • Share ›

hari6988 skulldude • 3 years ago

if mid was 0 , arr[mid-1] will reference arr[-1] and it will cause the error 1||x<arr[mid+1]

Reply • Share ›

Jatin Sanghvi • 3 years ago

shouldn't work. take arr = [1,3] and search for 2.

Reply • Share ›

GeeksforGeeks Jatin Sanghvi • 3 years ago

@Jatin Sanghvi: The given code works fine for your example. It returns present.

Reply • Share ›

foobar • 3 years ago

```
public static void CountNumberOfOccurences()
{

    /*
     * Count the number of occurrences in a sorted array
     * Given a sorted array arr[] and a number x, write a func
     * */
    int[] arr = {1, 1, 2, 2, 2, 2, 3,};
    int n = 2;

    int res = Array.BinarySearch(arr, 2);

    int found = res;
```

```
        int count = 0;

        if(res > 0)
        {
```

see more

Reply • Share ›

saji • 3 years ago

while invoking the method last , will using the args

last(arr, i, n-1, x, n) instead of last(arr, 0, n-1, x, n) cause any issues?

Reply • Share ›

GeeksforGeeks ➦ saji • 3 years ago

@saji: Thanks for suggesting the optimization. It doesn't cause any prc the suggested changes.

Reply • Share ›

nandini • 3 years ago

Isn't the complexity of the algorithm in **Method2 O(n)** in the worst case?

eg: {2,2,2,2,2}
program will start from mid-index and then will shift one-by-one towards left to find 'last' in each loop.

Reply • Share ›

thechamp ➦ nandini • 3 years ago

you have chosen an example where number of elements are so less tl is a binary search only

Reply • Share ›

Sandeep ➚ nandini • 3 years ago

@nandini: Please take a closer look at the program, it recursively divid
complexity is: T(n) = T(n/2) + C which is O(Logn)

^ | v • Reply • Share ›

WgpShashank • 3 years ago

Hi Sandeep It Can be done more simply using

Modified Binary Search Algorithm

calculate mid=low+high/2;
if(a[mid]>num)
search in right side & set low=mid+1 & return ;
else if(a[mid]
search in left side of mid set high=mid-1 & return ;
else //its important instead of just of printing the num or incrementing the coun
will be O(n) not O(logn) , si i will add 1 to recursively call for left side + recursiv
line executes we are incrementing the counter &
return 1+left_binsearch()+right_binsearch thus it will be in O(logn)

Here is the Code

```
 #include<stdio.h>
int Count(int a[], int value, int low, int high)
{
```

see more

^ | v • Reply • Share ›

Sandeep ➚ WgpShashank • 3 years ago

@WgpShashank: Thanks for suggesting a new method. The approach
worst case is O(n). Consider the case when all elements are same.

```
int a[] = {3, 3, 3, 3, 3, 3, 3};
int value = 3;
```

In this case, the time complexity can be written as

T(n) = 2*T(n/2) + C

which is O(n)

saurabh ➦ Sandeep • 2 years ago

i think Time compl. will be O(logn)

```
            T(n)
          /      \
      T(n/2)       T(n/2)
     /    \        /      \
  T(n/4) T(n/4)  T(n/4)  T(n/4)
.................................
```

Worst case will be longest height of above recursion tree....

let k will be max height.....and also we see terms at each level f
and through calculas stuff we know this will be a convergence s
is n(n/2+n/2)....n(1/2)^k<=1
k>=log2(n)..nearly....hence time compl will b O(logn)

WgpShashank ➦ Sandeep • 3 years ago

@sandeep Thanks For Pointing out ..yes it will O(n) in this scer

Yogesh ➦ WgpShashank • 3 years ago

i think that is the only case which is making its time com

that can be covered by adding line

```
if (a[high] == value && a[low] == value) return
```

at the beginning of the count function

Correct me if i am wrong somewhere.