GeeksQuiz

Computer science mock tests for geeks

Count 1’s in a sorted binary array

Given a binary array sorted in non-increasing order, count the number of 1’s in it.

Examples:

```
Input: arr[] = {1, 1, 0, 0, 0, 0, 0}
Output: 2

Input: arr[] = {1, 1, 1, 1, 1, 1, 1}
Output: 7

Input: arr[] = {0, 0, 0, 0, 0, 0, 0}
Output: 0
```

A simple solution is to linearly traverse the array. The time complexity of the simple solution is O(n). We can use Binary Search to find count in O(Logn) time. The idea is to look for last occurrence of 1 using Binary Search. Once we find the index last occurrence, we return index + 1 as count.

The following is C++ implementation of above idea.

```
// C++ program to count one's in a boolean array
#include <iostream>
using namespace std;

/* Returns counts of 1's in arr[low..high].  The array is
   assumed to be sorted in non-increasing order */
int countOnes(bool arr[], int low, int high)
{
  if (high >= low)
  {
    // get the middle index
    int mid = low + (high - low)/2;

    // check if the element at middle index is last 1
    if ( (mid == high || arr[mid+1] == 0) && (arr[mid] == 1))
      return mid+1;
    // If element is not last 1, recur for right side
    if (arr[mid] == 1)
      return countOnes(arr, (mid + 1), high);

    // else recur for left side
    return countOnes(arr, low, (mid -1));
  }
  return 0;
}

/* Driver program to test above functions */
int main()
{
   bool arr[] = {1, 1, 1, 1, 0, 0, 0};
   int n = sizeof(arr)/sizeof(arr[0]);
   cout << "Count of 1's in given array is " << countOnes(arr, 0, n-1);
   return 0;
}
```

Output:

```
Count of 1's in given array is 4
```

Time complexity of the above solution is O(Logn)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Category: Algorithms

Tweet 2 g+1 1

9 Comments GeeksQuiz 1 Login

Recommend 1 Share Sort by Best

Join the discussion…

coder • 3 months ago

An iterative version -
public static int countOnes(int[] a) {

```
int l = 0;

int r = a.length - 1;

int mid = 0;

int ctr = 0;

while(l <= r) {

mid = (l+r)/2;

if(a[mid] == 1){

ctr += (r - mid)+1;

r = mid - 1;
```

see more

2 • Reply • Share ›

Hengameh • 17 days ago

in the problem it is said: "non-increasing order", and in the code it is "non-decreasing order ".

1 • Reply • Share ›

GeeksforGeeks Mod Hengameh • 4 days ago

Thanks for pointing this out. We have updated the comment in code.

1 • Reply • Share ›

Guest • 17 days ago

"non-increasing order" means "decreasing order". So, the first example is wrong, it is not in non-increasing order.

• Reply • Share ›

Hengameh Guest • 4 days ago

the problem is typo in the code

• Reply • Share ›

Hengameh Guest • 4 days ago

You mean this one? It is correct :)
Input: arr[] = {1, 1, 0, 0, 0, 0, 0}

• Reply • Share ›

RK • 3 months ago

We can also find the first occurrence of 0 and return the index.

• Reply • Share ›

thevagabond85 RK • a month ago

and for

Input: arr[] = {1, 1, 1, 1, 1, 1, 1}

• Reply • Share ›

Goku thevagabond85 • a month ago

actually we can do it by finding the last occurence since it is sorted in decreasing order.So by finding the last occurence we know that all the elements from index 0 to index found are 1.

• Reply • Share ›

Iconic One Theme | Powered by Wordpress