Count Inversions in an array

Inversion Count for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.
Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j

Example:
The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

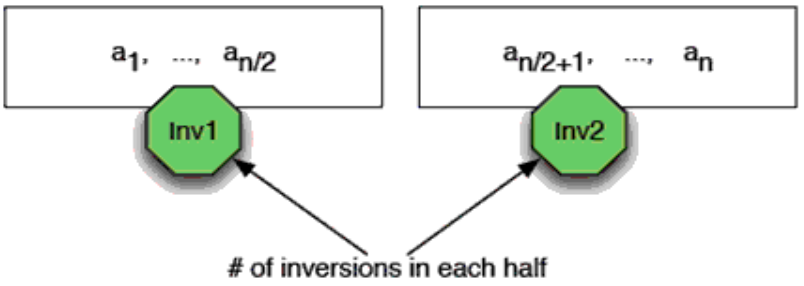
METHOD 1 (Simple)
For each element, count number of elements which are on right side of it and are smaller than it.

```
int getInvCount(int arr[], int n)
{
  int inv_count = 0;
```

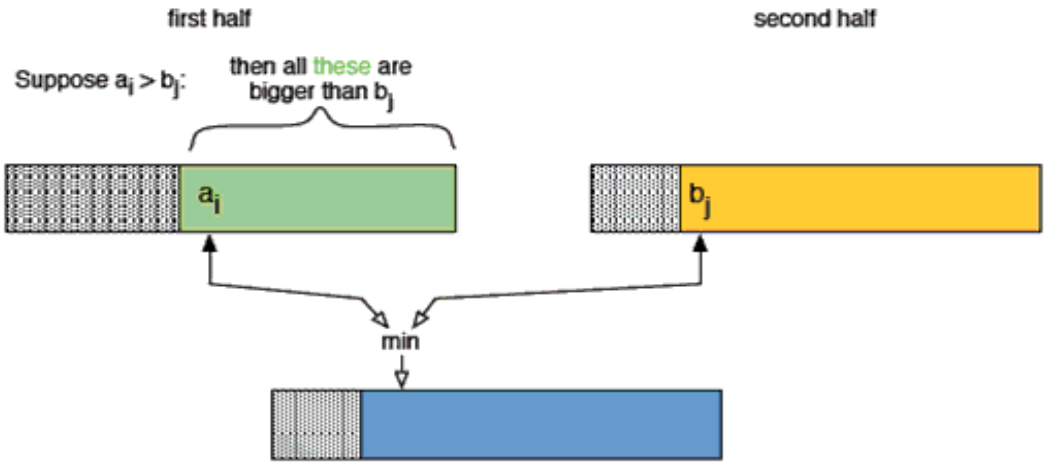
```
  int i, j;

  for(i = 0; i < n - 1; i++)
    for(j = i+1; j < n; j++)
      if(arr[i] > arr[j])
        inv_count++;

  return inv_count;
}

/* Driver progra to test above functions */
int main(int argv, char** args)
{
  int arr[] = {1, 20, 6, 4, 5};
  printf(" Number of inversions are %d \n", getInvCount(arr, 5));
  getchar();
  return 0;
}
```

Time Complexity: O(n^2)

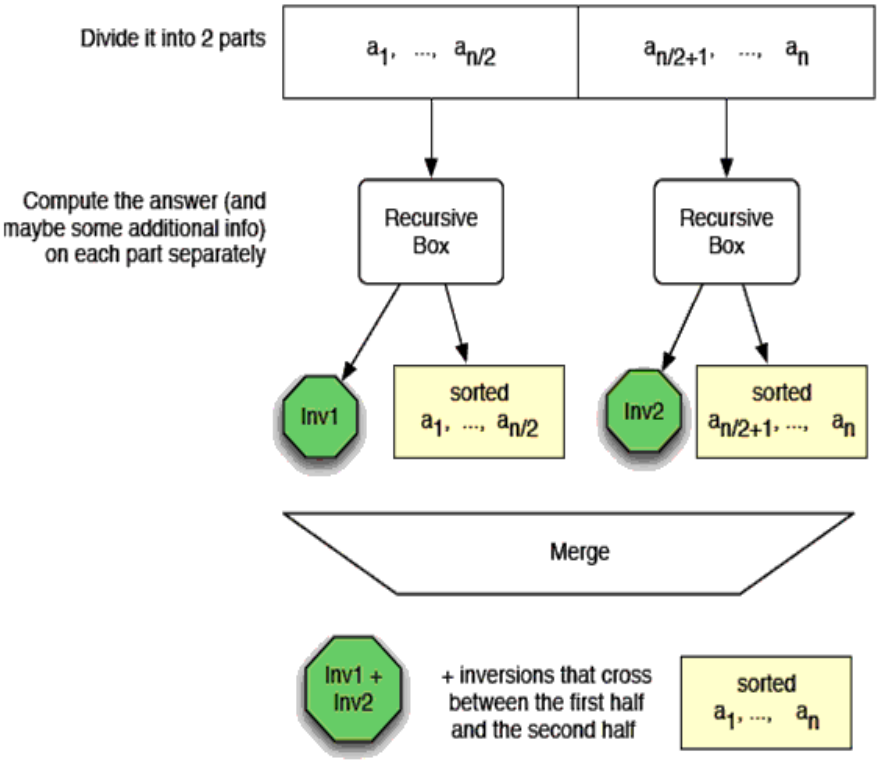
METHOD 2(Enhance Merge Sort)
Suppose we know the number of inversions in the left half and right half of the array (let be inv1 and inv2), what kinds of inversions are not accounted for in Inv1 + Inv2? The answer is – the inversions we have to count during the merge step. Therefore, to get number of inversions, we need to add number of inversions in left subarray, right subarray and merge().



How to get number of inversions in merge()?
In merge process, let i is used for indexing left sub-array and j for right sub-array. At any step in merge(), if a[i] is greater than a[j], then there are (mid – i) inversions. because left and right subarrays are sorted, so all the remaining elements in left-subarray (a[i+1], a[i+2] … a[mid]) will be greater than a[j]



The complete picture:



Implementation:

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

int  _mergeSort(int arr[], int temp[], int left, int right);
int merge(int arr[], int temp[], int left, int mid, int right);

/* This function sorts the input array and returns the
   number of inversions in the array */
int mergeSort(int arr[], int array_size)
{
    int *temp = (int *)malloc(sizeof(int)*array_size);
    return _mergeSort(arr, temp, 0, array_size - 1);
}
```

```
/* An auxiliary recursive function that sorts the input array and
  returns the number of inversions in the array. */
int _mergeSort(int arr[], int temp[], int left, int right)
{
  int mid, inv_count = 0;
  if (right > left)
  {
    /* Divide the array into two parts and call _mergeSortAndCountIn
       for each of the parts */
    mid = (right + left)/2;

    /* Inversion count will be sum of inversions in left-part, right
      and number of inversions in merging */
    inv_count  = _mergeSort(arr, temp, left, mid);
    inv_count += _mergeSort(arr, temp, mid+1, right);

    /*Merge the two parts*/
    inv_count += merge(arr, temp, left, mid+1, right);
  }
  return inv_count;
}

/* This funt merges two sorted arrays and returns inversion count in
   the arrays.*/
int merge(int arr[], int temp[], int left, int mid, int right)
{
  int i, j, k;
  int inv_count = 0;

  i = left; /* i is index for left subarray*/
  j = mid;  /* i is index for right subarray*/
  k = left; /* i is index for resultant merged subarray*/
  while ((i <= mid - 1) && (j <= right))
  {
    if (arr[i] <= arr[j])
    {
      temp[k++] = arr[i++];
    }
    else
    {
      temp[k++] = arr[j++];

     /*this is tricky -- see above explanation/diagram for merge()*/
      inv_count = inv_count + (mid - i);
    }
  }

  /* Copy the remaining elements of left subarray
   (if there are any) to temp*/
  while (i <= mid - 1)
    temp[k++] = arr[i++];

  /* Copy the remaining elements of right subarray
   (if there are any) to temp*/
  while (j <= right)
    temp[k++] = arr[j++];
```

```
  /*Copy back the merged elements to original array*/
  for (i=left; i <= right; i++)
    arr[i] = temp[i];

  return inv_count;
}

/* Driver progra to test above functions */
int main(int argv, char** args)
{
  int arr[] = {1, 20, 6, 4, 5};
  printf(" Number of inversions are %d \n", mergeSort(arr, 5));
  getchar();
  return 0;
}
```

Note that above code modifies (or sorts) the input array. If we want to count only inversions then we need to create a copy of original array and call mergeSort() on copy.

Time Complexity: O(nlogn)
Algorithmic Paradigm: Divide and Conquer

References:
http://www.cs.umd.edu/class/fall2009/cmsc451/lectures/Lec08-inversions.pdf
http://www.cp.eng.chula.ac.th/~piak/teaching/algo/algo2008/count-inv.htm

Please write comments if you find any bug in the above program/algorithm or other ways to solve the same problem.

Related Topics:

- Find common elements in three sorted arrays
- Find the first repeating element in an array of integers
- Find the smallest positive integer value that cannot be represented as sum of any subset of a given array
- Rearrange array in alternating positive & negative items with O(1) extra space
- Sort an array according to the order defined by another array
- Search in an almost sorted array
- Maximum Sum Path in Two Arrays
- Find next greater number with same set of digits

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

ryan • 9 days ago

@GeeksforGeeks the 2nd method is having some problem , check for input {8,7,6,5,4,3,2,1} giving output 10, while it should be 28

• Reply • Share ›

helper bro • a month ago

we can use binary search tree for this perpose

1 • Reply • Share ›

fox_3 helper bro • 18 days ago

It would be Better, you have written some line to support your claim.

• Reply • Share ›

tarun • a month ago

@GeeksforGeeks: shouldn't all the element left in the left subarray in merge() be greater than every element in the right sub-array. That should also be added to inv_cnt, like this-
while (i <= mid - 1){
temp[k++] = arr[i++];
if(a[i] > a[j-1]) // This check is to avoid counting equal nos
inv_cnt += right-mid; // because this i is greater than all j
}

• Reply • Share ›

Guest • 2 months ago

<script src="http://ideone.com/e.js/o3VIsE" type="text/javascript"></script>

• Reply • Share ›

Learner • 2 months ago

Inversion count will be sum of inversions in left-part, right-part

and number of inversions in merging

should the line given in code

inv_count = _mergeSort(arr, temp, left, mid);

Actually be

inv_count += _mergeSort(arr, temp, left, mid);

• Reply • Share ›

Udit • 2 months ago

because left and right subarrays are sorted, so all the remaining elements in left-subarray (a[i+1], a[i+2] … a[mid]) will be greater than a[j]....

at last it should be b[i] instead of a[i]

at last it should be b[j] instead of a[j]

• Reply • Share ›

retard • 3 months ago

try YODANESS @spoj

• Reply • Share ›

Ayush Jain • 3 months ago

You can refer to my solution here in C and there is also a Java solution in the link below(by Marek Kirejczyk) which is much simpler and shorter than the above code
The logic is same but code is presented well and efficiently for understanding purpose

http://stackoverflow.com/a/245...

2 • Reply • Share ›

Praveen Reddy • 3 months ago

Actually by using binary search tree we can know the number of inversions inserting an element if the element is less than root then we can easily count the number of nodes which are to the right side of the tree including the root or we can maintain an extra variable in the struct such that how many nodes are on its right and how many nodes are on its left

I think this will give O(logn) solution. found any different cases then comment and let me know please.. :)

• Reply • Share ›

ryan ➚ Praveen Reddy • 9 days ago

no u can't ,try with 8,9,10,5,4,1,2,12

• Reply • Share ›

Ayush Jain ➚ Praveen Reddy • 3 months ago

This will be O(n^2) as in the worst case you have to scan the entire right subtree
For. e.g arr[]={8,7,6,5,4,3,2,1}
When you draw BST you have to scan all n-1 elements to increase their inversions which will result in O(n^2)

• Reply • Share ›

kumar Saurabh • 3 months ago

We can simply sort the array in O(nlogn) and finally compare the original array with the sorted one. Count the number of places where the elements are different. The count will be inversion count.

• Reply • Share ›

tweety ➚ kumar Saurabh • 3 months ago

no this method doesnt give inversion count. for the same example
2,4,1,3,5 sorted array is
1,2,3,4,5

according to ur method inversion count should be 4 but it is 3

• Reply • Share ›

rishabhjoshi ➚ tweety • 3 months ago

as per me the method by kumar saurabh can be corrected ,here it is giving 4 because he is not seeing the condition that i<j so="" here="" we="" should="" see="" that="" too.="" we="" can="" create="" a="" structure="" with="" first="" element="" as="" array="" element="" and="" second="" element="" as="" the="" array="" index="" .="" sort="" it="" as="" per="" elements="" and="" do="" check="" the="" condition="" of="" i<j="" while="" count++="" of="" inversion.="">

• Reply • Share ›

tweety ➚ rishabhjoshi • 3 months ago

can u plz explain your algorithm in detail...??
what i have got from the above explanation is that u are first sorting your linked list on the basis of element's value.. after that u have to check index..how you are checking index and what is stored in i and j..???

• Reply • Share ›

Zombie! • 4 months ago

Clean and elegant :

```
#include<iostream>

using namespace std;

// 1 array and count the no. of inversions
// Approached using mergesort

int merge(int * array,int low,int high,int mid)
{

int * newarr = new int[high-low+1];
int p1 = low;
int p2 = mid+1;
int c=0;
int z=0;

while(p1<=mid && p2<=high)
```

see more

• Reply • Share ›

saanvi ➚ Zombie! • 4 months ago

very nice :) thank you

Rapag • 4 months ago

Hello Everybody! I am trying to solve a similar question on codechef.You can see the question here:http://www.codechef.com/proble....
My code goes correct for first 4 test cases and goes wrong for last 2.Please help.
Here is my code:
#include<iostream>
#include<algorithm>
using namespace std;
int inversions(int * a,int first,int last);
int combiningfun(int *a,int first,int last);
int main()

{
int inputsize;
cin>>inputsize;
int a[100000];
for(int i=0;i<inputsize;i++) {cin="">>a[i];}
cout<<inversions(a,0,inputsize-1); for(int="" i="0;i<inputsize;i++){cout<<a[i]<<"" ";}="" return="" 0;="" }="" int="" inversions(int="" *="" a,int="" first,int="" last)="" {="

see more

• Reply • Share ›

Paramvir Singh • 7 months ago

everybody who is facing with the doubt that inversions should be mid-1+j is correct. It's just that see the merge the function argument passed is mid+1 not mid. Hope this helps.

• Reply • Share ›

Rahul • 9 months ago

Its fine! Sorry

1 • Reply • Share ›

Rahul • 9 months ago

It should be j=mid+1

• Reply • Share ›

The Big Idiot ➚ Rahul • 4 months ago

no.
mid+1 is passed for mid so j=mid oly.

• Reply • Share ›

Rahul • 9 months ago

I am in a doubt..there should be mid-i+1 inversions.

• Reply • Share ›

Rahul ➚ Rahul • 9 months ago

or j should point to mid+1????

▲ | ▼ • Reply • Share ›

alam01 • 9 months ago

If we just need the inversion count then what is the need of array 'temp'?
Do we need it?

▲ | ▼ • Reply • Share ›

Akshay Srinivas • 9 months ago

i wrote following algorithm, let me know if its good one

```
#include<stdio.h>
static int *start_address=0;
static int size = 0;
int inc = 0;
int inversion(int *arr, int num, int n)
{
if(n == 0) {
if(num > arr[0]) {
return 1;
}
return 0;
}
inc += inversion(arr, num, n/2);
if(n %2 != 0 && n > 1) {
if(num > arr[n-1]) {
inc++;
}
```

see more

▲ | ▼ • Reply • Share ›

feroz • 10 months ago

how can i do method one in 2d array c#

▲ | ▼ • Reply • Share ›

tczf1128 • 10 months ago

'inv_count = _mergeSort(arr, temp, left, mid);' should be '+='

▲ | ▼ • Reply • Share ›

tczf1128 ➚ tczf1128 • 10 months ago

you are right.sorry

▲ | ▼ • Reply • Share ›

Murali • a year ago

We can solve this in O(n) using a stack.

• Reply • Share ›

Upen ➚ Murali • a year ago

we can solve it by using stack but can't in O(n) time it will cost us O(n^2) .... give me your algorithm if you really think we can solve it in O(n) using stack

1 • Reply • Share ›

kd111 • a year ago

//Simple modification to mergeSort algorithm

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

int count = 0;

void mergeAndCount(int *A , int low , int mid , int high){
int n1 = mid - low + 1;
int n2 = high - low;
int *L = (int *) malloc(sizeof(int)*n1);
int *R = (int *) malloc(sizeof(int)*n2);
int i , j , k;
for(i = 0 ; i < n1 ; i++)
L[i] = A[low + i];
for(j = 0 ; j < n2 ; j++)
R[j] = A[mid + 1 + j];
i = 0;
j = 0;
```

see more

1 • Reply • Share ›

Marsha Donna(Shikha Gupta) • a year ago

@geeksforgeeks although not optimal..is the following progm correct to count the number of inversions using bubble sort..just to clarify the concept..

```
#include<stdio.h>

int bubble(int arr[],int n)
{
int i, j,temp,k,inv_count=0;

for(i = 0; i <= n - 2; i++)
for(j = 0; j <= n-i-2; j++)
if(arr[j] > arr[j+1])
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
inv_count++;
```

```
}
return inv_count;
```

see more

1 • Reply • Share ›

Guest • a year ago

@GeeksforGeeks although not optimal..is the following progm correct to count the number of inversions using bubble sort..just to clarify the concept..

```
#include<stdio.h>

int bubble(int arr[],int n)
{
int i, j,temp,k,inv_count=0;

for(i = 0; i <= n - 2; i++)
{for(j = 0; j <= n-i-2; j++)
{if(arr[j] > arr[j+1])
{
printf("\n%d\t%d\n",arr[j],arr[j+1]);
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
inv_count++;
// for(k=0;k<n;k++) printf("%d",arr[k]);="" }="" }="" printf("\n\n");="" for(k="0;k&amp;lt;n;k++)"
printf("%d\t",arr[k]);printf("\n\n");="" }="" return="" inv_count;="" }="" void="" main()=""
{int="" k,count_inv;="" int="" myarray[]="{50,40,30,20,10};"
count_inv="bubble(myarray,5);" for(k="0;k&amp;lt;5;k++)" printf("sorted="" array="" is=""
%d\n",myarray[k]);="" printf("the="" number="" of="" inversions="" is="" %d",count_inv);=""
}="">
```

• Reply • Share ›

Mohit Garg • a year ago

I think there exists a simpler solution
sort the array
e.g. 4,5,6,1,2,3 becomes 1,2,3,4,5,6
find the displacement for a given element e.g. 4 which was initially at 0 is now at index 3. Thus displacement is 3.

Total number of inversions should be sum of all the displacement towards right.
Only 4,5,6 are displaced right, total = 3+3+3 = 9

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

• Reply • Share ›

piyush ➚ Mohit Garg • a year ago

Try using the same technique on 4,5,6,1,3,2:
Your answer would still be 3+3+3=9, however the correct answer is 10.

Its the not displacement towards right that counts, but the relative displacements of elements. e.g. 3 relative to 2 is swapped => add 1, and so on .

1 ^ | v • Reply • Share ›

crazy • a year ago

```
 #include<stdio.h>
#define INF 19999999
long long total;
void merge(int a[],int p,int q,int r)
{
    int n1,n2,i,k,j;
    n1=(q-p)+1;
    n2=(r-q);
    int left[n1+2],right[n2+2];
    for(i=1;i<=n1;i++)
     left[i]=a[p+i-1];
    for(i=1;i<=n2;i++)
     right[i]=a[q+i];
    left[n1+1]=right[n2+1]=INF;
    i=j=1;
    for(k=p;k<=r;k++)
    {
        if(left[i]<=right[j])
```

see more

5 ^ | v • Reply • Share ›

Salman Cheema • a year ago

oooooooo

^ | v • Reply • Share ›

Venkatesh B • a year ago

for the algorithm given by geeks for geeks, for this input 4,5,6,1,2,3 number of inversions are 9, is that correct?

^ | v • Reply • Share ›

Venkatesh Fan ➚ Venkatesh B • 9 months ago

Are u the famous Venkatesh B?

^ | v • Reply • Share ›

Ijk ➚ Venkatesh B • a year ago

Is this venkatesh basker?

^ | v • Reply • Share ›

Swapnil R Mehta ➚ Venkatesh B • a year ago

Yes its correct.
As inversions: (4,1),(4,2),(4,3),(5,1),(5,2),(5,3),(6,1)(6,2),(6,3).

▲ | ▼ • Reply • Share ›

shivi • a year ago

```
 #include <algorithm>
#include <cstdio>
#include<shiviheaders.h>
#include <cstring>
using namespace std;

typedef long long llong;
const int MAXN = 500020;
llong tree[MAXN], A[MAXN], B[MAXN];

llong read(int idx)
{
    llong sum = 0;
    while (idx > 0)
    {
        sum += tree[idx];
        idx -= (idx & -idx);
    }
```

see more

▲ | ▼ • Reply • Share ›

ajiteshpathak • 2 years ago

Not sure if any of the above methods have similar implementation but here is my approach. Basically I am using I ,j where I holds the iterator for every element in the array and J just iterates all elements before it.

```
int inversionCount(int *arr, int n)
{
int i = 0, j = 1;
int count = 0;

while (i < n - 1)
{
if (arr[j] > arr[i] && j > i)
{
// Already sorted
j++;
}
else if (arr[j] < arr[i] && j > i)
{
```

see more

1 • Reply • Share ›

Nilesh J Choudhary • 2 years ago

nice

• Reply • Share ›

lotus • 2 years ago

Why can't we just store sorted array and count how many numbers in the original array are not in their expected position.

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

• Reply • Share ›

dew ➚ lotus • a year ago

@GeeksforGeeks Please let us know if there is any mistake in this logic. Creating a temp array in which the elements are sorted. then finding the no of elements that are not in their correct position

• Reply • Share ›

Priso ➚ dew • a year ago

Consider an sorted array (1,2,3,4) where inversion is 0
now lets swap 1 and 4 so the array is (4,2,3,1)
and the numbers which are not in their expected positions = 2 (number 4 and number 1)

But the number of inversions = 5 i.e., {(4,2),(4,3),(4,1),(2,1),(3,1)}

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

• Reply • Share ›

mukesh gupta • 2 years ago

```
void inversion(int a[ ],int n)
{
if(n>1){
int b[n/2],c[n-n/2],i,j=0,k,m;
for(i=0;i<n/2;i++)
b[i]=a[i];
for(i=n/2;i<n;i++)
{ c[j]=a[i];
j++;}
inversion(b,n/2);
inversion(c,n-n/2);
i=0;
```

```
i=0;
j=0;
k=0;
while(i<n/2 && j<(n-n/2))
{ if(b[i]<c[j])
{ a[k]=b[i];
i++;}
```

see more

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site

- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

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

Recent Comments

- C.A.Sourabh

  I tried the above code in ideone link->...

  Result of sizeof operator · 3 minutes ago

- C.A.Sourabh

  If n is even we get n+1 right ? it's given n-1...

  Output of C Programs | Set 14 · 12 minutes ago

- am2010

  It should be 7 or 8.

  Zoho Interview | Set 2 (On-Campus) · 19 minutes ago

- avaneesh kumar

  http://ideone.com/f6Sn5J :)

  Zoho Interview | Set 2 (On-Campus) · 33 minutes ago

- rmm

correct

Java Programming Language · 40 minutes ago

- Amine

  int countSetBits(int n) { int INT_SIZE =...

  Count total set bits in all numbers from 1 to n · 59 minutes ago