Find the smallest missing number

Given a **sorted** array of n integers where each integer is in the range from 0 to m-1 and m > n. Find the smallest number that is missing from the array.

Examples
Input: {0, 1, 2, 6, 9}, n = 5, m = 10
Output: 3

Input: {4, 5, 10, 11}, n = 4, m = 12
Output: 0

Input: {0, 1, 2, 3}, n = 4, m = 5
Output: 4

Input: {0, 1, 2, 3, 4, 5, 6, 7, 10}, n = 9, m = 11
Output: 8

Thanks to Ravichandra for suggesting following two methods.

Method 1 (Use Binary Search)
For i = 0 to m-1, do binary search for i in the array. If i is not present in the array then return i.

Time Complexity: O(m log n)

Method 2 (Linear Search)
If arr[0] is not 0, return 0. Otherwise traverse the input array starting from index 1, and for each pair of elements a[i] and a[i+1], find the difference between them. if the difference is greater than 1 then a[i]+1 is the missing number.

Time Complexity: O(n)

Method 3 (Use Modified Binary Search)
Thanks to yasein and Jams for suggesting this method.
In the standard Binary Search process, the element to be searched is compared with the middle element and on the basis of comparison result, we decide whether to search is over or to go to left half or right half.
In this method, we modify the standard Binary Search algorithm to compare the middle element with its index and make decision on the basis of this comparison.

…1) If the first element is not same as its index then return first index
…2) Else get the middle index say mid
……a) If arr[mid] greater than mid then the required element lies in left half.
……b) Else the required element lies in right half.

```
#include<stdio.h>

int findFirstMissing(int array[], int start, int end) {

    if(start  > end)
      return end + 1;

    if (start != array[start])
      return start;

    int mid = (start + end) / 2;

    if (array[mid] > mid)
      return findFirstMissing(array, start, mid);
    else
      return findFirstMissing(array, mid + 1, end);
}

// driver program to test above function
int main()
{
  int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 10};
  int n = sizeof(arr)/sizeof(arr[0]);
  printf(" First missing element is %d",
           findFirstMissing(arr, 0, n-1));
  getchar();
  return 0;
}
```

Geometric Algorithms

Popular Posts

Note: This method doesn't work if there are duplicate elements in the array.

Time Complexity: O(Logn)

Source: http://geeksforgeeks.org/forum/topic/commvault-interview-question-for-software-engineerdeveloper-2-5-years-about-algorithms

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.

70 Comments GeeksforGeeks

Sort by Newest

Marsha Donna • 8 months ago

@jams can u pls explain how u gt the condition if(start > end)
return end + 1;

bani ➦ Marsha Donna • 3 months ago

u can understand this by taking array elements as 0,1,2,3,4.
apply the above algo on this input array..u ll eventually understand the (

Marsha Donna • 8 months ago

also in method 3 should'nt the condition be like if (array[mid] > mid)

return findFirstMissing(array, start, MID-1);

bani ➦ Marsha Donna • 3 months ago

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

both the cases would work ...
take input as 0,1,3,4,5....make use of both the conditions it will give cor
if(start > end)
return end + 1;
makes up for everything

1 • Reply • Share ›

Marsha Donna • 8 months ago

@GeeksforGeeks method 2 wont work for input {0,2,3,4}where m=5
the input array must be traversed from index 0 itself

• Reply • Share ›

Guest • 8 months ago

can u pls validate the follwing code
void smalmis(int arr[],int n,int m)
{
int i;
for(i=0;i<m;i++) {="" if(arr[i]!="i)break;" }="" printf("the="" smalest="" misin="" r

• Reply • Share ›

Guest • 8 months ago

@GeeksforGeeks can u pls validate the following code with linear complexity

```
void smalmis(int arr[],int n,int m)
{
    int i;
    for(i=0;i<m;i++) {="" if(arr[i]!="i)break;" }="" printf("the="" sr
```

1 • Reply • Share ›

Guest • 8 months ago

toperform linear search easir method would be

Reply • Share ›

nehamahajan • 8 months ago

Another way is to use hashmap:-

http://mahajanneha.blogspot.co...

Complexity: O(m+n)

Reply • Share ›

Avinash Abhi • 10 months ago

GeeksforGeeks yeah got it, i didnt see the note there at the end.

Reply • Share ›

GeeksforGeeks • 10 months ago

Please take a closer look at the question and the note at the bottom of the pos
are not valid. The first array is not sorted, second array has duplicates.

Reply • Share ›

Avinash Abhi • 10 months ago

I think method 3 is not correct.

Flaw 1: for condition arr[mid]=mid;.

suppose the array is {0,1,1,1,4,5,6,9,12}.

so after 1st iteration of binary search mid=(0+8)/2=4.

arr[mid]=mid=4 so the next subset to be searched will be (mid+1, end).
according to the solution, which is wrong.
the smallest missing number is in subset (0, mid-1) which is 2.

Flaw 2: for condition arr[mid]<mid;.

suppose array is arr={0,1,1,2,2,3,4,5,6}.

mid=(0+8)/2=4;
arr[mid]<mid;
so the next recursion will be.
findFirstMissing(array, mid + 1, end);.

see more

Reply • Share ›

abhishek08aug • a year ago

O(n) solution:

```
#include<stdio.h>

int findFirstMissing(int array[], int n, int m) {
  int i;
  for(i=0; i<m-1; i++) {
    if(i<=n-1) {
      if(array[i] != i) {
        return i;
      }
    }
    else {
      return i;
    }
  }
}
```

see more

1 • Reply • Share ›

Marsha Donna abhishek08aug • 8 months ago

@abhishek08aug according to exampl 3 given above u should not che
if(i<=n-1)....also the loop must run as for(i=0;i

Reply • Share ›

Guest abhishek08aug • 8 months ago

@abhishek08aug according to exampl 3 given above u should not che
as for(i=0;i<m;i++) and="" in="" else="" part="" it="" should="" not="" re
display="" that="" no="" number="" is="" missin="">

Reply • Share ›

will smith abhishek08aug • 11 months ago

intelligent :D

Reply • Share ›

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]

```
/**
* Given a sorted array of n integers where each integer is in the range from 0 t
* Find the smallest number that is missing from the array.
* Example: Input: {0, 1, 2, 6, 9}, n = 5, m = 10 Output: 3
* @author GAPIITD
*
*/
public class FindTheSmallestMissingNumber {

public static void main(String[] args) {
int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 10};
System.out.println(findSmallestMissingNo(arr));
```

}

private static int findSmallestMissingNo(int[] arr) {
int start = 0, end = arr.length - 1;

see more

• Reply • Share ›

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]
/**
* Given a sorted array of n integers where each integer is in the range from 0 t
* Find the smallest number that is missing from the array.
* Example: Input: {0, 1, 2, 6, 9}, n = 5, m = 10 Output: 3
* @author GAPIITD
*
*/
public class FindTheSmallestMissingNumber {

public static void main(String[] args) {
int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 10};
System.out.println(findSmallestMissingNo(arr));
}

private static int findSmallestMissingNo(int[] arr) {
int start = 0, end = arr.length - 1;

see more

• Reply • Share ›

Steve • 2 years ago

For better understanding in solution 3 we can make a small change in the func

```
int findFirstMissing(int array[], int start, int end)
{
int mid;
if(start>end)
return end+1;
if(start!=a[start])
return start;
mid=(start+end)/2;
if(a[mid]==mid) //If element in array equal to index then missing elemt is in righ
return bin(a,mid+1,end);
else
return bin(a,start,mid);
}
```


divya • 2 years ago

```
package Array;
//assume duplicate elements in the array are not present
public class Sift {
public static void main(String args[])
{
int arr[]={6,2,1,11,5,7};
int p1=0,p2=arr.length-1,temp;
while(p1<arr.length)
{
if(p1==p2)
break;
if(arr[p1]==p1+1)
{
p1++;
continue;
}
```

if(arr[p1]arr[p2])
{

see more

Reply • Share ›

algobard • 2 years ago

There's a flaw in the 2nd method.
It'll not work for inputs similar to:

2,3,4,5,6 (m=7, n=5)

Correct me if I'm wrong :)

Reply • Share ›

GeeksforGeeks algobard • 2 years ago

Thanks for pointing this out. We have added a line for cases like this.

Reply • Share ›

algobard • 2 years ago

Can't we do something like this:

Traverse the array->

If arr[index]!=index
return index;
else index++;

return sizeof(arr); // If no such element exists (we've reached the end of our ar

Again...this would be O(n)...but I just want to know if there's a flaw in this appr

Reply • Share ›

harsh jain • 2 years ago

I try to solve this problem in linear **time**

```
#include<stdio.h>


int main()
{
    int n , m;
    scanf("%d %d" , &n  , &m );

    int arr[n];
    int store[n];
    int i , k;
    for (  i = 0; i < n; i++ ) {
        scanf("%d" , &k );
        arr[i] = k;
        store[i] = k;
    }
```

see more

• Reply • Share ›

harsh jain • 2 years ago

```
#include<stdio.h>


int main()
{
    int n , m;
    scanf("%d %d" , &n  , &m );
```

```
    int arr[n];
    int store[n];
    int i , k;
    for (  i = 0; i < n; i++ ) {
        scanf("%d" , &k );
        arr[i] = k;
        store[i] = k;
    }
```

see more

• Reply • Share ›

sanki • 2 years ago

```
#include<stdio.h>
int main()
{
int i,n,m;
scanf("%d %d",&n,&m);
int a[n];
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<m;i++)
{
if(a[i]==i)
continue;
else
{
printf("%d\n",i);
break;
}
return 0;
```

```
}

}
```

Reply • Share ›

radhakrishna • 3 years ago

my solution : works irrespective of array is sorted or not

maintain bit array : 0 to m-1 bits. populate status of integer in that correspondi

status would be computed as follows
if number falls in range - 1
outside range - 0

after that scan array to find first zero.

Reply • Share ›

giri ➦ radhakrishna • 3 years ago

this solution also works and it is simplem, but the time complexity is O
Method 3 is Log(n).

Reply • Share ›

radhakrishna ➦ giri • 3 years ago

time complexity is O(n). but space complexity is not O(n) as ge
n* 2/4 bytes based on type of storage. but here we are maintair
terms of bits. i would say O(1).

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

Reply • Share ›

Abhijeet Sinha ➦ radhakrishna • 2 years ago

@radhakrishna - As far as my understanding goes, you
this.
int checker=0 (4 bytes or 32 bits)
and now as the you scan the array you will be doing che
character from left to 1.
My question is what if the range goes beyond 32 or ever

Please correct me if I am wrong and provide me with a

Abhijeet Sinha Abhijeet Sinha • 2 years ago

Thanks Vibhaj. I understood the concept :-)

tr4n2uil Abhijeet Sinha • 2 years ago

An int represents 32 bits (on 32 bit systems). So in case
an array of integers and use (base + offset) method to s

Say you implement 256 bitvector as follows :

```
 #define SIZE 8
#define MAXBIT 32

int bits[SIZE];

void set(int bits[], N){
    int base = N/SIZE;
    int offset = N % MAXBIT;
    bits[base] |= (1 << offset);
}

int check(int bits[], int N){
```

```
    int base = N/SIZE;
    int offset = N % MAXBIT;
    return bits[base] & (1 << offset);
}
```


sanki ➦ radhakrishna • 2 years ago

[sourcecode]
#include<stdio.h>
int main()
{
int i,n,m;
scanf("%d %d",&n,&m);
int a[n];
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<m;i++)
{
if(a[i]==i)
continue;
else
{
printf("%d\n",i);
break;
}
return 0;
}

}

Anant Upadhyay • 3 years ago

@R.Srinivasan - The no of comaprisons can be reduced by using a break sta

R.Srinivasan • 3 years ago

//The following 0(n) algo works for duplicates and m is unused.

```
int findFirstMissing(int arr[],int m,int n)
{
int i,missing=0;
for(i=0;i<n;i++)
if(arr[i]==missing)
missing++;
return missing;
}

int main()
{
int arr[] = {0, 0,0,2,2,2};
int n = sizeof(arr)/sizeof(arr[0]);
printf("First missing element is %d",findFirstMissing(arr,0, n));
getchar();
return 0;
}
```


asd • 3 years ago

Are the no.s distinct ?

ddfd ➦ asd • 3 years ago

They need not be distinct. But it they are distinct,then the modified bina

R.Srinivasan ddfd • 3 years ago

The modified binary search algo works for only distinct number distinct numbers as well as duplicates.

But the advantage of modified binary search algo is, it runs in 0 0(n) time.

Reply • Share ›

Jin • 3 years ago

This problem can be extended to finding the kth missing number. Binary searc

Reply • Share ›

Satyam • 3 years ago

I think it may work for repititive values also. Correct me if I am wrong.

```
int missingNumber(int a[],int n)
{
int start=0,mid,end=n-1,rep=0,ans;
while(start<=end)
{
mid=(start+end)/2;
if(a[mid]<(mid-rep)){rep=mid-a[mid];start=mid+1;}
else if(a[mid]==(mid-rep))start=mid+1;
else end=mid-1;
}
if(mid==0)ans=0;
else if(start==mid+1)ans=a[mid]+1;
else ans=a[mid-1]+1;
return ans;
}
```

Reply • Share ›

R.Srinivasan ➚ Satyam • 3 years ago

For the input {0,0,0,2,2,2}, Your program gives the first missing numbe "1".

^ | v • Reply • Share ›

Satyam ➚ R.Srinivasan • 3 years ago

the code gives the answer as 1 in this test case.
here n is the length of the array that we are passing to the funct

^ | v • Reply • Share ›

Satyam ➚ Satyam • 3 years ago

the modification that needs to be done is:-

if(a[mid]<(mid-rep)){rep=mid-a[mid]+1;start=mid+1;}

^ | v • Reply • Share ›

foobar • 3 years ago

I d use XOR. Iterate through the array and stop at the first test where XOR fails

^ | v • Reply • Share ›

radhe ➚ foobar • 3 years ago

What is the benefit of this? You would use extra space and yet do linea

Correct me if I am wrong. Thanks.

^ | v • Reply • Share ›

Gopal • 3 years ago

Not found accurate solution for O(n) implementation, so thought to post one.

```
public static int usingLinearSearch(int[] array){
        int index = 0;
        for(int i=0;i<array.length;i++){
```

```
                if(array[i] > index){
                        return index;
                }

                if(i == 0 || array[i-1] != array[i]){
                        index++;
                }
        }
        return array.length;
}
```

Reply • Share ›

Gopal • 3 years ago

Though Binary Search not work for duplicates, but elegant solution.

Reply • Share ›

Prakash • 3 years ago

The binary search algorithm won't work if duplicates are allowed.

Reply • Share ›

GeeksforGeeks • 3 years ago

@Jams: Thanks for suggesting a simpler implementation for method 3, we ha
implementation. Keep it up!!

@Hima: thanks for pointing this out, we have added not for the same.

Reply • Share ›

Hima • 3 years ago

The algo described in Method 3 doesn't seem to wrok if there are duplicate ele

Eg: 1,1,2,3,5,7,8