Find the smallest positive number missing from an unsorted array

You are given an unsorted array with both positive and negative elements. You have to find the smallest positive number missing from the array in O(n) time using constant extra space. You can modify the original array.

Examples

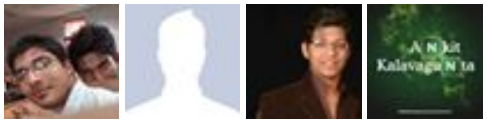
```
Input:  {2, 3, 7, 6, 8, -1, -10, 15}
Output: 1

Input:  { 2, 3, -7, 6, 8, 1, -10, 15 }
Output: 4

Input: {1, 1, 0, -1, -2}
Output: 2
```

Source: To find the smallest positive no missing from an unsorted array

A **naive method** to solve this problem is to search all positive integers, starting from 1 in the given array. We may have to search at most n+1 numbers in the given array. So this solution takes O(n^2) in worst case.

We can **use sorting** to solve it in lesser time complexity. We can sort the array in O(nLogn) time. Once the array is sorted, then all we need to do is a linear scan of the array. So this approach takes O(nLogn + n) time which is O(nLogn).

We can also **use hashing**. We can build a hash table of all positive elements in the given array.



Once the hash table is built. We can look in the hash table for all positive integers, starting from 1. As soon as we find a number which is not there in hash table, we return it. This approach may take O(n) time on average, but it requires O(n) extra space.

A O(n) time and O(1) extra space solution:
The idea is similar to this post. We use array elements as index. To mark presence of an element x, we change the value at the index x to negative. But this approach doesn't work if there are non-positive (-ve and 0) numbers. So we segregate positive from negative numbers as first step and then apply the approach.

Following is the two step algorithm.
1) Segregate positive numbers from others i.e., move all non-positive numbers to left side. In the following code, segregate() function does this part.
2) Now we can ignore non-positive elements and consider only the part of array which contains all positive elements. We traverse the array containing all positive numbers and to mark presence of an element x, we change the sign of value at index x to negative. We traverse the array again and print the first index which has positive value. In the following code, findMissingPositive() function does this part. Note that in findMissingPositive, we have subtracted 1 from the values as indexes start from 0 in C.

```
/* Program to find the smallest positive missing number */
#include <stdio.h>
#include <stdlib.h>

/* Utility to swap to integers */
void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a   = *b;
    *b   = temp;
}

/* Utility function that puts all non-positive (0 and negative) number
  side of arr[] and return count of such numbers */
int segregate (int arr[], int size)
{
    int j = 0, i;
    for(i = 0; i < size; i++)
    {
        if (arr[i] <= 0)
```


Popular Posts

```
        {
            swap(&arr[i], &arr[j]);
            j++;  // increment count of non-positive integers
        }
    }

    return j;
}

/* Find the smallest positive missing number in an array that contains
  all positive integers */
int findMissingPositive(int arr[], int size)
{
  int i;

  // Mark arr[i] as visited by making arr[arr[i] - 1] negative. Note t
  // 1 is subtracted because index start from 0 and positive numbers s
  for(i = 0; i < size; i++)
  {
    if(abs(arr[i]) - 1 < size && arr[abs(arr[i]) - 1] > 0)
      arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];
  }

  // Return the first index value at which is positive
  for(i = 0; i < size; i++)
    if (arr[i] > 0)
      return i+1;  // 1 is added becuase indexes start from 0

  return size+1;
}

/* Find the smallest positive missing number in an array that contains
  both positive and negative integers */
int findMissing(int arr[], int size)
{
   // First separate positive and negative numbers
   int shift = segregate (arr, size);

   // Shift the array and call findMissingPositive for
   // positive part
   return findMissingPositive(arr+shift, size-shift);
}

int main()
{
  int arr[] = {0, 10, 2, -10, -20};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
```

```
  int missing = findMissing(arr, arr_size);
  printf("The smallest positive missing number is %d ", missing);
  getchar();
  return 0;
}
```

Output:

```
The smallest positive missing number is 1
```

Note that this method modifies the original array. We can change the sign of elements in the segregated array to get the same set of elements back. But we still loose the order of elements. If we want to keep the original array as it was, then we can create a copy of the array and run this approach on the temp array.

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

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)

Recent Comments

- 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.

61 Comments GeeksforGeeks

Sort by Newest

Sumit Monga • 4 months ago

Another solution :

```
//this calculates maximum value in the array
int max(int arr[],int n)

{

int max = arr[0];

for(int i = 1; i < n; i++)

if(arr[i] > max)

max = arr[i];

return max;
```

}

//this function returns the minimum missing positive integer
int min_positive_missing(int arr[] int n)

see more

Reply • Share ›

devilsDen • 5 months ago

My O(n) time solution with O(1) space with very simple logic.

#include<limits.h>

int smallestMissing(int arr[], int size){

int min= 1;

int probableMin= 2;

int i;

for(i=0;i<size;i++){ if(arr[i]="">0){

if(arr[i]<min &&="" arr[i]!="1){" min="arr[i]-1;" probablemin="arr[i]+1;" }="" else=
else="" if(arr[i]="probableMin)" probablemin="arr[i]+1;" }="" }="" return="" min
arr[]="{2,3,7,6,8,-1,-10,15};" int="" arr_size="sizeof(arr)/sizeof(arr[0]);" int="" m
arr_size);="" printf("the="" smallest="" positive="" missing="" number="" is="
return="" 0;="" }="">

Reply • Share ›

vinod95300 • 5 months ago

This question cannot be solved in O(n) Space if range of integers is not given

1 Reply • Share ›

vamshi • 5 months ago

Why are we seperating negative and possitive...instead of that we can choose (i=0;i<n;i++) {="" if(a[i]<="0" ||="" a[i]="">=n) continue; a[a[i]] =99999 } In the ne a[index] != 99999.....Please correct me if iam missing anything...

 • Reply • Share ›

me.abhinav • 11 months ago

Incomplete information: There is no comment about the range of elements in t to +100 and the size of array is 10 then this method won't work!

 • Reply • Share ›

me.abhinav me.abhinav • 11 months ago

Sorry my bad....got it.... :)

2   • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

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

 • Reply • Share ›

ravisingh3531 • a year ago

correct me if i am wrong...but it can be solve by creating a BST.

create BST and takeout number in in order as the first number which is not co that no..

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

 • Reply • Share ›

Aditya J ravisingh3531 • 4 months ago

Reply • Share ›

BidhanPal • a year ago

To find the smallest positive number, run a loop from o to n-1, every time try to
of its value. exclude the elements which are already equal to index value or ou
1 or greater than n-1).

After placing all the element in its proper position run a loop to check which is
equal to the index. If none of elements are missing then n is the first element w

```
 int findSmallestMissingNumber(int a[], const int n)
{
    int i = 0;

    //Place all element at the position same as its value
    while (i<n)
    {
        //Skip the element if its already in correct position
        //Or its value is out of array index range
        //Consider zero is not the positive value
        if (a[i] < 1 || a[i] > n-1 || a[i] == i)
```

see more

Reply • Share ›

anonymous ➦ BidhanPal • 3 months ago

Super :)

Reply • Share ›

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]
import java.util.Arrays;

/**
* You are given an unsorted array with both positive and negative elements. Yo
number missing from the array in O(n) time using constant extra space. You c
* Example
* Input: { 2, 3, -7, 6, 8, 1, -10, 15 }
* Output: 4
* @author GAPIITD
*
*/
public class SmallestPositiveNumberMissingFromAnUnsortedArray {

public static void main(String[] args) {
int arr[] = {1, 1, 0, -1, -2};
int shiftPos = shiftNegativeNos(arr);

see more

▲ | ▼ • Reply • Share ›

arnavawasthi ➦ Ganesh • 8 months ago
Your solution return -1 for { 2, 0, 1, -1, -2 };

▲ | ▼ • Reply • Share ›

Qi Hui • a year ago
I add a java implementation. I also try to increase the index other than create a

import java.util.Random;
public class MinPositive {.

public static void main(String[] args){.
int n = 10; //array size.

```
int k = 3; //used to generate negative elements.
int[] array = new int[n];.
for(int i=0;i<n;i++){
array[i] = new Random().nextInt(n) - k;.
}
printArray(array);
int index = swap(array);.
printArray(array);
int missing = findMissing(array, index);.
System.out.println(missing);

}
```

see more

• Reply • Share ›

Ashwin • a year ago

what about this case? {-11,-10,-15,12,13,14}.
The smallest positive number here missing is 1 right? If we apply your solutior
you explain if I am missing anything here?

• Reply • Share ›

coderAce • a year ago

Would like to point out an optimisation. Right now even if A[i] and A[j] are the s
include an if statement before the swap method call inside the for loop of segr

Basically the for loop in segregate should like the following:

```
[sourcecode language="C++"]
int j = 0, i;
for(i = 0; i < N; i++)
{
if (A[i] <= 0)
```

```
{
if(i!=j)//To avoid swapping the same elements
{
swap_ele(&A[i], &A[j]);
}
j++; // increment count of non-positive integers
}
}
```

You can argue that at every step it performs a check, hence O(n) checks reqc
which involves 3 assignments, 3*N in total :)

coderAce coderAce • a year ago

Anyways, nice soln..the first time I was asked this problem in an intervi
guess) I gave the exact solution except that my implementation of the S
claimed my main algo is O(n). Unfortunately my interviewer caught me
come up with an O(n) algo.
This was the O(n^2) one I wrote

```
[sourcecode language="C++"]
int j=N-1;
for(int i=0;i<j;i++)
{
if(A[i]<=0)
{
while(A[j]<=0){j--;}
swap_ele(&A[i],&A[j--]);
}
}

return j+1;
```

sreeram • a year ago

```
int i=0;
for(i=0;i<n ;i++)
{
if(i< n && a[i] != i)
swap(&a[i],&a[a[i]])
}

for(i=0;i<n;i++)
if(a[i] != i )
return i
```

^ | v • Reply • Share ›

nitin gupta • 2 years ago

there is serious mistake in your code ..
here

[if(abs(arr[i]) - 1 0)
arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];]

index 0 in your ex. is 0
so first condition
arr[abs(arr[i])-1]>0 => arr[0-1]=>arr[-1] core dump

^ | v • Reply • Share ›

Kartik ➚ nitin gupta • 2 years ago

There is no problem in the code. Please take a closer look, the functior
an array with positive values. 0 cannot be there in array. See the follow

```
int shift = segregate (arr, size);

// Shift the array and call findMissingPositive for
// positive part
return findMissingPositive(arr+shift, size-shift);
```


lohith • 2 years ago

[sourcecode language="JAVA"]

```
public class SmallestPositiveMissingNumber {

public static void main(String str[]){
int array[] = { 2, 3, -7, 6, 8, 1, -10, 15 };

int i=0;
int answer=0;
while(array[i]<0)i++;
int minumum = array[i];
int maximum = array[i];
for(int j=i;j<array.length;j++){
if(array[j]>0 && minumum>array[j])
minumum = array[j];
if(array[j]>0 && maximum<array[j])
maximum=array[j];
}
```


NIpun • 2 years ago

#include

```
#include
#include
main()
{
int a[]={-1,0, 1, 1, 2, 3,4};
int n = 7 , i;
for(i=0;i=0 &&a[i]<=n)
a[a[i]] = INT_MIN;
}
for(i=1;i<=n;i++)
if(a[i]!=INT_MIN)
{
printf("%d\n",i);
break;
}
return 0;
}
```


NIpun • 2 years ago

Works great for repeated nos.

```
[sourcecode language=""]
#include<stdio.h>
main()
{
int a[]={-1, 1, 1, 2, 3};
const int n = 5;
int b[n+1]={0};
int i;
for(i=0;i<=n;i++)
{
```

```
if(a[i]>=0 &&a[i]<=n)
b[a[i]]++;
}
for(i=1;i<=n;i++)
if(b[i]==0)
{
printf("%d\n",i);
break;
}
return 0;
}
```

Reply • Share ›

cs ➦ NIpun • 2 years ago

But I think this will work only if numbers are in range 1-n.

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

Reply • Share ›

Game • 2 years ago

Guys,
I am sorry to point you "again", that "modifying" the array is equivalent to O(n) s
more sense if you guys use C++ as the coding language. With that said, the s

int findMissingPositive(const vector& numbers);

and the function not using any memory of order n.

The approach you have can be scrutinized heavily in any standard programmi

Reply • Share ›

do you have a better approach for this question?

Reply • Share ›

dees • 2 years ago

what happens in the program if i/p is {50,-10,20,-20,70}

Reply • Share ›

kartik ➔ dees • 2 years ago

The program will print 1.

1 Reply • Share ›

AnnabelIChan ➔ kartik • 3 months ago

why 1? I think it should be 21

Reply • Share ›

prakash • 2 years ago

```
#include
#include
#define max 100
int array[max];
int min=INT_MAX;
main()
{
int n,data,i=1;
scanf("%d",&n); //total no of terms
while(n--)
{
scanf("%d",&data);
if(data>0)
array[data]=data;
}
```

while(array[i]!=0)
i++;
printf("%d\n",i);

return 0;
}

plz any one tell me what would be the time complexit of this code....

• Reply • Share ›

tutum ➦ prakash • 2 years ago

both time and space complexity is O(n) ..same code as i mentioned be

• Reply • Share ›

roshu • 2 years ago

```
public static int getlowest(int[] arr){
            int counter=1;
            for(int i=0;i<arr.length;i++){
                if(arr[i] < counter)
                        continue;
                if(arr[i]==counter) counter++;
                else
                    return counter;


            }
            return 0;
        }
```

• Reply • Share ›

tutum ➦ roshu • 2 years ago

think if the array is sorted .

tutum • 2 years ago

i think its better than other in terms of space complexity...
if you find anything incorrect in this code . tell me

```
 /* #include<stdio.h>

int main()
{
    int i=0,size;
    //int A[]={2, 3, 7, 6, 8, -1, -10, 15};
    int A[]={2, 3, 7, 6, 8, -1, -10, 15};
    size=8;
    int B[8]={};
    for(i=0;i<size;i++){
                    if(A[i]<8 && A[i]>0){
                              B[ A[i]-1 ]=1;
                    }
    }
    for(i=0;i<size;i++){
                    if(B[i]==0){
                               printf("%d",i+1);
                               i=size;
                    }
    }
    return 0;
}
 */
```

hem agnihotri • 2 years ago

make a complete scan of array and find minimum and second minimum elem
then if minimum-1>second minimum then minimum-1 is acceptable

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

• Reply • Share ›

rathi ➦ hem agnihotri • 2 years ago

how can u say this . think before write something as a comment as so

what will be the answer in case 1 2 3 4 6 7
m=1 sm=2 0>2 so ans is 0 but ans is 5

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

• Reply • Share ›

nijju • 2 years ago

#include
#include

int small_positive(int A[16],int n) // Time= O(n) Space=O(1)
{
int i;
for(i=0;i<n;i++)
if(A[i]n)
A[i]=n;
A[n]=n;
for(i=0;i 0 && abs(A[i]) <n)
A[abs(A[i])] = -A[abs(A[i])];

for(i=1;i0 || A[i] == n)

{
printf("\n\nOutput : %d",i);
return 0;
}

see more

• Reply • Share ›

nijju • 2 years ago

#include
#include

int small_positive(int A[16],int n) // Time= O(n) Space=O(1)
{
int i;
for(i=0;i<n;i++)
if(A[i]n)
A[i]=n;
A[n]=n;
for(i=0;i 0 && abs(A[i]) <n)
A[abs(A[i])] = -A[abs(A[i])];

for(i=1;i0 || A[i] == n)
{
printf("\n\nOutput : %d",i);
return 0;
}

see more

• Reply • Share ›

chandan prakash • 2 years ago

Hi, this is another approach taking O(n) time complexity with no extra space .

```
 package pack;

public class SmalstPositivNoMissin {

        /**
         *You are given an unsorted array with both positive and negai
Eg:
Input = {2, 3, 7, 6, 8, -1, -10, 15}
Output = 1

Input = { 2, 3, -7, 6, 8, 1, -10, 15 }
Output = 4
         */

        public static void main(String[] args) {
```

see more

Reply • Share ›

Duke • 2 years ago

[sourcecode language="C++"]
/* Paste your code here (You may delete these lines if not writing code) */
#include
using namespace std;
int misNum(int *arr)
{
int num=10000000;
int len=sizeof(arr)/sizeof(int);
for(int i=0;i0)&&(arr[i]!=arr[i+1])&&(arr[i+1]!=arr[i]+1))

```
{
int small=arr[i]+1;
cout<<"i= "<<i<<" small= "<<small<<endl;
if(small<num)
num=small;
cout<<"i= "<<i<<" num= "<<num;
}

}
```

see more

• Reply • Share ›

Laddoo • 2 years ago

To make this fast..We can add one extra step before applying any other metho
As, We can search for 1 with Binary Search in log(N) Time..Then, any other m
complexity as O(N) in Worst Case..but will improve the average case..

• Reply • Share ›

GeeksforGeeks ➚ Laddoo • 2 years ago

@Laddoo: Binary Search can not be applied here as the input array is

• Reply • Share ›

Vivek Prakash ➚ GeeksforGeeks • 2 years ago

We can actually apply binary search after segregation. We kno
beginning and all positive numbers are at the end. This makes
you search for 0 (or 1) in this array, you will end up either at the
number, based on how you implement it.

For details you can read this tutorial at top coder: http://commu

This is better than the given alogorithm, even though it is still O
as the one given is O(2n).

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

• Reply • Share ›

Venki ➚ Vivek Prakash • 2 years ago

@Vivek,
We can apply binary search, but that would need anothe
auxiliary space, right?

I duplicated original array, and applied binary search, its

• Reply • Share ›

kartik ➚ Venki • 2 years ago

@Vivek & @Venki
I still don't get it. How can we apply binary search. Coul
please.

• Reply • Share ›

rashmi • 2 years ago

what if we have repeated nos i.e -1 1 1 2 3

• Reply • Share ›

GeeksforGeeks ➚ rashmi • 2 years ago

@rashmi: The approach works for repeated numbers as well. Take a

```
if(abs(arr[i]) - 1 < size && arr[abs(arr[i]) - 1] > 0)
 arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];
```

We change the sign of a number only if it is positive. So we make sure
become positive if arr[i] is present.

• Reply • Share ›

satya avv • 2 years ago

At First in the explanation , you said the first negative number, now in the code thing is this algo gives the maximum missing positive number which is min in computes the minimum of the list.

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

• Reply • Share ›

GeeksforGeeks ➚ satya avv • 2 years ago

@satya avv: Please take a closer look, both code and algo say that the positive should be looked for. The algo computes the smallest positive numbers by segregating the positive and negative numbers as a prepr Please let us know if you still have doubts. If you feel something is incc comment.

• Reply • Share ›

satya avv ➚ GeeksforGeeks • 2 years ago

Hi I am sorry for the comment. Yes it works the same.

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

1 • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site