GeeksforGeeks
A computer science portal for geeks
Practice IDE Q&A GeeksQuiz

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) numbers on left
  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)
       {
           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 that
  // 1 is subtracted because index start from 0 and positive numbers start from 1
  for(i = 0; i < size; i++)
  {
    if(abs(arr[i]) - 1 < size && arr[abs(arr[i]) - 1] > 0)
```

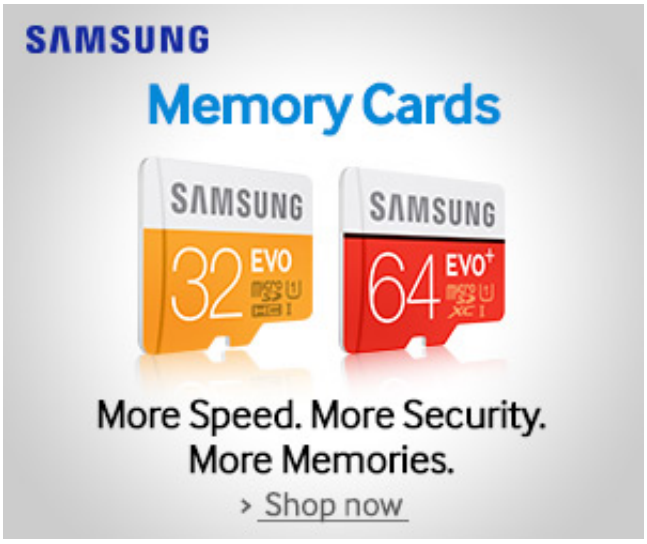
```
      arr[abs(arr[i]) - 1] = -arr[abs(arr[i]) - 1];
  }

  // 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.



128 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

4 Average Difficulty : **4/5.0**
Based on **3** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

@geeksforgeeks, Some rights reserved Contact Us! About Us! Advertise with us!