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;
}
```

Run on IDE

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.

113 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)

2.6 Average Difficulty : **2.6/5.0**
Based on **8** vote(s)

Add to TODO List

Mark as DONE

Like Share 4 people like this. Be the first of your friends.

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