Check if array elements are consecutive | Added Method 3

Given an unsorted array of numbers, write a function that returns true if array consists of consecutive numbers.

Examples:
a) If array is {5, 2, 3, 1, 4}, then the function should return true because the array has consecutive numbers from 1 to 5.

b) If array is {83, 78, 80, 81, 79, 82}, then the function should return true because the array has consecutive numbers from 78 to 83.

c) If the array is {34, 23, 52, 12, 3 }, then the function should return false because the elements are not consecutive.

d) If the array is {7, 6, 5, 5, 3, 4}, then the function should return false because 5 and 5 are not consecutive.

Method 1 (Use Sorting)
1) Sort all the elements.
2) Do a linear scan of the sorted array. If the difference between current element and next element is anything other than 1, then return false. If all differences are 1, then return true.

Time Complexity: O(nLogn)

Method 2 (Use visited array)
The idea is to check for following two conditions. If following two conditions are true, then return true.
1) *max – min + 1 = n* where max is the maximum element in array, min is minimum element in array and n is the number of elements in array.
2) All elements are distinct.

To check if all elements are distinct, we can create a visited[] array of size n. We can map the ith element of input array arr[] to visited array by using arr[i] – min as index in visited[].

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

/* Helper functions to get minimum and maximum in an array */
int getMin(int arr[], int n);
```

```
int getMax(int arr[], int n);

/* The function checks if the array elements are consecutive
  If elements are consecutive, then returns true, else returns
  false */
bool areConsecutive(int arr[], int n)
{
  if ( n <  1 )
    return false;

  /* 1) Get the minimum element in array */
  int min = getMin(arr, n);

  /* 2) Get the maximum element in array */
  int max = getMax(arr, n);

  /* 3) max - min + 1 is equal to n,  then only check all elements */
  if (max - min  + 1 == n)
  {
      /* Create a temp array to hold visited flag of all elements.
         Note that, calloc is used here so that all values are initialized
         as false */
      bool *visited = (bool *) calloc (n, sizeof(bool));
      int i;
      for (i = 0; i < n; i++)
      {
         /* If we see an element again, then return false */
         if ( visited[arr[i] - min] != false )
           return false;

         /* If visited first time, then mark the element as visited */
         visited[arr[i] - min] = true;
      }

      /* If all elements occur once, then return true */
      return true;
  }

  return false; // if (max - min  + 1 != n)
}

/* UTILITY FUNCTIONS */
int getMin(int arr[], int n)
{
  int min = arr[0];
  for (int i = 1; i < n; i++)
   if (arr[i] < min)
     min = arr[i];
  return min;
}
```

```
int getMax(int arr[], int n)
{
  int max = arr[0];
  for (int i = 1; i < n; i++)
   if (arr[i] > max)
     max = arr[i];
  return max;
}

/* Driver program to test above functions */
int main()
{
    int arr[]= {5, 4, 2, 3, 1, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    if(areConsecutive(arr, n) == true)
        printf(" Array elements are consecutive ");
    else
        printf(" Array elements are not consecutive ");
    getchar();
    return 0;
}
```

Time Complexity: O(n)
Extra Space: O(n)

Method 3 (Mark visited array elements as negative)
This method is O(n) time complexity and O(1) extra space, but it changes the original array and it works only if all numbers are positive. We can get the original array by adding an extra step though. It is an extension of method 2 and it has the same two steps.
1) *max – min + 1 = n* where max is the maximum element in array, min is minimum element in array and n is the number of elements in array.
2) All elements are distinct.

In this method, the implementation of step 2 differs from method 2. Instead of creating a new array, we modify the input array arr[] to keep track of visited elements. The idea is to traverse the array and for each index i (where 0 <= i < n), make arr[arr[i] - min]] as a negative value. If we see a negative value again then there is repetition.

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

/* Helper functions to get minimum and maximum in an array */
int getMin(int arr[], int n);
int getMax(int arr[], int n);

/* The function checks if the array elements are consecutive
  If elements are consecutive, then returns true, else returns
```

```
  false */
bool areConsecutive(int arr[], int n)
{

    if ( n <  1 )
        return false;

    /* 1) Get the minimum element in array */
    int min = getMin(arr, n);

    /* 2) Get the maximum element in array */
    int max = getMax(arr, n);

    /* 3) max - min + 1 is equal to n then only check all elements */
    if (max - min  + 1 == n)
    {
        int i;
        for(i = 0; i < n; i++)
        {
            int j;

            if (arr[i] < 0)
                j = -arr[i] - min;
            else
                j = arr[i] - min;

            // if the value at index j is negative then
            // there is repitition
            if (arr[j] > 0)
                arr[j] = -arr[j];
            else
                return false;
        }

        /* If we do not see a negative value then all elements
           are distinct */
        return true;
    }

    return false; // if (max - min  + 1 != n)
}

/* UTILITY FUNCTIONS */
int getMin(int arr[], int n)
{
    int min = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] < min)
            min = arr[i];
    return min;
```

```
}

int getMax(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}

/* Driver program to test above functions */
int main()
{
    int arr[]= {1, 4, 5, 3, 2, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    if(areConsecutive(arr, n) == true)
        printf(" Array elements are consecutive ");
    else
        printf(" Array elements are not consecutive ");
    getchar();
    return 0;
}
```

Note that this method might not work for negative numbers. For example, it returns false for {2, 1, 0, -3, -1, -2}.

Time Complexity: O(n)
Extra Space: O(1)

Source: http://geeksforgeeks.org/forum/topic/amazon-interview-question-for-software-engineerdeveloper-fresher-9

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

323 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)

1.9 Average Difficulty : **1.9/5.0**
Based on **10** 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.