Check for Majority Element in a sorted array

Question: Write a C function to find if a given integer x appears more than n/2 times in a sorted array of n integers.

Basically, we need to write a function say isMajority() that takes an array (arr[]), array's size (n) and a number to be searched (x) as parameters and returns true if x is a majority element (present more than n/2 times).

Examples:

```
Input: arr[] = {1, 2, 3, 3, 3, 3, 10}, x = 3
Output: True (x appears more than n/2 times in the given array)

Input: arr[] = {1, 1, 2, 4, 4, 4, 6, 6}, x = 4
Output: False (x doesn't appear more than n/2 times in the given array)

Input: arr[] = {1, 1, 1, 2, 2}, x = 1
Output: True (x appears more than n/2 times in the given array)
```

METHOD 1 (Using Linear Search)
Linearly search for the first occurrence of the element, once you find it (let at index i), check element at index i + n/2. If element is present at i+n/2 then return 1 else return 0.

C

```
/* C Program to check for majority element in a sorted array */
# include <stdio.h>
# include <stdbool.h>

bool isMajority(int arr[], int n, int x)
{
    int i;

    /* get last index according to n (even or odd) */
    int last_index = n%2? (n/2+1): (n/2);

    /* search for first occurrence of x in arr[]*/
    for (i = 0; i < last_index; i++)
    {
        /* check if x is present and is present more than n/2
```

```
           times */
        if (arr[i] == x && arr[i+n/2] == x)
            return 1;
    }
    return 0;
}

/* Driver program to check above function */
int main()
{
     int arr[] ={1, 2, 3, 4, 4, 4, 4};
     int n = sizeof(arr)/sizeof(arr[0]);
     int x = 4;
     if (isMajority(arr, n, x))
        printf("%d appears more than %d times in arr[]",
               x, n/2);
     else
        printf("%d does not appear more than %d times in arr[]",
                x, n/2);

   return 0;
}
```

Run on IDE

Java

```
/* Program to check for majority element in a sorted array */
import java.io.*;

class Majority {

    static boolean isMajority(int arr[], int n, int x)
    {
        int i, last_index = 0;

        /* get last index according to n (even or odd) */
        last_index = (n%2==0)? n/2: n/2+1;

        /* search for first occurrence of x in arr[]*/
        for (i = 0; i < last_index; i++)
        {
            /* check if x is present and is present more
               than n/2 times */
            if (arr[i] == x && arr[i+n/2] == x)
                return true;
        }
        return false;
    }

    /* Driver function to check for above functions*/
    public static void main (String[] args) {
        int arr[] = {1, 2, 3, 4, 4, 4, 4};
        int n = arr.length;
        int x = 4;
        if (isMajority(arr, n, x)==true)
           System.out.println(x+" appears more than "+
                              n/2+" times in arr[]");
        else
           System.out.println(x+" does not appear more than "+
                              n/2+" times in arr[]");
    }
}
/*This article is contributed by Devesh Agrawal*/
```

Run on IDE

Output:

```
4 appears more than 3 times in arr[]
```

Time Complexity: O(n)

METHOD 2 (Using Binary Search)

Use binary search methodology to find the first occurrence of the given number. The criteria for binary search is important here.

C

```
/* Program to check for majority element in a sorted array */
# include <stdio.h>
# include <stdbool.h>

/* If x is present in arr[low...high] then returns the index of
first occurrence of x, otherwise returns -1 */
int _binarySearch(int arr[], int low, int high, int x);

/* This function returns true if the x is present more than n/2
times in arr[] of size n */
bool isMajority(int arr[], int n, int x)
{
    /* Find the index of first occurrence of x in arr[] */
    int i = _binarySearch(arr, 0, n-1, x);

    /* If element is not present at all, return false*/
    if (i == -1)
        return false;

    /* check if the element is present more than n/2 times */
    if (((i + n/2) <= (n -1)) && arr[i + n/2] == x)
        return true;
    else
        return false;
}

/* If x is present in arr[low...high] then returns the index of
first occurrence of x, otherwise returns -1 */
int _binarySearch(int arr[], int low, int high, int x)
{
    if (high >= low)
    {
        int mid = (low + high)/2; /*low + (high - low)/2;*/

        /* Check if arr[mid] is the first occurrence of x.
            arr[mid] is first occurrence if x is one of the following
```

```
           is true:
           (i) mid == 0 and arr[mid] == x
           (ii) arr[mid-1] < x and arr[mid] == x
       */
       if ( (mid == 0 || x > arr[mid-1]) && (arr[mid] == x) )
           return mid;
       else if (x > arr[mid])
           return _binarySearch(arr, (mid + 1), high, x);
       else
           return _binarySearch(arr, low, (mid -1), x);
    }

    return -1;
}

/* Driver program to check above functions */
int main()
{
    int arr[] = {1, 2, 3, 3, 3, 3, 10};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 3;
    if (isMajority(arr, n, x))
        printf("%d appears more than %d times in arr[]",
               x, n/2);
    else
        printf("%d does not appear more than %d times in arr[]",
               x, n/2);
    return 0;
}
```

Run on IDE

Java

```
/* Program to check for majority element in a sorted array */
import java.io.*;

class Majority {

    /* If x is present in arr[low...high] then returns the index of
        first occurrence of x, otherwise returns -1 */
    static int  _binarySearch(int arr[], int low, int high, int x)
    {
        if (high >= low)
        {
            int mid = (low + high)/2;  /*low + (high - low)/2;*/

            /* Check if arr[mid] is the first occurrence of x.
                arr[mid] is first occurrence if x is one of the following
                is true:
                (i)  mid == 0 and arr[mid] == x
                (ii) arr[mid-1] < x and arr[mid] == x
            */
            if ( (mid == 0 || x > arr[mid-1]) && (arr[mid] == x) )
                return mid;
            else if (x > arr[mid])
                return _binarySearch(arr, (mid + 1), high, x);
            else
                return _binarySearch(arr, low, (mid -1), x);
        }

        return -1;
    }


    /* This function returns true if the x is present more than n/2
        times in arr[] of size n */
    static boolean isMajority(int arr[], int n, int x)
    {
        /* Find the index of first occurrence of x in arr[] */
        int i = _binarySearch(arr, 0, n-1, x);

        /* If element is not present at all, return false*/
        if (i == -1)
            return false;

        /* check if the element is present more than n/2 times */
        if (((i + n/2) <= (n -1)) && arr[i + n/2] == x)
            return true;
        else
            return false;
    }

    /*Driver function to check for above functions*/
    public static void main (String[] args)  {

        int arr[] = {1, 2, 3, 3, 3, 3, 10};
        int n = arr.length;
        int x = 3;
        if (isMajority(arr, n, x)==true)
            System.out.println(x + " appears more than "+
                                n/2 + " times in arr[]");
        else
            System.out.println(x + " does not appear more than " +
                                n/2 + " times in arr[]");
    }
}
/*This code is contributed by Devesh Agrawal*/
```

Run on IDE

Output:

```
3 appears more than 3 times in arr[]
```

Time Complexity: O(Logn)

Algorithmic Paradigm: Divide and Conquer

Please write comments if you find any bug in the above program/algorithm or a better way to solve the same problem.

160 Comments Category: Arrays Divide and Conquer Tags: Divide and Conquer

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 **9** vote(s)

Add to TODO List

Mark as DONE

Like Share 16 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.