Find a peak element

Given an array of integers. Find a peak element in it. An array element is peak if it is NOT smaller than its neighbors. For corner elements, we need to consider only one neighbor. For example, for input array {5, 10, 20, 15}, 20 is the only peak element. For input array {10, 20, 15, 2, 23, 90, 67}, there are two peak elements: 20 and 90. Note that we need to return any one peak element.

Following corner cases give better idea about the problem.
1) If input array is sorted in strictly increasing order, the last element is always a peak element. For example, 50 is peak element in {10, 20, 30, 40, 50}.
2) If input array is sorted in strictly decreasing order, the first element is always a peak element. 100 is the peak element in {100, 80, 60, 50, 20}.
3) If all elements of input array are same, every element is a peak element.

It is clear from above examples that there is always a peak element in input array in any input array.

A **simple solution** is to do a linear scan of array and as soon as we find a peak element, we return it. The worst case time complexity of this method would be O(n).

Can we find a peak element in worst time complexity better than O(n)?
We can use Divide and Conquer to find a peak in O(Logn) time. The idea is Binary Search based, we compare middle element with its neighbors. If middle element is greater than both of its neighbors, then we return it. If the middle element is smaller than the its left neighbor, then there is always a peak in left half (Why? take few examples). If the middle element is smaller than the its right neighbor, then there is always a peak in right half (due to same reason as left half). Following is C implementation of this approach.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
// A divide and conquer solution to find a peak element element
#include <stdio.h>

// A binary search based function that returns index of a peak element
int findPeakUtil(int arr[], int low, int high, int n)
{
    // Fin index of middle element
    int mid = low + (high - low)/2;  /* (low + high)/2 */

    // Compare middle element with its neighbours (if neighbours exist
    if ((mid == 0 || arr[mid-1] <= arr[mid]) &&
            (mid == n-1 || arr[mid+1] <= arr[mid]))
        return mid;

    // If middle element is not peak and its left neighbor is greater
    // then left half must have a peak element
    else if (mid > 0 && arr[mid-1] > arr[mid])
        return findPeakUtil(arr, low, (mid -1), n);

    // If middle element is not peak and its right neighbor is greater
    // then right half must have a peak element
    else return findPeakUtil(arr, (mid + 1), high, n);
}

// A wrapper over recursive function findPeakUtil()
int findPeak(int arr[], int n)
{
    return findPeakUtil(arr, 0, n-1, n);
}

/* Driver program to check above functions */
int main()
{
    int arr[] = {1, 3, 20, 4, 1, 0};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Index of a peak point is %d", findPeak(arr, n));
    return 0;
}
```

Output:

```
Index of a peak point is 2
```

Time Complexity: O(Logn) where n is number of elements in input array.

Exercise:

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Consider the following modified definition of peak element. An array element is peak if it is greater than its neighbors. Note that an array may not contain a peak element with this modified definition.

References:
http://courses.csail.mit.edu/6.006/spring11/lectures/lec02.pdf
http://www.youtube.com/watch?v=HtSuA80QTyo

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)
- 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

32 Tweet 3 1

Writing code in comment? Please use **ideone.com** and share the link here.

Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 4 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 7 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 32 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 34 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 59 minutes ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago