Problem | Editorial | Submissions

Peak element

Difficulty: **Basic** Accuracy: **38.86%** Submissions: **518K+** Points: **1**

Given an array **arr[]** where no two adjacent elements are same, find the **index** of a **peak** element. An element is considered to be a **peak** if it is greater than its adjacent elements (if they exist). If there are multiple peak elements, return index of any one of them. The output will be **"true"** if the index returned by your function is correct; otherwise, it will be "**false**".

Note: Consider the element **before** the **first** element and the element **after** the **last** element to be **negative infinity**.

Examples :

Input: arr = [1, 2, 4, 5, 7, 8, 3]
Output: true
Explanation: arr[5] = 8 is a peak element because arr[4] < arr[5] > arr[6].

Input: arr = [10, 20, 15, 2, 23, 90, 80]
Output: true
Explanation: arr[1] = 20 and arr[5] = 90 are peak elements because arr[0] < arr[1] > arr[2] and arr[4] < arr[5] > arr[6].

Java (1.8) | Average Time: 30m | Start Timer

```
1
49
50
51  class Solution {
52      public int peakElement(int[] arr) {
53          int n = arr.length;
54          if(n==1)
55              return 0;
56          if(arr[0]>arr[1])
57              return 0;
58          if(arr[n - 1] > arr[n - 2])
59              return n - 1;
60          int lo = 1, hi = n-2;
61          while(lo <= hi){
62              int mid = lo + (hi - lo) / 2;
63              if(arr[mid] > arr[mid - 1] && arr[mid] > arr[mid + 1])
64                  return mid;
65              if (arr[mid]< arr[mid + 1])
66                  lo = mid + 1 ;
67              else
68                  hi = mid - 1;
69          }
70          return 0;
71      }
72  }
73
```

Custom Input | Compile & Run | Submit