Find a Peak Element

Introduction

In this problem, we are given an array of unsorted integers. Our task is to find a peak element in the array. A peak element is an element whose neighbors have value smaller than that of the element.



Note: For corner elements, we need to consider only one neighbor.

Example:

Input: array[]= {5, 10, 20, 15}

Output: 20

Explanation: The element 20 has neighbors 10 and 15, both of them

are less than 20.

Input: array[] = {10, 20, 15, 2, 23, 90, 67}

Output: 20 or 90

Explanation: The element 20 has neighbors 10 and 15, both of them

are less than 20, similarly 90 has neighbors 23 and 67.

Naive Approach:

The idea is to traverse the array and return the element that has higher value than its neighbors.

Steps:

- If the element at index 0 is greater than the element at index 1, return arr[0].
- If the element at index n-1 is greater than the element at index n-2, return arr[n-1].
- If none of the above cases prevail, we traverse the remaining elements and check for a peak element.

Below is the implementation of above idea.



lava

Track Progress

all

```
int getPeak(int arr[], int n)
                                      \mathbf{m}
                                    Articles
    if(n == 1)
        return arr[0];
                                      if(arr[0] >= arr[1])
                                    Videos
        return arr[0];
    if(arr[n - 1] >= arr[n - 2])
                                      </>
        return arr[n - 1];
                                   Problems
    for(int i = 1; i < n - 1; i++)
        if(arr[i] >= arr[i - 1] && arr[i] >= arr[i + 1])
             return arr[i];
}
int main() {
 int arr[] = \{5, 10, 7, 8, 20, 12\}, n = 6;
 cout << getPeak(arr, n);</pre>
    return 0;
}
```

Output

10

Time Complexity: O(N), Where n is the number of elements in the input array. **Auxiliary Space:** O(1).

Find a peak element using iterative Binary search

Menu

The interesting fact about using Binary search in this unsorted array problem is that if the element to the left of the mid element is greater,

^

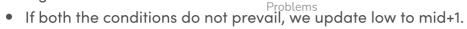
Track Progress

would be present in the right of the array. We will be using this logic to



- We calculate the mid as (low+high) 12 where low is the lower bound and high is the upper bound of the array.
- We run a loop while low is lesser than or equal to high.
- We check the condition: If the mid is 0 or arr [mid-1] is smaller than arr[mid] and if mid is n-1 and arr[mid+1] i smaller than arr[mid], we return mid.





Courses

Tutorials





```
Practice #include <iostream>
Contests sing namespace std;
```





Menu

Track Progress

,		
	int $arr[] = \{5, 20, 40, 30, 20, 50, 60\}, n = 7;$	
	All	
	return 0; Articles	

Time Complexity: O(log N), Where n is the number of elements in the input array. **Auxiliary Space:** O(1), No extra space is required, so the space complexity is constant.



</>

Problems

Mark as Read



Quiz Report An Issue

If you are facing any issue on this page. Please let us know.



Contest

Menu

^

Track Progress