Exp-2.6

Title:

Find a peak element's index in an array using O(log n) time.

Aim:

To design and implement a Python program to find the index of a peak element in an array, where a peak element is strictly greater than its neighbors, in logarithmic time complexity.

Procedure:

1. Read the input array nums.
2. Use a binary search approach to locate a peak element:
    - Start with pointers left at 0 and right at len(nums) - 1.
    - While left < right:
        - Compute mid = (left + right) // 2.
        - If nums[mid] < nums[mid + 1], then peak must be in the right half, set left = mid + 1.
        - Else, peak is in the left half (including mid), set right = mid.
3. After loop ends, left points to a peak element.
4. Return left as the peak element index.
5. Print the index.

Algorithm:

1. Start
2. Input array nums.
3. Initialize left = 0, right = len(nums) - 1.
4. While left < right:
    - mid = (left + right) // 2
    - If nums[mid] < nums[mid + 1], set left = mid + 1
    - Else, set right = mid

5. Return left as the peak index.
6. Stop.

Input:

4

1 2 3 1

7

1 2 1 3 5 6 4

Output:

2

1 or 0

Program:

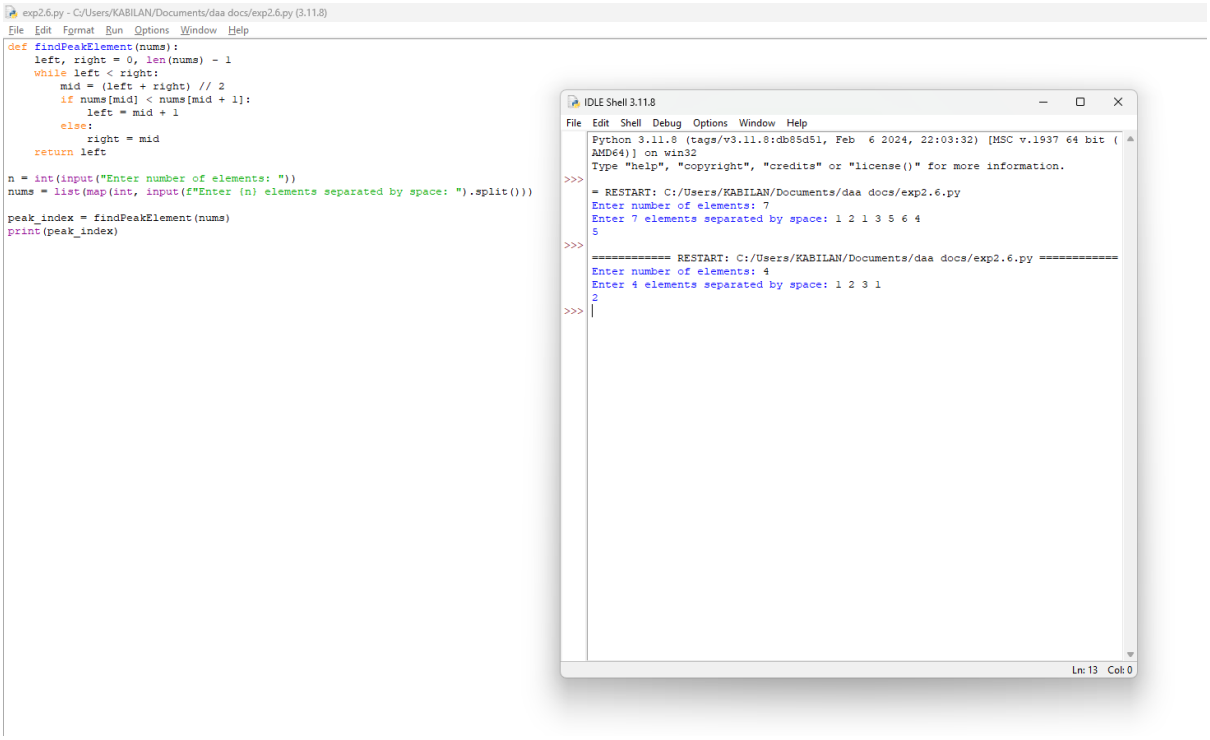
```
def findPeakElement(nums):
    left, right = 0, len(nums) - 1
    while left < right:
        mid = (left + right) // 2
        if nums[mid] < nums[mid + 1]:
            left = mid + 1
        else:
            right = mid
    return left
n = int(input("Enter number of elements: "))
nums = list(map(int, input(f"Enter {n} elements separated by space: ").split()))
peak_index = findPeakElement(nums)
print(peak_index)
```

Performance Analysis:

Time Complexity: O(log n)

Space Complexity: O(1)

Program Output:



Result:

Thus the given program Peak Element Finder is executed and got output successfully.