Problem 162: Find Peak Element

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

A peak element is an element that is strictly greater than its neighbors.

Given a

0-indexed

integer array

nums

, find a peak element, and return its index. If the array contains multiple peaks, return the index to

any of the peaks

.

You may imagine that

nums[-1] = nums[n] = -∞

. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in

O(log n)

time.

Example 1:

Input:

nums = [1,2,3,1]

Output:

2

Explanation:

3 is a peak element and your function should return the index number 2.

Example 2:

Input:

nums = [1,2,1,3,5,6,4]

Output:

5

Explanation:

Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

Constraints:

1 <= nums.length <= 1000

-2

31

<= nums[i] <= 2

31

- 1

nums[i] != nums[i + 1]

for all valid

i

.

Code Snippets

C++:

```
class Solution {
public:
int findPeakElement(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int findPeakElement(int[] nums) {

}
}
```

Python3:

```
class Solution:
def findPeakElement(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def findPeakElement(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var findPeakElement = function(nums) {

};
```

TypeScript:

```
function findPeakElement(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int FindPeakElement(int[] nums) {

}
}
```

C:

```
int findPeakElement(int* nums, int numsSize) {

}
```

Go:

```
func findPeakElement(nums []int) int {
}
```

Kotlin:

```
class Solution {
fun findPeakElement(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func findPeakElement(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_peak_element(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def find_peak_element(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findPeakElement($nums) {

}
}
```

Dart:

```
class Solution {
int findPeakElement(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def findPeakElement(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_peak_element(nums :: [integer]) :: integer
def find_peak_element(nums) do

end
end
```

Erlang:

```
-spec find_peak_element(Nums :: [integer()]) -> integer().
find_peak_element(Nums) ->
.
```

Racket:

```
(define/contract (find-peak-element nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int findPeakElement(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int findPeakElement(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Peak Element
Difficulty: Medium
Tags: array, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def findPeakElement(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findPeakElement(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @return {number}
*/
var findPeakElement = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findPeakElement(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FindPeakElement(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find Peak Element
* Difficulty: Medium
* Tags: array, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int findPeakElement(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Find Peak Element
// Difficulty: Medium
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findPeakElement(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findPeakElement(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func findPeakElement(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find Peak Element
// Difficulty: Medium
// Tags: array, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_peak_element(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def find_peak_element(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findPeakElement($nums) {

}
}
```

Dart Solution:

```
class Solution {
int findPeakElement(List<int> nums) {
}
}
```

Scala Solution:

```
object Solution {
def findPeakElement(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_peak_element(nums :: [integer]) :: integer
def find_peak_element(nums) do

end
end
```

Erlang Solution:

```
-spec find_peak_element(Nums :: [integer()]) -> integer().
find_peak_element(Nums) ->
.
```

Racket Solution:

```
(define/contract (find-peak-element nums)
(-> (listof exact-integer?) exact-integer?)
)
```