Problem 1150: Check If a Number Is Majority Element in a Sorted Array

Problem Information

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

Problem Description

Given an integer array

nums

sorted in non-decreasing order and an integer

target

, return

true

if

target

is a

majority

element, or

false

otherwise

.

A

majority

element in an array

nums

is an element that appears more than

nums.length / 2

times in the array.

Example 1:

Input:

nums = [2,4,5,5,5,5,5,6,6], target = 5

Output:

true

Explanation:

The value 5 appears 5 times and the length of the array is 9. Thus, 5 is a majority element because 5 > 9/2 is true.

Example 2:

Input:

nums = [10,100,101,101], target = 101

Output:

false

Explanation:

The value 101 appears 2 times and the length of the array is 4. Thus, 101 is not a majority element because 2 > 4/2 is false.

Constraints:

1 <= nums.length <= 1000

1 <= nums[i], target <= 10

9

nums

is sorted in non-decreasing order.

Code Snippets

C++:

```
class Solution {
public:
bool isMajorityElement(vector<int>& nums, int target) {

}
};
```

Java:

```
class Solution {
public boolean isMajorityElement(int[] nums, int target) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @param {number} target
* @return {boolean}
*/
var isMajorityElement = function(nums, target) {

};
```

TypeScript:

```
function isMajorityElement(nums: number[], target: number): boolean {

};
```

C#:

```
public class Solution {
public bool IsMajorityElement(int[] nums, int target) {

}
}
```

C:

```
bool isMajorityElement(int* nums, int numsSize, int target) {

}
```

Go:

```
func isMajorityElement(nums []int, target int) bool {

}
```

Kotlin:

```
class Solution {
fun isMajorityElement(nums: IntArray, target: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func isMajorityElement(_ nums: [Int], _ target: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_majority_element(nums: Vec<i32>, target: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} target
# @return {Boolean}
def is_majority_element(nums, target)

end
```

PHP:

```
class Solution {
```

```
/**
* @param Integer[] $nums
* @param Integer $target
* @return Boolean
*/
function isMajorityElement($nums, $target) {

}
}
```

Dart:

```
class Solution {
bool isMajorityElement(List<int> nums, int target) {

}
}
```

Scala:

```
object Solution {
def isMajorityElement(nums: Array[Int], target: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_majority_element(nums :: [integer], target :: integer) :: boolean
def is_majority_element(nums, target) do

end
end
```

Erlang:

```
-spec is_majority_element(Nums :: [integer()], Target :: integer()) ->
boolean().
is_majority_element(Nums, Target) ->
.
```

Racket:

```
(define/contract (is-majority-element nums target)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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:
bool isMajorityElement(vector<int>& nums, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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 boolean isMajorityElement(int[] nums, int target) {
}
}
```

Python3 Solution:

```
"""
Problem: Check If a Number Is Majority Element in a Sorted Array
Difficulty: Easy
Tags: array, sort, 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 isMajorityElement(self, nums: List[int], target: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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
* @param {number} target
* @return {boolean}
*/
var isMajorityElement = function(nums, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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 isMajorityElement(nums: number[], target: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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 bool IsMajorityElement(int[] nums, int target) {
}
}
```

C Solution:

```
/*
* Problem: Check If a Number Is Majority Element in a Sorted Array
* Difficulty: Easy
* Tags: array, sort, 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
*/

bool isMajorityElement(int* nums, int numsSize, int target) {

}
```

Go Solution:

```
// Problem: Check If a Number Is Majority Element in a Sorted Array
// Difficulty: Easy
// Tags: array, sort, 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 isMajorityElement(nums []int, target int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isMajorityElement(nums: IntArray, target: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isMajorityElement(_ nums: [Int], _ target: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check If a Number Is Majority Element in a Sorted Array
// Difficulty: Easy
// Tags: array, sort, 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 is_majority_element(nums: Vec<i32>, target: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} target
# @return {Boolean}
def is_majority_element(nums, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $target
* @return Boolean
*/
function isMajorityElement($nums, $target) {

}
}
```

Dart Solution:

```
class Solution {
bool isMajorityElement(List<int> nums, int target) {

}
}
```

Scala Solution:

```
object Solution {
def isMajorityElement(nums: Array[Int], target: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_majority_element(nums :: [integer], target :: integer) :: boolean
def is_majority_element(nums, target) do

end
end
```

Erlang Solution:

```
-spec is_majority_element(Nums :: [integer()], Target :: integer()) ->
boolean().
is_majority_element(Nums, Target) ->
.
```

Racket Solution:

```
(define/contract (is-majority-element nums target)
(-> (listof exact-integer?) exact-integer? boolean?)
)
```