Problem 2229: Check if an Array Is Consecutive

Problem Information

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

Problem Description

Given an integer array

nums

, return

true

if

nums

is

consecutive

, otherwise return

false

.

An array is

consecutive

if it contains every number in the range

[x, x + n - 1]

(

inclusive

), where

x

is the minimum number in the array and

n

is the length of the array.

Example 1:

Input:

nums = [1,3,4,2]

Output:

true

Explanation:

The minimum value is 1 and the length of nums is 4. All of the values in the range [x, x + n - 1] = [1, 1 + 4 - 1] = [1, 4] = (1, 2, 3, 4) occur in nums. Therefore, nums is consecutive.

Example 2:

Input:

nums = [1,3]

Output:

false

Explanation:

The minimum value is 1 and the length of nums is 2. The value 2 in the range [x, x + n - 1] = [1, 1 + 2 - 1], = [1, 2] = (1, 2) does not occur in nums. Therefore, nums is not consecutive.

Example 3:

Input:

nums = [3,5,4]

Output:

true

Explanation:

The minimum value is 3 and the length of nums is 3. All of the values in the range [x, x + n - 1] = [3, 3 + 3 - 1] = [3, 5] = (3, 4, 5) occur in nums. Therefore, nums is consecutive.

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean isConsecutive(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isConsecutive(nums: number[]): boolean {
```

```
};
```

C#:

```
public class Solution {
public bool IsConsecutive(int[] nums) {

}
}
```

C:

```
bool isConsecutive(int* nums, int numsSize) {

}
```

Go:

```
func isConsecutive(nums []int) bool {

}
```

Kotlin:

```
class Solution {
fun isConsecutive(nums: IntArray): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

```
}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
bool isConsecutive(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-consecutive nums)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public boolean isConsecutive(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if an Array Is Consecutive
Difficulty: Easy
Tags: array, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function isConsecutive(nums: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public bool IsConsecutive(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Check if an Array Is Consecutive
* Difficulty: Easy
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

bool isConsecutive(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Check if an Array Is Consecutive
// Difficulty: Easy
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func isConsecutive(nums []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isConsecutive(nums: IntArray): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check if an Array Is Consecutive
// Difficulty: Easy
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
bool isConsecutive(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-consecutive nums)
(-> (listof exact-integer?) boolean?)
)
```