Problem 2784: Check if Array is Good

Problem Information

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

Problem Description

You are given an integer array

nums

. We consider an array

good

if it is a permutation of an array

base[n]

.

base[n] = [1, 2, ..., n - 1, n, n]

(in other words, it is an array of length

n + 1

which contains

1

to

n - 1

exactly once, plus two occurrences of

n

). For example,

base[1] = [1, 1]

and

base[3] = [1, 2, 3, 3]

.

Return

true

if the given array is good, otherwise return

false

.

Note:

A permutation of integers represents an arrangement of these numbers.

Example 1:

Input:

nums = [2, 1, 3]

Output:

false

Explanation:

Since the maximum element of the array is 3, the only candidate n for which this array could be a permutation of base[n], is n = 3. However, base[3] has four elements but array nums has three. Therefore, it can not be a permutation of base[3] = [1, 2, 3, 3]. So the answer is false.

Example 2:

Input:

nums = [1, 3, 3, 2]

Output:

true

Explanation:

Since the maximum element of the array is 3, the only candidate n for which this array could be a permutation of base[n], is n = 3. It can be seen that nums is a permutation of base[3] = [1, 2, 3, 3] (by swapping the second and fourth elements in nums, we reach base[3]). Therefore, the answer is true.

Example 3:

Input:

nums = [1, 1]

Output:

true

Explanation:

Since the maximum element of the array is 1, the only candidate n for which this array could be a permutation of base[n], is n = 1. It can be seen that nums is a permutation of base[1] = [1, 1]. Therefore, the answer is true.

Example 4:

Input:

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

Output:

false

Explanation:

Since the maximum element of the array is 4, the only candidate n for which this array could be a permutation of base[n], is n = 4. However, base[4] has five elements but array nums has six. Therefore, it can not be a permutation of base[4] = [1, 2, 3, 4, 4]. So the answer is false.

Constraints:

1 <= nums.length <= 100

1 <= num[i] <= 200

Code Snippets

C++:

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

}
};
```

Java:

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

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Check if Array is Good
* 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 isGood(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Array is Good
* 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 isGood(int[] nums) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Check if Array is Good
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 isGood(self, nums: List[int]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if Array is Good
* 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 isGood = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Array is Good
* 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 isGood(nums: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Array is Good
* 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 IsGood(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Check if Array is Good
* 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 isGood(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Check if Array is Good
// 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 isGood(nums []int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Check if Array is Good
// 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_good(nums: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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