Problem 3046: Split the Array

Problem Information

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

Problem Description

You are given an integer array

nums

of

even

length. You have to split the array into two parts

nums1

and

nums2

such that:

nums1.length == nums2.length == nums.length / 2

.

nums1

should contain

distinct

elements.

nums2

should also contain

distinct

elements.

Return

true

if it is possible to split the array, and

false

otherwise

.

Example 1:

Input:

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

Output:

true

Explanation:

One of the possible ways to split nums is nums1 = [1,2,3] and nums2 = [1,2,4].

Example 2:

Input:

nums = [1,1,1,1]

Output:

false

Explanation:

The only possible way to split nums is nums1 = [1,1] and nums2 = [1,1]. Both nums1 and nums2 do not contain distinct elements. Therefore, we return false.

Constraints:

1 <= nums.length <= 100

nums.length % 2 == 0

1 <= nums[i] <= 100

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-possible-to-split nums)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 isPossibleToSplit(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 isPossibleToSplit(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Split the Array
Difficulty: Easy
Tags: array, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 isPossibleToSplit = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 isPossibleToSplit(nums: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 IsPossibleToSplit(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Split the Array
* Difficulty: Easy
* Tags: array, hash
*
* 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 isPossibleToSplit(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Split the Array
// Difficulty: Easy
// Tags: array, hash
//
// 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 isPossibleToSplit(nums []int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Split the Array
// Difficulty: Easy
// Tags: array, hash
//
// 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_possible_to_split(nums: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-possible-to-split nums)
(-> (listof exact-integer?) boolean?)
)
```