Problem 3151: Special Array I

Problem Information

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

Problem Description

An array is considered

special

if the

parity

of every pair of adjacent elements is different. In other words, one element in each pair

must

be even, and the other

must

be odd.

You are given an array of integers

nums

. Return

true

if

nums

is a

special

array, otherwise, return

false

.

Example 1:

Input:

nums = [1]

Output:

true

Explanation:

There is only one element. So the answer is

true

.

Example 2:

Input:

nums = [2,1,4]

Output:

true

Explanation:

There is only two pairs:

(2,1)

and

(1,4)

, and both of them contain numbers with different parity. So the answer is

true

.

Example 3:

Input:

nums = [4,3,1,6]

Output:

false

Explanation:

nums[1]

and

nums[2]

are both odd. So the answer is

false

.

Constraints:

1 <= nums.length <= 100

1 <= nums[i] <= 100

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: bool
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 isArraySpecial(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 isArraySpecial(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Special Array I
Difficulty: Easy
Tags: array
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 {boolean}
*/
var isArraySpecial = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 isArraySpecial(nums: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 IsArraySpecial(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Special Array I
* Difficulty: Easy
* Tags: array
*
* 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 isArraySpecial(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Special Array I
// Difficulty: Easy
// Tags: array
//
// 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 isArraySpecial(nums []int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Special Array I
// Difficulty: Easy
// Tags: array
```

```
//
// 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_array_special(nums: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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