Problem 3432: Count Partitions with Even Sum Difference

Problem Information

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

Problem Description

You are given an integer array

nums

of length

n

.

A

partition

is defined as an index

i

where

0 <= i < n - 1

, splitting the array into two

non-empty

subarrays such that:

Left subarray contains indices

[0, i]

.

Right subarray contains indices

[i + 1, n - 1]

.

Return the number of

partitions

where the

difference

between the

sum

of the left and right subarrays is

even

.

Example 1:

Input:

nums = [10,10,3,7,6]

Output:

4

Explanation:

The 4 partitions are:

[10]

,

[10, 3, 7, 6]

with a sum difference of

10 - 26 = -16

, which is even.

[10, 10]

,

[3, 7, 6]

with a sum difference of

20 - 16 = 4

, which is even.

[10, 10, 3]

,

[7, 6]

with a sum difference of

23 - 13 = 10

, which is even.

[10, 10, 3, 7]

,

[6]

with a sum difference of

30 - 6 = 24

, which is even.

Example 2:

Input:

nums = [1,2,2]

Output:

0

Explanation:

No partition results in an even sum difference.

Example 3:

Input:

nums = [2,4,6,8]

Output:

3

Explanation:

All partitions result in an even sum difference.

Constraints:

2 <= n == nums.length <= 100

1 <= nums[i] <= 100

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int countPartitions(int[] nums) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def countPartitions(self, nums):
```

```
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function countPartitions(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int CountPartitions(int[] nums) {

}
}
```

C:

```
int countPartitions(int* nums, int numsSize) {

}
```

Go:

```
func countPartitions(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun countPartitions(nums: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int countPartitions(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-partitions nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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:
int countPartitions(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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 int countPartitions(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Partitions with Even Sum Difference
Difficulty: Easy
Tags: array, math
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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 {number}
*/
var countPartitions = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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 countPartitions(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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 int CountPartitions(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Count Partitions with Even Sum Difference
* Difficulty: Easy
* Tags: array, math
*
* 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
```

```
*/

int countPartitions(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Count Partitions with Even Sum Difference
// Difficulty: Easy
// Tags: array, math
//
// 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 countPartitions(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countPartitions(nums: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Partitions with Even Sum Difference
// Difficulty: Easy
// Tags: array, math
```

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countPartitions(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-partitions nums)
(-> (listof exact-integer?) exact-integer?)
)
```