Problem 1442: Count Triplets That Can Form Two Arrays of Equal XOR

Problem Information

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

Problem Description

Given an array of integers

arr

.

We want to select three indices

i

,

j

and

k

where

(0 <= i < j <= k < arr.length)

.

Let's define

a

and

b

as follows:

a = arr[i] ^ arr[i + 1] ^ ... ^ arr[j - 1]

b = arr[j] ^ arr[j + 1] ^ ... ^ arr[k]

Note that

^

denotes the

bitwise-xor

operation.

Return

the number of triplets

(

i

,

j

and

k

) Where

a == b

.

Example 1:

Input:

arr = [2,3,1,6,7]

Output:

4

Explanation:

The triplets are (0,1,2), (0,2,2), (2,3,4) and (2,4,4)

Example 2:

Input:

arr = [1,1,1,1,1]

Output:

10

Constraints:

1 <= arr.length <= 300

1 <= arr[i] <= 10

8

Code Snippets

C++:

```
class Solution {
public:
int countTriplets(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public int countTriplets(int[] arr) {

}
}
```

Python3:

```
class Solution:
def countTriplets(self, arr: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countTriplets(arr: number[]): number {

};
```

C#:

```
public class Solution {
public int CountTriplets(int[] arr) {

}
}
```

C:

```
int countTriplets(int* arr, int arrSize) {

}
```

Go:

```
func countTriplets(arr []int) int {

}
```

Kotlin:

```
class Solution {
fun countTriplets(arr: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func countTriplets(_ arr: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_triplets(arr: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer}
def count_triplets(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function countTriplets($arr) {

}
}
```

Dart:

```
class Solution {
int countTriplets(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def countTriplets(arr: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_triplets(arr :: [integer]) :: integer
def count_triplets(arr) do

end
end
```

Erlang:

```
-spec count_triplets(Arr :: [integer()]) -> integer().
count_triplets(Arr) ->
.
```

Racket:

```
(define/contract (count-triplets arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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:
int countTriplets(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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 int countTriplets(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Triplets That Can Form Two Arrays of Equal XOR
Difficulty: Medium
Tags: array, math, 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 countTriplets(self, arr: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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[]} arr
* @return {number}
*/
var countTriplets = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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 countTriplets(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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 int CountTriplets(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Count Triplets That Can Form Two Arrays of Equal XOR
* Difficulty: Medium
* Tags: array, math, 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
*/

int countTriplets(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Count Triplets That Can Form Two Arrays of Equal XOR
// Difficulty: Medium
// Tags: array, math, 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
```

```go
func countTriplets(arr []int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun countTriplets(arr: IntArray): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func countTriplets(_ arr: [Int]) -> Int {

}
}
```

Rust Solution:

```rust
// Problem: Count Triplets That Can Form Two Arrays of Equal XOR
// Difficulty: Medium
// Tags: array, math, 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 count_triplets(arr: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```ruby
# @param {Integer[]} arr
# @return {Integer}
def count_triplets(arr)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function countTriplets($arr) {

}
}
```

Dart Solution:

```
class Solution {
int countTriplets(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def countTriplets(arr: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_triplets(arr :: [integer]) :: integer
def count_triplets(arr) do

end
end
```

Erlang Solution:

```
-spec count_triplets(Arr :: [integer()]) -> integer().
count_triplets(Arr) ->
.
```

Racket Solution:

```
(define/contract (count-triplets arr)
(-> (listof exact-integer?) exact-integer?)
)
```