Problem 3215: Count Triplets with Even XOR Set Bits II

Problem Information

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

Problem Description

Given three integer arrays

a

,

b

, and

c

, return the number of triplets

(a[i], b[j], c[k])

, such that the bitwise

XOR

between the elements of each triplet has an

even

number of

set bits

.

Example 1:

Input:

a = [1], b = [2], c = [3]

Output:

1

Explanation:

The only triplet is

(a[0], b[0], c[0])

and their

XOR

is:

1 XOR 2 XOR 3 = 00

2

.

Example 2:

Input:

a = [1,1], b = [2,3], c = [1,5]

Output:

4

Explanation:

Consider these four triplets:

(a[0], b[1], c[0])

:

1 XOR 3 XOR 1 = 011

2

(a[1], b[1], c[0])

:

1 XOR 3 XOR 1 = 011

2

(a[0], b[0], c[1])

:

1 XOR 2 XOR 5 = 110

2

(a[1], b[0], c[1])

:

1 XOR 2 XOR 5 = 110

2

Constraints:

1 <= a.length, b.length, c.length <= 10

5

0 <= a[i], b[i], c[i] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
long long tripletCount(vector<int>& a, vector<int>& b, vector<int>& c) {

}
};
```

Java:

```
class Solution {
public long tripletCount(int[] a, int[] b, int[] c) {

}
}
```

Python3:

```
class Solution:
def tripletCount(self, a: List[int], b: List[int], c: List[int]) -> int:
```

Python:

```
class Solution(object):
def tripletCount(self, a, b, c):
"""
:type a: List[int]
```

```
:type b: List[int]
:type c: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} a
* @param {number[]} b
* @param {number[]} c
* @return {number}
*/
var tripletCount = function(a, b, c) {

};
```

TypeScript:

```
function tripletCount(a: number[], b: number[], c: number[]): number {

};
```

C#:

```
public class Solution {
public long TripletCount(int[] a, int[] b, int[] c) {

}
}
```

C:

```
long long tripletCount(int* a, int aSize, int* b, int bSize, int* c, int
cSize) {

}
```

Go:

```
func tripletCount(a []int, b []int, c []int) int64 {
```

```
}
```

Kotlin:

```
class Solution {
fun tripletCount(a: IntArray, b: IntArray, c: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func tripletCount(_ a: [Int], _ b: [Int], _ c: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn triplet_count(a: Vec<i32>, b: Vec<i32>, c: Vec<i32>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} a
# @param {Integer[]} b
# @param {Integer[]} c
# @return {Integer}
def triplet_count(a, b, c)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $a
```

```
* @param Integer[] $b
* @param Integer[] $c
* @return Integer
*/
function tripletCount($a, $b, $c) {

}
}
```

Dart:

```
class Solution {
int tripletCount(List<int> a, List<int> b, List<int> c) {

}
}
```

Scala:

```
object Solution {
def tripletCount(a: Array[Int], b: Array[Int], c: Array[Int]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec triplet_count(a :: [integer], b :: [integer], c :: [integer]) ::
integer
def triplet_count(a, b, c) do

end
end
```

Erlang:

```
-spec triplet_count(A :: [integer()], B :: [integer()], C :: [integer()]) ->
integer().
triplet_count(A, B, C) ->
.
```

Racket:

```
(define/contract (triplet-count a b c)
(-> (listof exact-integer?) (listof exact-integer?) (listof exact-integer?)
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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:
long long tripletCount(vector<int>& a, vector<int>& b, vector<int>& c) {

}
};
```

Java Solution:

```
/**
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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 long tripletCount(int[] a, int[] b, int[] c) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Triplets with Even XOR Set Bits II
Difficulty: Medium
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 tripletCount(self, a: List[int], b: List[int], c: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def tripletCount(self, a, b, c):
"""
:type a: List[int]
:type b: List[int]
:type c: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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[]} a
* @param {number[]} b
* @param {number[]} c
* @return {number}
*/
var tripletCount = function(a, b, c) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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 tripletCount(a: number[], b: number[], c: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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 long TripletCount(int[] a, int[] b, int[] c) {

}
}
```

C Solution:

```
/*
* Problem: Count Triplets with Even XOR Set Bits II
* Difficulty: Medium
* 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
*/

long long tripletCount(int* a, int aSize, int* b, int bSize, int* c, int
cSize) {

}
```

Go Solution:

```
// Problem: Count Triplets with Even XOR Set Bits II
// Difficulty: Medium
// 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 tripletCount(a []int, b []int, c []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun tripletCount(a: IntArray, b: IntArray, c: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func tripletCount(_ a: [Int], _ b: [Int], _ c: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Triplets with Even XOR Set Bits II
// Difficulty: Medium
// 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 triplet_count(a: Vec<i32>, b: Vec<i32>, c: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} a
# @param {Integer[]} b
# @param {Integer[]} c
# @return {Integer}
def triplet_count(a, b, c)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $a
* @param Integer[] $b
* @param Integer[] $c
* @return Integer
*/
function tripletCount($a, $b, $c) {

}
}
```

Dart Solution:

```
class Solution {
int tripletCount(List<int> a, List<int> b, List<int> c) {

}
}
```

Scala Solution:

```
object Solution {
def tripletCount(a: Array[Int], b: Array[Int], c: Array[Int]): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec triplet_count(a :: [integer], b :: [integer], c :: [integer]) ::
integer
def triplet_count(a, b, c) do

end
end
```

Erlang Solution:

```
-spec triplet_count(A :: [integer()], B :: [integer()], C :: [integer()]) ->
integer().
triplet_count(A, B, C) ->
.
```

Racket Solution:

```
(define/contract (triplet-count a b c)
(-> (listof exact-integer?) (listof exact-integer?) (listof exact-integer?)
exact-integer?)
)
```