Problem 1835: Find XOR Sum of All Pairs Bitwise AND

Problem Information

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

Problem Description

The

XOR sum

of a list is the bitwise

XOR

of all its elements. If the list only contains one element, then its

XOR sum

will be equal to this element.

For example, the

XOR sum

of

[1,2,3,4]

is equal to

1 XOR 2 XOR 3 XOR 4 = 4

, and the

XOR sum

of

[3]

is equal to

3

.

You are given two

0-indexed

arrays

arr1

and

arr2

that consist only of non-negative integers.

Consider the list containing the result of

arr1[i] AND arr2[j]

(bitwise

AND

) for every

(i, j)

pair where

0 <= i < arr1.length

and

0 <= j < arr2.length

.

Return

the

XOR sum

of the aforementioned list

.

Example 1:

Input:

arr1 = [1,2,3], arr2 = [6,5]

Output:

0

Explanation:

The list = [1 AND 6, 1 AND 5, 2 AND 6, 2 AND 5, 3 AND 6, 3 AND 5] = [0,1,2,0,2,1]. The XOR sum = 0 XOR 1 XOR 2 XOR 0 XOR 2 XOR 1 = 0.

Example 2:

Input:

arr1 = [12], arr2 = [4]

Output:

4

Explanation:

The list = [12 AND 4] = [4]. The XOR sum = 4.

Constraints:

1 <= arr1.length, arr2.length <= 10

5

0 <= arr1[i], arr2[j] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int getXORSum(vector<int>& arr1, vector<int>& arr2) {

}
};
```

Java:

```
class Solution {
public int getXORSum(int[] arr1, int[] arr2) {

}
}
```

Python3:

```
class Solution:
def getXORSum(self, arr1: List[int], arr2: List[int]) -> int:
```

Python:

```
class Solution(object):
def getXORSum(self, arr1, arr2):
"""
:type arr1: List[int]
:type arr2: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} arr1
* @param {number[]} arr2
* @return {number}
*/
var getXORSum = function(arr1, arr2) {

};
```

TypeScript:

```
function getXORSum(arr1: number[], arr2: number[]): number {

};
```

C#:

```
public class Solution {
public int GetXORSum(int[] arr1, int[] arr2) {

}
}
```

C:

```
int getXORSum(int* arr1, int arr1Size, int* arr2, int arr2Size) {

}
```

Go:

```
func getXORSum(arr1 []int, arr2 []int) int {

}
```

Kotlin:

```
class Solution {
fun getXORSum(arr1: IntArray, arr2: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func getXORSum(_ arr1: [Int], _ arr2: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_xor_sum(arr1: Vec<i32>, arr2: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr1
# @param {Integer[]} arr2
# @return {Integer}
def get_xor_sum(arr1, arr2)
```

```
end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr1
* @param Integer[] $arr2
* @return Integer
*/
function getXORSum($arr1, $arr2) {

}
}
```

Dart:

```
class Solution {
int getXORSum(List<int> arr1, List<int> arr2) {

}
}
```

Scala:

```
object Solution {
def getXORSum(arr1: Array[Int], arr2: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_xor_sum(arr1 :: [integer], arr2 :: [integer]) :: integer
def get_xor_sum(arr1, arr2) do

end
end
```

Erlang:

```
-spec get_xor_sum(Arr1 :: [integer()], Arr2 :: [integer()]) -> integer().
get_xor_sum(Arr1, Arr2) ->
.
```

Racket:

```
(define/contract (get-xor-sum arr1 arr2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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 getXORSum(vector<int>& arr1, vector<int>& arr2) {

}
};
```

Java Solution:

```
/**
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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 getXORSum(int[] arr1, int[] arr2) {

}
}
```

Python3 Solution:

```
"""
Problem: Find XOR Sum of All Pairs Bitwise AND
Difficulty: Hard
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 getXORSum(self, arr1: List[int], arr2: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getXORSum(self, arr1, arr2):
"""
:type arr1: List[int]
:type arr2: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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[]} arr1
* @param {number[]} arr2
* @return {number}
*/
var getXORSum = function(arr1, arr2) {

};
```

TypeScript Solution:

```
/**
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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 getXORSum(arr1: number[], arr2: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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 GetXORSum(int[] arr1, int[] arr2) {

}
}
```

C Solution:

```
/*
* Problem: Find XOR Sum of All Pairs Bitwise AND
* Difficulty: Hard
* 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 getXORSum(int* arr1, int arr1Size, int* arr2, int arr2Size) {

}
```

Go Solution:

```
// Problem: Find XOR Sum of All Pairs Bitwise AND
// Difficulty: Hard
// 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 getXORSum(arr1 []int, arr2 []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getXORSum(arr1: IntArray, arr2: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func getXORSum(_ arr1: [Int], _ arr2: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find XOR Sum of All Pairs Bitwise AND
// Difficulty: Hard
// 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 get_xor_sum(arr1: Vec<i32>, arr2: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr1
# @param {Integer[]} arr2
# @return {Integer}
def get_xor_sum(arr1, arr2)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr1
* @param Integer[] $arr2
* @return Integer
*/
function getXORSum($arr1, $arr2) {

}
}
```

Dart Solution:

```
class Solution {
int getXORSum(List<int> arr1, List<int> arr2) {

}
}
```

Scala Solution:

```
object Solution {
def getXORSum(arr1: Array[Int], arr2: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_xor_sum(arr1 :: [integer], arr2 :: [integer]) :: integer
def get_xor_sum(arr1, arr2) do

end
end
```

Erlang Solution:

```
-spec get_xor_sum(Arr1 :: [integer()], Arr2 :: [integer()]) -> integer().
get_xor_sum(Arr1, Arr2) ->
.
```

Racket Solution:

```
(define/contract (get-xor-sum arr1 arr2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```