Problem 1803: Count Pairs With XOR in a Range

Problem Information

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

Problem Description

Given a

(0-indexed)

integer array

nums

and two integers

low

and

high

, return

the number of

nice pairs

.

A

nice pair

is a pair

(i, j)

where

0 <= i < j < nums.length

and

low <= (nums[i] XOR nums[j]) <= high

.

Example 1:

Input:

nums = [1,4,2,7], low = 2, high = 6

Output:

6

Explanation:

All nice pairs (i, j) are as follows: - (0, 1): nums[0] XOR nums[1] = 5 - (0, 2): nums[0] XOR nums[2] = 3 - (0, 3): nums[0] XOR nums[3] = 6 - (1, 2): nums[1] XOR nums[2] = 6 - (1, 3): nums[1] XOR nums[3] = 3 - (2, 3): nums[2] XOR nums[3] = 5

Example 2:

Input:

nums = [9,8,4,2,1], low = 5, high = 14

Output:

8

Explanation:

All nice pairs (i, j) are as follows: - (0, 2): nums[0] XOR nums[2] = 13 - (0, 3): nums[0] XOR nums[3] = 11 - (0, 4): nums[0] XOR nums[4] = 8 - (1, 2): nums[1] XOR nums[2] = 12 - (1, 3): nums[1] XOR nums[3] = 10 - (1, 4): nums[1] XOR nums[4] = 9 - (2, 3): nums[2] XOR nums[3] = 6 - (2, 4): nums[2] XOR nums[4] = 5

Constraints:

1 <= nums.length <= 2 * 10

4

1 <= nums[i] <= 2 * 10

4

1 <= low <= high <= 2 * 10

4

Code Snippets

C++:

```
class Solution {
public:
int countPairs(vector<int>& nums, int low, int high) {

}
};
```

Java:

```
class Solution {
public int countPairs(int[] nums, int low, int high) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @param {number} low
* @param {number} high
* @return {number}
*/
var countPairs = function(nums, low, high) {

};
```

TypeScript:

```
function countPairs(nums: number[], low: number, high: number): number {

};
```

C#:

```
public class Solution {
public int CountPairs(int[] nums, int low, int high) {
```

```
}
}
```

C:

```
int countPairs(int* nums, int numsSize, int low, int high){

}
```

Go:

```
func countPairs(nums []int, low int, high int) int {

}
```

Kotlin:

```
class Solution {
fun countPairs(nums: IntArray, low: Int, high: Int): Int {

}
}
```

Swift:

```
class Solution {
func countPairs(_ nums: [Int], _ low: Int, _ high: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_pairs(nums: Vec<i32>, low: i32, high: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} low
# @param {Integer} high
# @return {Integer}
def count_pairs(nums, low, high)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $low
* @param Integer $high
* @return Integer
*/
function countPairs($nums, $low, $high) {

}
}
```

Scala:

```
object Solution {
def countPairs(nums: Array[Int], low: Int, high: Int): Int = {

}
}
```

Racket:

```
(define/contract (count-pairs nums low high)
(-> (listof exact-integer?) exact-integer? exact-integer? exact-integer?)

)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
* 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:
int countPairs(vector<int>& nums, int low, int high) {

}
};
```

Java Solution:

```
/**
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
* 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 int countPairs(int[] nums, int low, int high) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Pairs With XOR in a Range
Difficulty: Hard
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 countPairs(self, nums: List[int], low: int, high: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
* 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
* @param {number} low
* @param {number} high
* @return {number}
*/
var countPairs = function(nums, low, high) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
* 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 countPairs(nums: number[], low: number, high: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
* 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 int CountPairs(int[] nums, int low, int high) {

}
}
```

C Solution:

```
/*
* Problem: Count Pairs With XOR in a Range
* Difficulty: Hard
```

```
* 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
*/



int countPairs(int* nums, int numsSize, int low, int high){

}
```

Go Solution:

```
// Problem: Count Pairs With XOR in a Range
// Difficulty: Hard
// 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 countPairs(nums []int, low int, high int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countPairs(nums: IntArray, low: Int, high: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countPairs(_ nums: [Int], _ low: Int, _ high: Int) -> Int {

}
```

```
}
```

Rust Solution:

```
// Problem: Count Pairs With XOR in a Range
// Difficulty: Hard
// 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 count_pairs(nums: Vec<i32>, low: i32, high: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} low
# @param {Integer} high
# @return {Integer}
def count_pairs(nums, low, high)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $low
* @param Integer $high
* @return Integer
*/
function countPairs($nums, $low, $high) {

}
```

```
}
```

Scala Solution:

```
object Solution {
def countPairs(nums: Array[Int], low: Int, high: Int): Int = {

}
}
```

Racket Solution:

```
(define/contract (count-pairs nums low high)
(-> (listof exact-integer?) exact-integer? exact-integer? exact-integer?)

)
```