Problem 3158: Find the XOR of Numbers Which Appear Twice

Problem Information

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

Problem Description

You are given an array

nums

, where each number in the array appears

either

once

or

twice.

Return the bitwise

XOR

of all the numbers that appear twice in the array, or 0 if no number appears twice.

Example 1:

Input:

nums = [1,2,1,3]

Output:

1

Explanation:

The only number that appears twice in

nums

is 1.

Example 2:

Input:

nums = [1,2,3]

Output:

0

Explanation:

No number appears twice in

nums

.

Example 3:

Input:

nums = [1,2,2,1]

Output:

3

Explanation:

Numbers 1 and 2 appeared twice.

1 XOR 2 == 3

.

Constraints:

1 <= nums.length <= 50

1 <= nums[i] <= 50

Each number in

nums

appears either once or twice.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (duplicate-numbers-xor nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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 duplicateNumbersXOR(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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 duplicateNumbersXOR(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the XOR of Numbers Which Appear Twice
Difficulty: Easy
Tags: array, 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 duplicateNumbersXOR(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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[]} nums
* @return {number}
*/
var duplicateNumbersXOR = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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 duplicateNumbersXOR(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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 DuplicateNumbersXOR(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find the XOR of Numbers Which Appear Twice
* Difficulty: Easy
* Tags: array, 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 duplicateNumbersXOR(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Find the XOR of Numbers Which Appear Twice
// Difficulty: Easy
// Tags: array, 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

func duplicateNumbersXOR(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Find the XOR of Numbers Which Appear Twice
// Difficulty: Easy
// Tags: array, 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 duplicate_numbers_xor(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (duplicate-numbers-xor nums)
(-> (listof exact-integer?) exact-integer?)
)
```