Problem 3132: Find the Integer Added to Array II

Problem Information

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

Problem Description

You are given two integer arrays

nums1

and

nums2

.

From

nums1

two elements have been removed, and all other elements have been increased (or decreased in the case of negative) by an integer, represented by the variable

x

.

As a result,

nums1

becomes

equal

to

nums2

. Two arrays are considered

equal

when they contain the same integers with the same frequencies.

Return the

minimum

possible integer

x

that achieves this equivalence.

Example 1:

Input:

nums1 = [4,20,16,12,8], nums2 = [14,18,10]

Output:

-2

Explanation:

After removing elements at indices

[0,4]

and adding -2,

nums1

becomes

[18,14,10]

.

Example 2:

Input:

nums1 = [3,5,5,3], nums2 = [7,7]

Output:

2

Explanation:

After removing elements at indices

[0,3]

and adding 2,

nums1

becomes

[7,7]

.

Constraints:

3 <= nums1.length <= 200

nums2.length == nums1.length - 2

0 <= nums1[i], nums2[i] <= 1000

The test cases are generated in a way that there is an integer

x

such that

nums1

can become equal to

nums2

by removing two elements and adding

x

to each element of

nums1

.

Code Snippets

C++:

```
class Solution {
public:
int minimumAddedInteger(vector<int>& nums1, vector<int>& nums2) {

}
};
```

Java:

```
class Solution {
public int minimumAddedInteger(int[] nums1, int[] nums2) {

}
}
```

Python3:

```
class Solution:
def minimumAddedInteger(self, nums1: List[int], nums2: List[int]) -> int:
```

Python:

```
class Solution(object):
def minimumAddedInteger(self, nums1, nums2):
"""
:type nums1: List[int]
:type nums2: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums1
* @param {number[]} nums2
* @return {number}
*/
var minimumAddedInteger = function(nums1, nums2) {

};
```

TypeScript:

```
function minimumAddedInteger(nums1: number[], nums2: number[]): number {

};
```

C#:

```
public class Solution {
public int MinimumAddedInteger(int[] nums1, int[] nums2) {

}
}
```

C:

```
int minimumAddedInteger(int* nums1, int nums1Size, int* nums2, int nums2Size)
{

}
```

Go:

```
func minimumAddedInteger(nums1 []int, nums2 []int) int {

}
```

Kotlin:

```
class Solution {
fun minimumAddedInteger(nums1: IntArray, nums2: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minimumAddedInteger(_ nums1: [Int], _ nums2: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_added_integer(nums1: Vec<i32>, nums2: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums1
# @param {Integer[]} nums2
# @return {Integer}
def minimum_added_integer(nums1, nums2)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums1
* @param Integer[] $nums2
* @return Integer
*/
function minimumAddedInteger($nums1, $nums2) {

}
}
```

Dart:

```
class Solution {
int minimumAddedInteger(List<int> nums1, List<int> nums2) {

}
}
```

Scala:

```
object Solution {
def minimumAddedInteger(nums1: Array[Int], nums2: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_added_integer(nums1 :: [integer], nums2 :: [integer]) ::
```

```
integer
def minimum_added_integer(nums1, nums2) do

end
end
```

Erlang:

```
-spec minimum_added_integer(Nums1 :: [integer()], Nums2 :: [integer()]) ->
integer().
minimum_added_integer(Nums1, Nums2) ->
.
```

Racket:

```
(define/contract (minimum-added-integer nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
* Tags: array, sort
*
* 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 minimumAddedInteger(vector<int>& nums1, vector<int>& nums2) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
* Tags: array, sort
*
* 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 minimumAddedInteger(int[] nums1, int[] nums2) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Integer Added to Array II
Difficulty: Medium
Tags: array, sort

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 minimumAddedInteger(self, nums1: List[int], nums2: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minimumAddedInteger(self, nums1, nums2):
"""
:type nums1: List[int]
:type nums2: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
* Tags: array, sort
*
* 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[]} nums1
* @param {number[]} nums2
* @return {number}
*/
var minimumAddedInteger = function(nums1, nums2) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
* Tags: array, sort
*
* 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 minimumAddedInteger(nums1: number[], nums2: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
```

```
* Tags: array, sort
*
* 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 MinimumAddedInteger(int[] nums1, int[] nums2) {

}
}
```

C Solution:

```
/*
* Problem: Find the Integer Added to Array II
* Difficulty: Medium
* Tags: array, sort
*
* 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 minimumAddedInteger(int* nums1, int nums1Size, int* nums2, int nums2Size)
{

}
```

Go Solution:

```
// Problem: Find the Integer Added to Array II
// Difficulty: Medium
// Tags: array, sort
//
// 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 minimumAddedInteger(nums1 []int, nums2 []int) int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun minimumAddedInteger(nums1: IntArray, nums2: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumAddedInteger(_ nums1: [Int], _ nums2: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Integer Added to Array II
// Difficulty: Medium
// Tags: array, sort
//
// 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 minimum_added_integer(nums1: Vec<i32>, nums2: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums1
# @param {Integer[]} nums2
# @return {Integer}
def minimum_added_integer(nums1, nums2)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums1
* @param Integer[] $nums2
* @return Integer
*/
function minimumAddedInteger($nums1, $nums2) {

}
}
```

Dart Solution:

```
class Solution {
int minimumAddedInteger(List<int> nums1, List<int> nums2) {

}
}
```

Scala Solution:

```
object Solution {
def minimumAddedInteger(nums1: Array[Int], nums2: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_added_integer(nums1 :: [integer], nums2 :: [integer]) ::
integer
def minimum_added_integer(nums1, nums2) do

end
```

```
end
```

Erlang Solution:

```
-spec minimum_added_integer(Nums1 :: [integer()], Nums2 :: [integer()]) ->
integer().
minimum_added_integer(Nums1, Nums2) ->
.
```

Racket Solution:

```
(define/contract (minimum-added-integer nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```