Problem 3400: Maximum Number of Matching Indices After Right Shifts

Problem Information

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

Problem Description

You are given two integer arrays,

nums1

and

nums2

, of the same length.

An index

i

is considered

matching

if

nums1[i] == nums2[i]

.

Return the

maximum

number of

matching

indices after performing any number of

right shifts

on

nums1

.

A

right shift

is defined as shifting the element at index

i

to index

(i + 1) % n

, for all indices.

Example 1:

Input:

nums1 = [3,1,2,3,1,2], nums2 = [1,2,3,1,2,3]

Output:

6

Explanation:

If we right shift

nums1

2 times, it becomes

[1, 2, 3, 1, 2, 3]

. Every index matches, so the output is 6.

Example 2:

Input:

nums1 = [1,4,2,5,3,1], nums2 = [2,3,1,2,4,6]

Output:

3

Explanation:

If we right shift

nums1

3 times, it becomes

[5, 3, 1, 1, 4, 2]

. Indices 1, 2, and 4 match, so the output is 3.

Constraints:

nums1.length == nums2.length

1 <= nums1.length, nums2.length <= 3000

1 <= nums1[i], nums2[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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


}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-matching-indices nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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:
int maximumMatchingIndices(vector<int>& nums1, vector<int>& nums2) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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 int maximumMatchingIndices(int[] nums1, int[] nums2) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Matching Indices After Right Shifts
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 maximumMatchingIndices(self, nums1: List[int], nums2: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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[]} nums1
* @param {number[]} nums2
* @return {number}
*/
var maximumMatchingIndices = function(nums1, nums2) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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 maximumMatchingIndices(nums1: number[], nums2: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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 int MaximumMatchingIndices(int[] nums1, int[] nums2) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Matching Indices After Right Shifts
* 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
*/

int maximumMatchingIndices(int* nums1, int nums1Size, int* nums2, int
nums2Size) {

}
```

Go Solution:

```
// Problem: Maximum Number of Matching Indices After Right Shifts
// 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 maximumMatchingIndices(nums1 []int, nums2 []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Matching Indices After Right Shifts
// 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 maximum_matching_indices(nums1: Vec<i32>, nums2: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-matching-indices nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```