Problem 2657: Find the Prefix Common Array of Two Arrays

Problem Information

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

Problem Description

You are given two

0-indexed

integer

permutations

A

and

B

of length

n

.

A

prefix common array

of

A

and

B

is an array

C

such that

C[i]

is equal to the count of numbers that are present at or before the index

i

in both

A

and

B

.

Return

the

prefix common array

of

A

and

B

.

A sequence of

n

integers is called a

permutation

if it contains all integers from

1

to

n

exactly once.

Example 1:

Input:

A = [1,3,2,4], B = [3,1,2,4]

Output:

[0,2,3,4]

Explanation:

At i = 0: no number is common, so C[0] = 0. At i = 1: 1 and 3 are common in A and B, so C[1] = 2. At i = 2: 1, 2, and 3 are common in A and B, so C[2] = 3. At i = 3: 1, 2, 3, and 4 are common in A and B, so C[3] = 4.

Example 2:

Input:

A = [2,3,1], B = [3,1,2]

Output:

[0,1,3]

Explanation:

At i = 0: no number is common, so C[0] = 0. At i = 1: only 3 is common in A and B, so C[1] = 1. At i = 2: 1, 2, and 3 are common in A and B, so C[2] = 3.

Constraints:

1 <= A.length == B.length == n <= 50

1 <= A[i], B[i] <= n

It is guaranteed that A and B are both a permutation of n integers.

Code Snippets

C++:

```
class Solution {
public:
vector<int> findThePrefixCommonArray(vector<int>& A, vector<int>& B) {

}
};
```

Java:

```
class Solution {
public int[] findThePrefixCommonArray(int[] A, int[] B) {
```

```
}
}
```

Python3:

```
class Solution:
def findThePrefixCommonArray(self, A: List[int], B: List[int]) -> List[int]:
```

Python:

```
class Solution(object):
def findThePrefixCommonArray(self, A, B):
"""
:type A: List[int]
:type B: List[int]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} A
* @param {number[]} B
* @return {number[]}
*/
var findThePrefixCommonArray = function(A, B) {

};
```

TypeScript:

```
function findThePrefixCommonArray(A: number[], B: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] FindThePrefixCommonArray(int[] A, int[] B) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findThePrefixCommonArray(int* A, int ASize, int* B, int BSize, int*
returnSize) {

}
```

Go:

```
func findThePrefixCommonArray(A []int, B []int) []int {

}
```

Kotlin:

```
class Solution {
fun findThePrefixCommonArray(A: IntArray, B: IntArray): IntArray {

}
}
```

Swift:

```
class Solution {
func findThePrefixCommonArray(_ A: [Int], _ B: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn find_the_prefix_common_array(a: Vec<i32>, b: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} a
# @param {Integer[]} b
# @return {Integer[]}
def find_the_prefix_common_array(a, b)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $A
* @param Integer[] $B
* @return Integer[]
*/
function findThePrefixCommonArray($A, $B) {

}
}
```

Dart:

```
class Solution {
List<int> findThePrefixCommonArray(List<int> A, List<int> B) {

}
}
```

Scala:

```
object Solution {
def findThePrefixCommonArray(A: Array[Int], B: Array[Int]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_the_prefix_common_array(a :: [integer], b :: [integer]) ::
[integer]
def find_the_prefix_common_array(a, b) do
```

```
end
end
```

Erlang:

```
-spec find_the_prefix_common_array(A :: [integer()], B :: [integer()]) ->
[integer()].
find_the_prefix_common_array(A, B) ->
.
```

Racket:

```
(define/contract (find-the-prefix-common-array A B)
(-> (listof exact-integer?) (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
* 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:
vector<int> findThePrefixCommonArray(vector<int>& A, vector<int>& B) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
* 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[] findThePrefixCommonArray(int[] A, int[] B) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Prefix Common Array of Two Arrays
Difficulty: Medium
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 findThePrefixCommonArray(self, A: List[int], B: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findThePrefixCommonArray(self, A, B):
"""
:type A: List[int]
:type B: List[int]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
* 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[]} A
* @param {number[]} B
* @return {number[]}
*/
var findThePrefixCommonArray = function(A, B) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
* 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 findThePrefixCommonArray(A: number[], B: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
```

```
* 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[] FindThePrefixCommonArray(int[] A, int[] B) {

}
}
```

C Solution:

```
/*
* Problem: Find the Prefix Common Array of Two Arrays
* Difficulty: Medium
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findThePrefixCommonArray(int* A, int ASize, int* B, int BSize, int*
returnSize) {

}
```

Go Solution:

```
// Problem: Find the Prefix Common Array of Two Arrays
// Difficulty: Medium
// 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 findThePrefixCommonArray(A []int, B []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun findThePrefixCommonArray(A: IntArray, B: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func findThePrefixCommonArray(_ A: [Int], _ B: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Find the Prefix Common Array of Two Arrays
// Difficulty: Medium
// 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 find_the_prefix_common_array(a: Vec<i32>, b: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} a
# @param {Integer[]} b
# @return {Integer[]}
def find_the_prefix_common_array(a, b)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $A
* @param Integer[] $B
* @return Integer[]
*/
function findThePrefixCommonArray($A, $B) {

}
}
```

Dart Solution:

```
class Solution {
List<int> findThePrefixCommonArray(List<int> A, List<int> B) {

}
}
```

Scala Solution:

```
object Solution {
def findThePrefixCommonArray(A: Array[Int], B: Array[Int]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_the_prefix_common_array(a :: [integer], b :: [integer]) ::
[integer]
```

```
def find_the_prefix_common_array(a, b) do

end
end
```

Erlang Solution:

```
-spec find_the_prefix_common_array(A :: [integer()], B :: [integer()]) ->
[integer()].
find_the_prefix_common_array(A, B) ->
.
```

Racket Solution:

```
(define/contract (find-the-prefix-common-array A B)
(-> (listof exact-integer?) (listof exact-integer?) (listof exact-integer?))
)
```