Problem 2670: Find the Distinct Difference Array

Problem Information

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

Problem Description

You are given a

0-indexed

array

nums

of length

n

.

The

distinct difference

array of

nums

is an array

diff

of length

n

such that

diff[i]

is equal to the number of distinct elements in the suffix

nums[i + 1, ..., n - 1]

subtracted from

the number of distinct elements in the prefix

nums[0, ..., i]

.

Return

the

distinct difference

array of

nums

.

Note that

nums[i, ..., j]

denotes the subarray of

nums

starting at index

i

and ending at index

j

inclusive. Particularly, if

i > j

then

nums[i, ..., j]

denotes an empty subarray.

Example 1:

Input:

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

Output:

[-3,-1,1,3,5]

Explanation:

For index i = 0, there is 1 element in the prefix and 4 distinct elements in the suffix. Thus, diff[0] = 1 - 4 = -3. For index i = 1, there are 2 distinct elements in the prefix and 3 distinct elements in the suffix. Thus, diff[1] = 2 - 3 = -1. For index i = 2, there are 3 distinct elements in the prefix and 2 distinct elements in the suffix. Thus, diff[2] = 3 - 2 = 1. For index i = 3, there are 4 distinct elements in the prefix and 1 distinct element in the suffix. Thus, diff[3] = 4 - 1 = 3. For index i = 4, there are 5 distinct elements in the prefix and no elements in the suffix. Thus, diff[4] = 5 - 0 = 5.

Example 2:

Input:

nums = [3,2,3,4,2]

Output:

[-2,-1,0,2,3]

Explanation:

For index i = 0, there is 1 element in the prefix and 3 distinct elements in the suffix. Thus, diff[0] = 1 - 3 = -2. For index i = 1, there are 2 distinct elements in the prefix and 3 distinct elements in the suffix. Thus, diff[1] = 2 - 3 = -1. For index i = 2, there are 2 distinct elements in the prefix and 2 distinct elements in the suffix. Thus, diff[2] = 2 - 2 = 0. For index i = 3, there are 3 distinct elements in the prefix and 1 distinct element in the suffix. Thus, diff[3] = 3 - 1 = 2. For index i = 4, there are 3 distinct elements in the prefix and no elements in the suffix. Thus, diff[4] = 3 - 0 = 3.

Constraints:

1 <= n == nums.length <= 50

1 <= nums[i] <= 50

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* distinctDifferenceArray(int* nums, int numsSize, int* returnSize) {

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun distinctDifferenceArray(nums: IntArray): IntArray {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer[]}
```

```
def distinct_difference_array(nums)

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (distinct-difference-array nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Distinct Difference Array
* 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:
vector<int> distinctDifferenceArray(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Distinct Difference Array
* 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[] distinctDifferenceArray(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Distinct Difference Array
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 distinctDifferenceArray(self, nums: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Distinct Difference Array
* 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 distinctDifferenceArray = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Distinct Difference Array
* 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 distinctDifferenceArray(nums: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find the Distinct Difference Array
* 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[] DistinctDifferenceArray(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find the Distinct Difference Array
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* distinctDifferenceArray(int* nums, int numsSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find the Distinct Difference Array
// 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 distinctDifferenceArray(nums []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun distinctDifferenceArray(nums: IntArray): IntArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Distinct Difference Array
// 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 distinct_difference_array(nums: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (distinct-difference-array nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```