Problem 1852: Distinct Numbers in Each Subarray

Problem Information

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

Problem Description

You are given an integer array

nums

of length

n

and an integer

k

. Your task is to find the number of

distinct

elements in

every

subarray of size

k

within

nums

.

Return an array

ans

such that

ans[i]

is the count of distinct elements in

nums[i..(i + k - 1)]

for each index

0 <= i < n - k

.

Example 1:

Input:

nums = [1,2,3,2,2,1,3], k = 3

Output:

[3,2,2,2,3]

Explanation:

The number of distinct elements in each subarray goes as follows: - nums[0..2] = [1,2,3] so ans[0] = 3 - nums[1..3] = [2,3,2] so ans[1] = 2 - nums[2..4] = [3,2,2] so ans[2] = 2 - nums[3..5] = [2,2,1] so ans[3] = 2 - nums[4..6] = [2,1,3] so ans[4] = 3

Example 2:

Input:

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

Output:

[1,2,3,4]

Explanation:

The number of distinct elements in each subarray goes as follows: - nums[0..3] = [1,1,1,1] so ans[0] = 1 - nums[1..4] = [1,1,1,2] so ans[1] = 2 - nums[2..5] = [1,1,2,3] so ans[2] = 3 - nums[3..6] = [1,2,3,4] so ans[3] = 4

Constraints:

1 <= k <= nums.length <= 10

5

1 <= nums[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
vector<int> distinctNumbers(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int[] distinctNumbers(int[] nums, int k) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function distinctNumbers(nums: number[], k: number): number[] {

};
```

C#:

```
public class Solution {
public int[] DistinctNumbers(int[] nums, int k) {

}
```

```
}
```

C:

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

}
```

Go:

```
func distinctNumbers(nums []int, k int) []int {

}
```

Kotlin:

```
class Solution {
fun distinctNumbers(nums: IntArray, k: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func distinctNumbers(_ nums: [Int], _ k: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn distinct_numbers(nums: Vec<i32>, k: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer[]}
def distinct_numbers(nums, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer[]
*/
function distinctNumbers($nums, $k) {

}
}
```

Scala:

```
object Solution {
def distinctNumbers(nums: Array[Int], k: Int): Array[Int] = {

}
}
```

Racket:

```
(define/contract (distinct-numbers nums k)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))

)
```

Solutions

C++ Solution:

```
/*
* Problem: Distinct Numbers in Each Subarray
* 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> distinctNumbers(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Distinct Numbers in Each Subarray
* 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[] distinctNumbers(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Distinct Numbers in Each Subarray
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 distinctNumbers(self, nums: List[int], k: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Distinct Numbers in Each Subarray
* 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[]} nums
* @param {number} k
* @return {number[]}
*/
var distinctNumbers = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Distinct Numbers in Each Subarray
* 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 distinctNumbers(nums: number[], k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Distinct Numbers in Each Subarray
* 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[] DistinctNumbers(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Distinct Numbers in Each Subarray
* 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* distinctNumbers(int* nums, int numsSize, int k, int* returnSize){

}
```

Go Solution:

```
// Problem: Distinct Numbers in Each Subarray
// 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 distinctNumbers(nums []int, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun distinctNumbers(nums: IntArray, k: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func distinctNumbers(_ nums: [Int], _ k: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Distinct Numbers in Each Subarray
// 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 distinct_numbers(nums: Vec<i32>, k: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Integer[]}
def distinct_numbers(nums, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return Integer[]
*/
function distinctNumbers($nums, $k) {

}
}
```

Scala Solution:

```
object Solution {
def distinctNumbers(nums: Array[Int], k: Int): Array[Int] = {

}
}
```

Racket Solution:

```
(define/contract (distinct-numbers nums k)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))

)
```