Problem 3729: Count Distinct Subarrays Divisible by K in Sorted Array

Problem Information

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

Problem Description

You are given an integer array

nums

sorted

in

non-descending

order and a positive integer

k

.

A

subarray

of

nums

is

good

if the sum of its elements is

divisible

by

k

.

Return an integer denoting the number of

distinct

good

subarrays of

nums

.

Subarrays are

distinct

if their sequences of values are. For example, there are 3

distinct

subarrays in

[1, 1, 1]

, namely

[1]

,

[1, 1]

, and

[1, 1, 1]

.

Example 1:

Input:

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

Output:

3

Explanation:

The good subarrays are

[1, 2]

,

[3]

, and

[1, 2, 3]

. For example,

[1, 2, 3]

is good because the sum of its elements is

1 + 2 + 3 = 6

, and

6 % k = 6 % 3 = 0

.

Example 2:

Input:

nums = [2,2,2,2,2,2], k = 6

Output:

2

Explanation:

The good subarrays are

[2, 2, 2]

and

[2, 2, 2, 2, 2, 2]

. For example,

[2, 2, 2]

is good because the sum of its elements is

2 + 2 + 2 = 6

, and

6 % k = 6 % 6 = 0

.

Note that

[2, 2, 2]

is counted only once.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

nums

is sorted in non-descending order.

1 <= k <= 10

9

Code Snippets

C++:

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

```
}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
long long numGoodSubarrays(int* nums, int numsSize, int k) {

}
```

Go:

```
func numGoodSubarrays(nums []int, k int) int64 {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn num_good_subarrays(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int numGoodSubarrays(List<int> nums, int k) {

}
}
```

Scala:

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

}
}
```

Elixir:

```
defmodule Solution do
@spec num_good_subarrays(nums :: [integer], k :: integer) :: integer
```

```
def num_good_subarrays(nums, k) do

end
end
```

Erlang:

```
-spec num_good_subarrays(Nums :: [integer()], K :: integer()) -> integer().
num_good_subarrays(Nums, K) ->
.
```

Racket:

```
(define/contract (num-good-subarrays nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
* Tags: array, hash, sort
*
* 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:
long long numGoodSubarrays(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
* Tags: array, hash, sort
*
* 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 long numGoodSubarrays(int[] nums, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Distinct Subarrays Divisible by K in Sorted Array
Difficulty: Hard
Tags: array, hash, sort

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 numGoodSubarrays(self, nums: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
* Tags: array, hash, sort
*
* 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 numGoodSubarrays = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
* Tags: array, hash, sort
*
* 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 numGoodSubarrays(nums: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
```

```
* Tags: array, hash, sort
*
* 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 long NumGoodSubarrays(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Count Distinct Subarrays Divisible by K in Sorted Array
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

long long numGoodSubarrays(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Count Distinct Subarrays Divisible by K in Sorted Array
// Difficulty: Hard
// Tags: array, hash, sort
//
// 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 numGoodSubarrays(nums []int, k int) int64 {
```

```
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Distinct Subarrays Divisible by K in Sorted Array
// Difficulty: Hard
// Tags: array, hash, sort
//
// 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 num_good_subarrays(nums: Vec<i32>, k: i32) -> i64 {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int numGoodSubarrays(List<int> nums, int k) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_good_subarrays(nums :: [integer], k :: integer) :: integer
def num_good_subarrays(nums, k) do

end
end
```

Erlang Solution:

```
-spec num_good_subarrays(Nums :: [integer()], K :: integer()) -> integer().
num_good_subarrays(Nums, K) ->
.
```

Racket Solution:

```
(define/contract (num-good-subarrays nums k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```