# Problem 2916: Subarrays Distinct Element Sum of Squares II

## Problem Information

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

## Problem Description

You are given a

0-indexed

integer array

nums

.

The

distinct count

of a subarray of

nums

is defined as:

Let

nums[i..j]

be a subarray of

nums

consisting of all the indices from

i

to

j

such that

0 <= i <= j < nums.length

. Then the number of distinct values in

nums[i..j]

is called the distinct count of

nums[i..j]

.

Return

the sum of the

squares

of

distinct counts

of all subarrays of

nums

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

A subarray is a contiguous

non-empty

sequence of elements within an array.

Example 1:

Input:

nums = [1,2,1]

Output:

15

Explanation:

Six possible subarrays are: [1]: 1 distinct value [2]: 1 distinct value [1]: 1 distinct value [1,2]: 2 distinct values [2,1]: 2 distinct values [1,2,1]: 2 distinct values The sum of the squares of the distinct counts in all subarrays is equal to 1

2

+ 1

2

+ 1

2

+ 2

2

+ 2

2

+ 2

2

+ 2

2

= 15.

Example 2:

Input:

nums = [2,2]

Output:

3

Explanation:

Three possible subarrays are: [2]: 1 distinct value [2]: 1 distinct value [2,2]: 1 distinct value
The sum of the squares of the distinct counts in all subarrays is equal to 1

2

+ 1

2

+ 1

2

= 3.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

5

## Code Snippets

**C++:**

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

    }
};
```

**Java:**

```java
class Solution {
    public int sumCounts(int[] nums) {

    }
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
int sumCounts(int* nums, int numsSize) {

}
```

**Go:**

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

}
```

## Kotlin:

```
class Solution {
fun sumCounts(nums: IntArray): Int {

}
}
```

## Swift:

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

}
}
```

## Rust:

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

}
}
```

## Ruby:

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

end
```

## PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
```

```
*/
function sumCounts($nums) {

}
}
```

**Dart:**

```dart
class Solution {
int sumCounts(List<int> nums) {

}
}
```

**Scala:**

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

}
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

```racket
(define/contract (sum-counts nums)
(-> (listof exact-integer?) exact-integer?)
)
```

# Solutions

**C++ Solution:**

```cpp
/*
 * Problem: Subarrays Distinct Element Sum of Squares II
 * Difficulty: Hard
 * Tags: array, tree, dp
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

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


}
};
```

**Java Solution:**

```java
/**
 * Problem: Subarrays Distinct Element Sum of Squares II
 * Difficulty: Hard
 * Tags: array, tree, dp
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

class Solution {
public int sumCounts(int[] nums) {


}
}
```

**Python3 Solution:**

```
"""
Problem: Subarrays Distinct Element Sum of Squares II
Difficulty: Hard
Tags: array, tree, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def sumCounts(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

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

**JavaScript Solution:**

```
/**
* Problem: Subarrays Distinct Element Sum of Squares II
* Difficulty: Hard
* Tags: array, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

```
    };
```

**TypeScript Solution:**

```typescript
/**
 * Problem: Subarrays Distinct Element Sum of Squares II
 * Difficulty: Hard
 * Tags: array, tree, dp
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

function sumCounts(nums: number[]): number {

};
```

**C# Solution:**

```csharp
/*
 * Problem: Subarrays Distinct Element Sum of Squares II
 * Difficulty: Hard
 * Tags: array, tree, dp
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

public class Solution {
public int SumCounts(int[] nums) {

}
}
```

**C Solution:**

```c
/*
 * Problem: Subarrays Distinct Element Sum of Squares II
 * Difficulty: Hard
```

```
* Tags: array, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int sumCounts(int* nums, int numsSize) {


}
```

**Go Solution:**

```go
// Problem: Subarrays Distinct Element Sum of Squares II
// Difficulty: Hard
// Tags: array, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func sumCounts(nums []int) int {


}
```

**Kotlin Solution:**

```kotlin
class Solution {
fun sumCounts(nums: IntArray): Int {


}
}
```

**Swift Solution:**

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


}
}
```

**Rust Solution:**

```rust
// Problem: Subarrays Distinct Element Sum of Squares II
// Difficulty: Hard
// Tags: array, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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


}
}
```

**Ruby Solution:**

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


end
```

**PHP Solution:**

```php
class Solution {

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


}
}
```

**Dart Solution:**

```dart
class Solution {
int sumCounts(List<int> nums) {
```

```
  }
}
```

## Scala Solution:

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


}
}
```

## Elixir Solution:

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

end
end
```

## Erlang Solution:

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

## Racket Solution:

```racket
(define/contract (sum-counts nums)
(-> (listof exact-integer?) exact-integer?)
)
```