Problem 3405: Count the Number of Arrays with K Matching Adjacent Elements

Problem Information

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

Problem Description

You are given three integers

n

,

m

,

k

. A

good array

arr

of size

n

is defined as follows:

Each element in

arr

is in the

inclusive

range

[1, m]

.

Exactly

k

indices

i

(where

1 <= i < n

) satisfy the condition

arr[i - 1] == arr[i]

.

Return the number of

good arrays

that can be formed.

Since the answer may be very large, return it modulo

10

9

+ 7

.

Example 1:

Input:

n = 3, m = 2, k = 1

Output:

4

Explanation:

There are 4 good arrays. They are

[1, 1, 2]

,

[1, 2, 2]

,

[2, 1, 1]

and

[2, 2, 1]

.

Hence, the answer is 4.

Example 2:

Input:

n = 4, m = 2, k = 2

Output:

6

Explanation:

The good arrays are

[1, 1, 1, 2]

,

[1, 1, 2, 2]

,

[1, 2, 2, 2]

,

[2, 1, 1, 1]

,

[2, 2, 1, 1]

and

[2, 2, 2, 1]

.

Hence, the answer is 6.

Example 3:

Input:

n = 5, m = 2, k = 0

Output:

2

Explanation:

The good arrays are

[1, 2, 1, 2, 1]

and

[2, 1, 2, 1, 2]

. Hence, the answer is 2.

Constraints:

1 <= n <= 10

5

1 <= m <= 10

5

0 <= k <= n - 1

Code Snippets

C++:

```
class Solution {
public:
int countGoodArrays(int n, int m, int k) {

}
};
```

Java:

```
class Solution {
public int countGoodArrays(int n, int m, int k) {

}
}
```

Python3:

```
class Solution:
def countGoodArrays(self, n: int, m: int, k: int) -> int:
```

Python:

```
class Solution(object):
def countGoodArrays(self, n, m, k):
"""
:type n: int
:type m: int
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} m
* @param {number} k
* @return {number}
*/
```

```
var countGoodArrays = function(n, m, k) {

};
```

TypeScript:

```
function countGoodArrays(n: number, m: number, k: number): number {

};
```

C#:

```
public class Solution {
public int CountGoodArrays(int n, int m, int k) {

}
}
```

C:

```
int countGoodArrays(int n, int m, int k) {

}
```

Go:

```
func countGoodArrays(n int, m int, k int) int {

}
```

Kotlin:

```
class Solution {
fun countGoodArrays(n: Int, m: Int, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func countGoodArrays(_ n: Int, _ m: Int, _ k: Int) -> Int {
}
}
```

Rust:

```
impl Solution {
pub fn count_good_arrays(n: i32, m: i32, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} m
# @param {Integer} k
# @return {Integer}
def count_good_arrays(n, m, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @param Integer $k
* @return Integer
*/
function countGoodArrays($n, $m, $k) {

}
}
```

Dart:

```
class Solution {
int countGoodArrays(int n, int m, int k) {
}
}
```

Scala:

```
object Solution {
def countGoodArrays(n: Int, m: Int, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_good_arrays(n :: integer, m :: integer, k :: integer) :: integer
def count_good_arrays(n, m, k) do

end
end
```

Erlang:

```
-spec count_good_arrays(N :: integer(), M :: integer(), K :: integer()) ->
integer().
count_good_arrays(N, M, K) ->
.
```

Racket:

```
(define/contract (count-good-arrays n m k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int countGoodArrays(int n, int m, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countGoodArrays(int n, int m, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Arrays with K Matching Adjacent Elements
Difficulty: Hard
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
```

```
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countGoodArrays(self, n: int, m: int, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countGoodArrays(self, n, m, k):
"""
:type n: int
:type m: int
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} m
* @param {number} k
* @return {number}
*/
var countGoodArrays = function(n, m, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countGoodArrays(n: number, m: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountGoodArrays(int n, int m, int k) {

}
}
```

C Solution:

```
/*
* Problem: Count the Number of Arrays with K Matching Adjacent Elements
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int countGoodArrays(int n, int m, int k) {

}
```

Go Solution:

```
// Problem: Count the Number of Arrays with K Matching Adjacent Elements
// Difficulty: Hard
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countGoodArrays(n int, m int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countGoodArrays(n: Int, m: Int, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countGoodArrays(_ n: Int, _ m: Int, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count the Number of Arrays with K Matching Adjacent Elements
// Difficulty: Hard
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_good_arrays(n: i32, m: i32, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} m
# @param {Integer} k
# @return {Integer}
def count_good_arrays(n, m, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @param Integer $k
* @return Integer
*/
function countGoodArrays($n, $m, $k) {

}
}
```

Dart Solution:

```
class Solution {
int countGoodArrays(int n, int m, int k) {
}
}
```

Scala Solution:

```
object Solution {
def countGoodArrays(n: Int, m: Int, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_good_arrays(n :: integer, m :: integer, k :: integer) :: integer
def count_good_arrays(n, m, k) do

end
end
```

Erlang Solution:

```
-spec count_good_arrays(N :: integer(), M :: integer(), K :: integer()) ->
integer().
count_good_arrays(N, M, K) ->
.
```

Racket Solution:

```
(define/contract (count-good-arrays n m k)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```