Problem 3339: Find the Number of K-Even Arrays

Problem Information

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

Problem Description

You are given three integers

n

,

m

, and

k

.

An array

arr

is called

k-even

if there are

exactly

k

indices such that, for each of these indices

i

(

0 <= i < n - 1

):

(arr[i] * arr[i + 1]) - arr[i] - arr[i + 1]

is

even

.

Return the number of possible

k-even

arrays of size

n

where all elements are in the range

[1, m]

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

n = 3, m = 4, k = 2

Output:

8

Explanation:

The 8 possible 2-even arrays are:

[2, 2, 2]

[2, 2, 4]

[2, 4, 2]

[2, 4, 4]

[4, 2, 2]

[4, 2, 4]

[4, 4, 2]

[4, 4, 4]

Example 2:

Input:

n = 5, m = 1, k = 0

Output:

1

Explanation:

The only 0-even array is

[1, 1, 1, 1, 1]

.

Example 3:

Input:

n = 7, m = 7, k = 5

Output:

5832

Constraints:

1 <= n <= 750

0 <= k <= n - 1

1 <= m <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def countOfArrays(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 countOfArrays = function(n, m, k) {

};
```

}

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int countOfArrays(int n, int m, int k) {

}
}
```

Scala:

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

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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 countOfArrays(int n, int m, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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 countOfArrays(int n, int m, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Number of K-Even Arrays
Difficulty: Medium
Tags: array, 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 countOfArrays(self, n: int, m: int, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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} n
* @param {number} m
* @param {number} k
* @return {number}
*/
var countOfArrays = function(n, m, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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 countOfArrays(n: number, m: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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 CountOfArrays(int n, int m, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Number of K-Even Arrays
* Difficulty: Medium
* Tags: array, 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 countOfArrays(int n, int m, int k) {

}
```

Go Solution:

```
// Problem: Find the Number of K-Even Arrays
// Difficulty: Medium
// Tags: array, 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 countOfArrays(n int, m int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Number of K-Even Arrays
// Difficulty: Medium
// Tags: array, 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 count_of_arrays(n: i32, m: i32, k: i32) -> i32 {
}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countOfArrays(int n, int m, int k) {

}
}
```

Scala Solution:

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

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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