Problem 2338: Count the Number of Ideal Arrays

Problem Information

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

Problem Description

You are given two integers

n

and

maxValue

, which are used to describe an

ideal

array.

A

0-indexed

integer array

arr

of length

n

is considered

ideal

if the following conditions hold:

Every

arr[i]

is a value from

1

to

maxValue

, for

0 <= i < n

.

Every

arr[i]

is divisible by

arr[i - 1]

, for

0 < i < n

.

Return

the number of

distinct

ideal arrays of length

n

. Since the answer may be very large, return it modulo

10

9

+ 7

.

Example 1:

Input:

n = 2, maxValue = 5

Output:

10

Explanation:

The following are the possible ideal arrays: - Arrays starting with the value 1 (5 arrays): [1,1], [1,2], [1,3], [1,4], [1,5] - Arrays starting with the value 2 (2 arrays): [2,2], [2,4] - Arrays starting with the value 3 (1 array): [3,3] - Arrays starting with the value 4 (1 array): [4,4] - Arrays starting with the value 5 (1 array): [5,5] There are a total of 5 + 2 + 1 + 1 + 1 = 10 distinct ideal arrays.

Example 2:

Input:

n = 5, maxValue = 3

Output:

11

Explanation:

The following are the possible ideal arrays: - Arrays starting with the value 1 (9 arrays): - With no other distinct values (1 array): [1,1,1,1,1] - With 2

nd

distinct value 2 (4 arrays): [1,1,1,1,2], [1,1,1,2,2], [1,1,2,2,2], [1,2,2,2,2] - With 2

nd

distinct value 3 (4 arrays): [1,1,1,1,3], [1,1,1,3,3], [1,1,3,3,3], [1,3,3,3,3] - Arrays starting with the value 2 (1 array): [2,2,2,2,2] - Arrays starting with the value 3 (1 array): [3,3,3,3,3] There are a total of 9 + 1 + 1 = 11 distinct ideal arrays.

Constraints:

2 <= n <= 10

4

1 <= maxValue <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int idealArrays(int n, int maxValue) {

}
};
```

Java:

```
class Solution {
public int idealArrays(int n, int maxValue) {

}
}
```

Python3:

```
class Solution:
def idealArrays(self, n: int, maxValue: int) -> int:
```

Python:

```
class Solution(object):
def idealArrays(self, n, maxValue):
"""
:type n: int
:type maxValue: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} maxValue
* @return {number}
*/
var idealArrays = function(n, maxValue) {

};
```

TypeScript:

```
function idealArrays(n: number, maxValue: number): number {

};
```

C#:

```
public class Solution {
public int IdealArrays(int n, int maxValue) {

}
}
```

C:

```
int idealArrays(int n, int maxValue) {

}
```

Go:

```
func idealArrays(n int, maxValue int) int {

}
```

Kotlin:

```
class Solution {
fun idealArrays(n: Int, maxValue: Int): Int {

}
}
```

Swift:

```
class Solution {
func idealArrays(_ n: Int, _ maxValue: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn ideal_arrays(n: i32, max_value: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} max_value
# @return {Integer}
def ideal_arrays(n, max_value)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $maxValue
* @return Integer
*/
function idealArrays($n, $maxValue) {

}
}
```

Dart:

```
class Solution {
int idealArrays(int n, int maxValue) {

}
}
```

Scala:

```
object Solution {
def idealArrays(n: Int, maxValue: Int): Int = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec ideal_arrays(n :: integer, max_value :: integer) :: integer
def ideal_arrays(n, max_value) do

end
end
```

Erlang:

```
-spec ideal_arrays(N :: integer(), MaxValue :: integer()) -> integer().
ideal_arrays(N, MaxValue) ->
.
```

Racket:

```
(define/contract (ideal-arrays n maxValue)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 idealArrays(int n, int maxValue) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 idealArrays(int n, int maxValue) {

}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Ideal Arrays
Difficulty: Hard
Tags: array, dp, math

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 idealArrays(self, n: int, maxValue: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def idealArrays(self, n, maxValue):
"""
:type n: int
:type maxValue: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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} maxValue
* @return {number}
*/
var idealArrays = function(n, maxValue) {

};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 idealArrays(n: number, maxValue: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 IdealArrays(int n, int maxValue) {

}
}
```

C Solution:

```
/*
* Problem: Count the Number of Ideal Arrays
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 idealArrays(int n, int maxValue) {

}
```

Go Solution:

```
// Problem: Count the Number of Ideal Arrays
// Difficulty: Hard
```

```
// Tags: array, dp, math
//
// 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 idealArrays(n int, maxValue int) int {

}
```

Kotlin Solution:

```
class Solution {
fun idealArrays(n: Int, maxValue: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func idealArrays(_ n: Int, _ maxValue: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count the Number of Ideal Arrays
// Difficulty: Hard
// Tags: array, dp, math
//
// 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 ideal_arrays(n: i32, max_value: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} max_value
# @return {Integer}
def ideal_arrays(n, max_value)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $maxValue
* @return Integer
*/
function idealArrays($n, $maxValue) {

}
}
```

Dart Solution:

```
class Solution {
int idealArrays(int n, int maxValue) {

}
}
```

Scala Solution:

```
object Solution {
def idealArrays(n: Int, maxValue: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec ideal_arrays(n :: integer, max_value :: integer) :: integer
def ideal_arrays(n, max_value) do

end
end
```

Erlang Solution:

```
-spec ideal_arrays(N :: integer(), MaxValue :: integer()) -> integer().
ideal_arrays(N, MaxValue) ->
.
```

Racket Solution:

```
(define/contract (ideal-arrays n maxValue)
(-> exact-integer? exact-integer? exact-integer?)
)
```