Problem 3130: Find All Possible Stable Binary Arrays II

Problem Information

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

Problem Description

You are given 3 positive integers

zero

,

one

, and

limit

.

A

binary array

arr

is called

stable

if:

The number of occurrences of 0 in

arr

is

exactly

zero

.

The number of occurrences of 1 in

arr

is

exactly

one

.

Each

subarray

of

arr

with a size greater than

limit

must contain

both

0 and 1.

Return the

total

number of

stable

binary arrays.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

zero = 1, one = 1, limit = 2

Output:

2

Explanation:

The two possible stable binary arrays are

[1,0]

and

[0,1]

.

Example 2:

Input:

zero = 1, one = 2, limit = 1

Output:

1

Explanation:

The only possible stable binary array is

[1,0,1]

.

Example 3:

Input:

zero = 3, one = 3, limit = 2

Output:

14

Explanation:

All the possible stable binary arrays are

[0,0,1,0,1,1]

,

[0,0,1,1,0,1]

,

[0,1,0,0,1,1]

,

[0,1,0,1,0,1]

,

[0,1,0,1,1,0]

,

[0,1,1,0,0,1]

,

[0,1,1,0,1,0]

,

[1,0,0,1,0,1]

,

[1,0,0,1,1,0]

,

[1,0,1,0,0,1]

,

[1,0,1,0,1,0]

,

[1,0,1,1,0,0]

,

[1,1,0,0,1,0]

, and

[1,1,0,1,0,0]

.

Constraints:

1 <= zero, one, limit <= 1000

Code Snippets

C++:

```
class Solution {
public:
int numberOfStableArrays(int zero, int one, int limit) {

}
};
```

Java:

```
class Solution {
public int numberOfStableArrays(int zero, int one, int limit) {

}
```

```
}
```

Python3:

```
class Solution:
def numberOfStableArrays(self, zero: int, one: int, limit: int) -> int:
```

Python:

```
class Solution(object):
def numberOfStableArrays(self, zero, one, limit):
"""
:type zero: int
:type one: int
:type limit: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} zero
* @param {number} one
* @param {number} limit
* @return {number}
*/
var numberOfStableArrays = function(zero, one, limit) {

};
```

TypeScript:

```
function numberOfStableArrays(zero: number, one: number, limit: number):
number {

};
```

C#:

```
public class Solution {
public int NumberOfStableArrays(int zero, int one, int limit) {
```

```
}
}
```

C:

```
int numberOfStableArrays(int zero, int one, int limit) {

}
```

Go:

```
func numberOfStableArrays(zero int, one int, limit int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfStableArrays(zero: Int, one: Int, limit: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfStableArrays(_ zero: Int, _ one: Int, _ limit: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_stable_arrays(zero: i32, one: i32, limit: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} zero
# @param {Integer} one
# @param {Integer} limit
# @return {Integer}
def number_of_stable_arrays(zero, one, limit)

end
```

PHP:

```
class Solution {

/**
* @param Integer $zero
* @param Integer $one
* @param Integer $limit
* @return Integer
*/
function numberOfStableArrays($zero, $one, $limit) {

}
}
```

Dart:

```
class Solution {
int numberOfStableArrays(int zero, int one, int limit) {

}
}
```

Scala:

```
object Solution {
def numberOfStableArrays(zero: Int, one: Int, limit: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_stable_arrays(zero :: integer, one :: integer, limit ::
```

```
integer) :: integer
def number_of_stable_arrays(zero, one, limit) do

end
end
```

Erlang:

```
-spec number_of_stable_arrays(Zero :: integer(), One :: integer(), Limit ::
integer()) -> integer().
number_of_stable_arrays(Zero, One, Limit) ->
.
```

Racket:

```
(define/contract (number-of-stable-arrays zero one limit)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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 numberOfStableArrays(int zero, int one, int limit) {

}
};
```

Java Solution:

```
/**
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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 numberOfStableArrays(int zero, int one, int limit) {

}
}
```

Python3 Solution:

```
"""
Problem: Find All Possible Stable Binary Arrays II
Difficulty: Hard
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 numberOfStableArrays(self, zero: int, one: int, limit: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfStableArrays(self, zero, one, limit):
"""
:type zero: int
:type one: int
:type limit: int
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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} zero
* @param {number} one
* @param {number} limit
* @return {number}
*/
var numberOfStableArrays = function(zero, one, limit) {

};
```

TypeScript Solution:

```
/**
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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 numberOfStableArrays(zero: number, one: number, limit: number):
number {

};
```

C# Solution:

```
/*
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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 NumberOfStableArrays(int zero, int one, int limit) {

}
}
```

C Solution:

```
/*
* Problem: Find All Possible Stable Binary Arrays II
* Difficulty: Hard
* 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 numberOfStableArrays(int zero, int one, int limit) {

}
```

Go Solution:

```
// Problem: Find All Possible Stable Binary Arrays II
// Difficulty: Hard
// 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 numberOfStableArrays(zero int, one int, limit int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfStableArrays(zero: Int, one: Int, limit: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfStableArrays(_ zero: Int, _ one: Int, _ limit: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find All Possible Stable Binary Arrays II
// Difficulty: Hard
// 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 number_of_stable_arrays(zero: i32, one: i32, limit: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} zero
# @param {Integer} one
```

```
# @param {Integer} limit
# @return {Integer}
def number_of_stable_arrays(zero, one, limit)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $zero
* @param Integer $one
* @param Integer $limit
* @return Integer
*/
function numberOfStableArrays($zero, $one, $limit) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfStableArrays(int zero, int one, int limit) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfStableArrays(zero: Int, one: Int, limit: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_stable_arrays(zero :: integer, one :: integer, limit ::
```

```
integer) :: integer
def number_of_stable_arrays(zero, one, limit) do

end
end
```

Erlang Solution:

```
-spec number_of_stable_arrays(Zero :: integer(), One :: integer(), Limit ::
integer()) -> integer().
number_of_stable_arrays(Zero, One, Limit) ->
.
```

Racket Solution:

```
(define/contract (number-of-stable-arrays zero one limit)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```