Problem 1955: Count Number of Special Subsequences

Problem Information

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

Problem Description

A sequence is

special

if it consists of a

positive

number of

0

s, followed by a

positive

number of

1

s, then a

positive

number of

2

s.

For example,

[0,1,2]

and

[0,0,1,1,1,2]

are special.

In contrast,

[2,1,0]

,

[1]

, and

[0,1,2,0]

are not special.

Given an array

nums

(consisting of

only

integers

0

,

1

, and

2

), return

the

number of different subsequences

that are special

. Since the answer may be very large,

return it modulo

10

9

+ 7

.

A

subsequence

of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements. Two subsequences are

different

if the

set of indices

chosen are different.

Example 1:

Input:

nums = [0,1,2,2]

Output:

3

Explanation:

The special subsequences are bolded [

0

,

1

,

2

,2], [

0

,

1

,2,

2

], and [

0

,

1

,

2

,

2

].

Example 2:

Input:

nums = [2,2,0,0]

Output:

0

Explanation:

There are no special subsequences in [2,2,0,0].

Example 3:

Input:

nums = [0,1,2,0,1,2]

Output:

7

Explanation:

The special subsequences are bolded: - [

0

,

1

,

2

,0,1,2] - [

0

,

1

,2,0,1,

2

] - [

0

,

1

,

2

,0,1,

2

] - [

0

,

1

,2,0,

1

,

2

] - [

0

,1,2,

0

,

1

,

2

] - [

0

,1,2,0,

1

,

2

] - [0,1,2,

0

,

1

,

2

]

Constraints:

1 <= nums.length <= 10

5

0 <= nums[i] <= 2

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int countSpecialSubsequences(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-special-subsequences nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Number of Special Subsequences
* 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 countSpecialSubsequences(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Count Number of Special Subsequences
* 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 countSpecialSubsequences(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Number of Special Subsequences
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 countSpecialSubsequences(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Count Number of Special Subsequences
* 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[]} nums
* @return {number}
*/
var countSpecialSubsequences = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Number of Special Subsequences
* 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 countSpecialSubsequences(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Number of Special Subsequences
* 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 CountSpecialSubsequences(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Count Number of Special Subsequences
* 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 countSpecialSubsequences(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Count Number of Special Subsequences
// 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 countSpecialSubsequences(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Number of Special Subsequences
// 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 count_special_subsequences(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countSpecialSubsequences(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-special-subsequences nums)
(-> (listof exact-integer?) exact-integer?)
)
```