Problem 3686: Number of Stable Subsequences

Problem Information

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

Problem Description

You are given an integer array

nums

.

A

subsequence

is

stable

if it does not contain

three consecutive

elements with the

same

parity when the subsequence is read

in order

(i.e., consecutive

inside the subsequence

).

Return the number of stable subsequences.

Since the answer may be too large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [1,3,5]

Output:

6

Explanation:

Stable subsequences are

[1]

,

[3]

,

[5]

,

[1, 3]

,

[1, 5]

, and

[3, 5]

.

Subsequence

[1, 3, 5]

is not stable because it contains three consecutive odd numbers. Thus, the answer is 6.

Example 2:

Input:

nums =

[2,3,4,2]

Output:

14

Explanation:

The only subsequence that is not stable is

[2, 4, 2]

, which contains three consecutive even numbers.

All other subsequences are stable. Thus, the answer is 14.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Number of Stable 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 countStableSubsequences(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Stable 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 countStableSubsequences(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Stable 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 countStableSubsequences(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Stable 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 countStableSubsequences = function(nums) {
};
```

TypeScript Solution:

```
/**
* Problem: Number of Stable 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 countStableSubsequences(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Stable 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 CountStableSubsequences(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Number of Stable 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 countStableSubsequences(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Number of Stable 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 countStableSubsequences(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Stable 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_stable_subsequences(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countStableSubsequences(List<int> nums) {
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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