Problem 903: Valid Permutations for DI Sequence

Problem Information

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

Problem Description

You are given a string

s

of length

n

where

s[i]

is either:

'D'

means decreasing, or

'I'

means increasing.

A permutation

perm

of

n + 1

integers of all the integers in the range

[0, n]

is called a

valid permutation

if for all valid

i

:

If

s[i] == 'D'

, then

perm[i] > perm[i + 1]

, and

If

s[i] == 'I'

, then

perm[i] < perm[i + 1]

.

Return

the number of

valid permutations

perm

. Since the answer may be large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

s = "DID"

Output:

5

Explanation:

The 5 valid permutations of (0, 1, 2, 3) are: (1, 0, 3, 2) (2, 0, 3, 1) (2, 1, 3, 0) (3, 0, 2, 1) (3, 1, 2, 0)

Example 2:

Input:

s = "D"

Output:

1

Constraints:

n == s.length

1 <= n <= 200

s[i]

is either

'I'

or

'D'

.

Code Snippets

C++:

```
class Solution {
public:
int numPermsDISequence(string s) {

}
};
```

Java:

```
class Solution {
public int numPermsDISequence(String s) {
```

```
}
}
```

Python3:

```
class Solution:
def numPermsDISequence(self, s: str) -> int:
```

Python:

```
class Solution(object):
def numPermsDISequence(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var numPermsDISequence = function(s) {

};
```

TypeScript:

```
function numPermsDISequence(s: string): number {

};
```

C#:

```
public class Solution {
public int NumPermsDISequence(string s) {

}
}
```

C:

```
int numPermsDISequence(char* s) {

}
```

Go:

```
func numPermsDISequence(s string) int {

}
```

Kotlin:

```
class Solution {
fun numPermsDISequence(s: String): Int {

}
}
```

Swift:

```
class Solution {
func numPermsDISequence(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_perms_di_sequence(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def num_perms_di_sequence(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function numPermsDISequence($s) {

}
}
```

Dart:

```
class Solution {
int numPermsDISequence(String s) {

}
}
```

Scala:

```
object Solution {
def numPermsDISequence(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_perms_di_sequence(s :: String.t) :: integer
def num_perms_di_sequence(s) do

end
end
```

Erlang:

```
-spec num_perms_di_sequence(S :: unicode:unicode_binary()) -> integer().
num_perms_di_sequence(S) ->
.
```

Racket:

```
(define/contract (num-perms-di-sequence s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 numPermsDISequence(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 numPermsDISequence(String s) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Valid Permutations for DI Sequence
Difficulty: Hard
Tags: array, string, 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 numPermsDISequence(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numPermsDISequence(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 {string} s
* @return {number}
*/
var numPermsDISequence = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 numPermsDISequence(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 NumPermsDISequence(string s) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Valid Permutations for DI Sequence
* Difficulty: Hard
* Tags: array, string, 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 numPermsDISequence(char* s) {

}
```

Go Solution:

```
// Problem: Valid Permutations for DI Sequence
// Difficulty: Hard
// Tags: array, string, 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 numPermsDISequence(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numPermsDISequence(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numPermsDISequence(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Valid Permutations for DI Sequence
// Difficulty: Hard
// Tags: array, string, 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 num_perms_di_sequence(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def num_perms_di_sequence(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function numPermsDISequence($s) {

}
}
```

Dart Solution:

```
class Solution {
int numPermsDISequence(String s) {

}
}
```

Scala Solution:

```
object Solution {
def numPermsDISequence(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_perms_di_sequence(s :: String.t) :: integer
def num_perms_di_sequence(s) do

end
end
```

Erlang Solution:

```
-spec num_perms_di_sequence(S :: unicode:unicode_binary()) -> integer().
num_perms_di_sequence(S) ->
.
```

Racket Solution:

```
(define/contract (num-perms-di-sequence s)
(-> string? exact-integer?)
)
```