Problem 2147: Number of Ways to Divide a Long Corridor

Problem Information

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

Problem Description

Along a long library corridor, there is a line of seats and decorative plants. You are given a

0-indexed

string

corridor

of length

n

consisting of letters

'S'

and

'P'

where each

'S'

represents a seat and each

'P'

represents a plant.

One room divider has

already

been installed to the left of index

0

, and

another

to the right of index

n - 1

. Additional room dividers can be installed. For each position between indices

i - 1

and

i

(

1 <= i <= n - 1

), at most one divider can be installed.

Divide the corridor into non-overlapping sections, where each section has

exactly two seats

with any number of plants. There may be multiple ways to perform the division. Two ways are

different

if there is a position with a room divider installed in the first way but not in the second way.

Return

the number of ways to divide the corridor

. Since the answer may be very large, return it

modulo

10

9

+ 7

. If there is no way, return

0

.

Example 1:

Input:

corridor = "SSPPSPS"

Output:

3

Explanation:

There are 3 different ways to divide the corridor. The black bars in the above image indicate the two room dividers already installed. Note that in each of the ways,

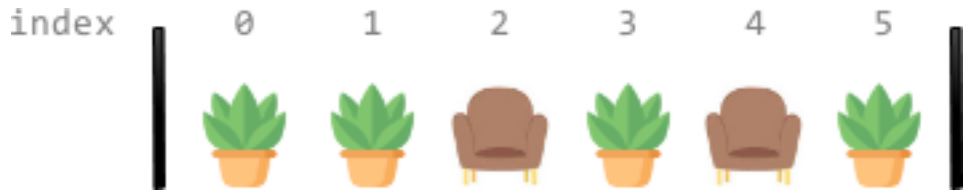
each

section has exactly

two

seats.

Example 2:



Input:

corridor = "PPSPSP"

Output:

1

Explanation:

There is only 1 way to divide the corridor, by not installing any additional dividers. Installing any would create some section that does not have exactly two seats.

Example 3:

Input:

corridor = "S"

Output:

0

Explanation:

There is no way to divide the corridor because there will always be a section that does not have exactly two seats.

Constraints:

n == corridor.length

1 <= n <= 10

5

corridor[i]

is either

'S'

or

'P'

.

Code Snippets

C++:

```
class Solution {
public:
int numberOfWays(string corridor) {

}
};
```

Java:

```
class Solution {
public int numberOfWays(String corridor) {

}
}
```

Python3:

```
class Solution:
def numberOfWays(self, corridor: str) -> int:
```

Python:

```
class Solution(object):
def numberOfWays(self, corridor):
"""
:type corridor: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} corridor
```

```
 * @return {number}
 */
var numberOfWays = function(corridor) {

};
```

TypeScript:

```
function numberOfWays(corridor: string): number {

};
```

C#:

```
public class Solution {
public int NumberOfWays(string corridor) {

}
}
```

C:

```
int numberOfWays(char* corridor) {

}
```

Go:

```
func numberOfWays(corridor string) int {

}
```

Kotlin:

```
class Solution {
fun numberOfWays(corridor: String): Int {

}
}
```

Swift:

```
class Solution {
func numberOfWays(_ corridor: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_ways(corridor: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} corridor
# @return {Integer}
def number_of_ways(corridor)

end
```

PHP:

```
class Solution {

/**
* @param String $corridor
* @return Integer
*/
function numberOfWays($corridor) {

}
}
```

Dart:

```
class Solution {
int numberOfWays(String corridor) {

}
}
```

Scala:

```
object Solution {
def numberOfWays(corridor: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_ways(corridor :: String.t) :: integer
def number_of_ways(corridor) do

end
end
```

Erlang:

```
-spec number_of_ways(Corridor :: unicode:unicode_binary()) -> integer().
number_of_ways(Corridor) ->
.
```

Racket:

```
(define/contract (number-of-ways corridor)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int numberOfWays(string corridor) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int numberOfWays(String corridor) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Ways to Divide a Long Corridor
Difficulty: Hard
Tags: string, dp, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def numberOfWays(self, corridor: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def numberOfWays(self, corridor):
"""
:type corridor: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} corridor
* @return {number}
*/
var numberOfWays = function(corridor) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function numberOfWays(corridor: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* 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 NumberOfWays(string corridor) {

}
}
```

C Solution:

```
/*
* Problem: Number of Ways to Divide a Long Corridor
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int numberOfWays(char* corridor) {

}
```

Go Solution:

```
// Problem: Number of Ways to Divide a Long Corridor
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func numberOfWays(corridor string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfWays(corridor: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfWays(_ corridor: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Ways to Divide a Long Corridor
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// 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_ways(corridor: String) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {String} corridor
# @return {Integer}
def number_of_ways(corridor)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $corridor
* @return Integer
*/
function numberOfWays($corridor) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfWays(String corridor) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfWays(corridor: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_ways(corridor :: String.t) :: integer
def number_of_ways(corridor) do

end
end
```

Erlang Solution:

```
-spec number_of_ways(Corridor :: unicode:unicode_binary()) -> integer().
number_of_ways(Corridor) ->
.
```

Racket Solution:

```
(define/contract (number-of-ways corridor)
(-> string? exact-integer?)
)
```