Problem 2833: Furthest Point From Origin

Problem Information

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

Problem Description

You are given a string

moves

of length

n

consisting only of characters

'L'

,

'R'

, and

'_'

. The string represents your movement on a number line starting from the origin

0

.

In the

i

th

move, you can choose one of the following directions:

move to the left if

moves[i] = 'L'

or

moves[i] = '_'

move to the right if

moves[i] = 'R'

or

moves[i] = '_'

Return

the

distance from the origin

of the

furthest

point you can get to after

n

moves

.

Example 1:

Input:

moves = "L_RL__R"

Output:

3

Explanation:

The furthest point we can reach from the origin 0 is point -3 through the following sequence of moves "LLRLLLR".

Example 2:

Input:

moves = "_R__LL_"

Output:

5

Explanation:

The furthest point we can reach from the origin 0 is point -5 through the following sequence of moves "LRLLLLL".

Example 3:

Input:

moves = "_______"

Output:

7

Explanation:

The furthest point we can reach from the origin 0 is point 7 through the following sequence of moves "RRRRRRR".

Constraints:

1 <= moves.length == n <= 50

moves

consists only of characters

'L'

,

'R'

and

'_'

.

Code Snippets

C++:

```
class Solution {
public:
int furthestDistanceFromOrigin(string moves) {

}
};
```

Java:

```
class Solution {
public int furthestDistanceFromOrigin(String moves) {

}
}
```

Python3:

```
class Solution:
def furthestDistanceFromOrigin(self, moves: str) -> int:
```

Python:

```
class Solution(object):
def furthestDistanceFromOrigin(self, moves):
"""
:type moves: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} moves
* @return {number}
*/
var furthestDistanceFromOrigin = function(moves) {

};
```

TypeScript:

```
function furthestDistanceFromOrigin(moves: string): number {

};
```

C#:

```
public class Solution {
public int FurthestDistanceFromOrigin(string moves) {
}
}
```

C:

```
int furthestDistanceFromOrigin(char* moves) {

}
```

Go:

```
func furthestDistanceFromOrigin(moves string) int {

}
```

Kotlin:

```
class Solution {
fun furthestDistanceFromOrigin(moves: String): Int {

}
}
```

Swift:

```
class Solution {
func furthestDistanceFromOrigin(_ moves: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn furthest_distance_from_origin(moves: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} moves
# @return {Integer}
def furthest_distance_from_origin(moves)

end
```

PHP:

```
class Solution {

/**
* @param String $moves
* @return Integer
*/
function furthestDistanceFromOrigin($moves) {

}
}
```

Dart:

```
class Solution {
int furthestDistanceFromOrigin(String moves) {

}
}
```

Scala:

```
object Solution {
def furthestDistanceFromOrigin(moves: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec furthest_distance_from_origin(moves :: String.t) :: integer
def furthest_distance_from_origin(moves) do

end
end
```

Erlang:

```
-spec furthest_distance_from_origin(Moves :: unicode:unicode_binary()) ->
integer().
furthest_distance_from_origin(Moves) ->
.
```

Racket:

```
(define/contract (furthest-distance-from-origin moves)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int furthestDistanceFromOrigin(string moves) {

}
};
```

Java Solution:

```
/**
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int furthestDistanceFromOrigin(String moves) {

}
}
```

Python3 Solution:

```
"""
Problem: Furthest Point From Origin
Difficulty: Easy
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def furthestDistanceFromOrigin(self, moves: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def furthestDistanceFromOrigin(self, moves):
"""
:type moves: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} moves
* @return {number}
*/
var furthestDistanceFromOrigin = function(moves) {

};
```

TypeScript Solution:

```
/**
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function furthestDistanceFromOrigin(moves: string): number {

};
```

C# Solution:

```
/*
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FurthestDistanceFromOrigin(string moves) {

}
}
```

C Solution:

```
/*
* Problem: Furthest Point From Origin
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int furthestDistanceFromOrigin(char* moves) {

}
```

Go Solution:

```
// Problem: Furthest Point From Origin
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func furthestDistanceFromOrigin(moves string) int {

}
```

Kotlin Solution:

```
class Solution {
fun furthestDistanceFromOrigin(moves: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func furthestDistanceFromOrigin(_ moves: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Furthest Point From Origin
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn furthest_distance_from_origin(moves: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} moves
# @return {Integer}
def furthest_distance_from_origin(moves)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $moves
* @return Integer
*/
function furthestDistanceFromOrigin($moves) {

}
}
```

Dart Solution:

```
class Solution {
int furthestDistanceFromOrigin(String moves) {

}
}
```

Scala Solution:

```
object Solution {
def furthestDistanceFromOrigin(moves: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec furthest_distance_from_origin(moves :: String.t) :: integer
def furthest_distance_from_origin(moves) do

end
end
```

Erlang Solution:

```
-spec furthest_distance_from_origin(Moves :: unicode:unicode_binary()) ->
integer().
furthest_distance_from_origin(Moves) ->
.
```

Racket Solution:

```
(define/contract (furthest-distance-from-origin moves)
(-> string? exact-integer?)
)
```