Problem 2211: Count Collisions on a Road

Problem Information

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

Problem Description

There are

n

cars on an infinitely long road. The cars are numbered from

0

to

n - 1

from left to right and each car is present at a

unique

point.

You are given a

0-indexed

string

directions

of length

n

.

directions[i]

can be either

'L'

,

'R'

, or

'S'

denoting whether the

i

th

car is moving towards the

left

, towards the

right

, or

staying

at its current point respectively. Each moving car has the

same speed

.

The number of collisions can be calculated as follows:

When two cars moving in

opposite

directions collide with each other, the number of collisions increases by

2

.

When a moving car collides with a stationary car, the number of collisions increases by

1

.

After a collision, the cars involved can no longer move and will stay at the point where they collided. Other than that, cars cannot change their state or direction of motion.

Return

the

total number of collisions

that will happen on the road

.

Example 1:

Input:

directions = "RLRSLL"

Output:

5

Explanation:

The collisions that will happen on the road are: - Cars 0 and 1 will collide with each other. Since they are moving in opposite directions, the number of collisions becomes 0 + 2 = 2. - Cars 2 and 3 will collide with each other. Since car 3 is stationary, the number of collisions becomes 2 + 1 = 3. - Cars 3 and 4 will collide with each other. Since car 3 is stationary, the number of collisions becomes 3 + 1 = 4. - Cars 4 and 5 will collide with each other. After car 4 collides with car 3, it will stay at the point of collision and get hit by car 5. The number of collisions becomes 4 + 1 = 5. Thus, the total number of collisions that will happen on the road is 5.

Example 2:

Input:

directions = "LLRR"

Output:

0

Explanation:

No cars will collide with each other. Thus, the total number of collisions that will happen on the road is 0.

Constraints:

1 <= directions.length <= 10

5

directions[i]

is either

'L'

,

'R'

, or

'S'

.

Code Snippets

C++:

```
class Solution {
public:
int countCollisions(string directions) {

}
};
```

Java:

```
class Solution {
public int countCollisions(String directions) {

}
}
```

Python3:

```
class Solution:
def countCollisions(self, directions: str) -> int:
```

Python:

```
class Solution(object):
def countCollisions(self, directions):
"""
:type directions: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} directions
* @return {number}
*/
var countCollisions = function(directions) {

};
```

TypeScript:

```
function countCollisions(directions: string): number {

};
```

C#:

```
public class Solution {
public int CountCollisions(string directions) {

}
}
```

C:

```
int countCollisions(char* directions) {

}
```

Go:

```
func countCollisions(directions string) int {

}
```

Kotlin:

```
class Solution {
fun countCollisions(directions: String): Int {

}
}
```

Swift:

```
class Solution {
func countCollisions(_ directions: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_collisions(directions: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} directions
# @return {Integer}
def count_collisions(directions)

end
```

PHP:

```
class Solution {

/**
* @param String $directions
* @return Integer
*/
function countCollisions($directions) {

}
```

```
}
```

Dart:

```
class Solution {
int countCollisions(String directions) {

}
}
```

Scala:

```
object Solution {
def countCollisions(directions: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_collisions(directions :: String.t) :: integer
def count_collisions(directions) do

end
end
```

Erlang:

```
-spec count_collisions(Directions :: unicode:unicode_binary()) -> integer().
count_collisions(Directions) ->
.
```

Racket:

```
(define/contract (count-collisions directions)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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 countCollisions(string directions) {

}
};
```

Java Solution:

```
/**
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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 countCollisions(String directions) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Collisions on a Road
Difficulty: Medium
Tags: string, stack
```

```
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 countCollisions(self, directions: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countCollisions(self, directions):
"""
:type directions: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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} directions
* @return {number}
*/
var countCollisions = function(directions) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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 countCollisions(directions: string): number {

};
```

C# Solution:

```
/*
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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 CountCollisions(string directions) {

}
}
```

C Solution:

```
/*
* Problem: Count Collisions on a Road
* Difficulty: Medium
* Tags: string, stack
*
* 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 countCollisions(char* directions) {

}
```

Go Solution:

```
// Problem: Count Collisions on a Road
// Difficulty: Medium
// Tags: string, stack
//
// 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 countCollisions(directions string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countCollisions(directions: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countCollisions(_ directions: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Collisions on a Road
// Difficulty: Medium
// Tags: string, stack
```

```
//
// 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 count_collisions(directions: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} directions
# @return {Integer}
def count_collisions(directions)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $directions
* @return Integer
*/
function countCollisions($directions) {

}
}
```

Dart Solution:

```
class Solution {
int countCollisions(String directions) {

}
}
```

Scala Solution:

```
object Solution {
def countCollisions(directions: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_collisions(directions :: String.t) :: integer
def count_collisions(directions) do

end
end
```

Erlang Solution:

```
-spec count_collisions(Directions :: unicode:unicode_binary()) -> integer().
count_collisions(Directions) ->
.
```

Racket Solution:

```
(define/contract (count-collisions directions)
(-> string? exact-integer?)
)
```