Problem 856: Score of Parentheses

Problem Information

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

Problem Description

Given a balanced parentheses string

s

, return

the

score

of the string

.

The

score

of a balanced parentheses string is based on the following rule:

"()"

has score

1

.

AB

has score

A + B

, where

A

and

B

are balanced parentheses strings.

(A)

has score

2 * A

, where

A

is a balanced parentheses string.

Example 1:

Input:

s = "()"

Output:

1

Example 2:

Input:

s = "(())"

Output:

2

Example 3:

Input:

s = "()()"

Output:

2

Constraints:

2 <= s.length <= 50

s

consists of only

'('

and

')'

.

s

is a balanced parentheses string.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
```

```
var scoreOfParentheses = function(s) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int scoreOfParentheses(char* s) {

}
```

Go:

```
func scoreOfParentheses(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

```
}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (score-of-parentheses s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Score of Parentheses
* 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 scoreOfParentheses(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Score of Parentheses
* 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 scoreOfParentheses(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Score of Parentheses
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 scoreOfParentheses(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Score of Parentheses
* 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} s
* @return {number}
*/
var scoreOfParentheses = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Score of Parentheses
* 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 scoreOfParentheses(s: string): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Score of Parentheses
* 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 ScoreOfParentheses(string s) {

}
}
```

C Solution:

```
/*
* Problem: Score of Parentheses
* 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 scoreOfParentheses(char* s) {

}
```

Go Solution:

```
// Problem: Score of Parentheses
// 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 scoreOfParentheses(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Score of Parentheses
// 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 score_of_parentheses(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (score-of-parentheses s)
(-> string? exact-integer?)
)
```