Problem 678: Valid Parenthesis String

Problem Information

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

Problem Description

Given a string

s

containing only three types of characters:

'('

,

')'

and

'*'

, return

true

if

s

is

valid

.

The following rules define a

valid

string:

Any left parenthesis

'('

must have a corresponding right parenthesis

')'

.

Any right parenthesis

')'

must have a corresponding left parenthesis

'('

.

Left parenthesis

'('

must go before the corresponding right parenthesis

')'

.

'*'

could be treated as a single right parenthesis

')'

or a single left parenthesis

'('

or an empty string

""

.

Example 1:

Input:

s = "()"

Output:

true

Example 2:

Input:

s = "(*)"

Output:

true

Example 3:

Input:

s = "(*))"

Output:

true

Constraints:

1 <= s.length <= 100

s[i]

is

'('

,

')'

or

'*'

.

Code Snippets

C++:

```
class Solution {
public:
bool checkValidString(string s) {

}
};
```

Java:

```
class Solution {
public boolean checkValidString(String s) {

}
}
```

Python3:

```
class Solution:
def checkValidString(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def checkValidString(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var checkValidString = function(s) {

};
```

TypeScript:

```
function checkValidString(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool CheckValidString(string s) {
```

```
}
}
```

C:

```
bool checkValidString(char* s) {

}
```

Go:

```
func checkValidString(s string) bool {

}
```

Kotlin:

```
class Solution {
fun checkValidString(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func checkValidString(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_valid_string(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def check_valid_string(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function checkValidString($s) {

}
}
```

Dart:

```
class Solution {
bool checkValidString(String s) {

}
}
```

Scala:

```
object Solution {
def checkValidString(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_valid_string(s :: String.t) :: boolean
def check_valid_string(s) do

end
end
```

Erlang:

```
-spec check_valid_string(S :: unicode:unicode_binary()) -> boolean().
check_valid_string(S) ->
.
```

Racket:

```
(define/contract (check-valid-string s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Parenthesis String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* 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:
bool checkValidString(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Valid Parenthesis String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* 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 boolean checkValidString(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Valid Parenthesis String
Difficulty: Medium
Tags: string, dp, greedy, stack

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 checkValidString(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkValidString(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Valid Parenthesis String
* Difficulty: Medium
```

```
* Tags: string, dp, greedy, stack
*
* 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} s
* @return {boolean}
*/
var checkValidString = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Valid Parenthesis String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* 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 checkValidString(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Valid Parenthesis String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* 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 bool CheckValidString(string s) {

}
}
```

C Solution:

```
/*
* Problem: Valid Parenthesis String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* 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
*/

bool checkValidString(char* s) {

}
```

Go Solution:

```
// Problem: Valid Parenthesis String
// Difficulty: Medium
// Tags: string, dp, greedy, stack
//
// 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 checkValidString(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkValidString(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkValidString(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Valid Parenthesis String
// Difficulty: Medium
// Tags: string, dp, greedy, stack
//
// 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 check_valid_string(s: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def check_valid_string(s)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $s
* @return Boolean
*/
function checkValidString($s) {

}
}
```

Dart Solution:

```
class Solution {
bool checkValidString(String s) {

}
}
```

Scala Solution:

```
object Solution {
def checkValidString(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_valid_string(s :: String.t) :: boolean
def check_valid_string(s) do

end
end
```

Erlang Solution:

```
-spec check_valid_string(S :: unicode:unicode_binary()) -> boolean().
check_valid_string(S) ->
.
```

Racket Solution:

```
(define/contract (check-valid-string s)
(-> string? boolean?)
)
```