

Problem 921: Minimum Add to Make Parentheses Valid

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

A parentheses string is valid if and only if:

It is the empty string,

It can be written as

AB

$($

A

concatenated with

B

$)$, where

A

and

B

are valid strings, or

It can be written as

(A)

, where

A

is a valid string.

You are given a parentheses string

s

. In one move, you can insert a parenthesis at any position of the string.

For example, if

s = "())"

, you can insert an opening parenthesis to be

"(

(

)))"

or a closing parenthesis to be

"())

)

)"

.

Return

the minimum number of moves required to make

s

valid

.

Example 1:

Input:

s = "())"

Output:

1

Example 2:

Input:

s = "((("

Output:

3

Constraints:

1 <= s.length <= 1000

s[i]

is either

'('

or

')

.

Code Snippets

C++:

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

    }
};
```

Java:

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

    }
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
 * @param {string} s
 * @return {number}
 */
var minAddToMakeValid = function(s) {

};
```

TypeScript:

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

};
```

C#:

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

    }
}
```

C:

```
int minAddToMakeValid(char* s) {

}
```

Go:

```
func minAddToMakeValid(s string) int {

}
```

Kotlin:

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

    }
}
```

Swift:

```

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

  }
}

```

Rust:

```

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

  }
}

```

Ruby:

```

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

end

```

PHP:

```

class Solution {

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

  }
}

```

Dart:

```

class Solution {
  int minAddToMakeValid(String s) {

  }
}

```

Scala:

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

Elixir:

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

Erlang:

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

Racket:

```
(define/contract (min-add-to-make-valid s)  
  (-> string? exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Minimum Add to Make Parentheses Valid  
 * Difficulty: Medium  
 * Tags: string, greedy, 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 minAddToMakeValid(string s) {

    }

};

```

Java Solution:

```

/**
 * Problem: Minimum Add to Make Parentheses Valid
 * Difficulty: Medium
 * Tags: string, greedy, 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 minAddToMakeValid(String s) {

    }

}

```

Python3 Solution:

```

"""
Problem: Minimum Add to Make Parentheses Valid
Difficulty: Medium
Tags: string, greedy, 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 minAddToMakeValid(self, s: str) -> int:
        # TODO: Implement optimized solution

```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
 * Problem: Minimum Add to Make Parentheses Valid
 * Difficulty: Medium
 * Tags: string, greedy, 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 minAddToMakeValid = function(s) {

};
```

TypeScript Solution:

```
/**
 * Problem: Minimum Add to Make Parentheses Valid
 * Difficulty: Medium
 * Tags: string, greedy, 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 minAddToMakeValid(s: string): number {

};

```

C# Solution:

```

/*
 * Problem: Minimum Add to Make Parentheses Valid
 * Difficulty: Medium
 * Tags: string, greedy, 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 MinAddToMakeValid(string s) {

    }
}

```

C Solution:

```

/*
 * Problem: Minimum Add to Make Parentheses Valid
 * Difficulty: Medium
 * Tags: string, greedy, 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 minAddToMakeValid(char* s) {

}

```

Go Solution:

```
// Problem: Minimum Add to Make Parentheses Valid
// Difficulty: Medium
// Tags: string, greedy, 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 minAddToMakeValid(s string) int {

}
```

Kotlin Solution:

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

    }
}
```

Swift Solution:

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

    }
}
```

Rust Solution:

```
// Problem: Minimum Add to Make Parentheses Valid
// Difficulty: Medium
// Tags: string, greedy, 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 min_add_to_make_valid(s: String) -> i32 {

    }
}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

    }

}
```

Dart Solution:

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

  }
}
```

Scala Solution:

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

  }
}
```

Elixir Solution:

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

  end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (min-add-to-make-valid s)
  (-> string? exact-integer?)
)
```