')'

Problem 1249: Minimum Remove to Make Valid Parentheses

Problem Information

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

Problem Description

Given a string

s

of

'('

,

')'

and lowercase English characters.

Your task is to remove the minimum number of parentheses (

'('

or

')'

, in any positions) so that the resulting

parentheses string

is valid and return

any

valid string.

Formally, a

parentheses string

is valid if and only if:

It is the empty string, contains only lowercase characters, or

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.

Example 1:

Input:

s = "lee(t(c)o)de)"

Output:

"lee(t(c)o)de"

Explanation:

"lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.

Example 2:

Input:

s = "a)b(c)d"

Output:

"ab(c)d"

Example 3:

Input:

s = "))(("

Output:

""

Explanation:

An empty string is also valid.

Constraints:

1 <= s.length <= 10

5

s[i]

is either

'('

,

')'

, or lowercase English letter.

Code Snippets

C++:

```
class Solution {
public:
string minRemoveToMakeValid(string s) {

}
};
```

Java:

```
class Solution {
public String minRemoveToMakeValid(String s) {
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minRemoveToMakeValid(s: string): string {

};
```

C#:

```
public class Solution {
public string MinRemoveToMakeValid(string s) {

}
}
```

C:

```
char* minRemoveToMakeValid(char* s) {

}
```

Go:

```
func minRemoveToMakeValid(s string) string {

}
```

Kotlin:

```
class Solution {
fun minRemoveToMakeValid(s: String): String {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
String minRemoveToMakeValid(String s) {

}
}
```

Scala:

```
object Solution {
def minRemoveToMakeValid(s: String): String = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec min_remove_to_make_valid(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
min_remove_to_make_valid(S) ->
```

```
.
```

Racket:

```
(define/contract (min-remove-to-make-valid s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Remove to Make Valid 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:
string minRemoveToMakeValid(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Remove to Make Valid 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 String minRemoveToMakeValid(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Remove to Make Valid 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 minRemoveToMakeValid(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Remove to Make Valid 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 {string}
*/
var minRemoveToMakeValid = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Remove to Make Valid 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 minRemoveToMakeValid(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Minimum Remove to Make Valid 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 string MinRemoveToMakeValid(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Remove to Make Valid 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
*/

char* minRemoveToMakeValid(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Remove to Make Valid 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 minRemoveToMakeValid(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun minRemoveToMakeValid(s: String): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Remove to Make Valid 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 min_remove_to_make_valid(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
String minRemoveToMakeValid(String s) {

}
}
```

Scala Solution:

```
object Solution {
def minRemoveToMakeValid(s: String): String = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec min_remove_to_make_valid(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
min_remove_to_make_valid(S) ->
.
```

Racket Solution:

```
(define/contract (min-remove-to-make-valid s)
(-> string? string?)
)
```