Problem 1021: Remove Outermost Parentheses

Problem Information

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

Problem Description

A valid parentheses string is either empty

""

,

"(" + A + ")"

, or

A + B

, where

A

and

B

are valid parentheses strings, and

+

represents string concatenation.

For example,

""

,

"()"

,

"(())()"

, and

"(()(()))"

are all valid parentheses strings.

A valid parentheses string

s

is primitive if it is nonempty, and there does not exist a way to split it into

s = A + B

, with

A

and

B

nonempty valid parentheses strings.

Given a valid parentheses string

s

, consider its primitive decomposition:

s = P

1

+ P

2

+ ... + P

k

, where

P

i

are primitive valid parentheses strings.

Return

s

after removing the outermost parentheses of every primitive string in the primitive decomposition of

s

.

Example 1:

Input:

s = "(()())(())"

Output:

"()()()"

Explanation:

The input string is "(()())(())", with primitive decomposition "(()())" + "(())". After removing outer parentheses of each part, this is "()()" + "()" = "()()()".

Example 2:

Input:

s = "(()())(())(()(()))"

Output:

"()()()()(())"

Explanation:

The input string is "(()())(())(()(()))", with primitive decomposition "(()())" + "(())" + "(()(()))". After removing outer parentheses of each part, this is "()()" + "()" + "()(())" = "()()()()(())".

Example 3:

Input:

s = "()()"

Output:

""

Explanation:

The input string is "()()", with primitive decomposition "()" + "()". After removing outer parentheses of each part, this is "" + "" = "".

Constraints:

1 <= s.length <= 10

5

s[i]

is either

'('

or

')'

.

s

is a valid parentheses string.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public String removeOuterParentheses(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* removeOuterParentheses(char* s) {

}
```

Go:

```
func removeOuterParentheses(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (remove-outer-parentheses s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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 removeOuterParentheses(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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 removeOuterParentheses(String s) {
}
}
```

Python3 Solution:

```
"""
Problem: Remove Outermost Parentheses
Difficulty: Easy
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 removeOuterParentheses(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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 removeOuterParentheses = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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 removeOuterParentheses(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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 RemoveOuterParentheses(string s) {

}

}
```

C Solution:

```c
/*
* Problem: Remove Outermost Parentheses
* Difficulty: Easy
* 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* removeOuterParentheses(char* s) {

}
```

Go Solution:

```go
// Problem: Remove Outermost Parentheses
// Difficulty: Easy
// 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 removeOuterParentheses(s string) string {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun removeOuterParentheses(s: String): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Remove Outermost Parentheses
// Difficulty: Easy
// 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 remove_outer_parentheses(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (remove-outer-parentheses s)
(-> string? string?)
)
```