Problem 301: Remove Invalid Parentheses

Problem Information

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

Problem Description

Given a string

s

that contains parentheses and letters, remove the minimum number of invalid parentheses to make the input string valid.

Return

a list of

unique strings

that are valid with the minimum number of removals

. You may return the answer in

any order

.

Example 1:

Input:

s = "()())()"

Output:

["(())()","()()()"]

Example 2:

Input:

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

Output:

["(a())()","(a)()()"]

Example 3:

Input:

s = ")("

Output:

[""]

Constraints:

1 <= s.length <= 25

s

consists of lowercase English letters and parentheses

'('

and

')'

.

There will be at most

20

parentheses in

s

.

Code Snippets

C++:

```
class Solution {
public:
vector<string> removeInvalidParentheses(string s) {

}
};
```

Java:

```
class Solution {
public List<String> removeInvalidParentheses(String s) {

}
}
```

Python3:

```
class Solution:
def removeInvalidParentheses(self, s: str) -> List[str]:
```

Python:

```
class Solution(object):
def removeInvalidParentheses(self, s):
```

```
"""
:type s: str
:rtype: List[str]
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string[]}
*/
var removeInvalidParentheses = function(s) {

};
```

TypeScript:

```
function removeInvalidParentheses(s: string): string[] {

};
```

C#:

```
public class Solution {
public IList<string> RemoveInvalidParentheses(string s) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** removeInvalidParentheses(char* s, int* returnSize) {

}
```

Go:

```
func removeInvalidParentheses(s string) []string {
}
```

Kotlin:

```
class Solution {
fun removeInvalidParentheses(s: String): List<String> {

}
}
```

Swift:

```
class Solution {
func removeInvalidParentheses(_ s: String) -> [String] {

}
}
```

Rust:

```
impl Solution {
pub fn remove_invalid_parentheses(s: String) -> Vec<String> {

}
}
```

Ruby:

```
# @param {String} s
# @return {String[]}
def remove_invalid_parentheses(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String[]
*/
function removeInvalidParentheses($s) {

}
}
```

Dart:

```
class Solution {
List<String> removeInvalidParentheses(String s) {

}
}
```

Scala:

```
object Solution {
def removeInvalidParentheses(s: String): List[String] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec remove_invalid_parentheses(s :: String.t) :: [String.t]
def remove_invalid_parentheses(s) do

end
end
```

Erlang:

```
-spec remove_invalid_parentheses(S :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
remove_invalid_parentheses(S) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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:
vector<string> removeInvalidParentheses(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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 List<String> removeInvalidParentheses(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Remove Invalid Parentheses
Difficulty: Hard
Tags: string, search

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 removeInvalidParentheses(self, s: str) -> List[str]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def removeInvalidParentheses(self, s):
"""
:type s: str
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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 removeInvalidParentheses = function(s) {
};
```

TypeScript Solution:

```
/**
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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 removeInvalidParentheses(s: string): string[] {

};
```

C# Solution:

```
/*
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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 IList<string> RemoveInvalidParentheses(string s) {

}
}
```

C Solution:

```
/*
* Problem: Remove Invalid Parentheses
* Difficulty: Hard
* Tags: string, search
*
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** removeInvalidParentheses(char* s, int* returnSize) {

}
```

Go Solution:

```
// Problem: Remove Invalid Parentheses
// Difficulty: Hard
// Tags: string, search
//
// 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 removeInvalidParentheses(s string) []string {

}
```

Kotlin Solution:

```
class Solution {
fun removeInvalidParentheses(s: String): List<String> {

}
}
```

Swift Solution:

```
class Solution {
func removeInvalidParentheses(_ s: String) -> [String] {
}
}
```

Rust Solution:

```
// Problem: Remove Invalid Parentheses
// Difficulty: Hard
// Tags: string, search
//
// 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_invalid_parentheses(s: String) -> Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String[]}
def remove_invalid_parentheses(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String[]
*/
function removeInvalidParentheses($s) {

}
}
```

Dart Solution:

```
class Solution {
List<String> removeInvalidParentheses(String s) {

}
}
```

Scala Solution:

```
object Solution {
def removeInvalidParentheses(s: String): List[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec remove_invalid_parentheses(s :: String.t) :: [String.t]
def remove_invalid_parentheses(s) do

end
end
```

Erlang Solution:

```
-spec remove_invalid_parentheses(S :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
remove_invalid_parentheses(S) ->
.
```

Racket Solution:

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