1021. Remove Outermost Parentheses

Hint

Easy 2.2K 1.4K

Companies

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 = P₁ + P₂ + ... + Pₖ`, where `Pᵢ` 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 <= 10⁵`
- `s[i]` is either `'('` or `')'`.
- `s` is a valid parentheses string.

Accepted 226.7K | Submissions 280.1K | Acceptance Rate 80.9%

Approach : Using extra string 'ans' to return

Approach 1: Using extra string ans to return.

```
class Solution {
public:
    string removeOuterParentheses(string s) {
        string ans="";
        int count=0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='('){
                if(count!=0) ans+='(';
                count++;
            }
            else if(s[i]==')'){
                count--;
                if(count!=0) ans+=')';
            }
        }

        return ans;
    }
};
```

T: O(n)

S: O(1) becoz extra space used for ans does not count for S(n)

Approach 2: without using any extra string variable.

```
class Solution {
public:
    string removeOuterParentheses(string s) {
        int count=0;
        int n=s.length();
        for(int i=0;i<n;i++){
            if(s[i]=='('){
                if(count==0) s[i]=' ';
                count++;
            }
            else if(s[i]==')'){
                count--;
                if(count==0) s[i]=' ';
            }
        }

        int i=0;
        while(i<n){
            if(s[i]==' ') {
```

```
            if(s[i] == '*') {
                s.erase(i,1);
            }else{
                i++;
            }
        }

        return s;
    }
};
```

i: O(n)

s: O(1)