1021. Remove Outermost Parentheses

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<sub>1</sub> + P<sub>2</sub> + ... + P<sub>k</sub>`, where `P<sub>i</sub>` 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<sup>5</sup>`
- `s[i]` is either `'('` or `')'`.
- `s` is a valid parentheses string.

```
class Solution:
    def removeOuterParentheses(self, s: str) -> str:
        ans = ''
        start = 0
        end = 0
        count = 0
        while end<len(s):
            if s[end]=='(':
                count+=1
            else:
                count-=1
            if count==0 and end!=0:
                ans = ans+s[start+1:end]
                start = end+1
            end+=1
        return ans+s[start:end+1]
```

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

        ans = []
        stack = []

        for i in range(len(s)):
            ch = s[i]

            if ch == '(':
                if len(stack)>0:
                    ans.append(ch)
                stack.append(ch)
            else:
                stack.pop()
                if len(stack)>0:
                    ans.append(ch)
        return ''.join(ans)
```