# 题目

有效括号字符串为空("")、"(" + A + ")"或A + B，其中A和B都是有效的括号字符串，+代表字符串的连接。例如，""，"()"，"(())()"和"(()(()))"都是有效的括号字符串。

如果有效字符串S非空，且不存在将其拆分为S = A+B的方法，我们称其为原语（primitive），其中A 和B都是非空有效括号字符串。

给出一个非空有效字符串S，考虑将其进行原语化分解，使得：S = P\_1 + P\_2 + ... + P\_k，其中P\_i是有效括号字符串原语。

对S进行原语化分解，删除分解中每个原语字符串的最外层括号，返回S。

**示例 1：**

输入："(()())(())"

输出："()()()"

解释：

输入字符串为 "(()())(())"，原语化分解得到 "(()())" + "(())"，

删除每个部分中的最外层括号后得到 "()()" + "()" = "()()()"。

**示例 2：**

输入："(()())(())(()(()))"

输出："()()()()(())"

解释：

输入字符串为 "(()())(())(()(()))"，原语化分解得到 "(()())" + "(())" + "(()(()))"，

删除每个部分中的最外层括号后得到 "()()" + "()" + "()(())" = "()()()()(())"。

**示例 3：**

输入："()()"

输出：""

解释：

输入字符串为 "()()"，原语化分解得到 "()" + "()"，

删除每个部分中的最外层括号后得到 "" + "" = ""。

**提示：**

S.length <= 10000

S[i]为"(" 或")"

S是一个有效括号字符串

# 分析

## 方法一：栈

**思路：**

**代码：**

class Solution {

public:

string removeOuterParentheses(string S) {

string res;

//记录有效括号

stack<char> stack;

//i遍历 j下次操作开始位置

for(int i = 0,j = 0;i<S.size();i++){

if(S[i]=='(')

stack.push('(');

else

stack.pop();

if(stack.size()==0)

{

res+= S.substr(j+1,i-j-1);

j = i+1;

}

}

return res;

}

};

或：

class Solution {

public:

string removeOuterParentheses(string s) {

string ans;

stack<char> sk;

for (auto c: s) {

// 遇到')'则从栈中弹出'('

if (c == ')') sk.pop();

// 遇到'('则入栈

if (!sk.empty()) ans.push\_back(c);

// '('替换

if (c == '(') sk.emplace(c);

}

return ans;

}

};

**复杂度：**

时间复杂度：O(n)，其中n是输入s的长度。仅需遍历s一次。

空间复杂度：O(n)，其中n是输入s的长度。需要使用栈，长度最大为O(n)。