**难度: 简单**

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

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/remove-outermost-parentheses

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(使用栈,效率低):**

1). 判断栈是否是空,并且是(,入栈

2). 栈不为空且是(入栈,还有让(加入springBuilder

3). 判断如果是)直接让栈头出栈,并且判断栈是否为空,不空的话就把(加入

|  |
| --- |
| public String removeOuterParentheses(String S) {  char[] chars = S.toCharArray();  Deque<Character> stack = new ArrayDeque<>();  StringBuilder sb = new StringBuilder();  for (int i = 0; i < chars.length; i++) {  if (stack.isEmpty() && chars[i] == '(') {  stack.push(chars[i]);  }else if (!stack.isEmpty() && chars[i] == '(') {  stack.push(chars[i]);  sb.append("(");  }  if (chars[i] == ')') {  stack.pop();  if (!stack.isEmpty()) {  sb.append(")");  }  }  }  return sb.toString();  } |

**思路2(不使用栈,相对高):**

与上面使用栈的代码大致相同,只是把栈改成int l来代替栈模拟

|  |
| --- |
| public String removeOuterParentheses2(String S) {  int l = 0;  char[] chars = S.toCharArray();  StringBuilder sb = new StringBuilder();  for (int i = 0; i < chars.length; i++) {  if (chars[i] == '(') {  l++;  if (l > 1) {  sb.append("(");  }  }  if (chars[i] == ')') {  l--;  if (l != 0) {  sb.append(")");  }  }  }  return sb.toString();  } |