# 题目

给定一个由 '(' 和 ')' 括号组成的字符串S，我们需要添加最少的括号（ '(' 或是 ')'，可以在任何位置），以使得到的括号字符串有效。

从形式上讲，只有满足下面几点之一，括号字符串才是有效的：

它是一个空字符串，或者

它可以被写成 AB （A 与 B 连接）, 其中 A 和 B 都是有效字符串，或者

它可以被写作 (A)，其中 A 是有效字符串。

给定一个括号字符串，返回为使结果字符串有效而必须添加的最少括号数。

**示例 1：**

输入："())"

输出：1

**示例 2：**

输入："((("

输出：3

**示例 3：**

输入："()"

输出：0

**示例 4：**

输入："()))(("

输出：4

**提示：**

S.length <= 1000

S 只包含 '(' 和 ')' 字符。

# 分析

## 方法一：栈

**思路：**（推荐该方法，具有普遍性）

维护一个栈，保存待匹配的括号。贪心匹配就是栈顶匹配：

1、当栈顶为(而下一个字符为)，则匹配，即出栈

2、否则，压栈

最终，栈大小表示未/待匹配的括号数，即答案。

另外，还有个思路是“消消乐”，即不断地消除字符串中任意位置的()、直至不存在，这种做法就等同于栈顶的贪心匹配。

**代码：**

class Solution {

public:

int minAddToMakeValid(string S) {

stack<char> st;

for (char ch : S) {

if (!st.empty() && st.top() == '(' && ch == ')') { // 栈顶匹配

st.pop();

} else {

st.push(ch);

}

}

return st.size();

}

};

或：

class Solution {

public:

    int minAddToMakeValid(string s) {

        stack<char> stk;

        for(int i=0;i<s.length();i++)

        {

            if(!stk.empty() && stk.top()== '(' && s[i] == ')')

            {

                stk.pop();

            }

            else

            {

                stk.push(s[i]);

            }

        }

        return stk.size();

    }

};

**复杂度分析：**

时间复杂度：O(n)

空间复杂度：O(n)

## 方法二：平衡法

**思路：**

维护一个delta值，表示当前左括号的数量：

如果delta < 0，表示当前有右括号未匹配，则必须添加一个左括号来跟它配对，因为即使后面有左括号也无法跟该右括号匹配。

如果delta >= 0，表示当前有左括号未匹配，此时不需要马上添加右括号来跟它配对，因为后面可能会出现右括号。所以我们只需要在最后把delta算上就行。

**代码：**

class Solution {

public:

int minAddToMakeValid(string S) {

int delta = 0;

int res = 0;

for (char ch : S) {

if (ch == '(') {

++delta;

} else if (ch == ')') {

--delta;

if (delta == -1) { // 当前右括号更多，必须添加1个左括号

++res;

delta = 0;

}

}

}

res += delta;

return res;

}

};

**复杂度分析：**

时间复杂度：O(N)，其中 N 是 S 的长度。

空间复杂度：O(1)。