# 题目

给出由小写字母组成的字符串 S，重复项删除操作会选择两个相邻且相同的字母，并删除它们。

在 S 上反复执行重复项删除操作，直到无法继续删除。

在完成所有重复项删除操作后返回最终的字符串。答案保证唯一。

示例：

输入："abbaca"

输出："ca"

解释：

例如，在 "abbaca" 中，我们可以删除 "bb" 由于两字母相邻且相同，这是此时唯一可以执行删除操作的重复项。之后我们得到字符串 "aaca"，其中又只有 "aa" 可以执行重复项删除操作，所以最后的字符串为 "ca"。

提示：

1 <= S.length <= 20000

S 仅由小写英文字母组成。

# 分析

## 方法一：栈

class Solution {

public:

string removeDuplicates(string S) {

stack<char> st;

for (char s : S) {

if (st.empty() || s != st.top()) {

st.push(s);

} else {

st.pop(); // s 与 st.top()相等的情况

}

}

string result = "";

while (!st.empty()) { // 将栈中元素放到result字符串汇总

result += st.top();

st.pop();

}

reverse (result.begin(), result.end()); // 此时字符串需要反转一下

return result;

}

};

## 方法二：栈/字符串

class Solution {

public:

string removeDuplicates(string s) {

string ans;

for (auto c : s) {

if (!ans.empty() && ans.back() == c) {

ans.pop\_back();

} else {

ans.push\_back(c);

}

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n是字符串的长度。我们只需要遍历该字符串一次。

空间复杂度：O(n)或 O(1)，取决于使用的语言提供的字符串类是否提供了类似「入栈」和「出栈」的接口。注意返回值不计入空间复杂度。