# 题目

给你一个字符串 s ，请你去除字符串中重复的字母，使得每个字母只出现一次。需保证返回结果的字典序最小（要求不能打乱其他字符的相对位置）。

示例 1：

输入：s = "bcabc"

输出："abc"

示例 2：

输入：s = "cbacdcbc"

输出："acdb"

提示：

1 <= s.length <= 10^4

s 由小写英文字母组成

注意：该题与1081相同

<https://leetcode-cn.com/problems/smallest-subsequence-of-distinct-characters>

# 分析

## 方法一：栈

思路：

可以使用贪心算法来解决这个问题。首先统计每个字符出现的次数，然后使用一个栈来维护当前结果。遍历字符串，对于每个字符，如果字符没有在栈中出现过，则需要将其加入栈中。在加入之前，需要判断栈顶字符是否大于当前字符且后面还会出现，如果满足这个条件，则可以将栈顶字符弹出，然后再加入当前字符，以保证结果的字典序最小。具体实现如下：

代码：

class Solution {

public:

string removeDuplicateLetters(string s) {

vector<int> count(26);

vector<bool> inStack(26);

for (char c : s) {

count[c - 'a']++;

}

stack<char> stk;

for (char c : s) {

count[c - 'a']--;

if (inStack[c - 'a']) continue;

while (!stk.empty() && stk.top() > c && count[stk.top() - 'a'] > 0) {

inStack[stk.top() - 'a'] = false;

stk.pop();

}

stk.push(c);

inStack[c - 'a'] = true;

}

string result = "";

while (!stk.empty()) {

result = stk.top() + result;

stk.pop();

}

return result;

}

};

这个算法的时间复杂度是 O(n)，其中 n 是字符串 s 的长度。

class Solution {

public:

string smallestSubsequence(string s) {

vector<int> left(26), in\_ans(26);

for (char c: s)

left[c - 'a']++; // 统计每个字母的出现次数

string ans;

for (char c: s) {

left[c - 'a']--;

if (in\_ans[c - 'a']) // ans 中不能有重复字母

continue;

while (!ans.empty() && c < ans.back() && left[ans.back() - 'a']) {

// (设 x=ans.back()) 如果 c < x，且右边还有 x，那么可以把 x 去掉，

// 因为后面可以重新把 x 加到 ans 中

in\_ans[ans.back() - 'a'] = false; // 标记 x 不在 ans 中

ans.pop\_back();

}

ans += c; // 把 c 加到 ans 的末尾

in\_ans[c - 'a'] = true; // 标记 c 在 ans 中

}

return ans;

}

};