**难度: 简单**

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。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/remove-all-adjacent-duplicates-in-string

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).** 创建一个栈,当栈为空或者是栈顶的值不等于需要push的值就入栈

2). 如果栈顶的值等于需要push的值,就直接出栈

3). 最后把剩余栈值依次抛出到StringBuilder中

4). StringBuilder把值颠倒然后返回字符串格式

|  |
| --- |
| public String removeDuplicates(String S) {  //创建一个栈  Deque<Character> stack = new ArrayDeque<>();  //转换成字符数组  char[] chars = S.toCharArray();  //遍历  for (int i = 0; i < chars.length; i++) {  if (stack.isEmpty() || stack.peek() != chars[i]) {  stack.push(chars[i]);  }else if (stack.peek() == chars[i]) {  stack.pop();  }  }  StringBuilder sb = new StringBuilder();  while(!stack.isEmpty()) {  sb.append(stack.pop());  }  return sb.reverse().toString();  } |

**思路2(不使用栈,效率相对高):**

1). 如果字符串的长度0或1直接返回

2). 模拟一个栈顶来定位

3). 如果相同就top-1, 不同就存放然后top+1

4). 最后转换成字符串返回

|  |
| --- |
| public String removeDuplicates2(String S) {  if (S.length() == 0 || S.length() == 1) {  return S;  }  int top = -1; //栈顶  //转换成字符数组  char[] chars = S.toCharArray();  StringBuilder sb = new StringBuilder();  //遍历  for (int i = 0; i < chars.length; i++) {  if (top > -1 && sb.charAt(top) == chars[i]) {  sb.deleteCharAt(top);  top--;  }else {  top++;  sb.append(chars[i]);  }  }  return sb.toString();  } |