1. 题目
2. 题干

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）。

1. 示例

**示例 1：**

**输入：**s = "bcabc"

**输出：**"abc"

**示例 2：**

**输入：**s = "cbacdcbc"

**输出：**"acdb"

1. 题解
2. 思路

首先，从题目来看，要给字符串中的字母去重。乍一听，挺简单的，之前我们也做过数字去重啊，但是定睛一看，这个题目的要求是返回结果的字典序最小，且不能打乱所有字母的排列顺序。

总结，三个要求：

1. 字母去重
2. 排序不变
3. 字典序最小

字母去重这个要求很好实现，设置一个栈来存储字母，设置一个布尔数组来记录哪些字母已经进栈，已经进栈的字母就不能再进栈了。同样的，这样也能实现排序不变的要求。

那么啥叫字典序最小？字典序就是字母的排列顺序，如何保证字典序最小呢？这个就需要我们在插入字母的时候进行判断了，如果当前要插入的字母比栈中的字母要小，那么就需要考虑将比该字母大的字母弹出了。

但是，这样又可能导致有的字母全部被删除了，一次都不会出现，这里我们需要设置一个数组来记录每个字母的出现次数，在弹出判断时保证最少它在栈中存在一次。

这样，最终栈中的字母必定是满足题目的三个要求的，返回即可。

1. 代码实现

Java：

class Solution {

    public String removeDuplicateLetters(String s) {

        Stack<Character> stack = new Stack<>();

        //ASCII字符，256个够用了，count数组记录每个字符出现的次数

        int[] count = new int[256];

        for(int i = 0; i < s.length() ; ++i){

            count[s.charAt(i)]++;

        }

        //判断字符是否在栈中

        boolean[] inStack = new boolean[256];

        for(char c : s.toCharArray()){

            //每遍历一次字符，该字符的count就要减一

            count[c]--;

            //该字符在栈中

            if(inStack[c]){

                continue;

            }

            //该字符不在栈中，需要判断字典序,栈顶元素大于当前字符

            while(!stack.isEmpty() && stack.peek() > c){

                //如果后面还有栈顶字符

                if(count[stack.peek()] != 0){

                    //弹出栈顶元素

                    inStack[stack.peek()] = false;

                    stack.pop();

                }else{

                    //后面没有栈顶元素了，那么停止弹出，停止while循环

                    break;

                }

            }

            //将当前的字符加入栈中

            stack.push(c);

            inStack[c] = true;

        }

        //取出栈中的所有字符

        StringBuilder out = new StringBuilder();

        while(!stack.isEmpty()){

            out.append(stack.pop());

        }

        return out.reverse().toString();

    }

}