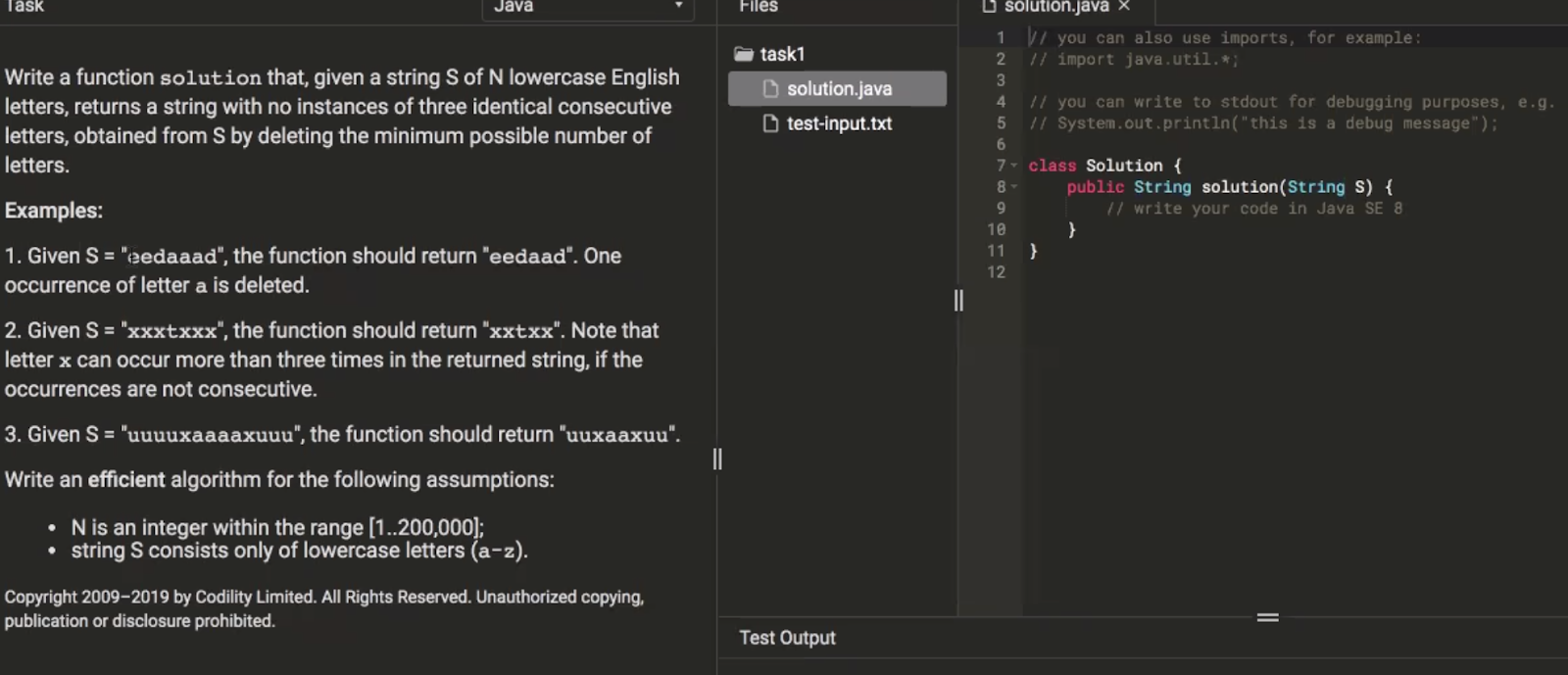
**1- 2如参电话面试或者视频面试，请助攻的同学，MUTE 自己。多谢大家！结束后，把题目贴上。谢谢大家**

**助攻同学1:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

****

**// time complexity: O(n), n is input string S's length;**

**// space complexity: O(1), only use an int count. StringBuilder is used for output, which is not counted for space complexity.**

**public String solution(String S) {**

**// write your code in Java SE 8**

**// if input string S is null or length is smaller or equal to 2, no need to check**

**if (S == null || S.length() <= 2) {**

**return S;**

**}**

**// use StringBuilder to save the character whose occurrence number is smaller or equal to 2**

**StringBuilder sb = new StringBuilder();**

**// use count to identify the current character's occurrence number**

**int count = 1;**

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

**// if StringBuilder's length is 0 or the current character is not the same as the last character in the StringBuilder**

**if (sb.length() == 0 || sb.charAt(sb.length() - 1) != S.charAt(i)) {**

**// add the current character to StringBuilder, and set count to 1**

**sb.append(S.charAt(i));**

**count = 1;**

**} else if (sb.charAt(sb.length() - 1) == S.charAt(i)) {**

**// if the current character is the same as the last character in the StringBuilder, add 1 to count**

**count++;**

**// if count is less than 3, we can add the current character to StringBuilder. Otherwise, ignore it**

**if (count < 3) {**

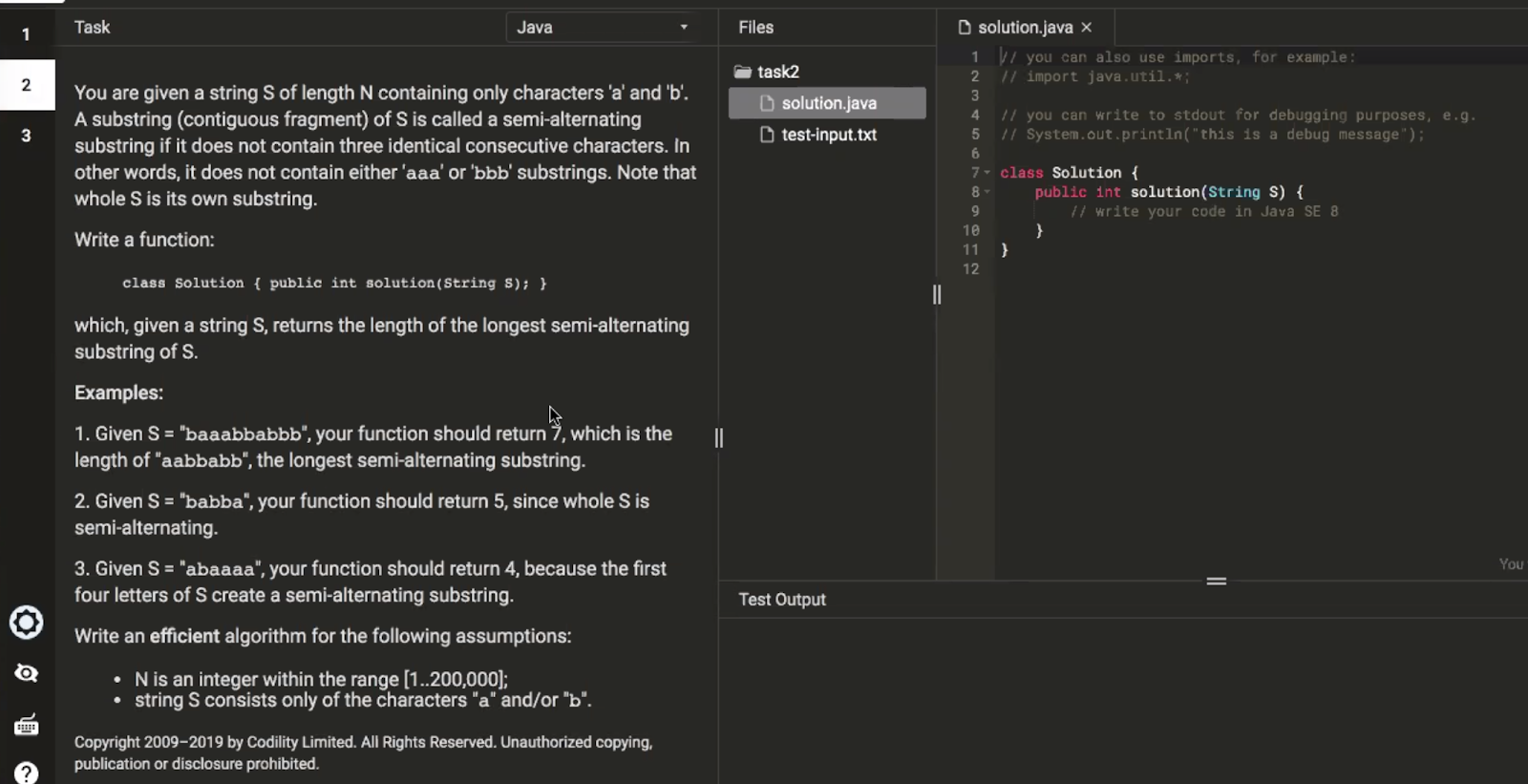
**sb.append(S.charAt(i));**

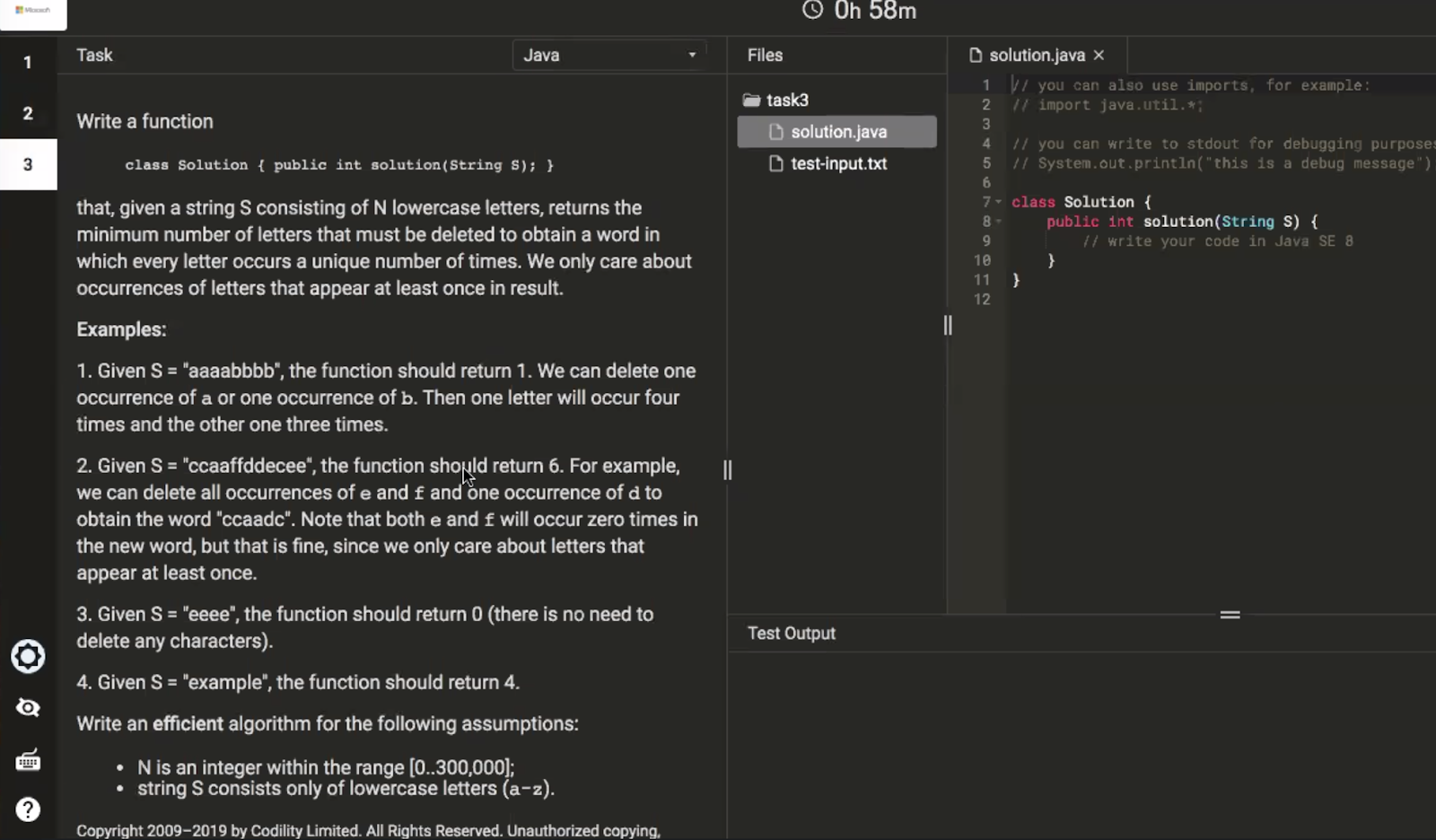
**}**

**}**

**}**

**return sb.toString();**

****

****

**//corner case**

**if(s == null || s.length() <= 1) return 0;**

**int[] counts = new int[26]; // the counts of each unique character**

**int maxCount = 0; //the count of most frequently appeared character**

**// first traverse the string to get the counts of each unique character and the number of max frequency**

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

**counts[c-’a’]++;**

**maxCount = Math.max(maxCount, counts[c-’a’]);**

**}**

**//then use a HashMap in which keys are frequency and values are count of unique character of same frequency**

**Map<Integer, Integer> map = new HashMap<>();**

**for(int i = 0; i < 26; i++) {**

**map.put(counts[i], map.getOrDefault(counts[i], 0) + 1);**

**}**

**int steps = 0;**

**// from max frequency to 0,  if more than one characters have same frequency, delete one and put it in the hashmap //with frequency minus 1; and iteratively do this**

**for(int i = MaxCount; i >0; i--) {**

**Integer curCount = map.get(i);**

**if(curCount != null && curCount > 1) {**

**steps += curCount - 1;**

**map.put(i-1, map.getOrDefault(i-1, 0) + curCount-1);**

**}**

**}**

**return steps;**

**Time O(n),**

**traverse the string one time to memorize the count of character costs O(n) , and traverse the count array with length of 26 cost O(1),  at last, worst case traverse the map from MaxCount to 1 cost O(n).**

**Space O(1) -> O(26)**

**the count array has size of 26, and the hashmap worst case will have size of 26. So space complexity is O(1);**

**助攻同学2:**

**if(s == null || s.length() == 0)**

**return 0;**

**if(s.length() < 3)**

**return s.length();**

**int cnt = 1, l = 0, lastSeen = 0;**

**int res = 0;**

**for(int r = 1;r < s.length();r++) {**

**char c = s.charAt(r);**

**if(s.charAt(r-1) == c) {**

**cnt++;**

**}else {**

**cnt = 1;**

**lastSeen = r;**

**}**

**if(cnt > 2 && l < lastSeen)**

**l = lastSeen;**

**while(cnt > 2) {**

**l++;**

**cnt--;**

**}**

**res = Math.max(res, r - l + 1);**

**}**

**return res;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**助攻同学3:**

**第二题贴了**

public int lengthOfLongestSubstringTwoDistinct(String s) {  
        if(s == null || s.length() == 0) return 0;  
        int idx1 = -1;  
        int idx2 = -1;  
        int maxLen = 0;  
        int start = 0;  
        for(int i = 0; i < s.length(); i++){  
            char ch = s.charAt(i);  
            if(idx1 == -1 || ch == s.charAt(idx1)){  
                idx1 = i;  
            }else if(idx2 == -1 || ch == s.charAt(idx2)){  
                idx2 = i;  
            }else{  
                if(idx1 < idx2){  
                    start = idx1 + 1; // 这里不能用idx2,因为只记录最后一个值  
                    idx1 = i;  
                }else if(idx1 > idx2){  
                    start = idx2 + 1;   
                    idx2 = i;  
                }  
            }  
            maxLen = Math.max(maxLen, i - start + 1);  
        }  
        return maxLen;  
    }

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**助攻同学4:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**助攻同学5：**

[**https://leetcode.com/discuss/interview-question/398023/Microsoft-Online-Assessment-Questions**](https://leetcode.com/discuss/interview-question/398023/Microsoft-Online-Assessment-Questions)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***