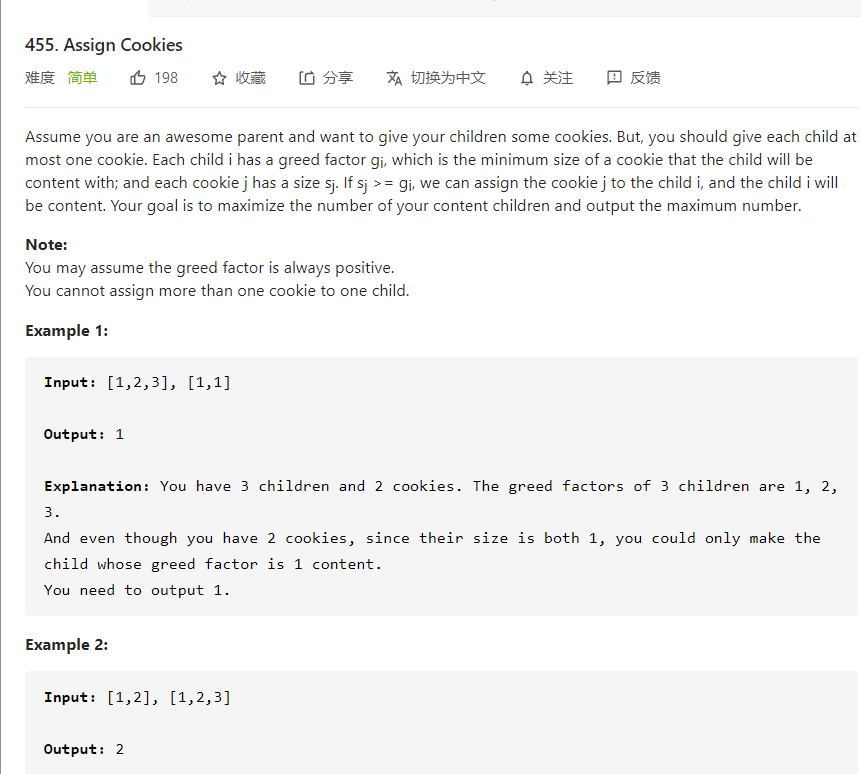
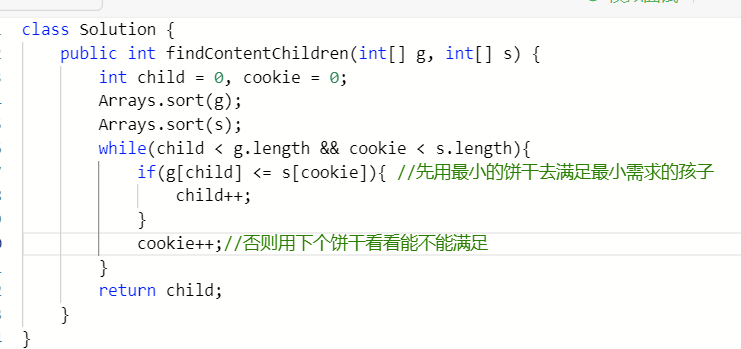
greed algorithm

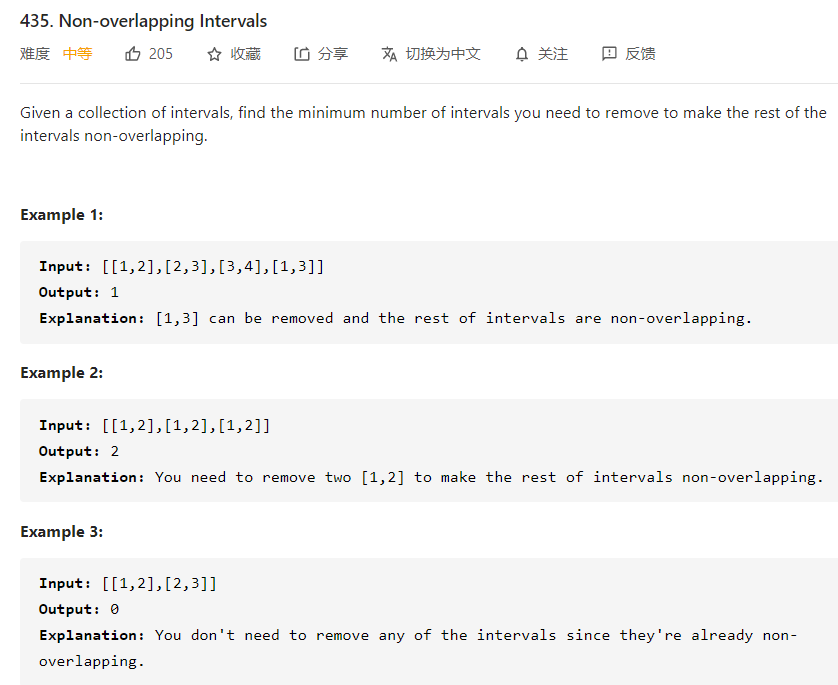
1.



解法：用尽量少的饼干去满足需求量小的孩子



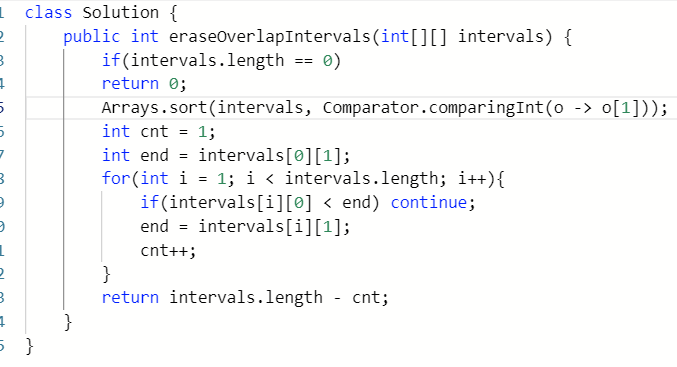
2.



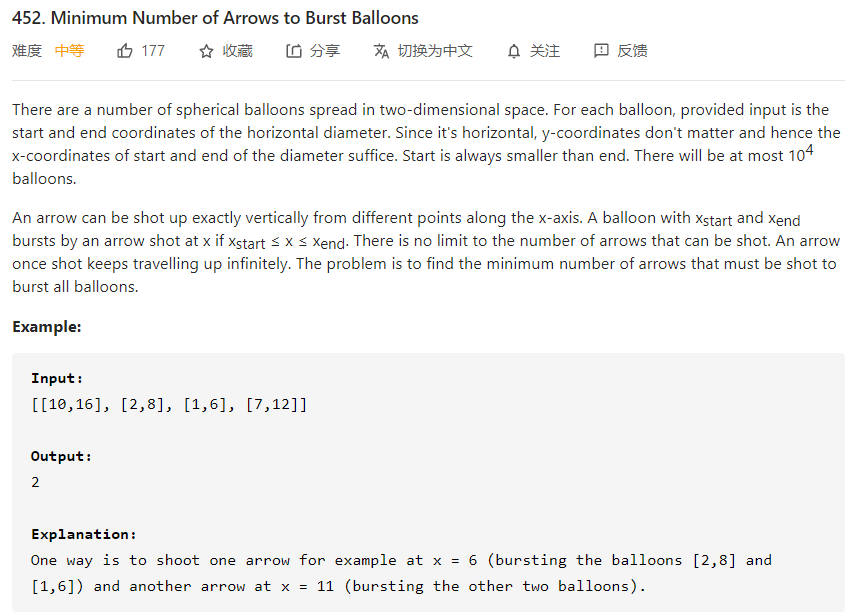
题目：计算去掉能使input的区间不重复的区间的数

解法：先计算最多能组成的不同的区间的数，然后用区间总数减去不重复的区间的数。每次选择中区间的结尾最重要，选择的区间的结尾越小，留给后面的区间的空间越大，则后面能选的区间的个数越大

先拿第一个区间的右区间开始，从第二个元素开始，如果他的左区间小于第一个元素的右区间，说明区间重复了，则cnt不加，否则把第二个元素的右区间去和后面元素的左区间比并且cnt+1.最后得到不重复的区间总数，被总长度减一下就得到了要去掉的会导致重复的区间



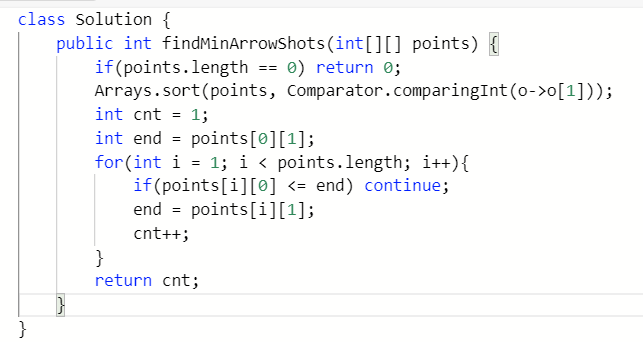
3.



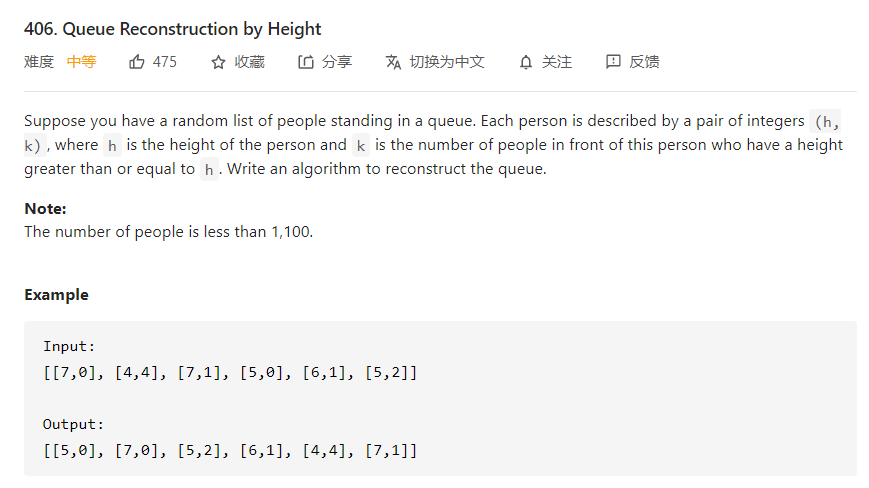
题目：最少的刺破气球的次数，比如x=6能同时刺破中间两个气球

解法：也是找到重复的区间数，但是[2,3]和[3,4]也算是重复区间

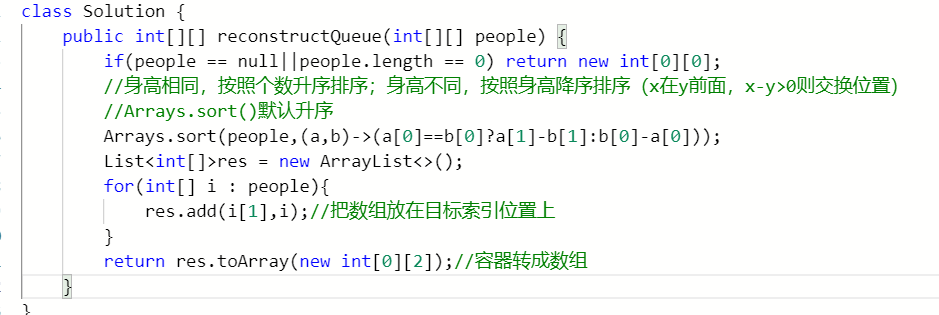
因为重复的区间就是可以用一次刺破



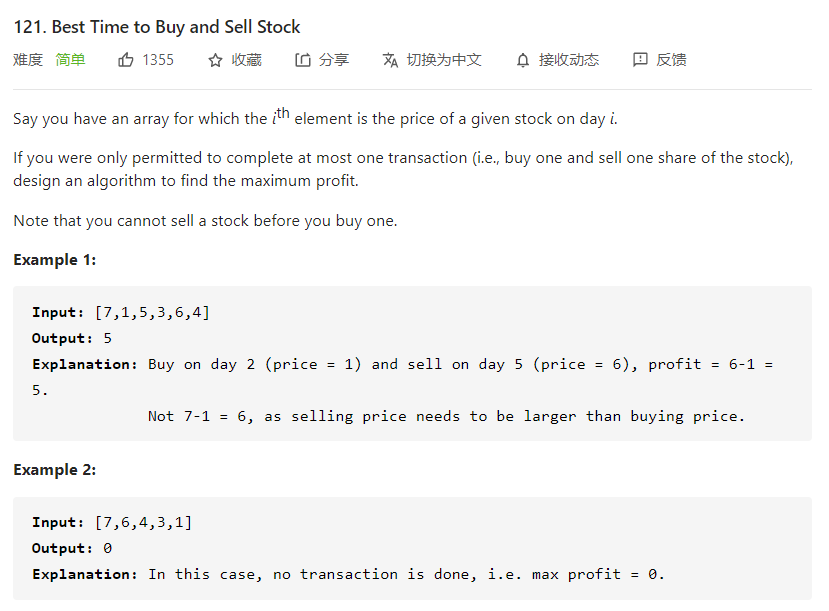
4.



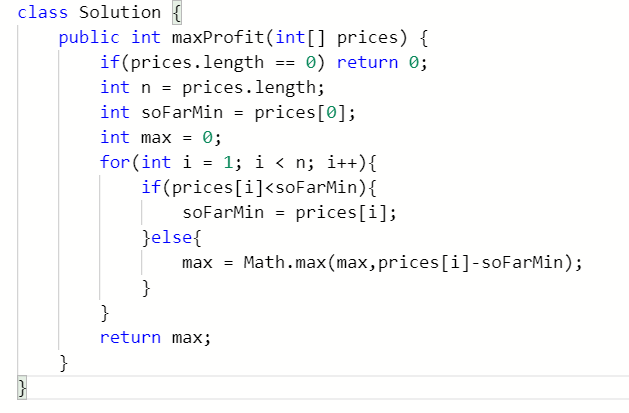
解法：为了使插入操作不影响后续的操作，身高较高的学生应该先做插入操作，否则身高较小的学生原先正确插入的第 k 个位置可能会变成第 k+1 个位置。



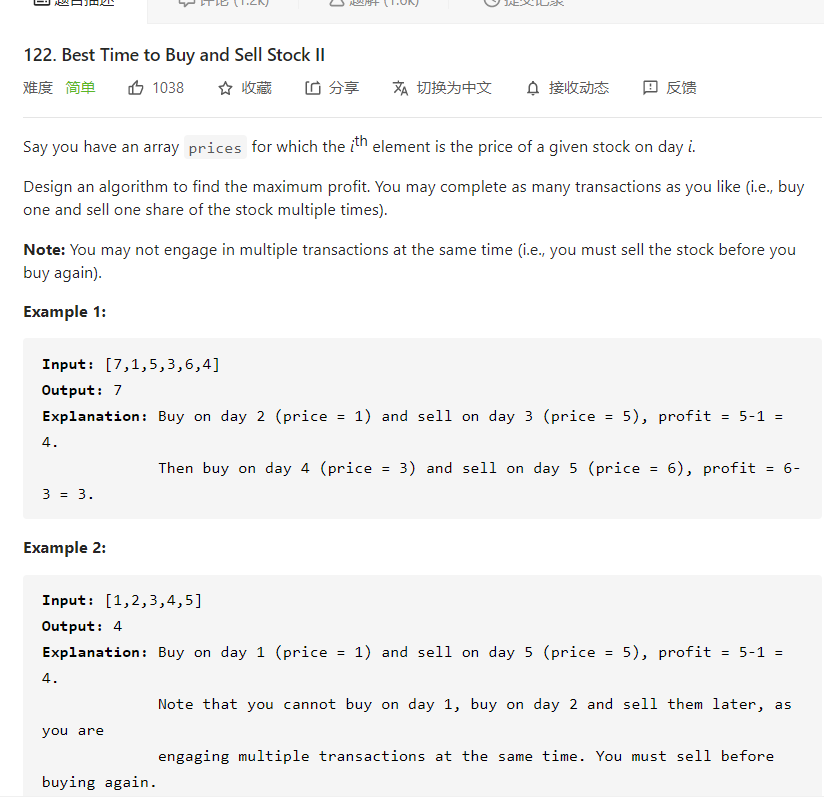
5.



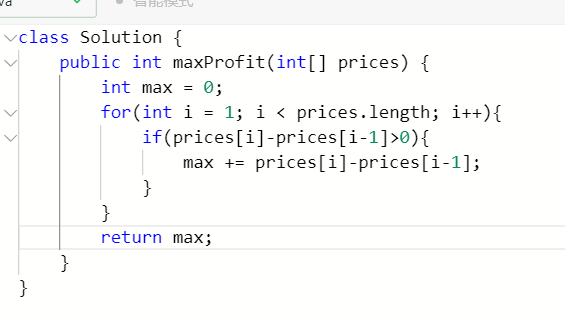
解法：只要记录前面的最小价格，将这个最小价格作为买入价格，然后将当前的价格作为售出价格，查看当前收益是不是最大收益。



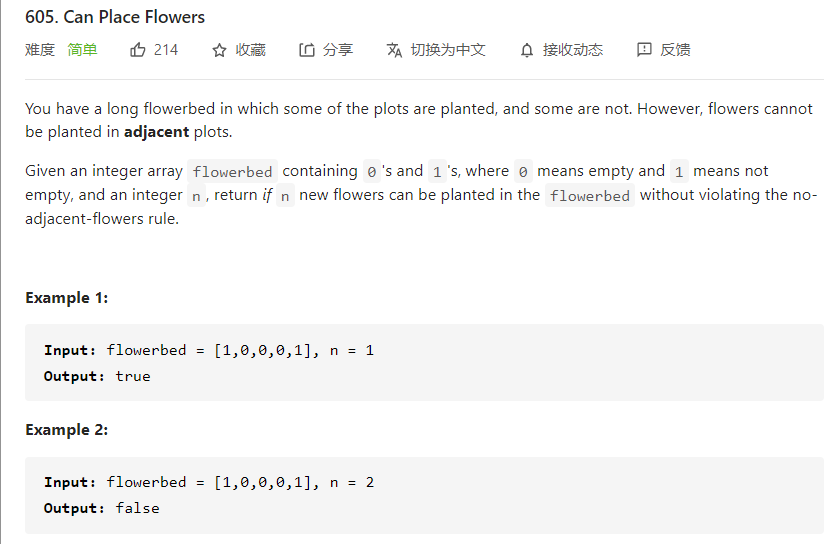
6.



解法：对于 [a, b, c, d]，如果有 a <= b <= c <= d ，那么最大收益为 d - a。而 d - a = (d - c) + (c - b) + (b - a) ，因此当访问到一个 prices[i] 且 prices[i] - prices[i-1] > 0，那么就把 prices[i] - prices[i-1] 添加到收益中。



7.

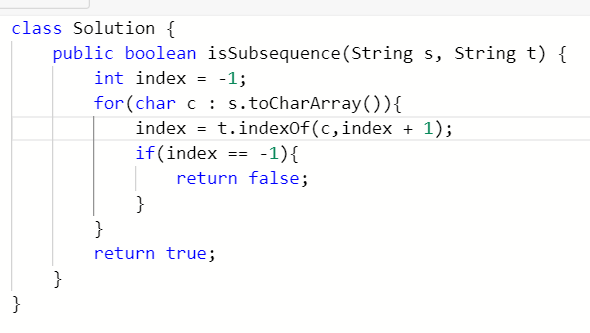


解法：遍历数组，如果是1就跳过，如果是0就检查前面和后面那个元素是不是0，都是0才种植并且cnt+1，检查的时候如果是第一个或者最后一个元素则循环到末尾或者开头

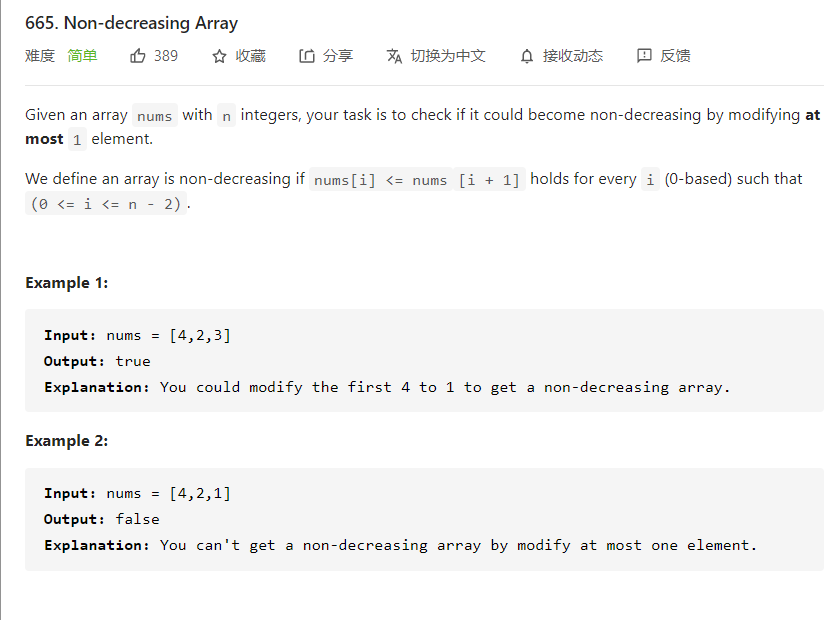




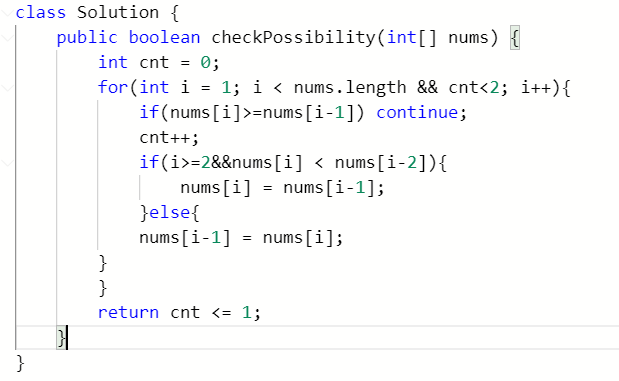
解法：在t里找s的每个字母在不在，t.indexOf(字母，fromIndex)因为abc和adcb return false所以字母次序也要一样



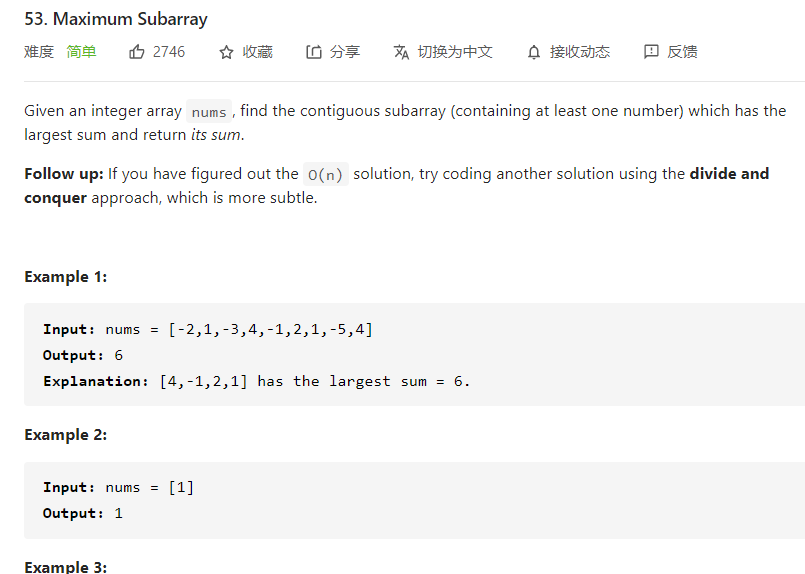
9.



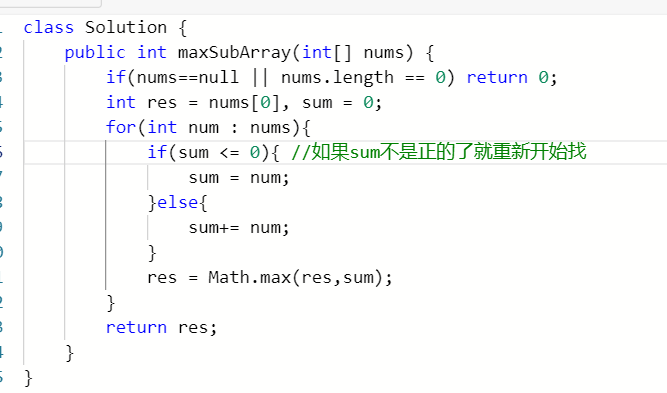
解法：遇到当前的数比之前小的去把之前的数改成和现在一样，这样就不影响之后操作，一个比较特别的情况就是 nums[i] < nums[i - 2]，修改 nums[i - 1] = nums[i] 不能使数组成为非递减数组，只能修改 nums[i] = nums[i - 1]。



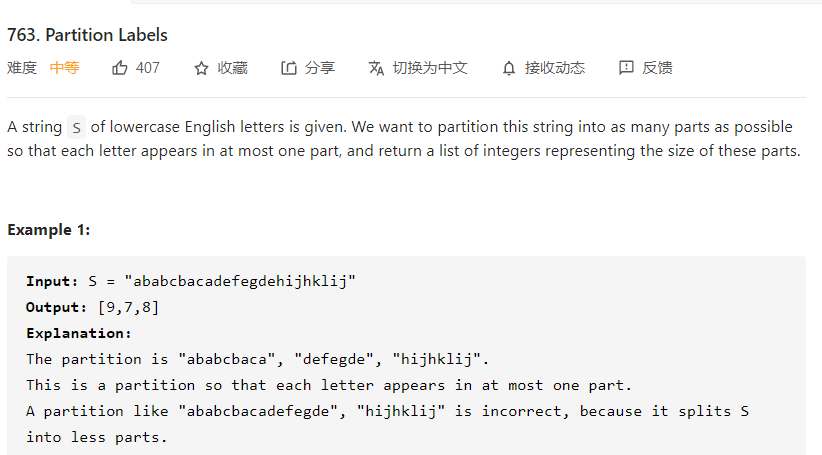
10.



解法：用一个sum去visit 用一个变量记录和最大的字符串当当前和成为负数时,则表明此前序列无法为后面提供最大子序列和,因此必须重新确定序列首项.



11.



解法：先把每个字母出现的最后一个index当到arraylist里，再遍历S，从第一个字母的last index开始看看在这之间的字母的最后一个index

