Programmer String:

|  |
| --- |
| public class ProgrammerString { |
|  |  |
|  | static final String str = "programmer"; |
|  | static final int len = 10; |
|  |  |
|  |  |
|  | public static void main(String[] args) { |
|  | String s = "programmerxxxprozmerqgram"; |
|  | System.out.println(programmerStrings(s)); |
|  | } |
|  |  |
|  | private static int programmerStrings(String s) { |
|  | if (s.length() < len) |
|  | return -1; |
|  |  |
|  | int endIndex = len; |
|  | int incrementIndex = 0; |
|  | int startIndex = len; |
|  |  |
|  |  |
|  | while (endIndex < s.length()) { |
|  |  |
|  | if (Isprogrammer(s.substring(startIndex, endIndex), str)) { |
|  | startIndex = endIndex + 1; |
|  | endIndex = endIndex + 10; |
|  | } else { |
|  | incrementIndex++; |
|  | endIndex++; |
|  | } |
|  | } |
|  |  |
|  | System.out.println(startIndex + " " + endIndex + " " + incrementIndex); |
|  | return incrementIndex; |
|  |  |
|  | } |
|  |  |
|  | private static boolean Isprogrammer(String substring, String programmerString) { |
|  | int len = substring.length(); |
|  | Character[] charArr = programmerString.chars().mapToObj(c -> (char) c).toArray(Character[]::new); |
|  | int minSt = -1; |
|  | int minend = -1; |
|  | int minlen = -1; |
|  | for (int st = 0; st <= len - 1; st++) { |
|  | Set<Character> copySet = new HashSet<Character>(); |
|  | Collections.addAll(copySet, charArr); |
|  | for (int end = st; end <= len - 1; end++) { |
|  | if (copySet.contains(substring.charAt(end))) { |
|  | copySet.remove(substring.charAt(end)); |
|  | if (copySet.size() == 0) { |
|  | if (minlen == -1 || minlen > (end - st)) { |
|  | minSt = st; |
|  | minend = end; |
|  | minlen = end - st; |
|  | return true; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | // System.out.println(str.substring(minSt, minend + 1)); |
|  | } |
|  | return false; |
|  | } |

Given an integer array, the values of the array need to separated into two subsets A and B whose intersection is null and whose unions the entire array. The sum of values in set A must be strictly greater than sum of values in set B, and number of elements in set A must be minimal. Return the values in set A.  
Complete following function definition for the same.

For eg. Given arr ={3,7,5,6,2}, here A would be {6,7}.  
Given arr = {2,3,4,4,5,9,7,8,6,10,4,5,10,10,8,4,6,4,10,1}, here A would be {8, 8, 9, 10, 10, 10, 10}.

**public** **List**<Integer> subsetA(**List**<Integer> arr) {

}

<https://github.com/Sandhya0223/interviewPrep/blob/7e798f70b491c922c0d81ea1a178274f1b48df74/CompanyTests/src/VMWare/SubsetSum.java>

#####################################################################

第一道是username system,很简单，用个hashmap记录出现次数作为后缀即可。

|  |
| --- |
| public int lastStoneWeight(int[] stones) { |
|  | PriorityQueue<Integer> pq = new PriorityQueue<>((o1, o2) -> o2.compareTo(o1)); |
|  | for(Integer in : stones){ |
|  | pq.add(in); |
|  | } |
|  | while(pq.size()>1){ |
|  | Integer poll = pq.poll(); |
|  | if (poll == pq.peek()) { |
|  | pq.remove(); |
|  | }else{ |
|  | pq.add(poll-pq.peek()); |
|  | pq.remove(); |
|  | } |
|  | } |
|  | Integer in= pq.poll(); |
|  | if(in==null) return 0; |
|  | else return in; |
|  | } |

第三道是even subarray，地里也有人整理过，我的做法是先构造一个dp记录当前为止单数出现的次数然后用双指针。因为要求distinct，所以用个set记录。时间复杂度上假设set查找为O(1),总体就O(n^2)了。

|  |
| --- |
| public static int evenSubarray(List<Integer> numbers, int k) { |
|  | int n = numbers.size(); |
|  | Set<List<Integer>> set = new HashSet<>(); |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = i + 1; j <= n; j++) { |
|  | List<Integer> list = numbers.subList(i, j); |
|  | if (valid(list, k)) { |
|  | set.add(list); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | return set.size(); |
|  |  |
|  | } |
|  |  |
|  | private static boolean valid(List<Integer> list, int k) { |
|  | int counter = 0; |
|  |  |
|  | for (int i: list) { |
|  | if ((i & 1) != 0) { |
|  | counter++; |
|  | } |
|  |  |
|  | if (counter > k) { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  | return true; |
|  | } |
|  |  |

#####################################################################

6个选择：  
1. find the inorder array with postorder and preorder arrays  
2. telephone: graph  
**3. Best data structure to manipulate a string with inserting, deleting: Linked list**4. circular linked list: 0 additional pointer  
5. try, catch, final: final  
6. time complexity of a recursion algo with sqrt inside: loglog(n)  
  
3个coding:  
1.  code friends: E=1, M=3, H=5:

easy难度的一道题，就是两个人每天比赛做leetcode题，easy/medium/hard题分别代表一定分值，看哪个人总分值高。地里应该有的。  
2. group anagram

|  |
| --- |
| public static List<String> funWithAnagrams(List<String> s) { |
|  | // Write your code here |
|  | Set<String> uniques = new HashSet<>(); |
|  | List<String> tmp = new ArrayList<>(); |
|  | List<String> result = new ArrayList<>(); |
|  | for (String s1 : s) { |
|  | char[] ta = s1.toCharArray(); |
|  | Arrays.sort(ta); |
|  | tmp.add(new String(ta)); |
|  | } |
|  |  |
|  | for (int i = 0; i < s.size(); i++) { |
|  | if (uniques.add(tmp.get(i))) { |
|  | result.add(s.get(i)); |
|  | } |
|  | } |
|  | Collections.sort(result); |
|  | return result; |
|  | } |

3. climb the hill

efficient janitor minimal trip: <https://leetcode.com/discuss/interview-question/490066/efficient-janitor-efficient-vineet-hackerrank-oa>

#####################################################################

2. build subsequence & sort: "aab" => ["a", "aa", "aab", "ab", "b"]

|  |
| --- |
| static String[] findSubSequences(String s) { |
|  |  |
|  | if(s.length() == 0) { |
|  | String[] ans = {""}; |
|  | return ans; |
|  | } |
|  |  |
|  | String smallAns[] = findSubSequences(s.substring(1)); |
|  | String ans[] = new String [2 \* smallAns.length]; |
|  |  |
|  | int k =0; |
|  | for(int i = 0; i < smallAns.length; i++) { |
|  | ans[k] = smallAns[i]; |
|  | k++; |
|  | } |
|  |  |
|  | for(int i = 0; i < smallAns.length; i++) { |
|  | ans[k] = s.charAt(0) + smallAns[i]; |
|  | k++; |
|  | } |
|  |  |
|  | return ans; |
|  | } |

#####################################################################

选择题：  
postorder, preorder find inorder DBEAFCG  
circular linked list extra pointers: 0  
data structure to operate string with insert, delete etc: linked list  
queue page replacement什么的，讲道理我题目没读很懂然后背的答案后来自己硬往上套的。。。答案196  
telephone : graph  
graph time complexity adjacent matrix and list

running time - logn  
running time - loglogn  
extra pointer for circular queue: 0  
try catch finally - output content in finally  
structure can insert, delete, re-arrange substring - linked list based  
telephone data structure - graph

#####################################################################

3)  Longest subarray with at most k normal letter.

#########################################################################选择题  
  
    #1 Queue using circular LL              答案 0  
  
    #2 Multiple Choice  
    FIFO system，4个page，一开始都没有load，先access100个不同的page，按某个顺序，然后再反着来。总共会有多少次page fault  
    4 pages frame   access 100 pages          
                                            答案 196  
  
    !!!!!!!!!!!!!!  
    #3 代码看时间复杂度   Procedure A(n)       答案 O(log log n)  
  
    !!!!!!!!!!!!!!  
    #4 DFS in a graph                       答案 O(n^2)    O(n+e)  
  
    #5 best data structure to represent telephone network。    答案???  m-ary tree  OR  graphs  
  
    #6 find the  output                          答案  4 errors  
  
    !!!!!!!!!!!!!!  
    # 给preorder和postorder的序列，求inorder        答案  D B E A F C G  
  
  
    !!!!!!!!!!!!!!  
    # 给出一个数据结构 能够支持string插入删除合并       答案 LL  
  
    !!!!!!!!!!!!!!  
    # mystery algorithm         答案 1  
  
    # try catch and finally block    
  
'''  
  
def mystreyAlg(a, b):  
    x = a  
    y = b  
    while x != y:  
        if x>y:  
            x = x-y  
        if x<y:  
            y = y-x  
    return x  
  
print("################## mystreyAlg ########################")  
print(mystreyAlg(2437, 875))  
  
##########################################################################  
'''  
1   Usernames System  
create the username portion of a registration sys that requires all usernames are unique.  
  
#输入list：[bob, alice, bob, alice,bob] 输出：[bob,alice,bob1,alice1,bob2]  
  
'''  
def usernamesSystem(inputList):  
uniqueNameMap = {} # map name to curr number  
resList = []  
for name in inputList:  
  if name not in uniqueNameMap:  
   uniqueNameMap[name] = 0  
  else:  
   uniqueNameMap[name] += 1  
   name = name + str(uniqueNameMap[name])  
  
  resList.append(name)  
return resList  
  
print("################## usernamesSystem ########################")  
print(usernamesSystem(["bob", "alice", "bob", "alice", "bob"]))  
  
  
  
# ####################################################################  
'''  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
2   Building the subseq  
a subseq of a string is obtained by deleting zero or more char from the string  
while maintaining order,     
EX  "xyz"   substring are. "x" "xy" "xz" "xyz" "y" "yz" "z"  
want to generate an array of all subseq of a given string, omitting the emoty string  
  
IF have to be continuous, use sliding windows two pointer solution  
[https://leetcode.com/problems/subarray-sum-equals-k/](https://link.1point3acres.com/?url=https%3A%2F%2Fleetcode.com%2Fproblems%2Fsubarray-sum-equals-k%2F)  
'''  
def buildSequences(string):  
    res = [""]  # init for appending single char  
    for char in string:  
        # res += [r + char for r in res] # this line is same as below  
        addition = []   # for extending substring  
        for existing\_str in res:  
            addition.append(existing\_str + char)  
        res = res + addition  
    res.remove("")  
    res.sort()  
    return res  
  
  
print("################## buildSequences ########################")  
print( buildSequences("abc"))  
print( buildSequences("ebda"))  
  
  
####################################################################  
# 3   even subarray    
  
def evenSubarray(numbers, k):  
    left = 0  
    subOddCount = 0  
    res = 0  
  
    for right, num in enumerate(numbers):  
  
        # find a odd num  
        if num % 2 == 1:  
            subOddCount += 1  
  
        if subOddCount > k:  
            while numbers[left] % 2 == 0:   # skip all even num  
                left += 1   # move left pointer to right until find the next odd  
  
            left += 1   # move left again to exclude the left number from subarray  
            subOddCount -= 1  
  
        # calc num of sub array ends with the right  
        res += (right - left + 1)  
  
    return res  
  
  
print("################## evenSubarray ########################")  
print (evenSubarray([1,2,3,3], 1)) # 6  
print (evenSubarray([1,2,3,4], 1)) # 8  
print (evenSubarray([1,2,3,4], 2)) # 10  
print (evenSubarray([1,2,3,4], 3)) # 10  
print (evenSubarray([1,2,2,3,3,4], 1)) # 2+4+6 = 12  
  
  
####################################################################  
'''  
# 4  
The perfect team。  
就给你一个字符串，每个char代表一个人擅长的科目，然后要组队，  
要求队伍里五个人各擅长一门科目，求可以最多可以组几个队伍。数数取最小。

|  |
| --- |
| static int differentTeams(String skills) { |
|  | Map<Character, Integer> skillCount = new HashMap<>(); |
|  |  |
|  | for (Character skill: skills.toCharArray()) { |
|  | if (skillCount.containsKey(skill)) { |
|  | skillCount.put(skill, skillCount.get(skill) + 1); |
|  | } else { |
|  | skillCount.put(skill, 1); |
|  | } |
|  | } |
|  |  |
|  | // if there is no 5 skills, then no groups can be formed |
|  | if (skillCount.size() < 5) return 0; |
|  |  |
|  | int maxPossibleGroup = Integer.MAX\_VALUE; |
|  |  |
|  | for (int count: skillCount.values()) { |
|  |  |
|  | if (count < maxPossibleGroup) { |
|  | maxPossibleGroup = count; |
|  | } |
|  | } |
|  | return maxPossibleGroup == Integer.MAX\_VALUE ? 0 : maxPossibleGroup; |
|  | } |

'''  
from collections import Counter     # produce a frequency table (key, freq)  
  
def differentTeams(skills):  
    freq\_table = Counter(skills)    #ordered  
    least\_common\_skill, least\_common\_count = freq\_table.most\_common()[-1]  
    return least\_common\_count  
  
# ############ OR ##############  
# def differentTeams(skills):  
  
#     skillSet = {}  
  
#     for skill in skills:  
#         if skill not in skillSet:  
#             skillSet[skill] = 1  
#         else:  
#             skillSet[skill] += 1  
#     return skillSet[min(m, key=m.get)]  
  
print("################## differentTeams ########################")  
print(differentTeams("pcmbzpcmbz"))  
  
  
##########################################################################  
'''  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
5   shift strings  
left shift:  
right shift:  
'''  
def getShiftedString(s, leftShifts, rightShfts):  
    # calc total shift  
    shiftLeft = leftShifts%len(s) - rightShfts%len(s)  
  
    return s[shiftLeft:] + s[:shiftLeft]  
  
print("################## getShiftedString ########################")  
print(getShiftedString('abcde', 3, 1))  
print(getShiftedString('abcde', 0, 2))  
  
**Shift JAVA:**

|  |
| --- |
|  |
| public static String getShiftedString(String str, int leftShifts, int rightShifts) { |
|  | if (str == null || str.length() == 0) { |
|  | return null; |
|  | } |
|  | if (leftShifts - rightShifts == 0) { |
|  | return str; |
|  | } |
|  |  |
|  | int shift = (leftShifts - rightShifts + str.length()) % str.length(); // get remainder |
|  | return str.substring(shift) + str.substring(0, shift); |
|  | } |
|  |  |

####################################################################  
'''  
6   Image Editing   === maximal square  
  
给一个0/1的二维数组 找到其中最大的全部为1的正方形

class Solution {

public int maximalSquare(char[][] matrix) {

int m = matrix.length;

if (m == 0) return 0;

int n = matrix[0].length;

int[][] dp = new int[m+1][n+1];

int ans = 0;

for (int i = 1; i <= m; i++){

for (int j = 1; j <= n; j++){

if (matrix[i-1][j-1] != '0'){

dp[i][j] = Math.min(dp[i-1][j], Math.min(dp[i-1][j-1], dp[i]

[j-1]))+1;

ans = Math.max(ans, dp[i][j]);

}

}

}

return ans \* ans;

}

[https://leetcode.com/problems/maximal-square/description/](https://link.1point3acres.com/?url=https%3A%2F%2Fleetcode.com%2Fproblems%2Fmaximal-square%2Fdescription%2F)  
  
'''  
########################################################  
def checkSquareIsWhite(arr, top, left, size) -> bool:  
    right = left+size  
    bottom = top+size  
    min\_el = 1  
  
    for i in range(top, bottom):  
        min\_row\_el = min(arr*[left:right])  
        min\_el = min(min\_el, min\_row\_el)  
  
    return min\_el > 0  
  
def largestMatrix(arr):  
    numrows = len(arr)  
    numcols = len(arr[0])  
    size = numrows  
  
    while size >= 1:  
        for top in range(0, numrows-size+1):  
            for left in range(0, numcols-size+1):  
                if checkSquareIsWhite(arr, top, left, size):  
                    return size  
        size -= 1  
  
    return 0 # submatrix not found  
  
print("################## image editing ########################")  
grid=[  [1,1,1,1,1],  
        [1,1,1,0,0],  
        [1,1,1,0,0],  
        [1,1,1,0,0],  
        [1,1,1,1,1] ]  
print(largestMatrix(grid))  #== 3  
  
grid=[  [1,1,1,1,1],  
        [1,1,1,0,0],  
        [1,1,0,0,0],  
        [1,1,1,0,0],  
        [1,1,1,1,1] ]  
print(largestMatrix(grid))  #== 2  
  
  
####################################################################  
# 7 group anagram*

*A screenshot of a cell phone

Description automatically generated  
'''  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
#*[*https://leetcode.com/problems/group-anagrams/description/*](https://link.1point3acres.com/?url=https%3A%2F%2Fleetcode.com%2Fproblems%2Fgroup-anagrams%2Fdescription%2F) *'''  
def groupAnagrams(strs):  
    # Assumption: string len is limited  
    # hash the string into an unique value, use multiple instead of add  
  
    hashmap = {} # map the sorted string to a list of the original strings  
  
    for s in strs:  
        # sort the string by char,     相当于根据string里面char出现的个数encode  
        temp = list(s)  
        temp.sort()  
        key = "".join(temp)  
  
        if key not in hashmap:  
            hashmap[key] =  
        else:  
            hashmap[key].append(s)  
  
    res =  [val for key, val in hashmap.items()]  
  
    return res  
  
print("################## groupAnagrams ########################")  
print(groupAnagrams(["eat", "tea", "tan", "ate", "nat", "bat"]))  
  
####################################################################  
# 8 team formation 2  
# Team formation2, 给个upper bound 和lower bound，至少选k个人组队，问至少有多少种  
# 组队方式  combination formular  
  
import math  
def countTeams(skills, k, l, r):  
  
    def skills\_in\_range(skill):  
        return skills <= r and skills >= l  
  
    filt = filter(skills\_in\_range, skills)  
  
    n = len(filt)  
    temp = k  
    res = 0  
    while temp <= n:  
        res += math.factorial(n) / (math.factorial(temp) \* math.factorial(n-temp))  
    return res  
  
  
print("################## countTeams ########################")  
  
  
####################################################################  
# 9 字符串计分  
# 给定字符串按照规则计分 给两个字符串,其中有只可能是EMH这三个字母的组合,E = 1, M = 3, H = 5, 比  
# 如EHH和EME分别相当于1 + 5 +5 = 11和1 + 3 + 1 = 5,返回代表数值更大的字符串。  
##################  
# 小明和小红比赛做leetcode，做了Hard 得5分，M 3分， E 1分， 两人每天做一题，给你两个长度相同的String 如 HHH，  
# HME 代表小明和小红这三天做的题目，求最后谁分数高，tie的话return “TIE”  
def encodeGame(question):  
    if question == 'H':  
        return 5  
    if question == 'M':  
        return 3  
    if question == 'E':  
        return 1  
  
def countScore(str1, str2):  
    diff = 0  
    for i in range(len(str1)):  
        diff += encodeGame(str1) - encodeGame(str2)  
  
    if diff > 0:  
        return str1  
    elif diff == 0:  
        return "TIE"  
    else:  
        return str2  
  
  
print("################## countScore ########################")  
print(countScore("EMM", "HEE"))  
print(countScore("MMM", "HEE"))  
  
  
####################################################################  
# 10 相似string  
# 给定一组string数组，排除其中多余的相似string， 如 abca 和 aabc 相似，  
# 每个字母数量一样就是相似，不用in place，很简单，就不说了  
  
def excludeSimilarString(stringList):  
    seen = set()  
    res = []  
  
    for string in stringList:  
        temp = list(string)  
        temp.sort()   # sort so that char  
        key = "".join(temp)  
  
        if key not in seen:  
            seen.add(key)  
            res.append(string)  
        # else, abandon  
  
    return res  
  
print("##################### excludeSimilarString #####################")  
print(excludeSimilarString(["abca", "aabc","abac","aaaabc", "app", "pap"]))  
  
  
  
####################################################################  
# 11  List 碰撞   ===  crashing the stone  
# 一个int数组，每次选两个最大的进行碰撞，如果相同则不加入原数组，如果不同则把差加入，持续这个过程，  
# 直到只剩一个数‍‌‍‌‌‍‍‍‍‌‌‌‍‌‍‌‌‍‍字或者没有数字，返回剩下的数字或者0，用个heap解决，很简单  
  
# PQ 解法  
import heapq  
def lastStoneWeight(arr):  
    pq = []  
    for num in arr:  
        heapq.heappush(pq, -num)    
    while len(pq) > 1:  
        first = heapq.heappop(pq)  
        second = heapq.heappop(pq)  
  
        if first != second:  
            heapq.heappush(pq, first-second);  
        print  
  
    if len(pq) == 1:  
        return -pq[0]  
    return 0  
  
# def lastStoneWeight(arr):  
#     while(len(arr) > 1):  
#         arr.sort(reverse = True)  
#         maxNum1 = arr[0]  
#         maxNum2 = arr[1]  
#         arr.remove(maxNum1)  
#         arr.remove(maxNum2)  
#         if maxNum1 != maxNum2:  
#             arr.append(maxNum1 - maxNum2)  
#     if len(arr) == 1:  
#         return arr[0]  
#     return 0*

|  |
| --- |
| public int lastStoneWeight(int[] stones) { |
|  | PriorityQueue<Integer> pq = new PriorityQueue<>((o1, o2) -> o2.compareTo(o1)); |
|  | for(Integer in : stones){ |
|  | pq.add(in); |
|  | } |
|  | while(pq.size()>1){ |
|  | Integer poll = pq.poll(); |
|  | if (poll == pq.peek()) { |
|  | pq.remove(); |
|  | }else{ |
|  | pq.add(poll-pq.peek()); |
|  | pq.remove(); |
|  | } |
|  | } |
|  | //Integer in= pq.poll(); |
|  | //if(in==null) return 0; |
|  | //else return in;  Return(pq.size()==0) ? pq.poll(); |
|  | } |

*print("################# listCollision ######################")  
print(listCollision([1,3,9]))           # 5  
print(listCollision([4,5,1,6,8,2]))     # 0  
  
  
####################################################################  
# 12 intelligent substring  
# Intelligent Substring。给一个字符串，只有小写英文字母。然后会对应1或者0。  
# 求最长的子字符串的长度，要求0的个数不超过k个。sliding window。  
  
def encode(c, charValue):  
    return 1 if charValue[ord(c) - ord('a')] == '0' else 0  
  
def getSpecialSubstring(s,k,charValue):  
    left = 0  
    subNormalCount = 0  
    maxLen = -1  
  
    for i, c in enumerate(s):  
        subNormalCount += encode(c,charValue)  
        if subNormalCount > k:  
            while encode(s[left],charValue) == 0: # move left until find the next closest odd number  
                left += 1  
            left += 1 # move left again to exclude the left number from subarray  
            subNormalCount -= 1  
        subLen = i - left + 1  
        maxLen = max(maxLen, subLen)  
    return maxLen  
  
  
print("################## getSpecialSubstring ########################")  
print (getSpecialSubstring("abcde", 2, "10101111111111111111111111")) # 5  
print (getSpecialSubstring("abcde", 1, "10101111111111111111111111")) # 3  
  
  
  
##########################################################################  
# 13  
# 前天晚上写的OA，75分钟6选择+3coding，一道coding新题，挂在上面了。。。  
# 以下内容需要积分高于 120 您已经可以浏览  
  
# e-mail thread: 题目很长，说是有一个list，每个element都是一封字符串表示的邮件，格式是 “*[*a@gmail.com*](mailto:a@gmail.com)*,*[*b@gmail.com*](mailto:b@gmail.com)*, abc（内容）”  
# 之后如果b对a回复了邮件，将会是“*[*b@gmail.com*](mailto:b@gmail.com)*,*[*a@gmail.com*](mailto:a@gmail.com)*, cde---abc"，邮件是已经按发送顺序排序好的  
# 主要是让你写字符串处理，让你把不同对话归类（thread）到一起并标序号，就像reddit或者老贴吧的楼中楼一样。  
# 举个例子，三封邮件：  
  
# 所以的例子都让你以为和后面的邮件内容无关，只用对话双方邮箱地址Hash的话能通过绝大部分test case，  
# 经过自己写test case测出来还是要用内容区分的。举个例子：  
# 以下内容需要积分高于 120 您已经可以浏览  
  
def email\_thread(emails):  
    """  
    把每个字符串处理成 a邮箱， b邮箱，新内容，老内容  
    先用对话双方的邮箱有序拼接字符串作为hash key值，  
    里面再嵌套一个dict，key值是thread id，value值是邮件内容 以及 对话id  
    用ab邮箱匹配到以后遍历里面的thread id，用老内容和每一个value的内容比较  
    匹配成功就修改内容 = 新内容+老内容，否则就是新thread  
    """  
    res = []  
    dialogue\_dict = {}  
    thread\_id = 0 # self-incrementing  
    for email in emails:  
        sender, receiver, content = email.split(", ")  
        content\_l = content.split("---")  
  
        if sender > receiver:  
            sender, receiver = receiver, sender  
        dialogue = (sender, receiver)  
  
        if dialogue not in dialogue\_dict:  
            thread\_id += 1  
            dialogue\_dict[dialogue] = {  
                thread\_id: content\_l  
            } # key is thread\_id, value is (mail\_content, session\_id)  
            res.append([thread\_id, len(content\_l)])  
        else: # for existing dialog  
            thread\_dict = dialogue\_dict[dialogue]  
            old\_contents = content\_l[1:]  
  
            found\_old = False  
  
            for (tid, thread\_content) in thread\_dict.items():  
                if thread\_content == old\_contents:  
                    found\_old = True  
                    thread\_dict[tid] = content\_l  
                    res.append([tid, len(content\_l)])  
                    break  
  
            if not found\_old:  
                thread\_id += 1 # create a new thread  
                thread\_dict[thread\_id] = content\_l  
                res.append([thread\_id, len(content\_l)])  
    return res  
  
  
print("###################### email\_thread #####################")  
  
emails = [  
    "a@gmail.com,*[*b@gmail.com*](mailto:b@gmail.com)*, abc",  
    "c@gmail.com,*[*d@gmail.com*](mailto:d@gmail.com)*, hhh",  
    "b@gmail.com,*[*a@gmail.com*](mailto:a@gmail.com)*, cde---abc",  
]  
print(email\_thread(emails))     # [[1,1],[2,1],[1,2]]  
  
emails = [  
    "a@gmail.com,*[*b@gmail.com*](mailto:b@gmail.com)*, abc",  
    "a@gmail.com,*[*b@gmail.com*](mailto:b@gmail.com)*, abc",  
    "b@gmail.com,*[*a@gmail.com*](mailto:a@gmail.com)*, cde---abc",  
    "b@gmail.com,*[*a@gmail.com*](mailto:a@gmail.com)*, def---cde---abc",  
]  
print(email\_thread(emails)) # [[1,1],[2,1],[1,2],[1,3]]  
  
  
  
# ##########################################################################  
# 14  
# break a palindrome。给定一个palindrome。要改变一个char，使得新的string不是palindrome。  
# 而且要是lexicocographically最小的，不然就返回"IMPOSSIBLE"。  
# solution: 找左半边不是"a"的, 只要不是中间的找到一个改成a 就成功  
#############################  
# 回文字串改一个字符使其lexicographically的value比原回文字串少，并且不能再是回文字串。如若不存在，return ”IMPOSSIBLE“。  
# 举例：输入 aba。算法算出aaa但仍是回文字串，所以输出IMPOSSIBLE。 输入abba，输出aaba。 super easy题  
  
def breakAPalindrome(palindrome):  
    res = ""  
    idx = -1  
    found = False  
    for i, c in enumerate(palindrome[:len(palindrome)//2]):    
        if c != 'a':  
            idx = i  
            found = True  
            break  
  
    # In Python, strings are immutable, so you can't change their characters in-place.  
    # So need to reconstruct the string ~!!!!!!          
    if found:  
        for i in range(len(palindrome)):  
            if i != idx:  
                res += palindrome  
            else:  
                res += "a"  
        return res  
    else:  
        return "IMPOSSIBLE"  
  
  
print("##################### breakAPalindrome ######################")  
print(breakAPalindrome("abbba"))  
print(breakAPalindrome("abba"))  
print(breakAPalindrome("aba"))  
print(breakAPalindrome("aaaamaaaa"))  
  
  
# ##########################################################################  
# 15    buy\_books  
# 第三道是比较少见的背包问题，大意是有n套丛书，每套书包含X本书，按套卖，每套价值Y元。  
# 所以有两个等长input array分别代表每套书包含几本书和每套书卖多少钱。  
# 另一个input是int，代表你的预算。求最多能买多少本书。  
  
# 和普通的 knapsack 不一样，普通knapsack是限制最多装的容量，求最大总和价值  
# 这题是限制最大总和价值（预算），求最多能带多少书  
# 解法没有不同，只需把成本数组当成重量，预算当成容量上限就行了  
  
def buy\_books(item\_sizes, item\_costs, budget):  
    num\_items = len(item\_sizes)  
  
    dp = [[0 for \_ in range(budget+1)] for \_ in range(num\_items+1)]  
  
    # Build table K[][] in up bottom manner  
    for first\_n\_items in range(1,num\_items+1):  
        for remaining\_budget in range(1,budget+1): #  
            # if first\_n\_items == 0 or remaining\_budget == 0: # no remaining budget or items  
            #     continue # dp[first\_n\_items][remaining\_budget] = 0 #  
            if item\_costs[first\_n\_items-1] > budget: # cost of nth item is at item\_costs[first\_n\_items-1] because `n` is one-indexed  
                dp[first\_n\_items][remaining\_budget] = dp[first\_n\_items-1][remaining\_budget] # don't buy this item if its cost is greater than budget  
            else:  
                nth\_cost = item\_costs[first\_n\_items-1]  
                no\_select\_num = dp[first\_n\_items-1][remaining\_budget]  
                select\_val = item\_sizes[first\_n\_items-1] # NOTE: item\_sizes[first\_n\_items-1] is the val we're interested in. it could be weight, could be anything that leads to the final result  
                select\_num = dp[first\_n\_items-1][remaining\_budget-nth\_cost] + select\_val  
                dp[first\_n\_items][remaining\_budget] = max(no\_select\_num, select\_num)  
  
    return dp[num\_items][budget] # the bottom-right most item in table  
  
print("##################### buy\_books ######################")  
print( buy\_books([60, 100, 120],[10, 20, 30],50) )  # 220  
  
  
  
#######################################################################  
'''  
16    climb the hill     经典*[*https://www.geeksforgeeks.org/mi ... ray-non-increasing/*](https://link.1point3acres.com/?url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fminimum-incrementdecrement-to-make-array-non-increasing%2F) *'''  
########################################################  
import heapq  
def climbTheHill(slope):  
    res1 = 0  
    res2 = 0  
    dif = 0  
  
    pq = [] # min-heap - descending order: sl can't be greater than smallest in first k-1 elements  
    for sl in slope:  
        if len(pq) > 0 and pq[0] < sl:  
            dif = sl - pq[0]  
            res1 += dif  
            heapq.heappop(pq)  
            heapq.heappush(pq, sl)  
        heapq.heappush(pq, sl)  
  
    pq = [] # max-heap - ascending order: sl can't be smaller than largest in first k-1 elements  
    for sl in slope:  
        if len(pq) > 0 and -pq[0] > sl:  
            dif = -pq[0] - sl  
            res2 += dif  
            heapq.heappop(pq)  
            heapq.heappush(pq, -sl)  
        heapq.heappush(pq, -sl)  
  
    return min(res1,res2)  
  
  
print("###############################climb the hill#############################")  
print( climbTheHill([0,1,2,5,5,4,4]) )  # 2  
print( climbTheHill([7,5,6,5,2,1,0]) )  # 1  
print( climbTheHill([9,8,7,2,3,3]) )    # 1  
  
##########################################################################  
# 17  
# greedy很简单。给一堆bag。  
# 重量大于1.01小于3.0。如[1.01, 1.01, 1.99, 2.5]，一次可以总共拿小与等于3.0重的东西。问最少拿多少次。  
# 注意这个至多每次只能拿2个，最少拿1个  
  
def minTimes(items):  
    items.sort()  
    i = 0  
    countBags = 0    
    while i < len(items):  
        j = i  
        tempTotal = 0  
  
        # creat one bag    
        while tempTotal + items <=3:  
            tempTotal += items[j]  
            j += 1  
  
        countBags += 1  
        i = j  
  
    return countBags  
  
print("############################### minTimes #############################")  
print(minTimes([1.01, 1.01, 1.99, 2.5]))*