<https://bobbyliujb.github.io/2018/08/21/vmware/>

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=543725&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26searchoption%5B3089%5D%5Bvalue%5D%5B5%5D%3D5%26searchoption%5B3089%5D%5Btype%5D%3Dcheckbox%26searchoption%5B3019%5D%5Bvalue%5D%3D1%26searchoption%5B3019%5D%5Btype%5D%3Dradio%26searchoption%5B3046%5D%5Bvalue%5D%3D40%26searchoption%5B3046%5D%5Btype%5D%3Dradio%26sortid%3D311%26orderby%3Ddateline>

1. 知道树的pre和post order 求 inorder

2. 给了pseudo code‍‍‍‍‍‍‍‌‌‍‌‌‌‌‌‍‍‌ 求最后答案，输入 a 与 b 开始 a 是 x b是y 大的数等于大的减小的 直到x和y相等

3. 哪个数据结构可以简单实现插入 删除 concatenate 重新排序

4. 一个while循环 终止条件是 a比n大， 每个循环 a\*2

5. try catch的结构最后的输出

6. 哪个结构适合模拟电话网络

1‍‍‍‍‍‍‍‌‌‍‌‌‌‌‌‍‍‌. Big O in Graph. 分adjacency matrix和adjacency list, V^2 和 V+E

2. Single circular linked list, 有head和tail，需要加几个指针做到O(1) push和poll的queue，0

3. FIFO policy for page faults, 196

4. 递归时间复杂度，https://www.1point3acres.com/bbs/thread-543077-1-1.html，loglogn5. Telephone data structure, Graph

6. try-catch-finally，finally中有的

#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

1. find the inorder array with postor‍‍‍‍‍‍‍‌‌‍‌‌‌‌‌‍‍‌der and preorder arrays

<https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/>

2. telephone: graph

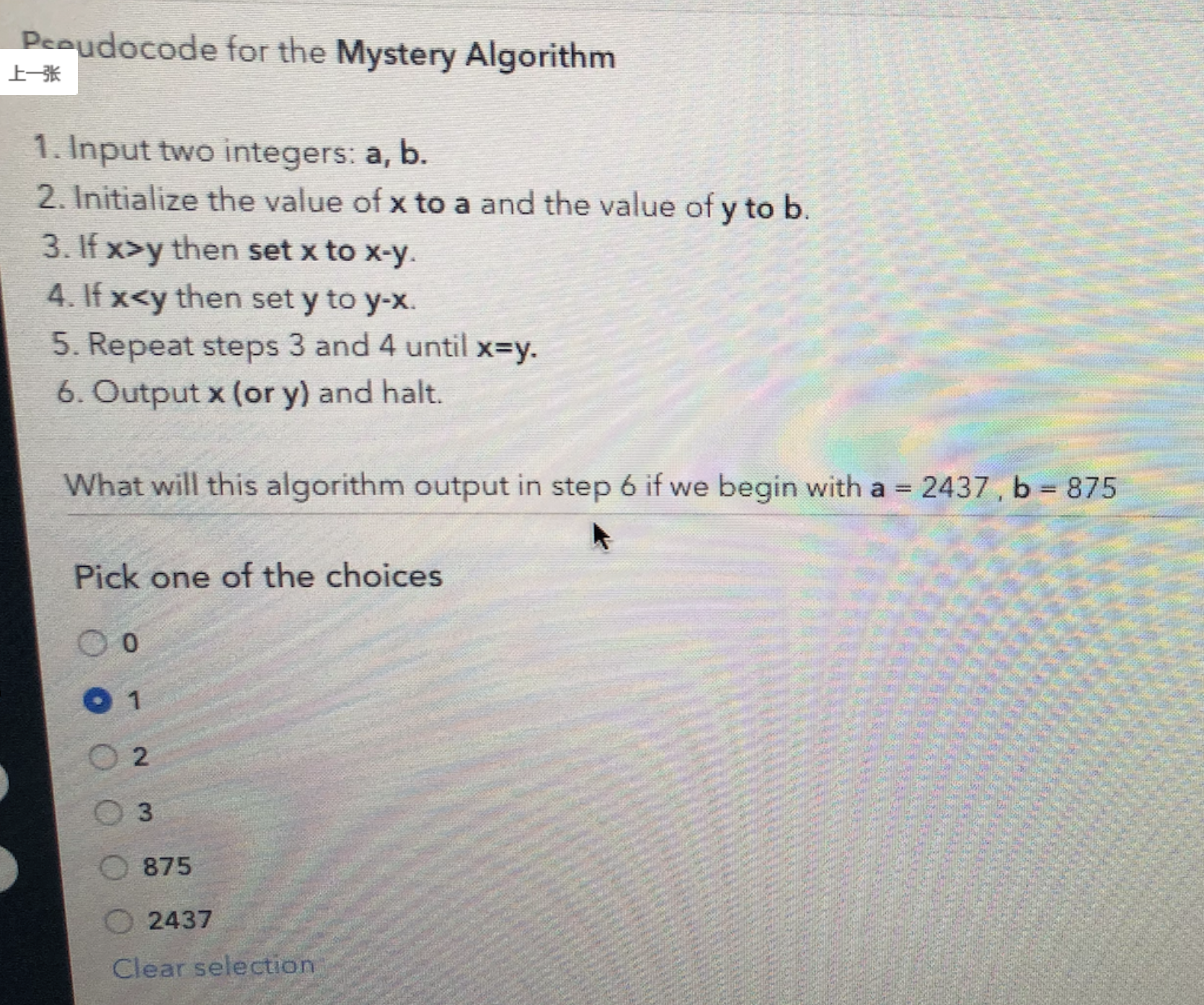
3. Best data structure to manipulate a string with inserting, deleting: Linked list

4. circular linked list: 0 additional pointer

5. try, catch, final: final

6. time complexity of a recursion algo with sqrt inside: loglog(n)

找最大公约数：这个题目是1



1. Number of subarrays 是变种题目，这个位置是要求的至多

<https://www.geeksforgeeks.org/number-subarrays-m-odd-numbers/>

public static int countSubarrays(int a[], int n, int m)

{

int count = 0;

int prefix[] = new int[n];

int odd = 0;

// traverse in the array

for (int i = 0; i < n; i++)

{

prefix[odd]++;

// if array element is odd

if (a[i] % 2 == 1)

odd++;

// when number of odd

// elements >= M

if (odd >= m)

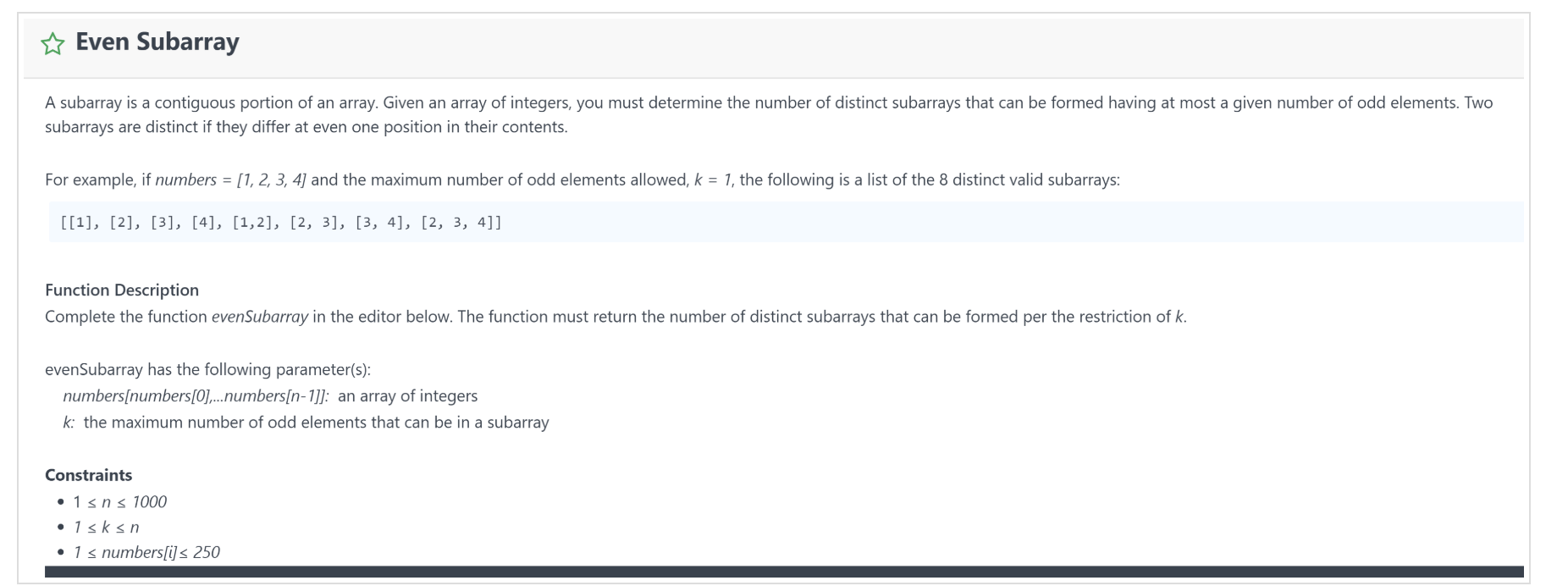
count += prefix[odd - m];

}

return count;

}

**Even subarray:**



java solution:

public int evenSubarray(int[] numbers, int k) {

int left = 0;

int subOddCount = 0;

int res = 0;

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

int right = numbers[i];

int num = numbers[i];

if(num % 2 == 1){subOddCount++;}

if(subOddCount > k){

while(numbers[left] % 2 == 0){left+=1;}

left+=1;

subOddCount-=1;

}

res += (right-left+1);

}

return res;

}

// python solution

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

1. 第三题是给一个0/1的二维数组 找到其中最大的全部为1的正方形
2. queue to implement linked list

|  |
| --- |
| // Java program for linked-list implementation of queue    // A linked list (LL) node to store a queue entry  class QNode {  int key;  QNode next;    // constructor to create a new linked list node  public QNode(int key)  {  this.key = key;  this.next = null;  }  }    // A class to represent a queue  // The queue, front stores the front node of LL and rear stores the  // last node of LL  class Queue {  QNode front, rear;    public Queue()  {  this.front = this.rear = null;  }    // Method to add an key to the queue.  void enqueue(int key)  {    // Create a new LL node  QNode temp = new QNode(key);    // If queue is empty, then new node is front and rear both  if (this.rear == null) {  this.front = this.rear = temp;  return; }    // Add the new node at the end of queue and change rear  this.rear.next = temp;  this.rear = temp;  }    // Method to remove an key from queue.  QNode dequeue()  {  // If queue is empty, return NULL.  if (this.front == null)  return null;    // Store previous front and move front one node ahead  QNode temp = this.front;  this.front = this.front.next;    // If front becomes NULL, then change rear also as NULL  if (this.front == null)  this.rear = null;  return temp;  }  }    // Driver class  public class Test {  public static void main(String[] args)  {  Queue q = new Queue();  q.enqueue(10);  q.enqueue(20);  q.dequeue();  q.dequeue();  q.enqueue(30);  q.enqueue(40);  q.enqueue(50);    System.out.println("Dequeued item is " + q.dequeue().key);  }  } |

1. Tree Traversal: given post order, pre order, figure out in order.

2.‍‍‍‍‍‍‍‌‌‍‌‌‌‌‌‍‍‌ Pseudo code for mystery algorithm

3. FiFo policy for page replacement

4. Guess the data structure. In my case the answer is linked list

5. Big O for DFS in a graph

6. Try, Catch, Final in JAVA

7-9 Coding

7. The coder friends, Easyyyyyyyyy someone posted before.

8. Efficient Janitor, Use greedy algo and sort the list at the first.

9. Crashing Stones, Use heapq(python)

Minimum cost, climb the hill

public int getMinimumCost(int[] input) {

if (input == null || input.length == 0) {

return 0;

}

return Math.min(getMinimumCost(input, false), getMinimumCost(input, true));

}

private int getMinimumCost(int[] input, boolean reverse) {

int[] sorted = Arrays.copyOf(input, input.length);

Arrays.sort(sorted);

// reverse the list, 单独抽出一个函数

if (reverse) {

sorted = Collections.reverse(Arrays.asList(a)).toArray(new int[input.length]);

}

int[][] dp = new int[input.length][sorted.length];

dp[0][0] = Math.abs(input[0] - sorted[0]);

for (int i = 1; i < input.length; i++) {

dp[i][0] = dp[i - 1][0] + Math.abs(input[i] - sorted[0]);

}

for (int j = 1; j < sorted.length; j++) {

dp[0][j] = Math.min(dp[0][j - 1], Math.abs(input[0] - sorted[j]));

}

for (int i = 1; i < input.length; i++) {

for (int j = 1; j < sorted.length; j++) {

dp[i][j] = Math.min(dp[i][j - 1], dp[i - 1][j] + Math.abs(input[i] - sorted[j]));

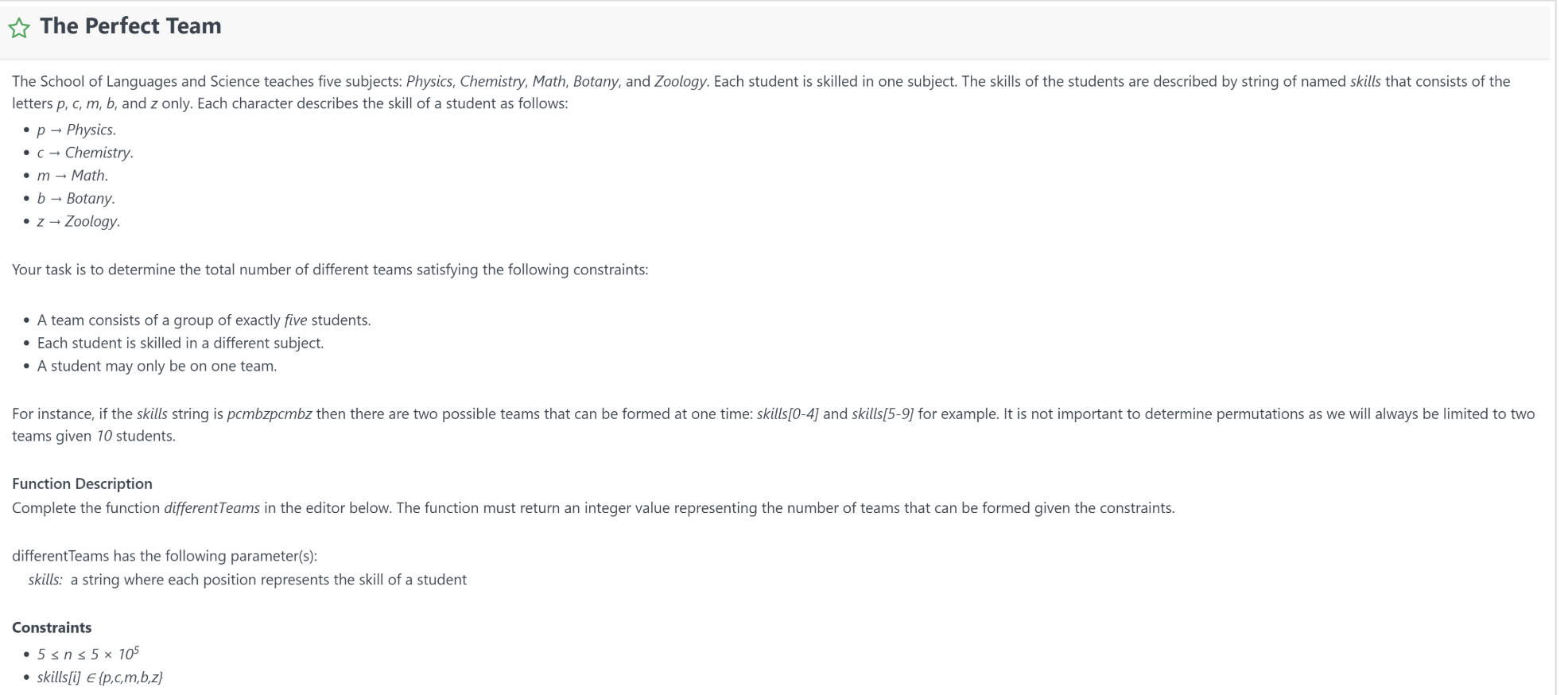
}

}

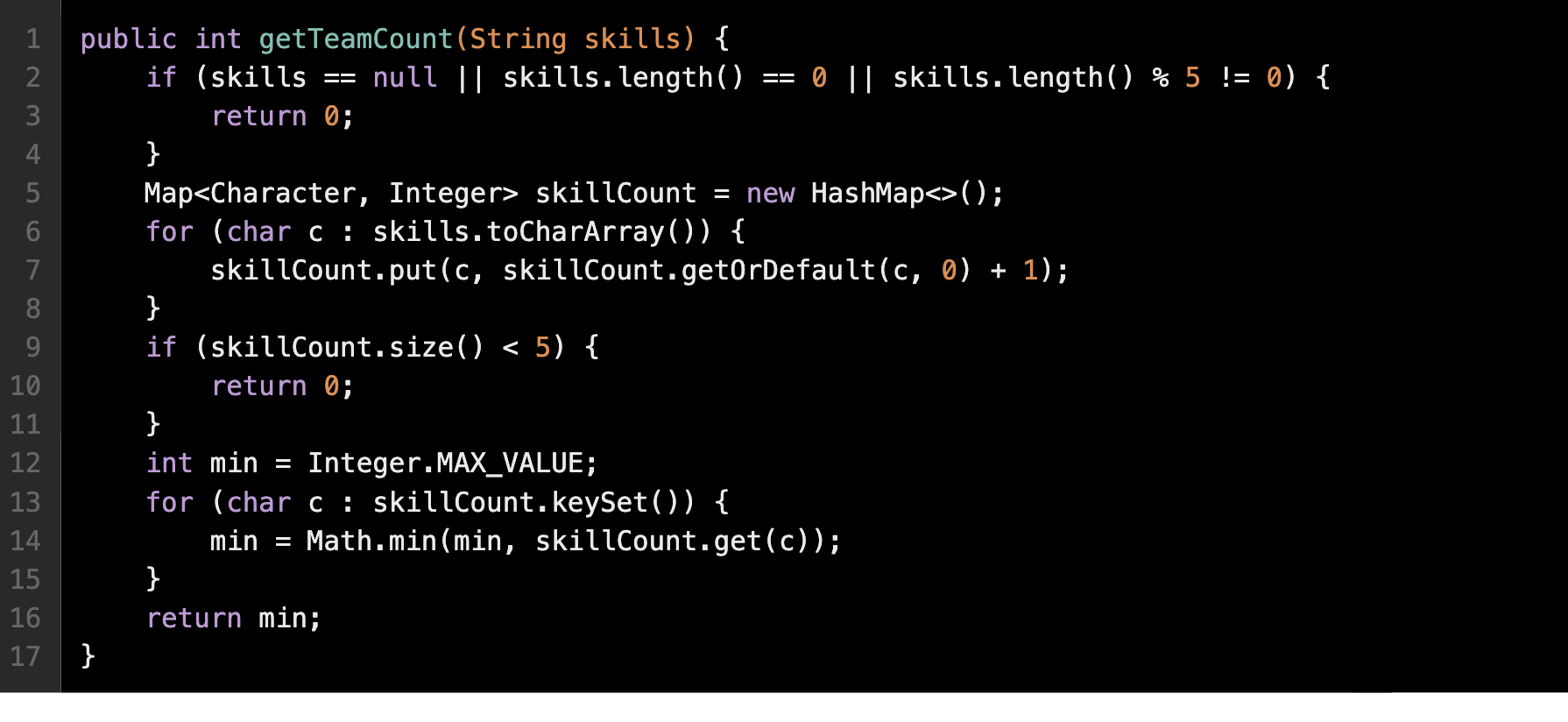
return dp[input.length - 1][sorted.length - 1];

}

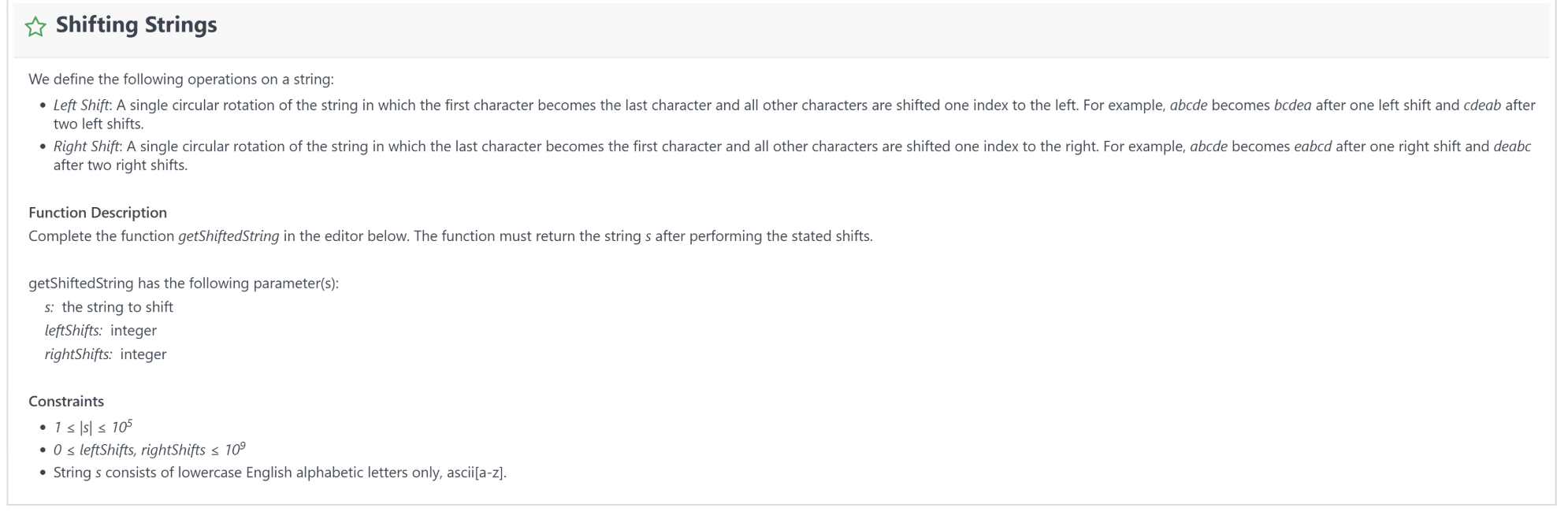
3. The perfect team



这个地方可以用entrySet() 去遍历



1. Shift strings



public String getShiftedString(String str, int leftShifts, int rightShifts) {

if (str == null || str.length() == 0) {

return null;

}

if (leftShifts - rightShifts == 0) {

return str;

}

leftShifts = (leftShifts - rightShifts) % str.length();

return str.substring(leftShifts) + str.substring(0, leftShifts);

}

1.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

2.Usernames System 。输入list：[bob, alice, bob, alice,bob] 输出：[bob,alice,bob1,alice1,bob2] super easy题

3. climb the hill！地里也有这道题

Minimum increment, 遍历相加即可

public int getMinSum(int[] nums) {

if (nums == null || nums.length == 0) {

return 0;

}

int sum = nums[0], prev = nums[0];

for (int i = 1; i < nums.length; i++) {

if (nums[i] <= prev) {

prev++;

sum += prev;

} else {

sum += nums[i];

prev = nums[i];

}

}

return sum;

}

#### [**孪生字符串**](http://www.1point3acres.com/bbs/thread-292218-1-1.html)

* 个字符串a和b，同字符串内，奇数位和奇数位的字符可以互相交换，偶数位和偶数位的字符也可以。判断能否通过这种交换操作使两个字符串相等。
* 分别统计奇数位和偶数位每个字母的出现次数，然后比较两个字符串的统计结果是否相等即可。

Image editing == Maximum squares

class Solution {

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

if(matrix == null || matrix.length == 0 || matrix[0].length == 0) return 0;

int row = matrix.length;

int col = matrix[0].length;

int[][] dp = new int[row+1][col+1];

int result = 0;

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

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

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

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

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

}

else dp[i][j] = 0;

}

}

return result\*result;

}

}

Group Anagram

class Solution {

public List<List<String>> groupAnagrams(String[] strs) {

if (strs.length == 0) return new ArrayList();

Map<String, List> ans = new HashMap<String, List>();

for (String s : strs) {

char[] ca = s.toCharArray();

Arrays.sort(ca);

String key = String.valueOf(ca);

if (!ans.containsKey(key)) ans.put(key, new ArrayList());

ans.get(key).add(s);

}

return new ArrayList(ans.values());

}

}

Climb the hill

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

3个coding:

1. code friends: E=1, M=3, H=5

3. climb the hill

# 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 formula

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:

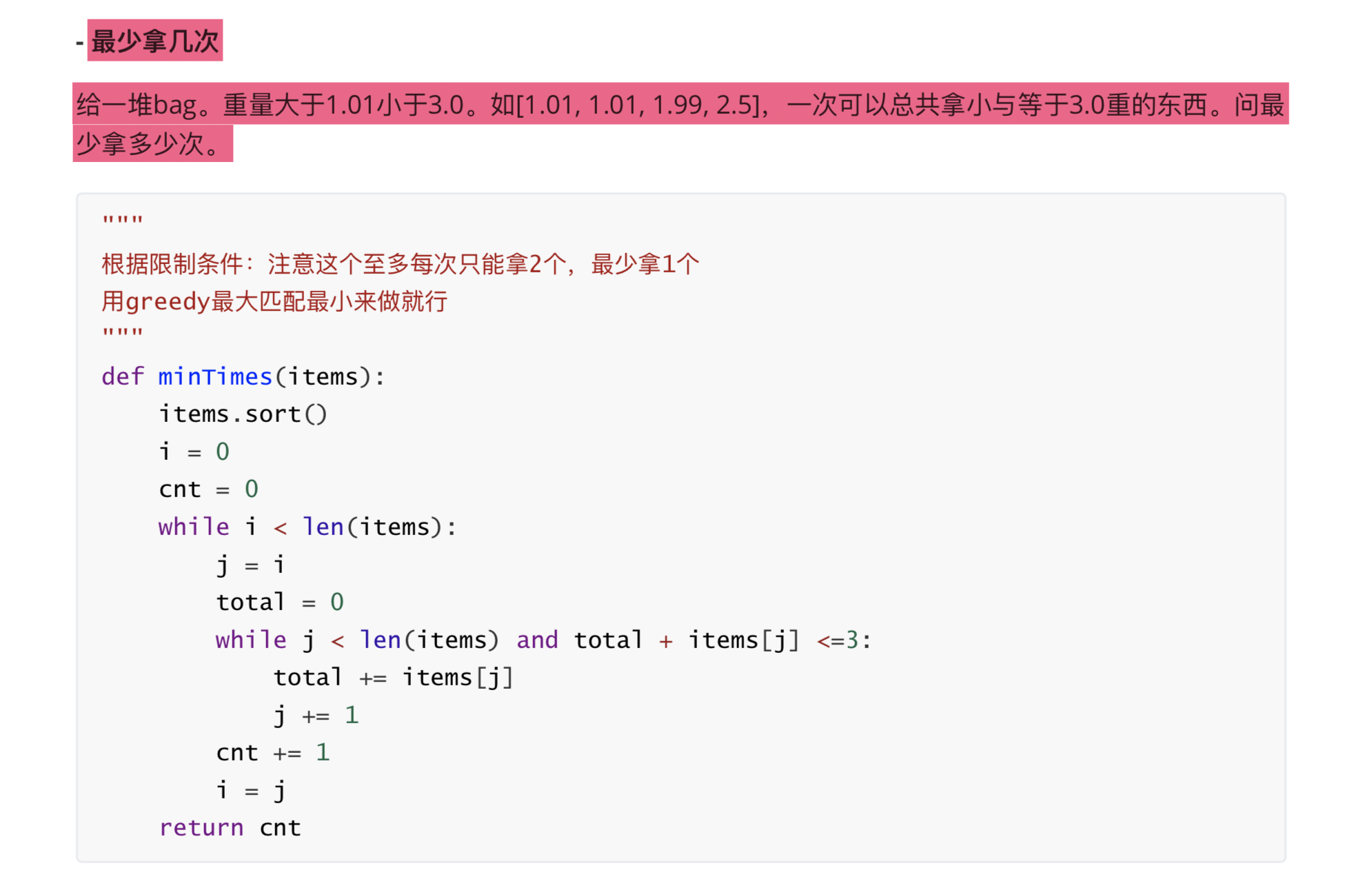
res += math.factorial(n) / (math.factorial(temp) \* math.factorial(n-temp))

return res

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



# 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

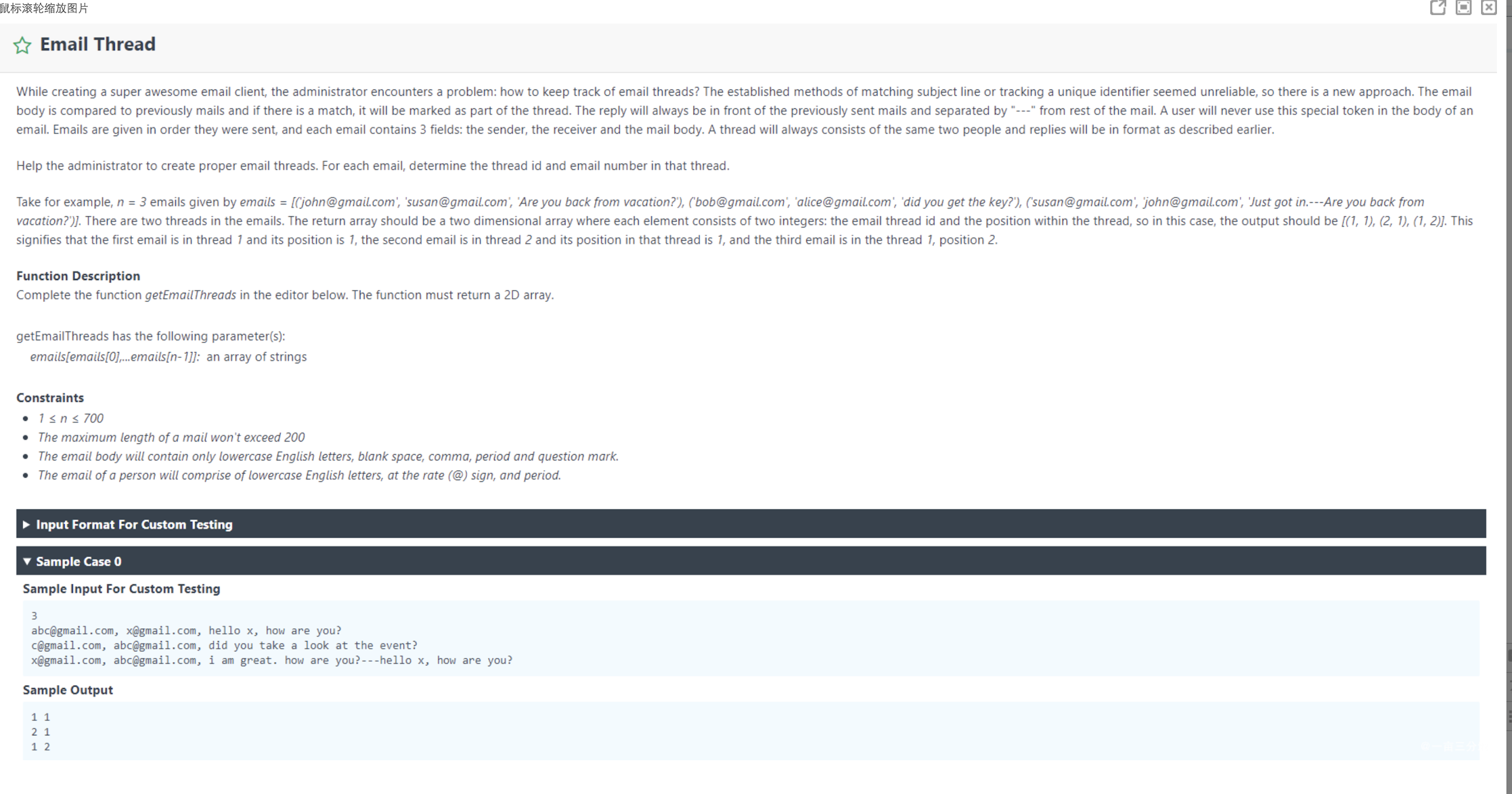
subLen = i - left + 1

maxLen = max(maxLen, subLen)

return maxLen

# 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, b@gmail.com, abc（内容）”

# 之后如果b对a回复了邮件，将会是“b@gmail.com, 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。

// 明天专门写一下

# 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：

my solution

**import** java.util.HashMap;

**import** java.util.\*;

**public class** Main {

**static class** Content{

**int sessionId**;

String **content**;

**public** Content(**int** sessionId, String content){

**this**.**sessionId** = sessionId;

**this**.**content** = content;

}

}

**public static int**[][] processEmail(String[] emails){

*// key: email pair, value: email threads maps*

*// inner map: key: thread id, value: Content*

**int** threadCount = 1;

**int**[][] result = **new int**[emails.**length**][2];

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

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

String email = emails[i];

String[] info = email.split(**", "**);

String email1 = info[0];

String email2 = info[1];

**if**(email1.compareTo(email2) > 0) {

String temp = email1;

email1 = email2;

email2 = temp;

}

*// 假设早发出来的在前面*

String emailKey = email1 + email2;

**if**(!map.containsKey(emailKey)) {

Map<Integer, List<Content>> innerMap = **new** HashMap<>();

List<Content> currentList = **new** ArrayList<>();

currentList.add(**new** Content(1, info[2]));

innerMap.put(threadCount, currentList);

map.put(emailKey, innerMap);

result[i][0] = threadCount;

result[i][1] = 1;

threadCount++;

} **else** {

*// first compare all the elements*

Map<Integer, List<Content>> innerMap = map.get(emailKey);

String[] emailContents = info[2].split(**"---"**);

**boolean** foundMatch = **false**;

**if** (emailContents.**length** > 1) {

**for**(Map.Entry<Integer, List<Content>> entry : innerMap.entrySet()) {

**int** thread = entry.getKey();

List<Content> contents = entry.getValue();

**if** (contents.size() == emailContents.**length**-1) {

**boolean** hasMatch = **true**;

**for**(**int** j = 0; j < contents.size(); j++) {

Content currentContent = contents.get(j);

**if** (currentContent.**content**.equals(emailContents[emailContents.**length**-j-1])){

**continue**;

} **else** {

hasMatch = **false**;

**break**;

}

}

**if** (hasMatch){

contents.add(**new** Content(contents.size(), emailContents[0]));

result[i][0] = thread;

result[i][1] = contents.size();

foundMatch = **true**;

**break**;

}

}

}

}

*// if still not matched*

**if**(!foundMatch){

**int** threadKey = threadCount;

List<Content> currentList = **new** ArrayList<>();

currentList.add(**new** Content(1, info[2]));

innerMap.put(threadCount, currentList);

map.put(emailKey, innerMap);

result[i][0] = threadKey;

result[i][1] = 1;

threadCount++;

}

}

}

**return** result;

}

**public static void** main(String[] args) {

String[] input = **new** String[3];

input[0] = **"a@gmail.com, b@gmail.com, first"**;

input[1] = **"b@gmail.com, c@gmail.com, first2"**;

input[2] = **"b@gmail.com, a@gmail.com, second---first"**;

**int**[][] result = *processEmail*(input);

**for**(**int**[] res : result) {

System.***out***.print(res[0] + **" "**);

System.***out***.println(res[1]);

}

}

}

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

# 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"

# 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

