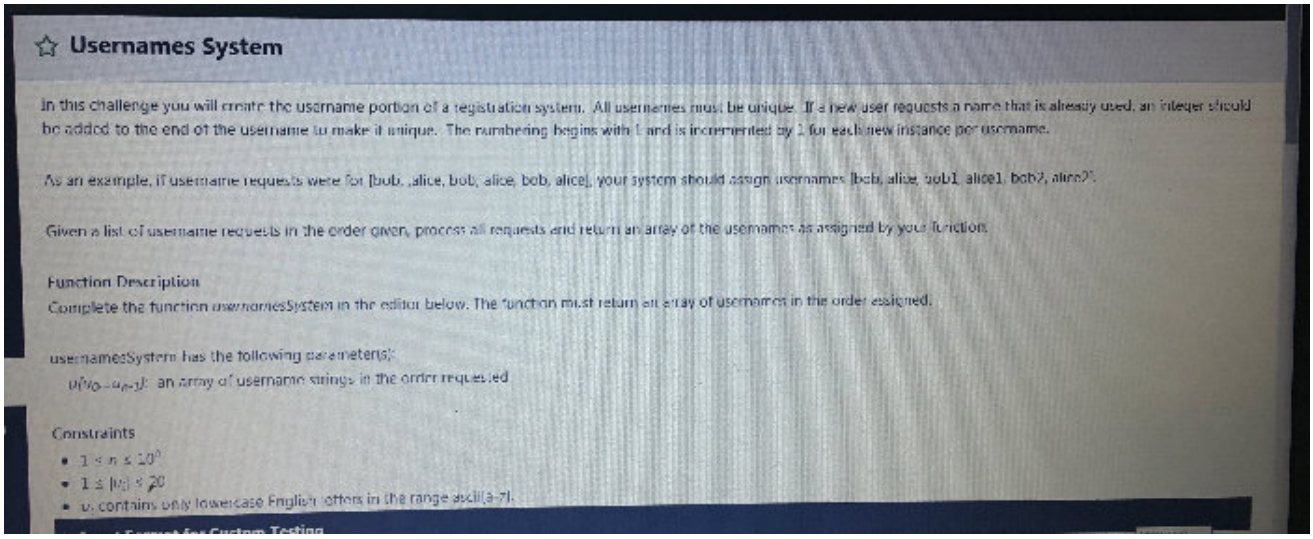
VMWare OA

LeetCode 1451, 76

Coding

- Username System



Code

```python
# user name system
def usernameSystem(names):
    count = dict()
    output = []
    for name in names:
        if name in count:
            output.append(name+str(count[name]))
            count[name]+=1
        else:
            count[name]=1
            output.append(name)
    return output
```

- Build Sequence

☆ Build the Subsequences

A subsequence of a string is obtained by deleting zero or more characters from the string while maintaining order. For example, the subsequences of string *s = "xyz"*, not including the empty string, are *"x"*, *"xy"*, *"xz"*, *"xyz"*, *"y"*, *"yz"*, and *"z"*. You will generate an array of all subsequences of a given string, omitting the empty string.

Function Description

Complete the function *buildSubsequences* in the editor below. The function must return an array of strings comprising all subsequences of the given string sorted alphabetically, ascending. Do not include the empty string in your results.

buildSubsequences has the following parameter(s):

s: the string to process

Constraints

- *1 < |s| < 16*
- *s* is a string of distinct lowercase English alphabetic letters *ascii[a-z]*.

▶ **Input Format for Custom Testing**

▼ **Sample Case 0**

Sample Input 0

```
ba
```

Sample Output 0

```
a
b
ba
```

Explanation 0

For *s = ba*, we assemble and return the following array of *3* alphabetically-ordered subsequences: *[a, b, ba]*.

- Even Subarrays

☆ Even Subarray

A subarray is a contiguous portion of an array. Given an array of integers, you must determine the number of distinct subarrays that can be formed having at most a given number of odd elements. Two subarrays are distinct if they differ at even one position in their contents.

For example, if *numbers = [1, 2, 3, 4]* and the maximum number of odd elements allowed, *k = 1*, the following is a list of the 8 distinct valid subarrays:

```
[[1], [2], [3], [4], [1,2], [2, 3], [3, 4], [2, 3, 4]]
```

Function Description

Complete the function *evenSubarray* in the editor below. The function must return the number of distinct subarrays that can be formed per the restriction of *k*.

evenSubarray has the following parameter(s):

numbers[numbers[0],...numbers[n-1]]: an array of integers
k: the maximum number of odd elements that can be in a subarray

Constraints

- $1 \leq n \leq 1000$
- $1 \leq k \leq n$
- $1 \leq numbers[i] \leq 250$

▼ Sample Case 0

Sample Input 0

```
4
6
3
5
8
1
```

Sample Output 0

```
6
```

Explanation 0

The distinct subarrays that can be formed are:

0 odd elements: *[6]* and *[8]*.

1 odd element: *[6, 3]*, *[3]*, *[5]*, and *[5, 8]*

note:

因为要distinct subarray去重，所以普通的sliding window并不能解决这道题。需要枚举出每个subarray, 根据数据规模只能有O(n^2)才能通过

Code

```
# even subarray
def evenSubarray(numbers, k):
    n = len(numbers)
    oddCnt = [0]*(n+1)
```

```
    for i in range(1, n+1):
        if numbers[i-1] % 2 != 0:
            oddCnt[i] = oddCnt[i-1] + 1
        else:
            oddCnt[i] = oddCnt[i-1]
    print (oddCnt)
    seen = set()
    output = []
    for i in range(n):
        sb = ""
        for j in range(i+1, n+1):
            sb += str(numbers[j-1])
            oddNums = oddCnt[j] - oddCnt[i]
            if oddNums > k:
                break
            if sb not in seen:
                seen.add(sb)
            output.append(numbers[i:j])
    return output
```

- The Perfect Team

☆ The Perfect Team

The School of Languages and Science teaches five subjects: *Physics, Chemistry, Math, Botany,* and *Zoology*. Each student is skilled in one subject. The skills of the students are described by string of named *skills* that consists of the letters *p, c, m, b,* and *z* only. Each character describes the skill of a student as follows:

- *p → Physics.*
- *c → Chemistry.*
- *m → Math.*
- *b → Botany.*
- *z → Zoology.*

Your task is to determine the total number of different teams satisfying the following constraints:

- A team consists of a group of exactly *five* students.
- Each student is skilled in a different subject.
- A student may only be on one team.

For instance, if the *skills* string is *pcmbzpcmbz* then there are two possible teams that can be formed at one time: *skills[0-4]* and *skills[5-9]* for example. It is not important to determine permutations as we will always be limited to two teams given *10* students.

Function Description

Complete the function *differentTeams* in the editor below. The function must return an integer value representing the number of teams that can be formed given the constraints.

differentTeams has the following parameter(s):

skills: a string where each position represents the skill of a student

Constraints

- $5 \leq n \leq 5 \times 10^5$
- *skills[i]* $\in$ *{p,c,m,b,z}*

Code

```
# the perfect team
def differentTeams(skills):
    cnt = dict()
    cnt['p'] = 0
    cnt['c'] = 0
    cnt['m'] = 0
    cnt['b'] = 0
```

```
    cnt['z'] = 0
    for i in range(len(skills)):
        char = skills[i]
        cnt[char] += 1
    mincnt = 2**31-1
    for _, v in cnt.items():
        mincnt = min(mincnt, v)
    return mincnt
```

- Shift String

☆ **Shifting Strings**

We define the following operations on a string:

- *Left Shift*: A single circular rotation of the string in which the first character becomes the last character and all other characters are shifted one index to the left. For example, *abcde* becomes *bcdea* after one left shift and *cdeab* after two left shifts.
- *Right Shift*: A single circular rotation of the string in which the last character becomes the first character and all other characters are shifted one index to the right. For example, *abcde* becomes *eabcd* after one right shift and *deabc* after two right shifts.

Function Description

Complete the function *getShiftedString* in the editor below. The function must return the string *s* after performing the stated shifts.

getShiftedString has the following parameter(s):

s: the string to shift
leftShifts: integer
rightShifts: integer

Constraints

- $1 \le |s| \le 10^5$
- $0 \le leftShifts, rightShifts \le 10^9$
- String *s* consists of lowercase English alphabetic letters only, ascii[a-z].

- Maximal Square (LC 221 Medium)

Code

```
// dp(i, j) = min(dp(i-1, j-1), dp(i-1, j), dp(i, j-1)) + 1 or 0
// 设dp(i, j)为以(i, j)为右下角的正方形的边长
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;
    }
```

```
}
```

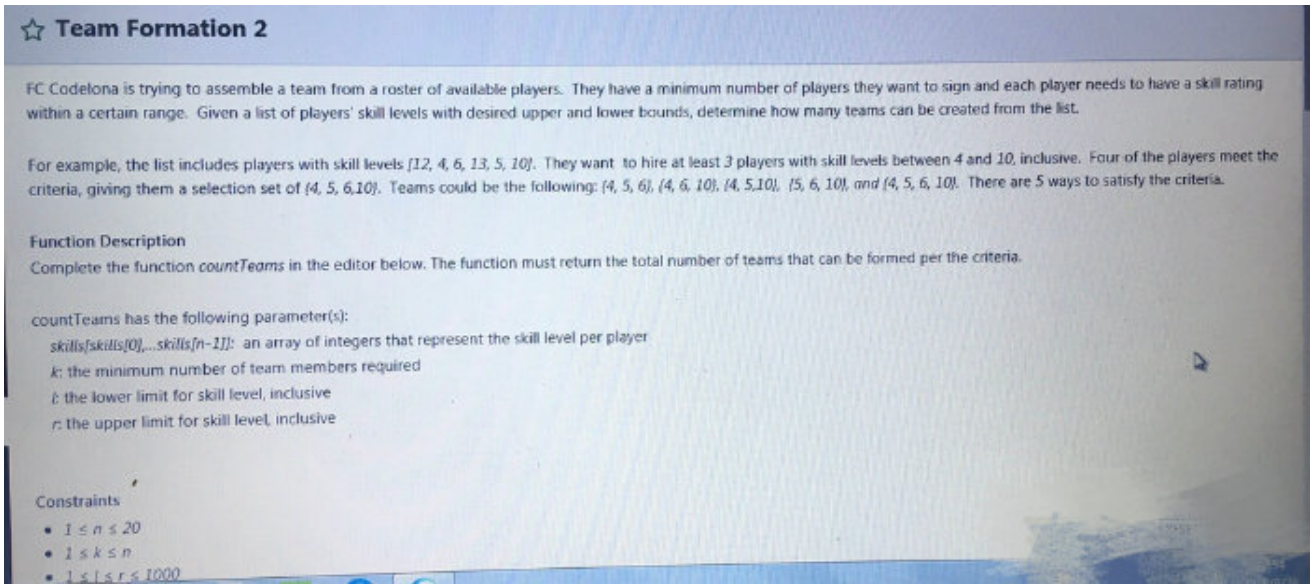
- Group Anagram (LC 49 Medium)

```
/*
use a hashmap to track the grouping of anagram
<encode, List <String> group>
for each strs:
    compute the encoding, if exists in the hashmap, add to group, else create an
entry
*/
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        List <List<String>> ans = new ArrayList <> ();
        if (strs.length == 0) return ans;
        Map <String, List<String>> map = new HashMap <> ();
        for (String s: strs){
            String encoded = encode(s);
            List <String> group = map.getOrDefault(encoded, null);
            if (group == null) group = new ArrayList <> ();
            group.add(s);
            map.put(encoded, group);
        }
        for (String key: map.keySet()){
            ans.add(map.get(key));
        }
        return ans;
    }

    // returns the encoding of a string
    public String encode (String s){
        int[] cnt = new int[26];
        for (int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int index = ch - 'a';
            cnt[index]++;
        }
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 26; i++){
            if (cnt[i] > 0){
                int offset = i;
                char ch = (char) ((int) 'a' + offset);
                sb.append(ch);
                sb.append(cnt[i]);
            }
        }
```

```
            return sb.toString();
        }
    }
```

- Team Formation2



☆ Team Formation 2

FC Codelona is trying to assemble a team from a roster of available players. They have a minimum number of players they want to sign and each player needs to have a skill rating within a certain range. Given a list of players' skill levels with desired upper and lower bounds, determine how many teams can be created from the list.

For example, the list includes players with skill levels *{12, 4, 6, 13, 5, 10}*. They want to hire at least *3* players with skill levels between *4* and *10*, inclusive. Four of the players meet the criteria, giving them a selection set of *{4, 5, 6,10}*. Teams could be the following: *{4, 5, 6}*, *{4, 6, 10}*, *{4, 5,10}*, *{5, 6, 10}*, and *{4, 5, 6, 10}*. There are *5* ways to satisfy the criteria.

Function Description

Complete the function *countTeams* in the editor below. The function must return the total number of teams that can be formed per the criteria.

countTeams has the following parameter(s):

skills[skills[0],...skills[n-1]]: an array of integers that represent the skill level per player
k: the minimum number of team members required
l: the lower limit for skill level, inclusive
r: the upper limit for skill level, inclusive

Constraints

- $1 \le n \le 20$
- $1 \le k \le n$
- $1 \le l \le r \le 1000$

- 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。

- 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

和group anagram一样。先用encode把每个string给encode了

```
Set <String > seen = new HashSet <> ();
List <String> ans = new ArrayList <> ();
for (String s: list){
  String encode = encode(s);
  if (seen.contains(encode)){
    continue;
  }
  ans.add(s);
}
return ans;
```

- list碰撞

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

```
 public static int listCollision(List<Integer> arr){
    PriorityQueue <Integer> pq = new PriorityQueue <> (new Comparator <Integer>()
{
      @Override
      public int compare(Integer a, Integer b){
        return b-a;
      }
    });
    for (Integer v: arr){
      pq.add(v);
    }
    while (pq.size() > 1){
      int first = pq.poll();
      int second = pq.poll();
      if (first != second){
        pq.offer(first-second);
      }
    }

    if (pq.size() == 1){
      return pq.poll();
    }
    return 0;
  }
```

- Intelligent Str

☆ Intelligent Substring

You are exploring a new language where there are two types of characters: *special* and *normal*. A character is *special* if its *value* is 1 and *normal* if its *value* is 0.

Given a string *s*, and an integer *k*, determine the length of the longest possible substring with at most *k* normal characters. There will be a 26 digit bit string *charValue* where each position represents the *special* or *normal* nature, or *value*, of the corresponding letter of the English alphabet.

For example, if *s = "abcde"* and *k = 2* , using the following *charValue*:

```
alphabet  = abcdefghijklmnopqrstuvwxyz
charValue = 10101111111111111111111111
```

the normal characters are in the set {b, d}. Since *k* is *2* and there are only two normal characters present in the string, any substring meets the criterion. The longest substring can be made up of the entire string and has a length of *5*. If instead *k = 1*, possible substrings are ['b', 'd', 'ab', 'bc', 'cd', 'de', 'abc', 'cde']. The longest substrings are *3* characters long.

Function Description

Complete the function *getSpecialSubstring* in the editor below. The function must return an integer that denotes the length of the longest substring of *s* with at most *k* normal characters.

getSpecialSubstring has the following parameter(s):

s: the input string
k: the maximum number of normal characters allowed in a substring
charValue: a string representing special or normal for each letter of the alphabet, ascii[a-z]

Constraints

- $1 \le |s| \le 10^5$
- $1 \le k \le |s|$
- $|charValue| = 26$
- $charValue[i] \in \{0,1\}$

▶ **Input Format For Custom Testing**

▼ **Sample Case 0**

Sample Input 0

Code

```
public static int getSpecialSubstring(String s, int k, String charValue){
    int ans = 0;
    // at most k normal characters
    int i = 0, j = 0, cnt = 0;
    while (j < s.length()){
      if (isNormal(s.charAt(j), charValue)) cnt++;
      j++;
      while (i < j && cnt > k){
        if (isNormal(s.charAt(i), charValue)) cnt--;
        i++;
      }
      if (cnt <= k){
        ans = Math.max(j-i, ans);
      }
    }
    return ans;
  }

  public static boolean isNormal(char ch, String charValue){
    return charValue.charAt(ch-'a') == '0';
  }
```

- Email Thread

给list of strings。每个string是“a@gmail.com, b@gmail.com, how are you?"。用数字表示threads。 例子： input: [a@gmail.com, b@gmail.com, how are you?] [b@gmail.com, a@gmail.com, I am good --- how are you?] [a@gmail.com, c@gmail.com, Hi] [a@gmail.com, b@gmail.com, what's your name?] ... output: [1,1] [1,2] [2,1] [3,1]

```
# email threads
def emailThreads(strs):
    msgs = dict()
    tids = 0
    output = []
    for s in strs:
        lst = s.split(',')
        sender, receiver, content = lst[0], lst[1], lst[2]
        if '---' not in content:
            # this is a new thread
            tids += 1
            msgs[tuple(lst)] = [tids, 1]
            output.append([tids,1])
        else:
            msg = content.split('---')
            prev = msg[1:]
            last_thread = (receiver, sender, '---'.join(prev))
            tid, msg_cnt = msgs[last_thread][0], msgs[last_thread][1]
            # output current data point
            output.append([tid, msg_cnt+1])
            msgs[tuple(lst)] = [tid, msg_cnt+1]
    return output
```

- Break Palindrome

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"。

把左半边不是a的改成a就行了

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。

- DP题 (2017年的面经)

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。

```
"""
knapsack with repetitions
记忆化解法
"""
def budgetShopping(n, quantities, costs):
    dp = [-1]*(n+1)
    dp[0] = 0
    return memo(dp, quantities, costs, n)

def memo(dp, quantities, costs, cur):
    if dp[cur] != -1:
        return dp[cur]

    num_stores = len(quantities)
    dp[cur] = 0
    for i in range(num_stores):
        if costs[i] <= cur:
            cnt = memo(dp, quantities, costs, cur-costs[i])
            dp[cur] = max(dp[cur], quantities[i]+cnt)
    return dp[cur]
```

- climb the hill

☆ Climb the hill!

Jack was trying to go up the hill. He does not have any problem in climbing up or coming down the hill if the slope is consistently either increasing or decreasing. Areas where the slope is constant do not bother him in either situation.

Given a list of heights along his path, find the minimum amount to add or subtract to each offending height to make the *slope* meet Jack's requirements. Heights may be increased or decreased as necessary. The value of a change is absolute. In other words, if a height 10 is increased or decreased to 13 or 7, the change is 3.

The following is an example of an array describing a generally increasing set of heights making a slope: [0, 1, 2, 5, 6, 5, 7]. The minimum changes required will result from making the slope increasing along its length. Even though the slope varies, it is always increasing over the subarray [0, 1, 2, 5, 6], so no changes are made along that range. The height at array position 5, value = 5, must be raised to at least 6, making the slope flat, so add 1. Now test against the remaining value, position 6, value = 7. The new height 6 < 7 and the rule holds. The sum of all changes necessary is 1.

Function Description
Complete the *climbTheHill* function in the editor below. The function must return an integer that denotes the minimum cost required to make the slope increasing or decreasing along its length.

climbTheHill has the following parameters:
slope[slope[0],...,slope[n-1]]: an array of integers representing heights along a path

Constraints
- $1 \le n \le 10^3$
- $1 \le slope[i] \le 10^9$

▸ **Input Format for Custom Testing**

▾ **Sample Case 0**

Sample Input 0

```
6
9
8
7
2
3
3
```

Sample Output 0

```
1
```

Explanation 0
The required sequence that can be formed is *[9, 8, 7, 3, 3, 3]* (decreasing). The minimum cost incurred is in changing *slope[3]* from 2 to 3.

```
"""
设dp(i, j) = 让前i个数不降序且第i个数不超过b_j的最小cost
```

```
设b为a的排序之后的sequence
感觉和这道题是一样的: https://blog.csdn.net/v5zsq/article/details/79605704
先用下面这个helper把不降序的解球出来，再把原array reverse,可以再利用下面这个helper求降序的解，最后取最小就行
"""
def climbHill(a):
    n = len(a)
    dp = [[0 for i in range(n+1)] for j in range(n+1)]
    b = sorted(a)
    dp[1][1] = abs(a[0]-b[0])
    # initialization
    for j in range(1, n+1):
        if j > 1:
            dp[1][j] = min(dp[1][j-1], abs(a[0] - b[j-1]))
    for i in range(1, n+1):
        if i > 1:
            dp[i][1] = dp[i-1][1] + abs(a[i-1] - b[0])
    # compute
    for i in range(2, n+1):
        for j in range(2, n+1):
            dp[i][j] = min(dp[i][j-1], dp[i-1][j]+abs(a[i-1]-b[j-1]))
    return dp[n][n]
```

- 最少拿几次

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。

```
"""
根据限制条件：注意这个至多每次只能拿2个，最少拿1个
用greedy最大匹配最小来做就行
"""
def minTimes(items):
    items.sort()
    i = 0
    cnt = 0
    while i < len(items):
        j = i
        total = 0
        while j < len(items) and total + items[j] <=3:
            total += items[j]
            j += 1
        cnt += 1
        i = j
    return cnt
```

Multiple Choice

1. 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 (196)
2. 一个无向图，n个点，e个边，选择当图用（1）邻接矩阵 O(n^2)（2）邻接列表时DFS的时间复杂度。O(n+3)
3. 求时间复杂度
4. 给postorder，preorder求inorder
5. 找一个数据结构，能够支持string插删合并啥的
6. best data structure to represent telephone network
7. mystery algorithm
8. try, catch & finally block