DrawBridge Interview
OA

Tommy and Jerry

Valid Tree

Diamond Mine

Magical Strings
第二题Magical Strings，规则是一个字符串只能由 a, e, i, o, u 中的字符组成，每个字符都可以用任意次，但是必须满足规则：
a后面只能跟e
e后面只能跟a或i
i后面可以跟a, e, o, u中任一
o后面只能跟i或u附件
u后面只能跟a
题目输入是一个数字n，求满足以上规则且长度为n的字符串一共有多少个，输出个数对于1000000007的模。

```
def numofCombStr(n):
    a, e, i, o, u = 1, 1, 1, 1, 1
    j = 1
    while j <= n:
        at = e
        et = a+i
        it = a+e+o+u
        ot = i+u
        ut = a
        a,e,i,o,u = at, et, it, ot, ut
        j += 1
    return a+e+i+o+u
```

```
/*
* Sort an array by number of 1s in the binary of integer, if number of 1s are same,
* the smaller one should be placed first.
*
* Example
*
* input [1,2,3,4,5]
* return [1,2,4,3,5]
```

```
*
* */

def reOrder(nums):
    nums = sorted(nums, key=lambda x: bin(x).count('1'))
    return nums

/*
* Given a positive integer target, count all the combinations of contiguous positive
* integers that sum up to the target.
*
* For Example,
* target = 15
* return 3
*
* since
* 15 = 4 + 5 + 6
* 15 = 1 + 2 + 3 + 4 + 5
* 15 = 7 + 8
* */
def numofComb(num):
    if num <= 0:
        return 0
    count = 0
    k = 2
    while k*(k-1)/2 < num:
        if (num-k*(k-1)/2)%k == 0:
            count += 1
        k += 1
    return count


/*
* Degree of an Array
*
* Given an array of n integers, we define its degree as the maximum frequency of
* any element in the array.
*
* For example, the array [1, 2, 3, 4, 2, 2, 3] has a degree of 3 because the number
* 2 occurs three times (which is more than any other number in the array).
* We want to know the size of the smallest subarray of our array such that the
* subarray's degree is equal to the array's degree.
*
```

```
* For example, the array [1, 2, 2, 3, 1] has a degree of 2 because 1 and 2 occur a
* maximal two times. There are two possible subarrays with this degree: [1, 2, 2, 3, 1]
* and [2, 2]. Our answer is the length of the smallest subarray, which is 2.
*
* Complete the function in the editor below. It has one parameter: an array of
* n integers, arr. The function must return an integer denoting the minimum size
* of the subarray such that the degree of the subarray is equal to the degree of the array.
* */
    def findShortestSubArray(self, nums):
        counter = collections.Counter(nums)
        temp = []
        for key in counter.keys():
            if counter[key] == max(counter.values()):
                temp.append(key)
        degree = max(counter.values())
        if degree == 1:
            return 1
        answer = len(nums)
        for each in temp:
            count = 0
            first = len(nums)-1
            last = 0
            for index, num in enumerate(nums):
                if num == each and count == 0:
                    first = index
                    count += 1
                elif num == each and count == degree-1:
                    last = index
                elif num == each:
                    count += 1
            answer = min(answer, last-first+1)
        return answer


/*
* Given a String contains of '(' and ')'
* and an integer k represent the maximum time of replacement
*
* replacement operation : replace ')' with '()'
*
* calculate whether can make it balanced
* */第一题是括号匹配，  function signature是 int[] balanceOrnot(String[] strs, int[]
maxReplacement), strs[i] 对应的是一个像“<<>>>>”这样字符串, 函数判断strs 是否可以通过
```

maxReplaement次的替换，变成括号匹配的形式。注意替换的规则是 >可以替换为<>, 这样就匹配了。但是 <无法替换为<>。函数返回一个数组，数组代表strs能否被成功替换。

```
    def validParenthesis(self, lists, number):
        if not lists:
            return True
        stack = []
        count = 0
        for each in lists:
            if each == '<':
                stack.append(each)
            elif each == '>':
                if not stack:
                    count += 1
                else:
                    stack.pop()
        if not stack and count <= number:
            return True
        return False
```

```
/*
* minMove
*
* 两个相同长度的正整数，每次操作可以对一个数的某一位digit加1或者减1，问最少操作次数使两数相等。
* 输入为两个数组a,m，输出两个数组对应位置上两个数的最小操作和。
* */

def minMove(self, a, b):
    count  = 0
    while a!= 0:
        a1 %= 10
        b1 %= 10
        count += abs(a1-b1)
        a /= 10
        b /= 10
    return count
```

```
/*
* Jungle Book
*
* 输入是一个list,index代表pray,value代表predator,value = -1 代表没有predator,
```

```
* 就是说list = <n个有向edge, n个node的tree(forest),题里说输入满足不成环,一个species只有一个predator。
* 问把这些动物最少分成几组， 使组内成员不互相伤害(A->B->C的话A,C也不能一组)
* */

def JungleBook(books):
    count = 0
    dictionary = {}
    for i in range(len(books)):
        count = max(count, calculate(books, i, dictionary))
    return count

def calculate(books, index, dictionary):
    predator = books[index]
    if predator == -1:
        dictionary[index] = 1
        return 1
    value = 0
    if predator in dictionary:
        value = dictionary[predator] + 1
    else:
        value = calculate(books, predator, dictionary) + 1
    dictionary[index] = value
    return value

/*
* Given a set of inclusive intervals
* Calculate the minimum size of a set of number
* where each interval contains at least two numbers in this set.
*
* Example
* given [1,3], [1,4], [2,5].
* return 2. ([2,3])
*
* given [2,4], [3,6], [0,2], [4,7].
* return 4. ([0,2,4,6] or [1,2,4,5])
* */

    def intersectionSizeTwo(self, intervals):
        intervals = sorted(intervals, key=lambda x: x[1])
        answer = []
        for interval in intervals:
            if not answer or answer[-1] < interval[0]:
```

```
            answer.append(interval[1]-1)
            answer.append(interval[1])
        elif answer[-2] < interval[0]:
            if answer[-1] == interval[1]:
                answer.append(interval[1]-1)
            else:
                answer.append(interval[1])
    return len(answer)
```