python刷题记录

1. 过河卒

```
n,m,x,y = map(int,input().split())
a = 0
b = 0
p1 = [x+1,y+2]
p2 = [x+2,y+1]
p3 = [x+2,y-1]
p4 = [x+1,y-2]
p5 = [x-1,y-2]
p6 = [x-2,y-1]
p7 = [x-2,y+1]
p8 = [x-1,y+2]
list_horse = [p1,p2,p3,p4,p5,p6,p7,p8,[x,y]]
list = []
def move(a,b,n,m,list):
    if ([a + 1, b] in list_horse) and ([a, b + 1] in list_horse):  # 这是完全不能动的情况!直接结束
        return
    if (a != n or b != m) and a <= n and b <= m:  #  判断条件：我能懂,但是我不能达到B点,而且没有走出边界,不然直接结束
        if [a + 1, b] in list_horse:  # 判断条件：右边有驻马点
            move(a,b+1,n,m,list)
        elif [a,b+1] in list_horse:  #  判断条件：下边有驻马点
            move(a+1,b,n,m,list)
        else:
            move(a+1,b,n,m,list)
            move(a,b+1,n,m,list)
    elif a==n and b==m:
        list.append(1)
        return
    elif a > n or b > m:
        return
move(a,b,n,m,list)
print(len(list))
```

最大的亮点: 这道题使用了递归,每一个成功的情况,会在list里面加一次元素

注意: 我最开始尝试的是 count += 1 , 但是很明显, 我没有成功

2. 铺地毯

```
n = int(input())
cover_list = []
for i in range(1,n+1):
    a,b,g,k = map(int,input().split())
    for x in range(a,a+g+1):
```

```
        for y in range(b,b+k+1):
            cover_list.append([x,y,i])
x,y = map(int,input().split())
target = 0
for element in reversed(cover_list):
    if element[0] == x and element[1] == y:
        target = element[2]
        break
if target == 0:
    print(-1)
else:
    print(target)
```

但是很遗憾的是, 这个程序所用的内存还是太大了

```
n = int(input())
list = []
for i in range(n):
    x,y,a,b = map(int,input().split())
    small_list = [x,y,a,b]
    list.insert(0,small_list)
x,y = map(int,input().split())
for element in list:
    if x in range(element[0],element[0]+element[2]+1):
        if y in range(element[1],element[1]+element[3]+1):
            print(len(list)-list.index(element))
            break
else:
    print(-1)
```

在这个程序里面,我们仅仅使用了一次遍历, 没有开过多列表,因此成功满足题目要求

3. 独木桥

```
# 两个人向碰面然后两个人背身继续走,相当于交换灵魂继续走
def main():
    l = int(input())
    n = int(input())
    if n == 0:
        print('0 0')
        return
    zuobiao = input().split()
    max_right = [0]
    max_left = [0]
    min_right = [0]
    min_left = [0]
    for i in zuobiao:
        if int(i) <= (l+1)/2:
            max_right.append(l+1-int(i))
            min_left.append(int(i))
        else:
```

```
            max_left.append(int(i))
            min_right.append(l+1-int(i))

 print(max(max(min_right),max(min_left)),max(max(max_right),max(max_left)))
    return
main()
```

这道题我面临过两个坑:

第一个就是如果n = 0呢?第三行是直接不用输入的

因此用函数的return进行解决:

第二个就是:如果士兵坐标全部一边倒呢? 将会有集合是空集, max函数会报错

因此干脆原先每个列表里面就放入0, 因为它的存在完全不影响正确结果的输出

4. 三连击

```
import itertools
digits = [1,2,3,4,5,6,7,8,9]
numbers = list(itertools.permutations(digits))
for i in range(len(numbers)):
    num0 = int(''.join([str(x) for x in numbers[i]]))
    num1 = num0//1000000
    num2 = (num0-num1*1000000)//1000
    num3 = num0-num1*1000000-num2*1000
    if num3 == 3 * num1 and num2 == 2 * num1:
        print(num1,num2,num3)
```

注意这里面的itertools模组的permutations函数,将digits里面的数字去排列组合,输出全部的含有这些数字的数字,但是这个数字的每一位是放在了元组里面

```
num0 = int(''.join([str(x) for x in numbers[i]]))
```

这句话是精髓中的精髓

5. 与指数相关的函数

```
#判断一个数是不是质数的函数
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
#输入一段区间，输出全部质数的函数
def find_primes(start, end):
    primes = []
```

```
    for num in range(start, end+1):
        if is_prime(num):
            primes.append(num)
    return primes
```

6. 生成螺旋矩阵

```
def generate_spiral_matrix(x,y):
    matrix = [[0] * y for _ in range(x)]
    top, bottom, left, right = 0, x-1, 0, y-1
    num = 1
    while num <= x * y:
        # fill top row
        for i in range(left, right+1):
            matrix[top][i] = num
            num += 1
        top += 1
        # fill right column
        if num <= x*y:
            for i in range(top, bottom+1):
                matrix[i][right] = num
                num += 1
            right -= 1
        else:
            break
        # fill bottom row
        if num <= x * y:
            for i in range(right, left-1, -1):
                matrix[bottom][i] = num
                num += 1
            bottom -= 1
        else:
            break
        # fill left column
        if num <= x * y:
            for i in range(bottom, top-1, -1):
                matrix[i][left] = num
                num += 1
            left += 1
        else:
            break
    return matrix
# 测试
x,y = map(int,input().split())
spiral_matrix = generate_spiral_matrix(x,y)
for row in spiral_matrix:
    print(row)
```

7. 判断输入的字符串中是否有:alphanumeric alphabetical digit lower upper

```
import re
def alphanumeric(string):
    x=0
    for char in string:
        if char.isalnum():
            x+=1
        else:
            continue
    if x != 0:
        print(True)
    else:
        print(False)

def alphabetical(string):
    x = 0
    for char in string:
        if char.isalpha():
            x += 1
        else:
            continue
    if x != 0:
        print(True)
    else:
        print(False)
def digits(string):
    x = 0
    for char in string:
        if char.isdigit():
            x += 1
        else:
            continue
    if x != 0:
        print(True)
    else:
        print(False)
def lower(string):
    x = 0
    for char in string:
        if char.islower():
            x += 1
        else:
            continue
    if x != 0:
        print(True)
    else:
        print(False)
def upper(string):
```

```
    x = 0
    for char in string:
        char = str(char)
        if char.isupper():
            x += 1
        else:
            continue
    if x != 0:
        print(True)
    else:
        print(False)
string=input()
alphanumeric(string)
alphabetical(string)
digits(string)
lower(string)
upper(string)
```

8. 列表综合操作

```
n = int(input())
num = input()                    #输入一行字符串
array1 = num.split()               #注意，现在列表中所有元素均是字符串
array = []
for i in array1:                        #将所有的元素转化为整数数据类型，然后放到新的列表里面
    array.append(int(i))
array.sort(reverse=True)          #这里是降序！
k = int(array[0])
for i in array:                          #寻找列表中第二大的数字！！（注意，最大的数字可能不止一个）
    i = int(i)
    if i >= k:
        continue
    else:
        print(i)
        break
```

9. 水洼地

```
n = int(input())
list = input().split()
new_list = []
times = 0
while list != new_list:
    times += 1
    new_list = [i for i in list]    #这步十分容易出错！不应该是new_list = list !
    for i in range(len(list)-1):    #微操！留出最后一位i不取
        if list[i] == list[i+1]:     #一次次的消去重复元素
```

```
            list.pop(i)
            break
num = 0
for x in range(1,len(list)-1):
    if int(list[x])<int(list[x-1]) and int(list[x])<int(list[x+1]):
        num += 1
    else:
        continue
print(num)
```

精髓在于: 相邻且相同的重复元素要删除

10. 算阶乘之和

```
n = int(input())
sum = 0
for x in range(1,n+1):
    tmp=1
    for m in range(1,x+1):
        tmp = tmp * m
    sum += tmp
print(sum)
```

11. 幂表示

```
def f1(x):
    ##获取一个数的幂
    str0 = bin(int(str(x), 10))
    str1 = str0[2:]
    list1 = []
    index = 0
    for i in str1[::-1]:
        if i == '1':
            list1.append(index)
        index += 1
    list1.reverse()
    return list1


def f2(list):
    ##格式化输出
    list1 = [str(i) for i in list]
    str2 = ''
    for i in range(len(list1)):
        if i < len(list1) - 1:
            if list1[i] == "1":
                str2 += "2+"
            else:
                if list[i] != 0:
                    str2 += "2({})+".format(f2(f1(list[i])))
```

```
                else:
                    str2 += "2(0)"
        if i == len(list1) - 1:
            if list1[i] == "1":
                str2 += "2"
            else:
                if list[i] != 0:
                    str2 += "2({})".format(f2(f1(list[i])))
                else:
                    str2 += "2(0)"
    return str2
n = int(input())
print(f2(f1(n)))
```

附: 这段代码中的25和33行至今我无法理解! 应该是语法上的不了解

12. 车站

```
a,n,m,x = map(int,input().split())
if x in [1,2] :
    print(a)
elif x == 3:
    print(2*a)
else:
    list_a = [1,1]
    list_b = [1,2]
    for _ in range(n-5):
        num = list_a[-1]+list_a[-2]
        list_a.append(num)
    for _ in range(n - 5):
        num = list_b[-1] + list_b[-2]
        list_b.append(num)
    b = (m-(list_a[-1]+1)*a)//(list_b[-1]-1)
    print(((list_a[x-3]+1)*a)+(list_b[x-3]-1)*b)
```

火车从始发站（称为第 1 站）开出，在始发站上车的人数为 a，然后到达第2 站，在第 2 站有人上、下车，但上、下车的人数相同，因此在第 2站开出时（即在到达第 3 站之前）车上的人数保持为 a 人。从第 3 站起（包括第 3 站）上、下车的人数有一定规律：上车的人数都是前两站上车人数之和，而下车人数等于上一站上车人数，一直到终点站的前一站（第 $(n-1)$ 站），都满足此规律。现给出的条件是：共有 n个车站，始发站上车的人数为 a，最后一站下车的人数是 m（全部下车）。试问 x 站开出时车上的人数是多少?

13. 拼接数字并排序

```python
import itertools
n = int(input())
list_num = input().split()
numbers = list(itertools.permutations(list_num))
number_list = []
for list_want in numbers:
    tmp = int(''.join(list_want))
    number_list.append(tmp)
number_list.sort(reverse = True)
print(number_list[0])
```

14. CANTOR表

```python
n = int(input())
tmp = 0
for x in range(n):
    if ((x-1)**2+(x-1))//2 < n and (x**2+x)//2 >= n:
        tmp = x
        break
# 如果是偶数项的列表,那么就是分母开始从大到小变化,分子从小到大变化
# 同样,如果是奇数项的列表,那么就是分母从小到大变化,分子从大到小变化
fenzi = [i for i in range(1,tmp+1)]
fenmu = [i for i in range(tmp,0,-1)]
if tmp%2 == 0:
    index = int(n - ((tmp-1)**2 + (tmp-1))//2)-1
    print(f'{fenzi[index]}/{fenmu[index]}')
else:
    index = int(n - ((tmp - 1) ** 2 + (tmp - 1))//2) - 1
    print(f'{fenzi[tmp-index-1]}/{fenmu[tmp-index-1]}')
```

15. 回文数

```python
n = int(input())
instr = input()
a = []
for ch in instr:
    try:
        a.append(int(ch,base=n))
    except Exception:
        # must handle exception !!!
        pass
step = 0
while step <= 30:
    b = a[::-1]
    if a == b:
        break
    # add b and b.revert()
```

```
    a = []
    up = 0
    for index in range(len(b)):
        s = b[index] + b[-1 - index] + up
        a.append(s % n)
        up = 1 if s >= n else 0
    if up == 1:
        a.append(1)
    step += 1
if step <= 30:
    print("STEP="+str(step))
else:
    print("Impossible!")
```

16. 数的计算

给出正整数 *n*，要求按如下方式构造数列：

1. 只有一个数字 n* 的数列是一个合法的数列。
2. 在一个合法的数列的末尾加入一个正整数，但是这个正整数不能超过该数列最后一项的一半，可以得到一个新的合法数列。

请你求出，一共有多少个合法的数列

```
n = int(input())
list = []
def add_half(x):
    if x == 0:
        list.append(1)
    elif x != 1:
        for i in range(0,x//2+1):
            add_half(i)
    else:
        list.append(1)
add_half(n)
print(len(list))
```

17.最大公约数和最小公倍数问题

```python
import numpy
x,y = map(int,input().split())
limit = int(numpy.sqrt(x*y)//1+1)
ans = 0
for p in range(1,limit+1):
    q = int(x*y/p)
    if numpy.gcd(p,q) == x and numpy.lcm(p,q) == y:
        if p != q:
            ans += 2
        else:
            ans += 1
print(ans)
```

值得注意的是这里import了numpy模组, 帮助我们直接找到最大公约数和最小公倍数

注意的是: 最小公约数和最大公倍数的成绩应该恰好就是两个数字的乘积

18. 均分卡牌

```python
n = int(input())
num_list = input().split() # 注意每个元素的类型是str
sum = 0
num_list = [int(i) for i in num_list]
for x in num_list:      # 现在就转化为了int
    sum += x
ave = int(sum/n)
time = 0
for i in range(len(num_list)-1):
    if num_list[i] != ave:
        num_list[i+1] += num_list[i] - ave
        time += 1
    else:
        continue
print(time)
```

注意思想: 可能你会说: 如果这个i项的牌不够分给下一个怎么办?

那么就让他是这么操作的! 虽然看上去有负数的出现, 但是其实可以反向看

是对面给了我正数数量的牌, 等价于我给对面复数数量的牌

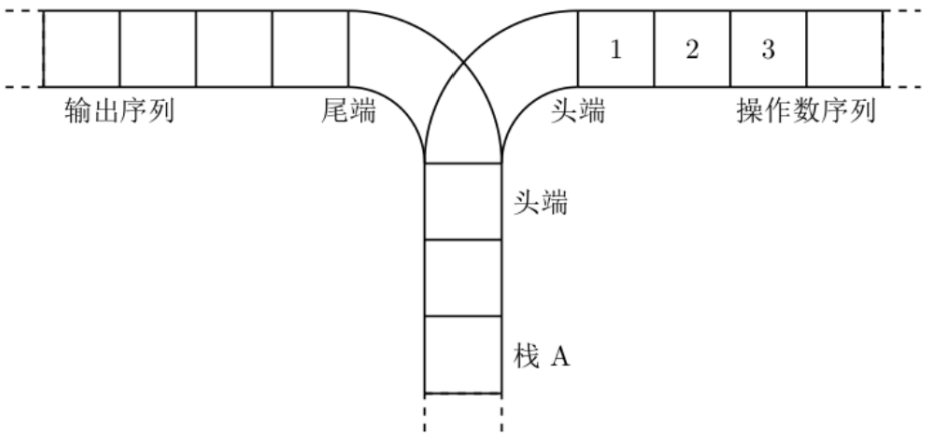
19. 选数 判断和是不是质数

```python
import numpy
from itertools import combinations
#输入和定义变量
nums_len,sum_nums = map(int,input().split())
nums = list(map(int,input().split()))
count = 0
```

```
array_sum = 0
def prime_number(number):
    for x in range(2,int(numpy.sqrt(number)//1) + 1):
        if number % x == 0:
            return False
    return True
#遍历所有组合
for tem_combin in combinations(nums,sum_nums):
    #将组合转化为数组
    array = numpy.array(tem_combin)
    #数组求和
    array_sum = array.sum()
    #如果为素数，就计数
    if prime_number(array_sum):
        count += 1
print(count)
```

20. 栈

```
n = int(input())
num_li = [i for i in range(2,n+1)]
zhai = [1]
count_list = []
# 只有两种操作：要么zhai移除一个数字，要么numlist移给zhai一个数字
# 慢慢判断，如果zhai是空的,那么只能numlist移入
# 如果zhai不是空的,两种操作都可以
# 如果numlist空了,这个分支就结束了
# 注意下面一定要先复制一遍列表,不然的话一个列表在一个地方动过了,它所有地方都是变化的
def operation(num_li,zhai):
    list1 = [_ for _ in num_li]
    list2 = [_ for _ in zhai]
    if num_li == []:
        count_list.append(1)
    elif zhai == []:
        tmp = num_li.pop(0)
        zhai.append(tmp)
        operation(num_li,zhai)
    else:
        tmp = num_li.pop(0)
        zhai.append(tmp)
        operation(num_li, zhai)
        list2.pop(0)
        operation(list1,list2)
operation(num_li,zhai)
print(len(count_list))
```

> **题目描述**
>
> 
>
>
> 洛谷
>
> 宁宁考虑的是这样一个问题：一个操作数序列，$1,2,\ldots,n$（图示为 1 到 3 的情况），栈 A 的深度大于 n。
>
> 现在可以进行两种操作，
>
> 1. 将一个数，从操作数序列的头端移到栈的头端（对应数据结构栈的 push 操作）
> 2. 将一个数，从栈的头端移到输出序列的尾端（对应数据结构栈的 pop 操作）

你的程序将对给定的 n，计算并输出由操作数序列 $1,2,\ldots,n$ 经过操作可能得到的输出序列的总数。

输入格式

输入文件只含一个整数 n（$1 \leq n \leq 18$）。

输出格式

输出文件只有一行，即可能输出序列的总数目。

```python
n = int(input())
mul1 = 1
mul2 = 1
for i in range(n+1,2*n+1):
    mul1 *= i
for x in range(1,n+1):
    mul2 *= x
num = int(mul1//(mul2*(n+1)))
print(num)
```

这道题我首先想到的是递归, 具体思路详见代码的注释

但是在看题解的过程中发现其背后是有玄机的, 实际上这道题更像是数学题

卡特兰公式 CATALAN

3、数论做法 卡特兰/$Catalan$

既然很多Dalao都说过，那我直接给式子了；

- **递推式1：**

$$f[n] = f[0] * f[n-1] + f[1] * f[n-2] + ... + f[n-1] * f[0] (n \geq 2)$$

然后按照这个递推式模拟就好了(代码后面给)

既然上面标了1，那就有递推式2~

- **递推式2：**

$$h[n] = h[n-1] * (4 * n - 2)/(n+1)$$

依旧按式子模拟(代码后面给)

既然有2，那再来个3吧~

- **递推式3：**

$h[n] = C[2n, n]/(n+1) (n = 0, 1, 2, ...)$,$C$是组合数

$PS: C[m, n] = C[m-1, n-1] + C[m-1, n]$:且规定：$C[n, 0] = 1 C[n, n] = 1 C[0, 0] = 1$

这个公式也叫组合数公式(下面那个也是)

（不知道组合数可以百度）

于是仍然把标程放到最后~

- **递推式4：**

$h[n] = C[2n, n] - C[2n, n-1] (n = 0, 1, 2, ...)$ 组合数C不解释了；

21. 采药-----dp背包题

```
# 定义一个二维矩阵，M[i][j]代表在采第i株药的时候,花费j时间可以获得的价值的最大值
# 在dp书包类问题中，i+1和j+1分别为表格的列和行，一共(i+1)*(j+1)个元素，表格最后是要填满的才能完成任务
# 为什么要加1? 是为了在i 和 j 为1的时候方便进行推算
# 如果时间不够了的话，M[i][j] = M[i-1][j]，这是显而易见的
# 如果时间有的话，可以选择拿还是不拿;拿的话，这个药草的价值是到手了
# 但是时间是要预留出来的，上一个状态是M[i-1][j-time[i]]+value[i]
# 上一个状态先是默认已知的,因此拿还是不拿两个状态的价值我都是知道的,因此需要进行比较
# 最初的状态，即我们直观已知的数据,是第一行和第一列全是0
t,m = map(int,input().split())
time = []
value = []
for _ in range(m):
    a,b = map(int,input().split())
    time.append(a)
    value.append(b)
matrix = [[0 for _ in range(t+1)] for _ in range(m+1)]
for i in range(1,m+1):
```

```
    for j in range(1,t+1):
        if j < time[i-1]:
            matrix[i][j] = matrix[i-1][j]
        else:     # 注意为什么time和value里面的index都是要加1的!看我for里面是怎么设的!
            if matrix[i-1][j] >= matrix[i-1][j-time[i-1]] + value[i-1]:
                matrix[i][j] = matrix[i-1][j]
            else:
                matrix[i][j] = matrix[i-1][j-time[i-1]] + value[i-1]
print(matrix[m][t])
```

第一行有 22 个整数 *T*（1≤*T*≤1000）和 *M*（1≤*M*≤100），用一个空格隔开，*T* 代表总共能够用来采药的时间，M* 代表山洞里的草药的数目。

接下来的 M行每行包括两个在 11 到 100100 之间（包括 11 和 100100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

输出在规定的时间内可以采到的草药的最大总价值

22. 乘积最大

```

import itertools
import numpy as np
n,k = map(int,input().split())
num = input()
num_list = [i for i in num]
a = num_list[0]
b = num_list[-1]
num_list.pop(0)
num_list.pop(-1)
for _ in range(k):
    num_list.append('*')
coarse_list = list(itertools.permutations(num_list))
key_list = []
target = np.array(num_list)
for element in coarse_list:
    for i in range(n-1):
        if element[i] == '*' and element[i+1] == '*':
            break
    else:
        key_list.append(element)
ves_list = []
for elements in key_list:
    elements = list(elements)
    elements.insert(0,a)
    elements.append(b)
    ves_list.append(elements)
sum_list = []
for member in ves_list:
    str = ''.join(member)
```

```
    tmp_list = str.split('*')
    if ''.join(tmp_list) == num:
        sum = 1
        for i in range(len(tmp_list)):
            sum *= int(tmp_list[i])
        sum_list.append(sum)
num_list.clear()
key_list.clear()
coarse_list.clear()
print(max(sum_list))
```

设有一个长度为 N 的数字串，要求选手使用 K 个乘号将它分成 $K+1$ 个部分，找出一种分法，使得这 $K+1$ 个部分的乘积能够为最大。

同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：

有一个数字串：312，当 $N=3, K=1$ 时会有以下两种分法：

1. $3 \times 12 = 36$
2. $31 \times 2 = 62$

这时，符合题目要求的结果是: $31 \times 2 = 62$

现在，请你帮助你的好朋友 XZ 设计一个程序，求得正确的答案。

输入格式

程序的输入共有两行：

第一行共有 2 个自然数 N, K

第二行是一个长度为 N 的数字串。

输出格式

结果显示在屏幕上，相对于输入，应输出所求得的最大乘积（一个自然数）。