第十四届国赛python_B组

> 代码及题目参考：
>
> 荷碧TongZJ
>
> 蓝桥杯【第14届国赛】Python B组
>
> https://blog.csdn.net/qq_55745968/article/details/131141353?spm=1001.2014.3001.5506

A：弹珠堆放

题目

> 小蓝有 20230610 颗磁力弹珠，他对金字塔形状尤其感兴趣，如下图所示： 高度为 1 的金字塔需要 1 颗弹珠；高度为 2 的金字塔需要 4 颗弹珠；高度为 3 的金字塔需要 10 颗弹珠；高度为 4 的金字塔需要 20 颗弹珠。小蓝想要知道用他手里的弹珠可以摆出的最高的金字塔高度是多少?

题解

> 等差数列求出每一层需要多少弹珠，算出不能堆出的最高层之后减1就是答案。

```
ans=20230610
for i in range(1,ans):
    need=(i*(i+1)*(i+2))/6
    if need>ans:
        break
print(i-1)
```

B：划分

题目

> 给定 40 个数，请将其任意划分成两组，每组至少一个元素。每组的权值为 组内所有元素的和。划分的权值为两组权值的乘积。请问对于以下 40 个数，划分的权值最大为多少。
>
> 5160 9191 6410 4657 7492 1531 8854 1253 4520 9231 1266 4801 3484 4323 5070 1789 2744 5959 9426 4433 4404 5291 2470 8533 7608 2935 8922 5273 8364 8819 7374 8077 5336 8495 5602 6553 3548 5267 9150 3309
>
> 在试题包中有一个名为 nums.txt 的文本文件，文件中的数与题面上的数 相同。

题解

> 就是分成正方形最大面积那样子类似于，所以肯定是两边相等，那么能不能凑出两边相等呢，堵他能相等就行。

```python
with open('nums.txt') as f:
    array = sorted(map(int, f.read().split()), reverse=True)
print((sum(array)//2)**2)
```

C：偶串

题目

> 小蓝特别喜欢偶数，当他看到字符串时，他总数要检查一下是不是每种字 符都是出现偶数次。给定一个字符串，请帮助小蓝检查一下该字符串是否满足 要求。

题解

> Counter无需多言

```python
from collections import Counter
def solve(s):
    c=Counter(list(s))
    for item in c.values():
        if item%2!=0:
            return False
    return True

s=input()
if solve(s):
    print('YES')
else:
    print('NO')
```

D：交易账本

题目

> 小蓝最近研发了一种新的记账方式，并邀请了一些用户参加测试。交易账 本可以看作是交易记录的集合，每条交易记录都有着一个独一无二的交易编号 txId （编号大小反映了交易记录产生的时间顺序，txId 小的交易记录先发生于 txId 大的交易记录），每条交易记录包含一个或多个输入信息以及一个或多个输 出信息。
>
> 其中输入来自于已经发生过的某比交易的某个输出，可以理解为这笔钱从 某比交易输出后继续输入到了当前这比交易中，输入信息主要包含以下数据： fromTxId、fromTxOutNumber ，这表示当前输入来自于交易编号为 fromT xId 的第 fromTxOutNumber (fromTxOutNumber = 0, 1, 2, · · ·) 个输出；输出信息主 要包含以下数据：account、val ，表示将 val 数目的钱转移到了账户编号为 account 的账户上。注意，当 fromTxId 和 fromTxOutNumber 都为 −1 时，表明这是一笔特殊交易，由系统账户直接产生输出，特殊交易只含有一个输入和 一个输出，可以认为系统账户拥有无限多数目的钱，特殊交易一定可以成功。

> 个合法的账本应满足以下条件：1）对于每笔交易记录，所有的输入中涉 及到的钱的总数目应和所有输出中钱的总数目相等；2）交易中的一个输出要么 不使用，要使用的话输出中的钱应该全部分配给下一个输入，而不能分配给多 个输入（特殊交易除外）；3）交易按照顺序进行，不可以在某比交易中引用还 未发生的交易。
>
> 现在已知一共有 N 个不同的账户，初始时所有账户钱数目都为 0，账本上 总计有 M 条交易记录（按照交易完成的顺序进行记录），请你来判断下账本上 的记录是否是合法的。

题解

> 大模拟题目，去年在D也是一道这样的题目，这两道题目需要好好看一下，准备准备。

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    memory = []
    for _ in range(m):
        __info = map(int, input().split())
        # fromTxId, fromTxOutNumber
        in_dat = [[next(__info) for _ in range(2)] for _ in range(next(__info))]
        # accout, val, exist
        out_dat = [[next(__info) for _ in range(2)] + [True] for _ in
range(next(__info))]
        # 存储该交易信息
        memory.append([in_dat, out_dat])
    # 如果不是特殊交易，判断是否合法
    try:
        for idx, (in_dat, out_dat) in enumerate(memory):
            if in_dat[0] != [-1, -1]:
                in_cnt = 0
                for f, i in in_dat:
                    assert f < idx, f'该交易还未发生 {f} -> {idx}'
                    tmp = memory[f][1][i]
                    assert tmp[2], f'该输出已被占用 {tmp}'
                    memory[f][1][i][2] = False
                    in_cnt += tmp[1]
                out_cnt = sum(x[1] for x in out_dat)
                assert in_cnt == out_cnt, f'输入输出不符 {in_cnt}, {out_cnt}'
        print('Yes')
    except Exception as e:
        print('NO')
```

E：背包问题

题目

> 小蓝是一位狂热的积木爱好者，家里堆满了自己用积木组装的建筑模型。 最近，有两款新出的积木组件上市，小蓝自然不会错过，他带上了自己的三个 背包来到了积木商城，打算将尽可能多的积木组件带回家，每个背包都有一个 固定的空间大小。小蓝只会购买这两种新出的积木组件 A 和 B，A 和 B 各自会 占用背包的一部分空间，但对于同一种类型的积木占用的空间是相同的。小蓝想知道自己最多能带走多少数量的积木组件。
> 可以认为小蓝有足够的货币，只要背包可以装下的积木他都有能力购买。 商场内的积木数量也是有限制的。

题解

> 这道题目没有写出来没有关系，把自己能看懂的努力学会就ok

F：翻转

题目

> 小蓝制作了 n 个工件，每个工件用一个由小写英文字母组成的，长度为 2 的字符串表示，第 i 个工件表示为 si 。小蓝想把 n 个工件拼接到一起，方便转 移到另一个地方完成下一道工序，而拼接后的工件用字符串 S = s1 + s2 +...+ sn 表示，其中 + 表示一种奇特的拼接方式：对于 c = a + b 来说，如果 a 的第二 个字符和 b 的第一个字符相同，则拼接后的结果 c 长度为 3 而不是 4 ，中间相同的字符可以省略一个，比如 xy + yz = xyz 而 xy + zy = xyzy 。小蓝为了让拼 接后的字符串 S 的长度尽量小，可以将若干个工件进行左右翻转之后再进行拼 接，请问拼接后的字符串 S 的最小长度是多少?
> 请注意所有工件必须按出现顺序依次拼接，可以翻转任意工件。

题解

> 首先自然是读入所有的工件，然后判断是否旋转的时候这样，遍历从第二个往后的所有，先判断前一个是否需要翻转，然后判断后面一个是否需要翻转。最后拼接的时候进行一个模拟计算即可

```python
n = int(input())
subs = [input() for _ in range(n)]
# e.g.: ax, ba, cb
for i in range(1, n):
    # 翻转上一个工件
    if subs[i - 1][0] in subs[i]:
        subs[i - 1] = subs[i - 1][::-1]
    # 翻转当前的工件
    if subs[i][1] == subs[i - 1][1]:
        subs[i] = subs[i][::-1]
# 拼接并得到答案
res = subs.pop(0)
for x in subs:
    res += x[1] if res[-1] == x[0] else x
print(len(res))
```

G：最大阶梯

题目

> 小蓝特别喜爱阶梯图案，阶梯图案可以看做是由若干个大小和颜色都相同 的方格组成的，对于大小为 N 的阶梯图案，包含了 N 个连续的列，其中第 i 列 恰好有 i（1 ≤ i ≤ N）个方格，将这 N 列的底部对齐后便组成了一个阶梯图案， 将其按照 90 度旋转若干次后仍是阶梯图案，下图展示了几个不同大小的阶梯图案小蓝有一块大小为 H × H 的布匹，由 H × H 个大小相同的方格区域组成， 每一个方格都有自己的颜色。小蓝可以沿着方格的边缘对布匹进行裁剪，他想 要知道自己能得到的最大的同色阶梯图案的大小是多少?

题解

> 看博客吧，这道题目也没看，应该能看懂的，看不懂也没事，在最后写一个总结看一个这套题目有多少会的，不会的就不写就行，没事。

```
h = int(input())
array = [list(map(int, input().split())) for _ in range(h)]


def valtrig(arr, idx):
    mid, v = len(arr) // 2, set()
    # 左、右半区
    if idx in (0, 3):
        for r in range(len(arr)):
            for c in range(r + 1 if r < mid else len(arr) - r):
                if idx == 3: c = len(arr) - 1 - c
                v.add(arr[r][c])
                if len(v) > 1: return False
    # 上、下半区
    elif idx in (1, 2):
        for c in range(len(arr)):
            for r in range(c + 1 if c < mid else len(arr) - c):
                if idx == 2: r = len(arr) - 1 - r
                v.add(arr[r][c])
                if len(v) > 1: return False
    return True


def search(x):
    # 枚举左上顶点，计算方阵边界
    for l in range(h):
        r = l + x
        if r <= h:
            for t in range(h):
                b = t + x
                if b <= h:
                    arr = [row[l: r] for row in array[t: b]]
                    # 左右任意半区 + 上下任意半区 = 阶梯图案
                    if (valtrig(arr, 0) or valtrig(arr, 3)) and (
                        valtrig(arr, 1) or valtrig(arr, 2)): return True
    return False


l, r = 1, h
# 二分答案
while l + 1 != r:
    mid = (l + r) // 2
    if search(mid):
        l = mid
    else:
        r = mid
print(l)
```

H：最长回文前后缀

题目

> 给定一个字符串 S ，请找出 S 的一个前缀和后缀，使得它们拼接后是一个 回文串。请输出这个串的最长长度

题解

> 暴力骗分,没有必要用其他技巧，我知道有其他板子算法，但是关键是板子算法记不住啊，自己也手残写不出来~

```
def main():
    s = input()
    for x in range(len(s) * 2, 1, -1):
        # 枚举前缀的长度
        for i in range(max(1, x - len(s)),
                       min(len(s) + 1, x)):
            # 后缀的长度: x - i
            tmp = s[:i] + s[i - x:]
            if tmp == tmp[::-1]: return x
    return 1

print(main())
```

I：贸易航线

题目

> 小蓝要带领一支船队依次经过 n 个地点。小蓝的船上可以携带 k 单位的商品，商品共有 m 种，在每个地点的价格各不相同，部分商品在部分地区无法交易。
>
> 一开始小蓝的船是空的，小蓝可以在每个地点任意地买入各种商品，然后 在之后经过的地点卖出。小蓝的钱很多，你可以认为小蓝买入商品时只会受到 船队的容量的限制。
>
> 问小蓝到达终点时的最大收益是多少。

题解

> 因为货物和钱是无限的，只有容量是有限的，所以只有两种状态：空载、满载某一种货物以 dp[i] 表示载满第 i 种货物时的收益，额外增加 dp[-1] 表示空载时的收益
>
> 每到达一个地点，先把所有的货物卖出，把满载某一种货物的状态 dp[:-1] 传递到 dp[-1]，得到空载时的最优状态
>
> 然后再买入货物，把 dp[-1] 传递到 dp[:-1] 得到满载某一种货物的最优状态

```python
n, m, k = map(int, input().split())

# dp[-1] 表空载, dp[i] 表载满第 i 种货物时的收益
dp = m * [-float('inf')] + [0]
for _ in range(n):
    price = tuple(map(lambda x: int(x) * k, input().split()))
    # 满载时, 卖出不买入
    for i, p in enumerate(price):
        if p > 0: dp[-1] = max(dp[-1], dp[i] + p)
    # 空载时再买入
    for i, p in enumerate(price):
        if p > 0: dp[i] = max(dp[i], dp[-1] - p)
print(dp[-1])
```

J：困局

题目

> 小蓝发现有个小偷在坐标 0 处，正在往 x 轴正方向逃离并保持逃离方向不 变。小蓝不想让他快速逃离（即走到坐标 n + 1 处），准备设立 k 道双向传送门 拖延其时间，每道传送门可以连接 x 轴上的两个不同的整点坐标 p ↔ q ，其中 p, q ∈ [1, n] ，同时每个坐标上最多作为一道传送门的端点。
> 当小偷达到一个整点时，如果其上有传送门，则会触发传送门到达传送门 另一端，当然同一个传送门不能连续触发，当无法传送时小偷会保持向 x 轴正 方向移动。小蓝想通过设置这些传送门使得小偷被至少传送 2k 次，请问有多少 种设置传送门的方式可以完成目标?

题解

> 最后一题不用写管啥呢

X：总结

> 能写的就是AB两个填空题，C考counter，D考大模拟，F算是遍历？能写但好像自己想不到，H直接暴力就行不管算法，H是动态规划不一定写得出来，但是代码很短

> 不用写的题目：E动态规划不用写，G是二分但是不止二分写出来有难度，J不用写。

看自己能力的话，现在应该是改到只有2道填空题，6道大题，总共8道题目了，填空题第一个要写出来，第二题写不出就算了，填空题第一道要写出来，第二道都是大模拟，沉下心来冷静写，再后面四道大题基本就是两个动态规划一个简单一个困难，然后一个字符串，一个路线或者图论之类的。后面四道题目能写出来两道就非常不错了，因为最后一道肯定不会写，剩下就是三道题目，有两道会那挺厉害了，本来就没学多少。