第十三次作业：

OJ02692：假币问题

```
def check(coins, case):
    for item in case:
        left, right, res = item.split()

        left_total = sum(coins[i] for i in left)
        right_total = sum(coins[i] for i in right)

        if left_total == right_total and res != 'even':
            return False
        elif left_total < right_total and res != 'down':
            return False
        elif left_total > right_total and res != 'up':
            return False

    return True

n = int(input())  # 输入测试案例数

for _ in range(n):
    test = [input().strip() for _ in range(3)]  # 获取当前测试用例的 3 行数据

    # 对每个可能的伪造币和重量组合进行测试
    for counterfeit in 'ABCDEFGHIJKL':
        found = False
        for weight in [-1, 1]:
            coins = {coin: 0 for coin in 'ABCDEFGHIJKL'}
            # 重置当前伪造币的重量
            coins[counterfeit] = weight

            # 检查当前的硬币分配是否符合所有条件
            if check(coins, test):
                found=True
                tag = 'light' if weight == -1 else 'heavy'
                print(f'{counterfeit} is the counterfeit coin and it is {tag}.')
                break  # 找到一个伪造币后，不再检查其他伪造币
        if found:
            break
```

注意循环的逻辑以及该 break 就 break；对字典的运用；生成器的运用

思路：

初始化：

coins 字典初始化为 {A: 0, B: 0, ..., L: 0}。

第一轮检查伪造币 A：

对 A 设置 weight = -1 或 weight = 1，其他硬币的重量保持为 0。
检查每个称重操作（例如 A B C even）。由于只修改了 A 的重量，检查会正确验证 A 是否符合假币的条件。
第二轮检查伪造币 B：
重新初始化 coins 字典，然后设置 B 的重量。其他硬币的状态保持为 0。
检查当前状态是否符合条件。
输出伪造币：
如果在某一轮找到了符合条件的伪造币（例如 A 是轻的），程序会输出 A is the counterfeit coin and it is light. 并终止检查。

OJ01088：滑雪

```
r, c = map(int, input().split())
ski = [list(map(int, input().split())) for _ in range(r)]

directions = [(-1, 0), (1, 0), (0, 1), (0, -1)]

# dp[i][j] 表示从 (i, j) 位置出发的最长下降路径长度
dp = [[-1] * c for _ in range(r)]  # 初始化为 -1，表示尚未计算


def dfs(maze, x, y):
    if dp[x][y] != -1:  # 如果已经计算过，直接返回结果
        return dp[x][y]

    max_step = 1  # 起始点的路径长度至少为 1
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < r and 0 <= ny < c and maze[nx][ny] < maze[x][y]:
            max_step = max(max_step, 1 + dfs(maze, nx, ny))

    dp[x][y] = max_step  # 缓存计算结果
    return dp[x][y]


# 计算所有格子的最长路径
res = 0
for i in range(r):
    for j in range(c):
        res = max(res, dfs(ski, i, j))

print(res)
```

| 比赛 | 题目 | 结果 | 内存 | 时间 | 代码长度 | 语言 | 提交时间 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 题库（包括计概、数算题目） | 01088: 滑雪 | Accepted | 4396kB | 42ms | 966 B | Python3 | 刚刚 |

（一开始用 visited 来标记访问过/没访问过，来回溯，但是超时了，于是设置了这个 dp 数组来缓存之前的值，避免重复计算路径。这个 dp 数组起到的作用有点像一个“前缀和”的作用，降低所需的时间）

OJ25572：螃蟹采蘑菇

```
from collections import deque

n = int(input())
mat=[list(map(int,input().split())) for _ in range(n)]

a = []
for i in range(n):
    for j in range(n):
        if mat[i][j] == 5:
            a.append([i, j])

lx = a[1][0] - a[0][0]
ly = a[1][1] - a[0][1]
dire = [[-1, 0], [0, 1], [1, 0], [0, -1]]
v = [[0] * n for i in range(n)]

def bfs(x, y):
    v[x][y] = 1
    queue = deque([(x, y)])
    while queue:
        x, y = queue.popleft()
        if (mat[x][y] == 9 and mat[x + lx][y + ly] != 1) or \
                (mat[x][y] != 1 and mat[x + lx][y + ly] == 9):
            return 'yes'
        for _ in range(4):
            dx = x + dire[_][0]
            dy = y + dire[_][1]
            if 0 <= dx < n and 0 <= dy < n and 0 <= dx + lx < n \
                    and 0 <= dy + ly < n and v[dx][dy] == 0 \
                    and mat[dx][dy] != 1 and mat[dx + lx][dy + ly] != 1:
                queue.append([dx, dy])
                v[dx][dy] = 1
    return 'no'
```

```
print(bfs(a[0][0], a[0][1]))
```

| 比赛 | 题目 | 结果 | 内存 | 时间 | 代码长度 | 语言 | 提交时间 |
|---|---|---|---|---|---|---|---|
| 题库（包括计概、数算题目） | 25572: 螃蟹采蘑菇 | Accepted | 3744kB | 22ms | 996 B | Python3 | 刚刚 |

本来想用 dfs 做的但一直 wa...所以最后还是看了题解的 bfs

OJ27373：最大整数

```
m=int(input())
n=int(input())
nums=input().split()

def f(string):
    if string == '':
        return 0
    else:
        return int(string)

for i in range(n-1):
    for j in range(i+1,n):
        if nums[i]+nums[j]<nums[j]+nums[i]:
            nums[i],nums[j]=nums[j],nums[i]

weight=[]
for num in nums:
    weight.append(len(num))
dp=[['']*(m+1) for _ in range(n+1)]
for _ in range(n+1):
    dp[_][0]=''
for _ in range(m+1):
    dp[0][_]=''

for k in range(1,n+1):
    for t in range(1,m+1):
        if weight[k-1]>t:
            dp[k][t]=dp[k-1][t]
        else:
            dp[k][t]=str(max(f(dp[k-1][t]),int(dp[k-1][t-weight[k-1]]+nums[k-1])))
print(dp[n][m])
```

| 比赛 | 题目 | 结果 | 内存 | 时间 | 代码长度 | 语言 | 提交时间 |
|---|---|---|---|---|---|---|---|
| 题库（包括计概、数算题目） | 27373: 最大整数 | Accepted | 31228kB | 586ms | 666 B | Python3 | 刚刚 |

OJ02811：熄灯问题

#1）第 2 次按下同一个按钮时，将抵消第 1 次按下时所产生的结果。因此，每个按钮最多只需要按下一次；2）各个按钮被按下的顺序对最终的结果没有影响；3）对第 1 行中每盏点亮的灯，按下第 2 行对应的按钮，就可以熄灭第 1 行的全部灯。如此重复下去，可以熄灭第 1、2、3、4 行的全部灯。同样，按下第 1、2、3、4、5 列的按钮，可以熄灭前 5 列的灯。

```
import copy
X=[[0,0,0,0,0,0,0,0]]
for _ in range(5):
    X.append([0]+[int(x) for x in input().split()]+[0])
X.append([0,0,0,0,0,0,0,0])
Y=[[0]*8 for _ in range(7)]

for a in (0,1):
    Y[1][1]=a
    for b in (0,1):
        Y[1][2]=b
        for c in (0, 1):
            Y[1][3] = c
            for d in (0, 1):
                Y[1][4] = d
                for e in (0, 1):
                    Y[1][5] = e
                    for f in (0, 1):
                        Y[1][6] = f

                        A=copy.deepcopy(X)
                        B=copy.deepcopy(Y)
                        for i in range(1, 7):
                            if B[1][i] == 1:
                                A[1][i] = abs(A[1][i] - 1)
                                A[1][i - 1] = abs(A[1][i - 1] - 1)
                                A[1][i + 1] = abs(A[1][i + 1] - 1)
                                A[2][i] = abs(A[2][i] - 1)
                        for i in range(2,6):
                            for j in range(1,7):
                                if A[i-1][j]==1:
                                    B[i][j]=1
                                    A[i][j]=abs(A[i][j]-1)
                                    A[i-1][j] = abs(A[i-1][j] - 1)
                                    A[i+1][j] = abs(A[i+1][j] - 1)
                                    A[i][j-1] = abs(A[i][j-1] - 1)
                                    A[i][j+1] = abs(A[i][j+1] - 1)

                        if all(A[5][_] == 0 for _ in range(1,7)):
```

```
for m in range(1, 6):
  print(" ".join(str(B[m][n]) for n in range(1, 7)))
```

| 比赛 | 题目 | 结果 | 内存 | 时间 | 代码长度 | 语言 | 提交时间 |
|---|---|---|---|---|---|---|---|
| 题库（包括计概、数算题目） | 02811: 熄灯问题 | Accepted | 3808kB | 28ms | 2075 B | Python3 | 刚刚 |

08210：河中跳房子

```
L,N,M=map(int,input().split())
distance=[0]
for _ in range(N):
    distance.append(int(input()))
distance.append(L)

n=N+2
def check(x):
    num=0
    now_rock=0
    for i in range(1,n):
        if distance[i]-now_rock<x:
            num+=1
        else:
            now_rock=distance[i]
    if num>M:
        return True
    else:
        return False

lo, hi = 0, L+1
shortest=-1
while lo<hi:
    mid = (lo+hi)//2
    if check(mid):
        hi=mid
    else:
        shortest=mid
        lo=mid+1

print(shortest)
```

| 比赛 | 题目 | 结果 | 内存 | 时间 | 代码长度 | 语言 | 提交时间 |
|---|---|---|---|---|---|---|---|
| 题库（包括计概、数算题目） | 08210: 河中跳房子 | Accepted | 6028kB | 259ms | 514 B | Python3 | 刚刚 |

学到了二分查找缩小区间的做题方式！

感想：复习中，正在 cheatsheet 上总结各种题型（尤其是 dp/bfs/dfs）的模板代码，希望机考善待我。。。。。