1. 题目

E04015: 邮箱验证

strings, http://cs101.openjudge.cn/practice/04015

思路：对输入依次检查不同条件，并用 try-except 来处理不确定结束输入的情况。

代码：

```
while True:
    try:
        mail=input().strip()
        if not mail:
            break
        if mail.count("@")!=1:
            print("NO")
        elif mail[0] in {'@', '.'} or mail[-1] in {'@', '.'}:
            print("NO")
        elif '.' not in mail[mail.index("@"):]:
            print("NO")
        elif mail[mail.index("@") -1] == '.' or mail[mail.index("@") + 1] == '.':
            print("NO")
        else:
            print("YES")
    except EOFError:
        break
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

大约用时：30 分钟

M02039: 反反复复

implementation, http://cs101.openjudge.cn/practice/02039/

思路：将输入的句子按照 col 分为多行并存入列表，并对奇数行反转，最后按列顺序输出每一列字符。

代码：

```
col=int(input())
sen=input()
word=len(sen)
res=[]
for i in range(word//col):
    ltr=sen[:col]
    sen=sen[col:]
    res.append(ltr)
for i,j in enumerate(res):
```

```
    if i % 2 != 0:
        res[i]=j[::-1]
for k in range(col):
    for q in range(len(res)):
        print(res[q][k],end="")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

大约用时：30 分钟

M02092: Grandpa is Famous

implementation, http://cs101.openjudge.cn/practice/02092/

思路：先用 times 字典记录每个 players 再所有比赛中出现的次数，之后将球员按出现次数分组，存入 dict 字典，keys 是出现次数，value 是球员列表，之后降序排序找出现次数第二多的 players 后去重再按升序排序输出。

代码：

```
while True:
    N,M=map(int,input().split())
    if N==0 and M==0:
        break
    times={}
    for _ in range(N):
        players=list(map(int,input().split()))
        for i in players:
            if i not in times:
                times[i]=1
            elif i in times:
                times[i]+=1
    dict={}

    for i,j in times.items():
        if j not in dict:
            dict[j]=[i]
        else:
            dict[j].append(i)
    sort_dict=sorted(dict.items(),key=lambda x:x[0],reverse=True)
    a=sorted(set(sort_dict[1][1]))
    print(" ".join(map(str,a)))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

大约用时：30 分钟

M04133: 垃圾炸弹

matrices, http://cs101.openjudge.cn/practice/04133/

思路：读取每条道路位置和垃圾数量后，通过两层循环遍历从(0, 0)到(1024, 1024)的所有位置，计算每个位置周围的垃圾量，然后对每个位置检查，如果该位置在炸弹范围内（abs(i - x) <= bomb 和 abs(j - y) <= bomb），就加上该道路的垃圾量，之后更新最大的垃圾量 garbage 和达到该垃圾量的次数 count，并输出。

代码：

```
bomb=int(input())
road=int(input())
total=[]
for _ in range(road):
    x,y,i=list(map(int,input().split()))
```

```
    total.append([x,y,i])
# min_x=min([x for x,_,_ in total])
# max_x=max([x for x,_,_ in total])
# min_y=min([y for _,y,_ in total])
# max_y=max([y for _,y,_ in total])
garbage=0
count=0
for i in range(1025):
    for j in range(1025):
        n_gar=0
        for x,y,c in total:
            # print(f"x,y:{i},{j}")
            # print(f"c:{c}")
            if abs(i - x) <= bomb and abs(j - y) <= bomb:
                n_gar+=c
        if n_gar>garbage:
            garbage=n_gar
            count=1
        elif n_gar == garbage:
            count+=1
print(count,garbage)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

大约用时：45 分钟

T02488: A Knight's Journey

backtracking, http://cs101.openjudge.cn/practice/02488/

思路：先定义所有合法动作，然后定义 tour 函数用于尝试从每一个位置开始进行回溯（step 函数），寻找访问所有格子的路径，如果找到路径，则返回该路径，否则返回 None。在 step 函数中，当路径的长度等于棋盘上的格子数时，说明已访问所有格子，返回当前路径，否则继续尝试访问尚未访问过的格子并将其加入 next_moves 列表中，之后对列表按列优先（字母顺序）对下一步位置进行排序并标记（已访问）当前位置加入到 path 中。如果路径不成功，则回溯并撤销当前格子的访问状态直到找到合适路径。最后通过 alphabet 函数将数字位置转换为所需的形式并输出，如果找不到则输出 impossible。

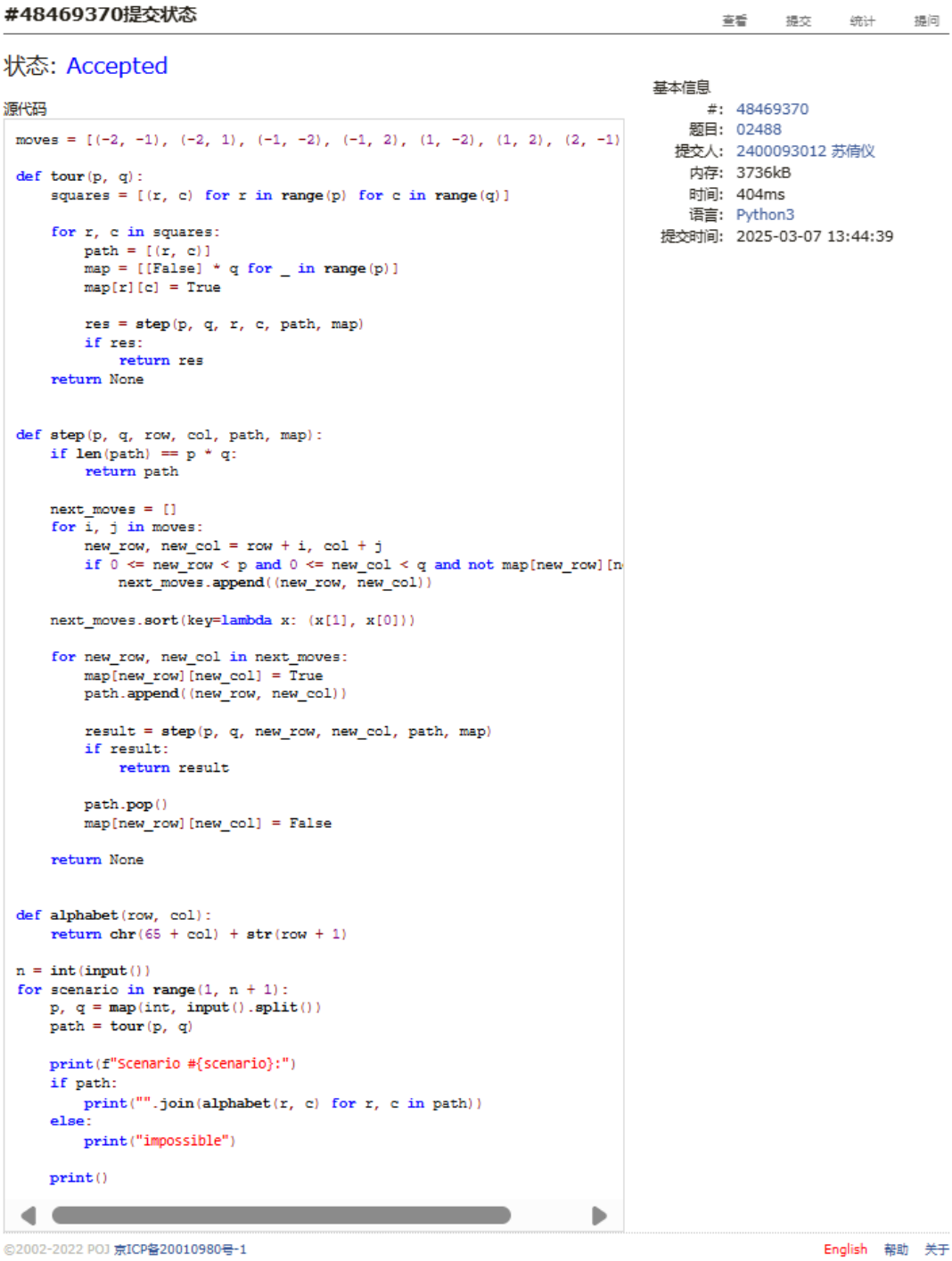
代码：

moves = [(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1), (2, 1)]

```
def tour(p, q):
    squares = [(r, c) for r in range(p) for c in range(q)]

    for r, c in squares:
        path = [(r, c)]
        map = [[False] * q for _ in range(p)]
        map[r][c] = True

        res = step(p, q, r, c, path, map)
        if res:
            return res
    return None


def step(p, q, row, col, path, map):
    if len(path) == p * q:
        return path

    next_moves = []
    for i, j in moves:
        new_row, new_col = row + i, col + j
        if 0 <= new_row < p and 0 <= new_col < q and not map[new_row][new_col]:
            next_moves.append((new_row, new_col))

    next_moves.sort(key=lambda x: (x[1], x[0]))

    for new_row, new_col in next_moves:
        map[new_row][new_col] = True
        path.append((new_row, new_col))

        result = step(p, q, new_row, new_col, path, map)
        if result:
            return result

        path.pop()
        map[new_row][new_col] = False

    return None
```

```
def alphabet(row, col):
    return chr(65 + col) + str(row + 1)

n = int(input())
for scenario in range(1, n + 1):
    p, q = map(int, input().split())
    path = tour(p, q)

    print(f"Scenario #{scenario}:")
    if path:
        print("".join(alphabet(r, c) for r, c in path))
    else:
        print("impossible")
    print()
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



#48469370提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 48469370
题目: 02488
提交人: 2400093012 苏倩仪
内存: 3736kB
时间: 404ms
语言: Python3
提交时间: 2025-03-07 13:44:39

源代码

```
moves = [(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1)

def tour(p, q):
    squares = [(r, c) for r in range(p) for c in range(q)]

    for r, c in squares:
        path = [(r, c)]
        map = [[False] * q for _ in range(p)]
        map[r][c] = True

        res = step(p, q, r, c, path, map)
        if res:
            return res
    return None


def step(p, q, row, col, path, map):
    if len(path) == p * q:
        return path

    next_moves = []
    for i, j in moves:
        new_row, new_col = row + i, col + j
        if 0 <= new_row < p and 0 <= new_col < q and not map[new_row][n
            next_moves.append((new_row, new_col))

    next_moves.sort(key=lambda x: (x[1], x[0]))

    for new_row, new_col in next_moves:
        map[new_row][new_col] = True
        path.append((new_row, new_col))

        result = step(p, q, new_row, new_col, path, map)
        if result:
            return result

        path.pop()
        map[new_row][new_col] = False

    return None


def alphabet(row, col):
    return chr(65 + col) + str(row + 1)

n = int(input())
for scenario in range(1, n + 1):
    p, q = map(int, input().split())
    path = tour(p, q)

    print(f"Scenario #{scenario}:")
    if path:
        print("".join(alphabet(r, c) for r, c in path))
    else:
        print("impossible")

    print()
```

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

大约用时：1 小时 30 分钟

T06648: Sequence

heap, http://cs101.openjudge.cn/practice/06648/

思路：初始化第一行（第一个序列），然后依次读取每个下一行的第一个元素并与第一行进行合并存入最小堆，并取出最小数存入 result，之后当 first_line 中的元素不足 n 个时则依次计算每一行的下一个元素并加入到 first_line 中，再调用 heappush 依次取出当前最小的数，合并新的 n 个最小和作为第一行用于下一轮合并，反复循环直到最后输出 n 个最小的和。

代码：

```
import heapq

t=int(input())
for _ in range(t):
    m,n=map(int,input().split())
    first_line=sorted(map(int,input().split()))

    for _ in range(m-1):
        next_line=sorted(map(int,input().split()))
        result=[]
        min_heap=[(first_line[i]+next_line[0],i,0) for i in range(n)]
        heapq.heapify(min_heap)

        for _ in range(n):
            min_s,i,j=heapq.heappop(min_heap)
            result.append(min_s)

            if j+1<len(next_line):
                heapq.heappush(min_heap,(first_line[i]+next_line[j+1],i,j+1))

        first_line=result
    print(*first_line)
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

#48466537提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

t=int(input())
for _ in range(t):
    m,n=map(int,input().split())
    first_line=sorted(map(int,input().split()))

    for _ in range(m-1):
        next_line=sorted(map(int,input().split()))
        result=[]
        min_heap=[(first_line[i]+next_line[0],i,0) for i in range(n)]
        heapq.heapify(min_heap)

        for _ in range(n):
            min_s,i,j=heapq.heappop(min_heap)
            result.append(min_s)

            if j+1<len(next_line):
                heapq.heappush(min_heap,(first_line[i]+next_line[j+1],i
                
        first_line=result
    print(*first_line)
```

基本信息

#: 48466537
题目: 06648
提交人: 2400093012 苏倩仪
内存: 7488kB
时间: 960ms
语言: Python3
提交时间: 2025-03-06 23:02:04

English 帮助 关于

大约用时：1 小时

2. 学习总结和收获

<mark>如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算 2025spring 每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。</mark>

感觉前三题比较基础，到第四题时以为遍历两次 1025 会超时所以原本想用最小和最大 xy 来做的，但是发现考虑的不够，后来用了 1025 也发现是对的。第五题我在去年的大作业也有做过类似的冰湖挑战，但是已经忘了七七八八了 hhh，再做一次还是觉得很难，尝试了很多次，第六题中因为之前没用过 heapq 包，导致一直 exceed memory limit，听朋友说了之后第一次尝试了 heapq，学到了利用最小堆来找出最小和，避免了暴力计算。