# Assignment #3: 惊蛰 Mock Exam

Updated 1641 GMT+8 Mar 5, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

> **说明：**
>
> 1. **惊蛰月考**：**AC4** 。考试题目都在"题库（包括计概、数算题目）"里面，按照数字题号能找到，可以重新提交。作业中提交自己最满意版本的代码和截图。
> 2. **解题与记录：**
>
>    对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示"Accepted"的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
> 3. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的"作业评论"区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且"作业评论"区包含上传的.md或.doc附件。
> 4. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

## 1. 题目

### E04015: 邮箱验证

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

代码：

```
def mail(s : str):
    pos, cnt = None, 0
    for i, c in enumerate(s):
        if c == "@":
            cnt += 1
            pos = i

    if cnt != 1 or s[0] in {".", "@"} or s[-1] in {".", "@"} :
        return "NO"
    if s[pos + 1] == "." or s[pos - 1] == ".":
        return "NO"
    for c in s[pos + 1:]:
        if c == ".":
            return "YES"
    return "NO"

```

```
while 1:
    try:
        s = input()
        print(mail(s))
    except EOFError:
        break
```

## 状态: Accepted

源代码

```
def mail(s : str):
    pos, cnt = None, 0
    for i, c in enumerate(s):
        if c == "@":
            cnt += 1
            pos = i

    if cnt != 1 or s[0] in {".", "@"} or s[-1] in {".", "@"} :
        return "NO"
    if s[pos + 1] == "." or s[pos - 1] == ".":
        return "NO"
    for c in s[pos + 1:]:
        if c == ".":
            return "YES"
    return "NO"

while 1:
    try:
        s = input()
        print(mail(s))
    except EOFError:
        break
```

## M02039: 反反复复

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

代码：Trivial

```
n = int(input())
s = input()
m = len(s) // n
mat = [[" "] * n for _ in range(m)]

for i in range(m * n):
    x = i // n
    y = i % n
    if x % 2 == 1:
        y = n - 1 - y
    mat[x][y] = s[i]
```

```
12
13 for i in range(n):
14     for j in range(m):
15         print(mat[j][i], end = "")
```

## 状态: Accepted

源代码

```
n = int(input())
s = input()
m = len(s) // n
mat = [[" "] * n for _ in range(m)]

for i in range(m * n):
    x = i // n
    y = i % n
    if x % 2 == 1:
        y = n - 1 - y
    mat[x][y] = s[i]

for i in range(n):
    for j in range(m):
        print(mat[j][i], end = "")
```

## M02092: Grandpa is Famous

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

代码：Trivial

```
1  while 1:
2      n, m = map(int, input().split())
3      if n == 0:
4          break
5      cnt = {}
6      for _ in range(n):
7          l = list(map(int, input().split()))
8          for num in l:
9              if num in cnt:
10                 cnt[num] += 1
11             else:
12                 cnt[num] = 1
13     players = []
14     for key, value in cnt.items():
15         players.append((value, key))
16     players = sorted(players, reverse=True)
17
18     Ans = []
19     for i, (pt1, num) in enumerate(players):
20         if pt1 == players[1][0]:
```

```
            Ans.append(num)
    Ans = sorted(Ans)
    print(*Ans, sep = " ")
```

## 状态: Accepted

源代码

```
while 1:
    n, m = map(int, input().split())
    if n == 0:
        break
    cnt = {}
    for _ in range(n):
        l = list(map(int, input().split()))
        for num in l:
            if num in cnt:
                cnt[num] += 1
            else:
                cnt[num] = 1
    players = []
    for key, value in cnt.items():
        players.append((value, key))
    players = sorted(players, reverse=True)


    Ans = []
    for i, (pt1, num) in enumerate(players):
        if pt1 == players[1][0]:
            Ans.append(num)
    Ans = sorted(Ans)
    print(*Ans, sep = " ")
```

## M04133: 垃圾炸弹

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

思路:

代码:

```
d = int(input())
n = int(input())
trash = []
for _ in range(n):
    x, y, i = map(int, input().split())
    trash.append((x, y, i))


def inrange(x):
```

```
    return -d <= x and x <= d

def cnt(nx, ny):
    sum = 0
    for x, y, i in trash:
        if inrange(nx - x) and inrange(ny - y):
            sum += i
    return sum

MAX, visited = 0, set()
for x, y, _ in trash:
    for i in range(- d, d + 1):
        for j in range(- d, d + 1):
            nx, ny = x + i, y + j
            if nx not in range(0, 1025) or ny not in range(1025):
                continue
            sum = cnt(nx, ny)
            if sum == MAX:
                visited.add((nx, ny))
            elif sum > MAX:
                visited = {(nx, ny)}
                MAX = sum

print(len(visited), MAX)
```

## 状态: Accepted

源代码

```
d = int(input())
n = int(input())
trash = []
for _ in range(n):
    x, y, i = map(int, input().split())
    trash.append((x, y, i))


def inrange(x):
    return -d <= x and x <= d

def cnt(nx, ny):
    sum = 0
    for x, y, i in trash:
        if inrange(nx - x) and inrange(ny - y):
            sum += i
    return sum

MAX, visited = 0, set()
for x, y, _ in trash:
    for i in range(- d, d + 1):
        for j in range(- d, d + 1):
            nx, ny = x + i, y + j
            if nx not in range(0, 1025) or ny not in range(1025):
                continue
            sum = cnt(nx, ny)
            if sum == MAX:
                visited.add((nx, ny))
            elif sum > MAX:
                visited = {(nx, ny)}
                MAX = sum

print(len(visited), MAX)
```

## T02488: A Knight's Journey

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

思路:

代码:

```
dir = [(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1), (2, 1)]

def isValid(board, x, y):
```

```python
    return x in range(len(board)) and y in range(len(board[0])) and board[x][y] == 0

def dfs(board, start): # return 在 board 基础上从 start 开始继续走的路径, 无路径则 return None
    sx, sy = start
    if board[sx][sy] == len(board) * len(board[0]):
        return board
    for dx, dy in dir:
        nx, ny = sx + dx, sy + dy
        if not isValid(board, nx, ny):
            continue
        board[nx][ny] = board[sx][sy] + 1
        new_board = dfs(board, (nx, ny))
        if new_board:
            return new_board
        board[nx][ny] = 0
    return None

def find_board(m, n):
    board = [[0] * m for _ in range(n)] # board 存储每个格子的 step, 0 表示暂时还未走到
    for i in range(n):
        for j in range(m):
            board[i][j] = 1
            new_board = dfs(board, (i, j))
            if new_board:
                return new_board
    return None

def num_to_pos(x, y):
    return chr(ord("A") + x) + str(y + 1)

def solution():
    m, n = map(int, input().split())
    board = find_board(m, n)
    if not board:
        print("impossible\n")
        return
    result = [0] * (len(board) * len(board[0]))
    for i in range(len(board)):
        for j in range(len(board[0])):
            step = board[i][j] - 1
            result[step] = num_to_pos(i, j)
    print(*result, sep = "", end = "\n\n")

if __name__ == "__main__":
    T = int(input())
    for i in range(1, T + 1):
        print(f"Scenario #{i}:")
        solution()
```

## 状态：Accepted

源代码

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

def isValid(board, x, y):
    return x in range(len(board)) and y in range(len(board[0])) and boa

def dfs(board, start): # return 在 board 基础上从 start 开始继续走的路径，无
    sx, sy = start
    if board[sx][sy] == len(board) * len(board[0]):
        return board
    for dx, dy in dir:
        nx, ny = sx + dx, sy + dy
        if not isValid(board, nx, ny):
            continue
        board[nx][ny] = board[sx][sy] + 1
        new_board = dfs(board, (nx, ny))
        if new_board:
            return new_board
        board[nx][ny] = 0
    return None

def find_board(m, n):
    board = [[0] * m for _ in range(n)] # board 存储每个格子的 step, 0 表示
    for i in range(n):
        for j in range(m):
            board[i][j] = 1
            new_board = dfs(board, (i, j))
            if new_board:
                return new_board
    return None

def num_to_pos(x, y):
    return chr(ord("A") + x) + str(y + 1)

def solution():
    m, n = map(int, input().split())
    board = find_board(m, n)
    if not board:
        print("impossible\n")
        return
    result = [0] * (len(board) * len(board[0]))
    for i in range(len(board)):
        for j in range(len(board[0])):
            step = board[i][j] - 1
            result[step] = num_to_pos(i, j)
    print(*result, sep = "", end = "\n\n")

if __name__ == "__main__":
    T = int(input())
    for i in range(1, T + 1):
        print(f"Scenario #{i}:")
        solution()
```

## T06648: Sequence

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

思路：关键在于如何将 `merge` 函数优化至 $O(n \log n)$

代码：

```python
from heapq import heapify, heappop, heappush

def merge(a, b, n):
    heap = [(a[i] + b[0], i, 0) for i in range(n)]
    heapify(heap)
    result = []
    for _ in range(n):
        sum_val, i, j = heappop(heap)
        result.append(sum_val)
        if j + 1 < n:
            heappush(heap, (a[i] + b[j + 1], i, j + 1))
    return result

def Solution():
    m, n = map(int, input().split())
    nums = [sorted(list(map(int, input().split()))) for _ in range(m)]
    for i in range(1, m):
        nums[i] = merge(nums[i - 1], nums[i], n)
    print(*nums[-1], sep = " ")

T = int(input())
for _ in range(T):
    Solution()
```

## 状态：Accepted

源代码

```
from heapq import heapify, heappop, heappush

def merge(a, b, n):
    heap = [(a[i] + b[0], i, 0) for i in range(n)]
    heapify(heap)
    result = []
    for _ in range(n):
        sum_val, i, j = heappop(heap)
        result.append(sum_val)
        if j + 1 < n:
            heappush(heap, (a[i] + b[j + 1], i, j + 1))
    return result

def Solution():
    m, n = map(int, input().split())
    nums = [sorted(list(map(int, input().split()))) for _ in range(m)]
    for i in range(1, m):
        nums[i] = merge(nums[i - 1], nums[i], n)
    print(*nums[-1], sep = " ")

T = int(input())
for _ in range(T):
    Solution()
```

# 2. 学习总结和收获

考场经验：

1. 考试时无法访问提交记录，**代码写完记得存档**，方便跳题后回来调试 (这回把代码重新写了一遍 😭) ①
2. 持续输入：①

```
import sys

for line in sys.stdin:
    s = line.strip()
    pass
```

或者

```
while 1:
    try:
        s = input()
        pass
    except EOFError:
        break
    # except : break OpenJudge 抽风时候使用
```

这样的持续输入技巧值得学习

3. ①题目中**'@'不能和'.'直接相连** 有歧义, 开始认为 `.` 不能在`@`后面, 但可以在其前面
4. 调试时可以用

```
import sys
sys.stdin = open("input", "r")
```

节省测试时间, 但是**提交代码时记得删掉这两行**

4. 英文题非常恶心, 其中 ③ 很长时间没有理解题意, 跳过后回来发现 **Considering that each appearance in a weekly ranking constitutes a point for the player** 介绍排序规则, 藏在`描述` 的一大段文字中很隐蔽.

~~下次考试可以带纸质英文词典~~ 😉

5. 注意数据范围

其中⑤只要求 `p * q <= 26`, 最傻瓜的搜索都可以过, 但考试时被吓唬到了

然而⑥考试时写出来一个 $O(n^2)$ 的 `merge` 模块 TLE , 考试后花了一定时间优化到 $O(n \log n)$