Assignment #3: 惊蛰 Mock Exam

Updated 1641 GMT+8 Mar 5, 2025

2025 spring, Complied by 叶靖、信管

> **说明：**
>
> 1. **惊蛰月考**：AC6（请改为同学的通过数） 。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

思路：

基础题，理论上直接将题目的要求都导入就可以，但有个坑在小考时没想到的点（'@'与'.'不能直接相连），考试时没考虑到'.'在'@'前面的情况，想通了一切就游刃有余。

代码：

while True: try: email = input().strip()

```
        ans = False
        if email.count('@') == 1 and '.' in email:
            if email[0] not in ('@', '.') and email[-1] not in ('@', '.'):
                if email.index('@') + 1 < email.rindex('.') and email[email.index('@') + 1] !=
                    ans = True

        if ans is True:
            print('YES')
        else:
            print('NO')
    except EOFError:
        break
```

代码运行截图 （至少包含有"Accepted"）

 03-01

M02039: 反反复复

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

思路： 基础题+1。将字符串切片，单数行由行尾反向取值，随着行数增加而截取字母。

代码：

char_num = int(input()) char_input = input() char_line = len(char_input) // char_num

char_ans = [] for i in range(char_num): for j in range(char_line): if j % 2 == 0: char = char_input[i + j * char_num] char_ans.append(char) else: char = char_input[(j + 1) * char_num - i - 1] char_ans.append(char) print(''.join(list(map(str, char_ans))))

代码运行截图 （至少包含有"Accepted"）

 03-02

M02092: Grandpa is Famous

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

思路：

将数目全部集中并计算其中的数量，最后找出第二多的数目，应用了哈希表的概念。此题最大的难题应该是在把控时间复杂度方面，尽可能确保不超时。

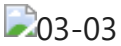
代码：

while True: N, M = list(map(int, input().split())) if N == 0 and M == 0: break

```
rankings = []
for _ in range(N):
    ranks = list(map(int, input().split()))
    rankings.extend(ranks)
count = {}
for num in rankings:
    if num in count:
        count[num] += 1
    else:
        count[num] = 1
counts = sorted(set(count.values()), reverse=True)
if len(counts) > 1:
    sec_max = counts[1]
else:
    sec_max = counts[0]

result = [num for num, times in count.items() if times == sec_max]
print(' '.join(map(str, sorted(result))))
```

代码运行截图 （至少包含有"Accepted"）



M04133: 垃圾炸弹

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

思路：

由1025 * 1025组成的网格，要找到最合适的炸弹投放点，使得被清楚的垃圾最大化，从而减少所需的炸弹投放数。 利用二维前缀和来优化计算炸弹影响范围的垃圾总数的过程，减少耗时。

代码：

def max_bomb(d, n, trash_location): grid = [[0] * 1025 for _ in range(1025)] for x, y, i in trash_location: grid[x][y] = i prefix_sum = [[0] * 1025 for _ in range(1025)] for i in range(1025): for j in range(1025): prefix_sum[i][j] = grid[i][j] + (prefix_sum[i - 1][j] if i > 0 else 0) + (prefix_sum[i][j - 1] if j > 0 else 0) - (prefix_sum[i - 1][j - 1] if i > 0 and j > 0 else 0)

```
max_num_bomb = 0
max_trash_location = 0
for x in range(1025):
    for y in range(1025):
        x1, y1 = max(0, x - d), max(0, y - d)
        x2, y2 = min(1024, x + d), min(1024, y + d)
        trash_cleared = prefix_sum[x2][y2]
        if x1 > 0:
            trash_cleared -= prefix_sum[x1 - 1][y2]
        if y1 > 0:
```

```
                trash_cleared -= prefix_sum[x2][y1 - 1]
            if x1 > 0 and y1 > 0:
                trash_cleared += prefix_sum[x1 - 1][y1 - 1]
            if trash_cleared > max_num_bomb:
                max_num_bomb = trash_cleared
                max_trash_location = 1
            elif trash_cleared == max_num_bomb:
                max_trash_location += 1
    return max_trash_location, max_num_bomb
```

d = int(input()) n = int(input()) trash_location = [] for _ in range(n): x, y, i = map(int, input().split()) trash_location.append((x, y, i)) max_locations, max_trash = max_bomb(d, n, trash_location) print(max_locations, max_trash)

代码运行截图 （至少包含有"Accepted"）

 03-04

T02488: A Knight's Journey

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

思路：

这题个人感觉很难。若真出现在期考，我认为放弃是最佳选择。解题过程中虽然使用了大模型，还是解不出来，最后还得靠朋友的指导才能勉强解题。 解题思路应该是找出一条哈密顿路径，骑士可以访问每个格子恰好一次，且按照字典序最小的顺序输出路径，过程中也涉及了回溯和字典序。

代码：

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

def dfs(x, y, path, step, visited, p, q, flag): if flag: return flag if step == p * q: flag = True print(''.join(path)) return flag for dx, dy in move: nx, ny = x + dx, y + dy if 0 <= nx < p and 0 <= ny < q and not visited[nx][ny]: visited[nx][ny] = True path.append(chr(ny + ord('A')) + str(nx + 1)) flag = dfs(nx, ny, path, step + 1, visited, p, q, flag) visited[nx][ny] = False path.pop() if flag: return flag return flag

n = int(input()) for m in range(1, n + 1): p, q = map(int, input().split()) visited = [[False] * q for _ in range(p)] flag = False print(f'Scenario #:')

```
    for i in range(p):
        for j in range(q):
            if flag:
                break
            visited[i][j] = True
```

```
            path = [chr(j + ord('A')) + str(i + 1)]
            flag = dfs(i, j, path, 1, visited, p, q, flag)
            visited[i][j] = False
        if flag:
            break
    if not flag:
        print("impossible")
    print()
```

代码运行截图 （至少包含有"Accepted"）

T06648: Sequence

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

思路：

达到题目要求不难，但总会出现内存空间不足的问题。曾尝试不用heapq包来解题，但总是出现内存不足的情况，暴力枚举是完全无法实现的。 因此，需要逐步合并序列，用优先队列加上贪心的概念，最后要归并成n个最小的和，过程中用上了heapq包来完成合并。

代码：

import heapq def merge(a, b): heap = [] n = len(a) for i in range(n): heapq.heappush(heap, (a[i] + b[0], i, 0)) res = [] for _ in range(n): val, i, j = heapq.heappop(heap) res.append(val) if j + 1 < len(b): next_val = a[i] + b[j + 1] heapq.heappush(heap, (next_val, i, j + 1)) return res t = int(input()) for _ in range(t): m, n = map(int, input().split()) nums = [] for __ in range(m): nums.append(sorted(map(int, input().split()))) sums = nums[0].copy() for i in range(1, m): sums = merge(sums, nums[i]) print(' '.join(map(str, sums[:n])))

代码运行截图 （AC代码截图，至少包含有"Accepted"）

2. 学习总结和收获

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。

此次的月考确实让我意识到了自己的不足，而刷题的重要性也深刻地体现出来了。有的题型的解答模板都是有蛮强烈的共同之处，所以还是得多刷题来累积解题经验。 就给自己定个小目标吧，先不说要把自己练到大神级别，就希望在下次的月考或是作业中，能将偏基础和中等级别的题型解出来吧。