Assignment #D: 图 & 散列表

Updated 2042 GMT+8 May 20, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院

> **说明：**
>
> 1. **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。
>
> 2. **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。
>
> 3. **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

M17975: 用二次探查法建立散列表

http://cs101.openjudge.cn/practice/17975/

<mark>需要用这样接收数据。因为输入数据可能分行了，不是题面描述的形式。OJ 上面有的题目是给 C++设计的，细节考虑不周全。</mark>

```python
import sys
input = sys.stdin.read
data = input().split()
index = 0
n = int(data[index])
```

```
index += 1
m = int(data[index])
index += 1
num_list = [int(i) for i in data[index:index+n]]
```

思路：调试了半天都无法 ac。。实在不知道该怎么处理 qwq（使用了上方的代码进行输入也还是不行，求助同学和 ai 也无果。。）

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M01258: Agri-Net

MST, http://cs101.openjudge.cn/practice/01258/

思路：用 prim 算法：先选择一个起始节点（第一个农场），将其加入生成树，并建立一个最小堆用于存储当前生成树到其他节点的边及其权重。
每次从堆中取出权重最小的边，如果连接的节点未被访问，则加入生成树，并将该节点的所有邻接边加入堆；并不断重复直到所有节点都被访问即可！
时间：10mins

代码：

```python

```
```
import sysimport heapq
```

```
def prim_mst(n, adjacency_matrix):
    visited = [False] * n
    min_heap = []
    # 从第一个农场开始
    heapq.heappush(min_heap, (0, 0))
    total_fiber = 0
    while min_heap:
        weight, farm = heapq.heappop(min_heap)
        if visited[farm]:
            continue
        visited[farm] = True
        total_fiber += weight
        for neighbor in range(n):
            if not visited[neighbor] and adjacency_matrix[farm][neighbor] != 0:
                heapq.heappush(min_heap,         (adjacency_matrix[farm][neighbor],
neighbor))
    return total_fiber
def main():
    input = sys.stdin.read().split()
    ptr = 0
    while ptr < len(input):
        n = int(input[ptr])
        ptr += 1
        adjacency_matrix = []
        for _ in range(n):
            row = list(map(int, input[ptr:ptr + n]))
            ptr += n
            adjacency_matrix.append(row)
        print(prim_mst(n, adjacency_matrix))
if __name__ == "__main__":
    main()
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M3552.网络传送门旅游

bfs, https://leetcode.cn/problems/grid-teleportation-traversal/

思路：用 BFS 方法，从起点出发，每次扩展格子并使用 deque 分层推进，每层步数加一，直到到达终点；如果当前格子是一个传送门，就立即传送到所有相同字母的其他位置。其中，传送不增加步数（在当前 BFS 层内完成）
需要注意的是，为避免走重复路径，要标记访问过的格子；这样既节省了内存，又避免了可能会发生的 TLE 或者 MLE~
时间：25mins

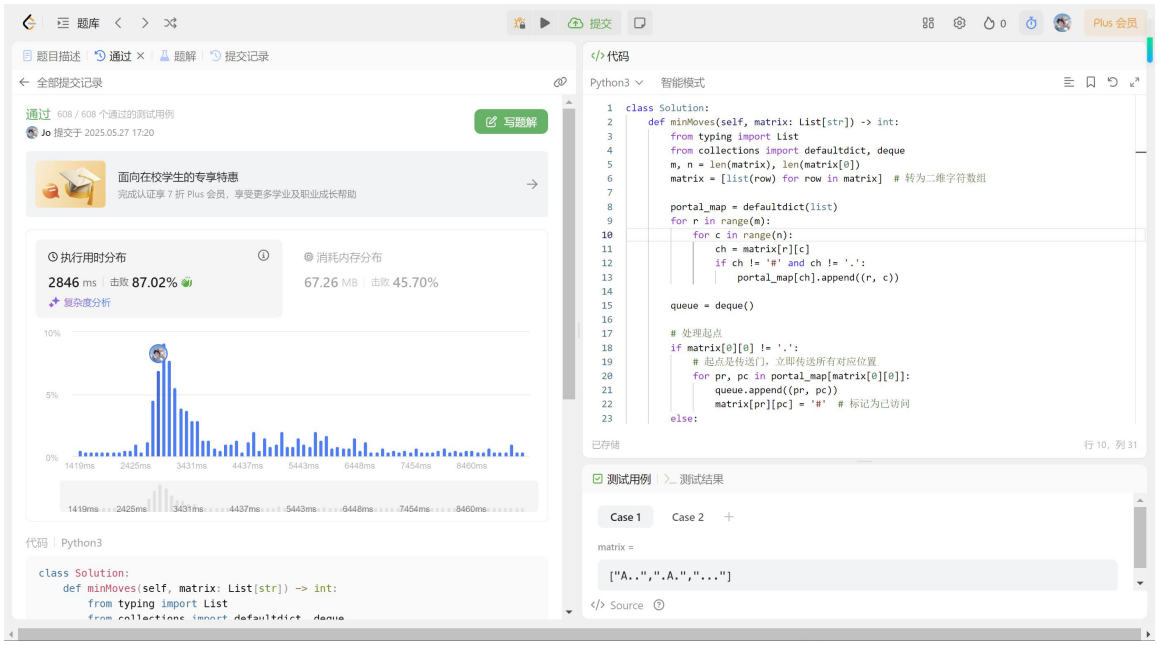
代码：

```python

```

```
class Solution:
    def minMoves(self, matrix: List[str]) -> int:
        from typing import List
        from collections import defaultdict, deque
        m, n = len(matrix), len(matrix[0])
        matrix = [list(row) for row in matrix]  # 转为二维字符数组

        portal_map = defaultdict(list)
```

```
        for r in range(m):
            for c in range(n):
                ch = matrix[r][c]
                if ch != '#' and ch != '.':
                    portal_map[ch].append((r, c))

        queue = deque()

        # 处理起点
        if matrix[0][0] != '.':
            # 起点是传送门，立即传送所有对应位置
            for pr, pc in portal_map[matrix[0][0]]:
                queue.append((pr, pc))
                matrix[pr][pc] = '#'  # 标记为已访问
        else:
            queue.append((0, 0))
            matrix[0][0] = '#'

        steps = 0
        while queue:
            for _ in range(len(queue)):
                r, c = queue.popleft()
                if (r, c) == (m - 1, n - 1):
                    return steps

                # 4 个方向移动
                for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
                    nr, nc = r + dr, c + dc
                    if 0 <= nr < m and 0 <= nc < n and matrix[nr][nc] != '#':
                        if matrix[nr][nc] == '.':
                            queue.append((nr, nc))
                            matrix[nr][nc] = '#'
                        else:
                            # 传送门
                            for pr, pc in portal_map[matrix[nr][nc]]:
                                queue.append((pr, pc))
                                matrix[pr][pc] = '#'

            steps += 1

        return -1
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



M787.K 站中转内最便宜的航班

Bellman Ford, https://leetcode.cn/problems/cheapest-flights-within-k-stops/

思路：用类似 dijkstra 算法的方法来做：使用一个最小堆来优先处理当前花费最低的路径，其中每个节点包含了到达某个城市所用的花费、当前城市、已经中转的次数。由于最多访问 K+1 个城市（包括起点和终点），因此当你到达目标城市并且中转次不大于 K 时，才返回当前累计的价格。最终，如果队列为空都找不到符合条件的路径，则返回-1 即可
时间：25mins

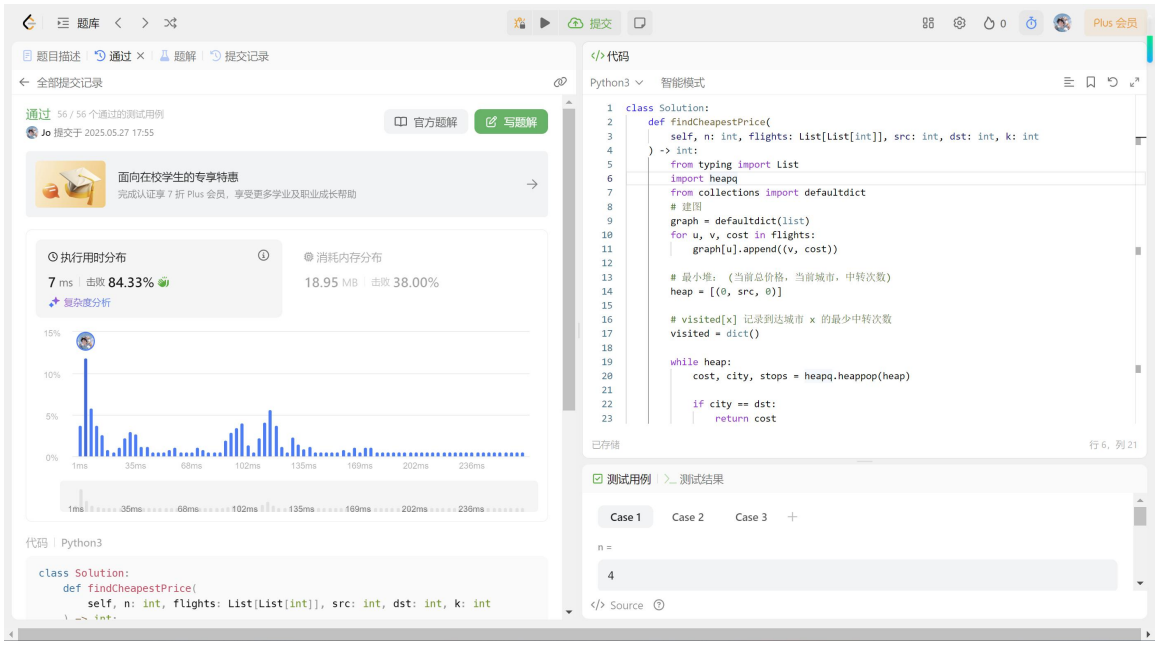
代码：

```python

```

```
class Solution:
    def findCheapestPrice(
        self, n: int, flights: List[List[int]], src: int, dst: int, k: int
    ) -> int:
        from typing import List
        import heapq
        from collections import defaultdict
        # 建图
```

```
        graph = defaultdict(list)
        for u, v, cost in flights:
            graph[u].append((v, cost))

        # 最小堆：（当前总价格，当前城市，中转次数）
        heap = [(0, src, 0)]

        # visited[x] 记录到达城市 x 的最少中转次数
        visited = dict()

        while heap:
            cost, city, stops = heapq.heappop(heap)

            if city == dst:
                return cost

            # 如果中转次数 > k，不再继续搜索
            if stops > k:
                continue

            # 剪枝：如果之前用更少中转次数访问过这个城市，则跳过
            if city in visited and visited[city] <= stops:
                continue
            visited[city] = stops

            for nei, price in graph[city]:
                heapq.heappush(heap, (cost + price, nei, stops + 1))

        return -1
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



M03424: Candies

Dijkstra, http://cs101.openjudge.cn/practice/03424/

思路：如果用普通的 dij 方法中的 heapq 来做这道题，几乎一定会 TLE；反之，如果用 deque，建立 status 列表避免城市重复入队，则可顺利 AC
时间：25mins

代码：

```python

```

```
# 上个代码用 heapq（经典 dij）会超时，现在改用 deque，并建立 status 列表避免重复入队
# queue 改为只记录城市，不断地剪枝优化即可

from collections import deque

n, m = map(int,input().split())
road = {} # 把每个人视为一个节点，每个关系视为一条路，记录以每个人为起点出发的路
for i in range(n):
    road[i + 1] = []
for i in range(m):
    a, b, c = map(int,input().split())
```

```
    road[a].append((c, b)) # 高度排在前面

def max_candies(n, road):
    good = [0] # 以 snoopy 为原点，记录“最矮的”“最高的高度”
    for _ in range(n - 1):
        good.append(float("inf"))

    queue = deque([1])

    status = ["True"] # 用于记录每个点此刻是否在队伍里，避免重复入队
    for _ in range(n - 1):
        status.append("False")

    while queue:
        location = queue.popleft()
        status[location - 1] = "False" # location 标记为出队

        for (s, t) in road[location]:
            if s + good[location - 1] < good[t - 1]: # 如果当前路径是最优路径，更新 good
                good[t - 1] = s + good[location - 1]
                # 此时，根据 t 是否是当下最好的城市（距离最短）来决定把它放在队列的哪个位置
                if status[t - 1] == "False": # 先检查 t 是否已经在队伍里；如果已经在则不要重复添加
                    if queue and good[t - 1] < good[queue[0] - 1]:
                        queue.appendleft(t)
                    else:
                        queue.append(t)
                    status[t - 1] = "True" # 最后，别忘了把状态设为 True

    print(good[-1])

max_candies(n, road)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M22508:最小奖金方案

topological order, http://cs101.openjudge.cn/practice/22508/

思路：

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

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>

这次作业题目难度参差不齐，感觉难度和题目的题号不成正相关；不过大部分题目还是比较模板化的，熟练度上来了之后做起来还是比较快的~
另外，感觉第五题在题干里最好要说明 c 大于等于 0，不然就很难处理。距离考试还剩一周了，正在整理 cheatsheet；继续加油！