21 第7讲 作业记录（一）

笔记本: 浙江大学《数据结构》
创建时间: 2025/4/26 20:09 **更新时间:** 2025/4/26 20:21
作者: panhengye@163.com
URL: https://pintia.cn/problem-sets/1873565885118418944/exam/problems/type/7?...

07-图4 哈利·波特的考试

【提交结果】

提交结果

题目	用户	提交时间
07-图4	飞翔的小师弟	2025/04/26 20:09:25
编译器	内存	用时
Python (python3)	3132 / 65536 KB	184 / 400 ms
状态	分数	评测时间
答案正确	25 / 25	2025/04/26 20:09:25

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0	sample换数字，只有唯一解	3108	15	答案正确	12 / 12
1	无解	2992	14	答案正确	1 / 1
2	最大N的等边长环，解不唯一，输出最小编号	3012	171	答案正确	4 / 4
3	最大N，最大M，随机完全图	3132	184	答案正确	8 / 8

【代码记录】

```
def init_graph(n, m):
    """
    初始化图

    Args:
        n: 节点数
        m: 边数

    Returns:
        graph: 邻接矩阵
    """
    graph = [[float('inf')] * n for _ in range(n)] # 初始化一个 n x n 的距离矩阵
    for i in range(n):
        graph[i][i] = 0  # 节点到自身的距离为0
    for _ in range(m):
        u, v, w = map(int, input().split())
        # 处理输入值和邻接表中对应的值有偏移的情况
        u = u - 1
        v = v - 1
```

```
        graph[u][v] = w # 无向图，所以需要设置两个方向的边
        graph[v][u] = w
    return graph  # 返回初始化好的图


def find_min_max_distance(graph, n):
    """
    寻找最大距离的最小节点

    Args:
        graph: 邻接矩阵
        n: 节点数
    """
    # Floyd算法求解任意两点之间的最短路径
    for k in range(n):
        for i in range(n):
            for j in range(n):
                if graph[i][j] > graph[i][k] + graph[k][j]:
                    graph[i][j] = graph[i][k] + graph[k][j]

    # 寻找最大距离的最小节点
    min_max_distance = float('inf')
    min_node = -1
    for i in range(n):
        max_dist = 0
        possible = True
        for j in range(n):
            if i!=j and graph[i][j] == float('inf'):
                possible = False
                break
            if graph[i][j] > max_dist:
                max_dist = graph[i][j]
        if possible and max_dist < min_max_distance:
            min_max_distance = max_dist
            min_node = i + 1 # 节点编号从1开始
        elif possible and max_dist == min_max_distance:
            min_node = min(min_node, i + 1)

    return min_node, min_max_distance


def main():

    # 处理输入数据
    n, m = map(int, input().split())
    graph = init_graph(n, m)

    # 寻找最大距离的最小节点
    min_node, min_max_distance = find_min_max_distance(graph, n)

    # 输出结果
    if min_node == -1:
        print("0")
    else:
        print(min_node, min_max_distance)


if __name__ == "__main__":
    main()
```

【整体思路】

这段代码解决的是"哈利·波特的考试"问题，主要是一个图论中的最短路径问题。核心思想是：

1. 图的构建

- 使用邻接矩阵表示一个无向带权图
- 初始化所有距离为无穷大
- 设置自环（节点到自身）距离为0

- 读取边信息并设置对应的权重

2. 最短路径计算

- 使用Floyd-Warshall算法计算任意两点间的最短路径
- 该算法通过三重循环，考虑所有可能的中间节点，来优化任意两点间的距离

3. 最优集合点查找

- 对每个节点，计算它到其他所有节点的最大距离
- 如果有节点不可达（距离为无穷大），则该节点不能选为集合点
- 选择"最大距离"最小的节点作为最优集合点
- 如果有多个节点满足条件，选择编号最小的

4. 结果输出

- 如果找不到有效的集合点，输出"0"
- 否则输出最优节点编号和对应的最大距离

这个算法的时间复杂度主要由Floyd-Warshall算法决定，为$O(n^3)$，其中n是节点数。

【关键点解析1】

```
min_max_distance = float('inf')
    min_node = -1
    for i in range(n):
        max_dist = 0
        possible = True
        for j in range(n):
            if i!=j and graph[i][j] == float('inf'):
                possible = False
                break
            if graph[i][j] > max_dist:
                max_dist = graph[i][j]
        if possible and max_dist < min_max_distance:
            min_max_distance = max_dist
            min_node = i + 1 # 节点编号从1开始
        elif possible and max_dist == min_max_distance:
            min_node = min(min_node, i + 1)
```

1. 对每个节点i，计算它到其他所有节点的最大距离（max_dist）
2. 如果节点i到任何其他节点不可达（距离为无穷大），则标记该节点不可选（possible = False）
3. 在所有可达的节点中，选择"最大距离"最小的节点作为最优集合点
4. 如果有多个节点的"最大距离"相同且最小，则选择编号最小的节点