Assignment #D: 图 & 散列表

Updated 2042 GMT+8 May 20, 2025

2025 spring, Complied by 叶靖、信管

> **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/

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。

```
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：

代码：

```
def quadratic_probe_insert(keys, M):
    table = [None] * M
    result = []

    for key in keys:
        pos = key % M
        if table[pos] is None or table[pos] == key:
            table[pos] = key
            result.append(pos)
            continue
        # 否则开始二次探查
        i = 1
        instered = False
        while not instered:
            for sign in [1, -1]:
                new_pos = (pos + sign * (i ** 2)) % M
                if table[new_pos] is None or table[new_pos] == key:
                    table[new_pos] = key
                    result.append(new_pos)
                    instered = True
                    break
            i += 1  # 探查次数增加
    return result

import sys
input = sys.stdin.read
data = input().split()
N = int(data[0])
M = int(data[1])
keys = list(map(int, data[2:2 + N]))

positions = quadratic_probe_insert(keys, M)
print(*positions)
```

代码运行截图 （至少包含有"Accepted"）

M01258: Agri-Net

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

思路：

代码：

```
def prim_mst(matrix):
    n = len(matrix)
```

```
        visited = [False] * n
        min_edge = [float('inf')] * n
        min_edge[0] = 0
        total_cost = 0

        for _ in range(n):
            u = -1
            for i in range(n):
                if not visited[i] and (u == -1 or min_edge[i] < min_edge[u]):
                    u = i
            visited[u] = True
            total_cost += min_edge[u]
            for v in range(n):
                if not visited[v] and matrix[u][v] < min_edge[v]:
                    min_edge[v] = matrix[u][v]
        return total_cost

    def read_case(input_lines):
        n = int(input_lines[0])
        numbers = []
        i = 1
        while len(numbers) < n * n:
            numbers.extend(map(int, input_lines[i].split()))
            i += 1
        matrix = [numbers[j * n:(j + 1) * n] for j in range(n)]
        return matrix, input_lines[i:]

    def solve_all_cases(lines):
        results = []
        while lines:
            matrix, lines = read_case(lines)
            results.append(str(prim_mst(matrix)))
        return '\n'.join(results)

    # 主函数：循环读取直到EOF（或遇到空行终止）
    def main():
        lines = []
        try:
            while True:
                line = input()
                if line.strip() == '':
                    break
                lines.append(line)
        except EOFError:
            pass
        print(solve_all_cases(lines))

    main()
```

代码运行截图 （至少包含有"Accepted"）

M3552.网络传送门旅游

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

思路：

代码：

```
from collections import deque, defaultdict

class Solution:
    def minMoves(self, matrix):
        if matrix[-1][-1] == '#':
            return -1

        m, n = len(matrix), len(matrix[0])
        pos = defaultdict(list)
        for i, row in enumerate(matrix):
            for j, c in enumerate(row):
                if c.isupper():
                    pos[c].append((i, j))

        DIRS = [(0, -1), (0, 1), (-1, 0), (1, 0)]
        dis = [[float('inf')] * n for _ in range(m)]
        dis[0][0] = 0
        q = deque([(0, 0)])

        while q:
            x, y = q.popleft()
            d = dis[x][y]

            if x == m - 1 and y == n - 1:
                return d

            c = matrix[x][y]
            if c.isupper() and c in pos:
                for px, py in pos[c]:
                    if d < dis[px][py]:
                        dis[px][py] = d
                        q.appendleft((px, py))
                del pos[c]

            for dx, dy in DIRS:
                nx, ny = x + dx, y + dy
                if 0 <= nx < m and 0 <= ny < n and matrix[nx][ny] != '#' and d + 1 < dis[n
                    dis[nx][ny] = d + 1
                    q.append((nx, ny))
```

```
        return -1
```

代码运行截图 （至少包含有"Accepted"）

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：

代码：

```
class Solution(object):
    def findCheapestPrice(self, n, flights, src, dst, k):
        INF = float('inf')
        dist = [INF] * n
        dist[src] = 0

        # 进行 k+1 次松弛
        for _ in range(k + 1):
            temp = dist[:]  # 复制上一轮的距离
            for u, v, w in flights:
                if dist[u] == INF:
                    continue
                if dist[u] + w < temp[v]:
                    temp[v] = dist[u] + w
            dist = temp

        return dist[dst] if dist[dst] != INF else -1
```

代码运行截图 （至少包含有"Accepted"）

M03424: Candies

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

思路：

代码：

```
import sys
import threading
import heapq
```

```
def main():
    input = sys.stdin.readline
    N, M = map(int, input().split())
    graph = [[] for _ in range(N+1)]
    for _ in range(M):
        A, B, c = map(int, input().split())
        graph[A].append((B, c))
    INF = 10**30
    dist = [INF] * (N+1)
    dist[1] = 0
    pq = [(0, 1)]  # (当前距离, 节点)
    while pq:
        d, u = heapq.heappop(pq)
        if d > dist[u]:
            continue
        if u == N:
            break    # 提前退出
        for v, w in graph[u]:
            nd = d + w
            if nd < dist[v]:
                dist[v] = nd
                heapq.heappush(pq, (nd, v))
    # 输出从 1 到 N 的最短路距离，即为最大可实现的 x_N - x_1
    print(dist[N])

if __name__ == "__main__":
    threading.Thread(target=main).start()
```

代码运行截图 （至少包含有"Accepted"）

M22508:最小奖金方案

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

思路：

代码：

```
from collections import deque

def min_total_prize(n, m, battles):
    graph = [[] for _ in range(n)]
    indeg = [0] * n

    for a, b in battles:
        graph[b].append(a)  # b被a打败，b -> a（因为a的奖金必须比b高）
        indeg[a] += 1
```

```
    # dp[i]: 从i出发最长路径长度（奖金差的增量）
    dp = [0] * n

    q = deque()
    # 入度为0的点，奖金差为0
    for i in range(n):
        if indeg[i] == 0:
            q.append(i)

    while q:
        u = q.popleft()
        for v in graph[u]:
            if dp[v] < dp[u] + 1:
                dp[v] = dp[u] + 1
            indeg[v] -= 1
            if indeg[v] == 0:
                q.append(v)

    # 每个队伍奖金 = 100 + dp[i]
    total = sum(100 + x for x in dp)
    return total

if __name__ == "__main__":
    n, m = map(int, input().split())
    battles = [tuple(map(int, input().split())) for _ in range(m)]
    print(min_total_prize(n, m, battles))
```

代码运行截图 （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。

这次的编程题目涵盖了图论最短路径、散列表冲突处理、动态规划等多个算法知识点，难度非常大。 每一道题都不仅考察了基本的数据结构掌握情况，还需要灵活的思维和对边界条件的精准把握,尤其是在处理带限制条件的最短路径和二次探查哈希插入时一直会出现超时或错误。 在考试中如果遇到这些题目，我真的只能投降认输。这次作业每一道对我来说都是地狱难度级别。😔😔😔