**## 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>

思路：先初始化哈希列表，初始为0.5表示空，然后对每个key计算其尝试插入的下标，如果为空/已有相同的key就直接插入，接下来进行冲突处理（位置已被占用），用sign控制方向（1：+，-1：-；a：控制偏移步数（平方数），只在sign恢复为正时加一），用平方探测法测试其他的空位，如果为空就继续加入，最后返回最终的插入位置。

代码：

**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]]

**def** **insert**(num\_list,m):

t=[0.5]\*m

pos=[]

**for** key **in** num\_list:

h=key%m

cur=t[h]

**if** cur == 0.5 **or** cur == key:

t[h]=key

pos.**append**(h)

**else**:

sign=1

a=1

**while** True:

np=(h+sign\*a\*a)%m

**if** t[np]==0.5 **or** t[np] == key:

t[np]=key

pos.**append**(np)

**break**

sign\*=-1

**if** sign==1:

a+=1

**return** pos

res=**insert**(num\_list,m)

**print**(\*res)

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mst最小生成树，先遍历所有未包含在生成树中的节点，找到min\_edge值最小的节点u并将其加入生成树，然后将节点u标记为已包含在生成树中与其边权值加入total中。然后继续更新所有未包含在生成树中的节点到生成树的最小边权值，对于每个未包含在生成树中的节点v，如果通过新加入的节点u到v的边权值小于当前记录的min\_edge[v]，则更新min\_edge[v]。最后输出总边权值。

代码：

**while** True:

**try**:

N=**int**(**input**())

**except**:

**break**

vals=[]

**for** \_ **in** **range**(N):

vals+=**input**().**split**()

vals=**list**(**map**(int,vals))

dist=[vals[i\*N:(i+1)\*N] **for** i **in** **range**(N)] *# 一维列表vals转换为二维邻接矩阵dist，dist[i][j]：第i个农场到第j个农场的光纤长度*

in\_mst=[False]\*N *# 标记每个节点是否已包含在最小生成树中。初始时，所有节点都未包含。*

inf=10\*\*9 *# 初始的最小边权值*

min\_edge=[inf]\*N *# 记录每个节点到当前生成树的最小边权值。初始时，所有值设为inf，表示未知。*

min\_edge[0]=0 *# 节点0作为起始点，其到生成树的边权值设为0。*

total=0 *# 累计最小生成树的总边权值。*

**for** \_ **in** **range**(N):

u=-1

**for** i **in** **range**(N):

**if** **not** in\_mst[i] **and** (u == -1 **or** min\_edge[i]<min\_edge[u]):

u=i

in\_mst[u]=True

total+=min\_edge[u]

**for** v **in** **range**(N):

**if** **not** in\_mst[v] **and** dist[u][v]<min\_edge[v]:

min\_edge[v]=dist[u][v]

**print**(total)

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：先处理可以进行瞬移操作的大写字符（用字典记录大写字符位置），初始化dis用于保存起点到任意位置的最小步数，然后从首节点开始进行移动，若当前位置是大写字母，则遍历所有大写字母的坐标，若用瞬移能更新更短距离则更新位置并放入队头，瞬移结束后删除大写字母避免重复；若当前位置可以普通移动则往四个方向尝试，若移动合法且不是障碍且通过这一步能用d+1更新更短距离便将其放入队尾（双端队列），最后到达终点时返回d。

代码：

class Solution:

    def minMoves(self, mat: List[str]) -> int:

        if mat[-1][-1]=='#':

            return -1

        m,n=len(mat),len(mat[0])

        visited={}

        for i,row in enumerate(mat):

            for j,col in enumerate(row):

                if col.isupper():

                    if col not in visited:

                        visited[col]=[]

                    visited[col].append((i,j))

        direct=[(0,-1),(0,1),(-1,0),(1,0)]

        dis=[[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=mat[x][y]

            if c in visited:

                for px,py in visited[c]:

                    if d<dis[px][py]:

                        dis[px][py]=d

                        q.appendleft((px,py))

                del visited[c]

            for dx,dy in direct:

                nx,ny=x+dx,y+dy

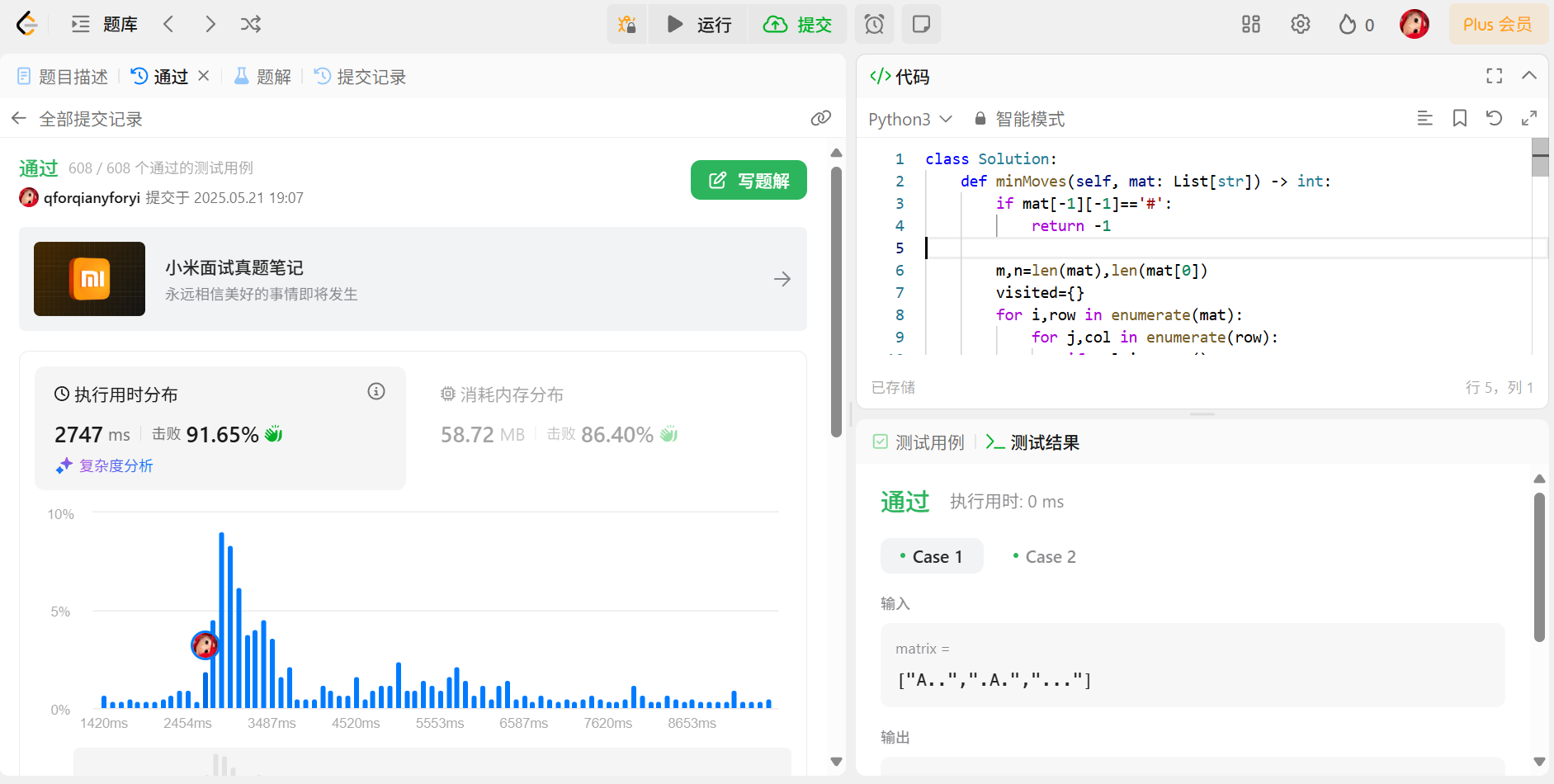
                if 0<=nx<m and 0<=ny<n and mat[nx][ny] != '#' and d+1<dis[nx][ny]:

                    dis[nx][ny]=d+1

                    q.append((nx,ny))

        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：需要用defaultdict构建邻接表：key=起点s，值是一个字典，记录从s到各个终点e的cost。然后将prev和curr初始化为默认值为无穷大的字典：表示“当前已知最短花费”，原点src的距离设为0。然后进行循环，对每条航班（s，e，cost）进行更新（若经由s再到e的花费小于之前记录的curr[e]则更新），更新后将curr复制给prev。最后若curr[dst]为无穷大说明路径不存在（-1），否则返回其花费。

代码：

class Solution:

    def findCheapestPrice(self, n: int, flights: List[List[int]], src: int, dst: int, k: int) -> int:

        from collections import defaultdict

        graph=collections.defaultdict(dict)

        for s,e,cost in flights:

            graph[s][e]=cost

        prev=collections.defaultdict(lambda:inf)

        curr=collections.defaultdict(lambda:inf)

        prev[src] = 0

        for i in range(k+1):

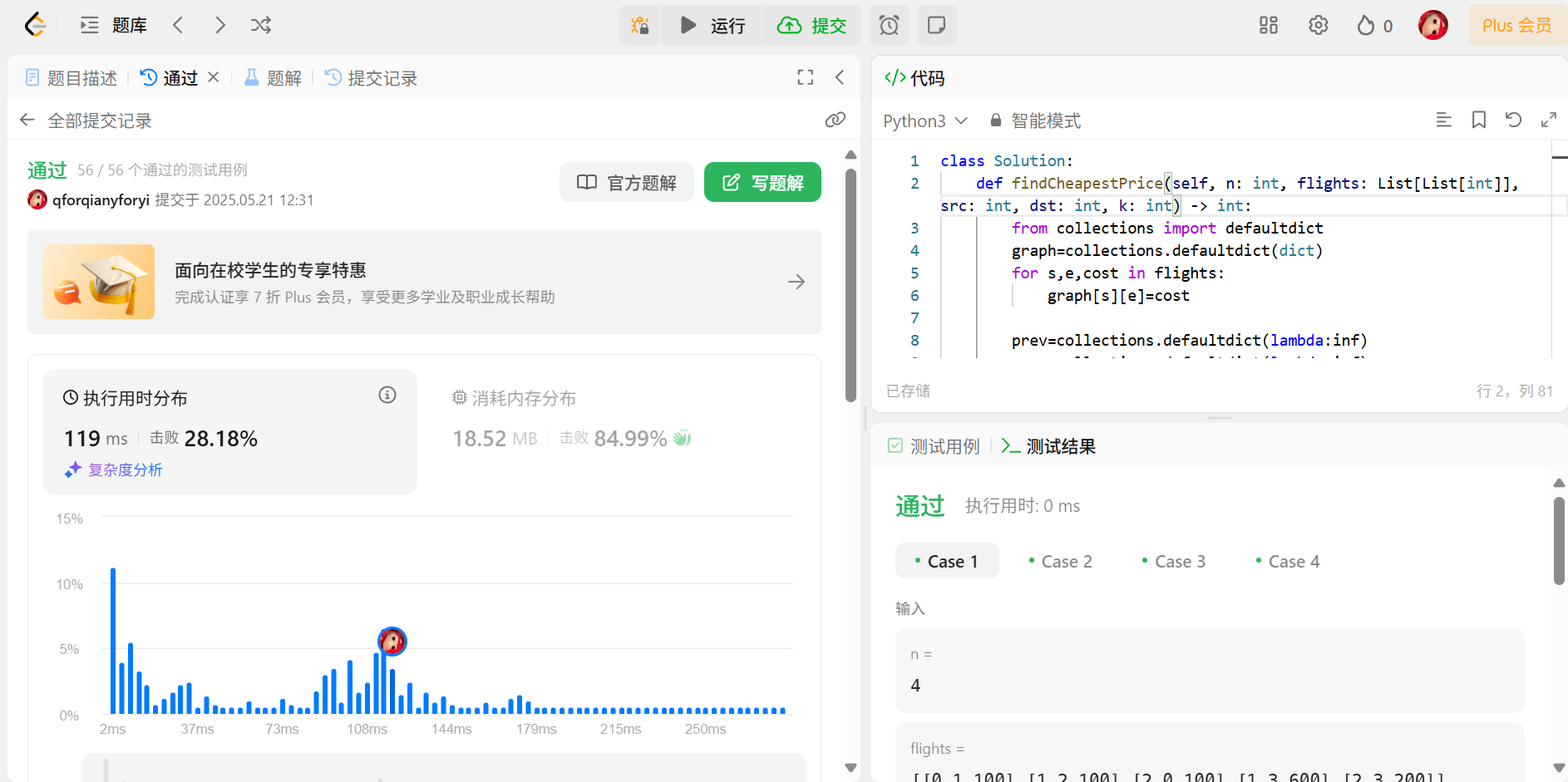
            for s,e,cost in flights:

                curr[e]=min(curr[e],prev[s]+cost)

            prev=curr.copy()

        return curr[dst] if curr[dst] != inf else -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：首先建图，然后进行dijikstra，先用candy出粗吗从点1到其他点的最短距离，heap用于挑选当前最短的点（当前距离，当前节点），然后进行循环，每次从堆中取出当前距离最小的节点，若d > candy[u]：非最短则跳过，然后遍历u的所有边，比较到达v的新路径长度，如果比之前的短则更新并推入堆中继续遍历直到循环结束返回结果。

代码：

**import** heapq

n,m=**map**(int,**input**().**split**())

g=[[] **for** \_ **in** **range**(n+1)]

**for** \_ **in** **range**(m):

a,b,c=**map**(int,**input**().**split**())

g[a].**append**((b,c))

**def** **dijikstra**(n,m):

candy=[**float**("inf")]\*(n+1)

candy[1]=0

*# q=[False]\*(n+1)*

heap=[(0,1)]

**while** heap:

d,u=heapq.**heappop**(heap)

**if** d > candy[u]:

**continue**

**for** v,w **in** g[u]:

nd=candy[u]+w

**if** nd<candy[v]:

candy[v]=nd

heapq.**heappush**(heap,(candy[v],v))

**return** candy[n]

**print**(**dijikstra**(n,m))

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：先建图+计算入度，然后用拓扑排序，先将所有入度为0的节点入队，逐个出队并移除其入度，dp[v]=max(dp[v],dp[u]+1)用于求最长路径，入度降为0的节点再入队，直至处理完所有节点，最后计算总奖金（每队基础100+额外层数score（最长路径，score[i]：i比它的最深胜者链低了多少层））

代码：

**from** collections **import** deque

n,m=**map**(int,**input**().**split**())

edges=[]

**for** \_ **in** **range**(m):

a,b=**map**(int,**input**().**split**())

edges.**append**((a,b))

g=[[] **for** \_ **in** **range**(n)]

indeg=[0]\*n

**for** a,b **in** edges:

g[b].**append**(a)

indeg[a] += 1

q=**deque**(i **for** i **in** **range**(n) **if** indeg[i] == 0)

dp=[0]\*n

**while** q:

u=q.**popleft**()

**for** v **in** g[u]:

dp[v]=**max**(dp[v],dp[u]+1)

indeg[v] -= 1

**if** indeg[v] == 0:

q.**append**(v)

**print**(n\*100+**sum**(dp))

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>

这次都好难，尤其是第五题…… 英文题字好多好难看懂（，全部基本都看了题解，感觉好挫败哈哈哈哈，第一题从接收输入就给我搞懵了，但是稍微理解一下感觉还可以理解但是不知道为什么debug了很久，第二题又看不懂题目（好多英文）了，让 AI 给我翻译的，（但是还是看不懂，就直接照着答案来看题目想要干嘛了）还因为没注意要处理多组数据 WA 了好多次… 第三题其实感觉还可以，只不过写起来总会忽略一些小细节，第四题因为没学过 defaultdict 所以本来打算不用的，结果做不出来，就看题解现学了（，第五题难平，我感觉靠我自己应该一辈子做不出来 TvT，相比之下第六题就还行。感觉这次好失败，很多题目都看不太懂，思路也都很模糊 TvT 有点被打击到了