Assignment #D: 图 & 散列表

Updated 2042 GMT+8 May 20, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

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>

```
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：

好坑，在插入元素时成功的条件不止是 **当前位置无元素**，还有可能是**当前位置已有与待插入元素相同的元素**，也就是已经插入过了，这个点一直导致 wa。

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
s = [int(i) for i in data[index:index+n]]
# n,m=map(int,input().split())
# s=list(map(int,input().split()))
hash_table = [-1] * m
idx = []
for i in s:
    base = 0
    while True:
        pos = (i % m + base ** 2) % m
        if hash_table[pos] in (-1, i):
            hash_table[pos] = i
            idx.append(pos)
            break
        pos = (i % m - base ** 2) % m
        if hash_table[pos] in (-1, i):
            hash_table[pos] = i
            idx.append(pos)
            break
        base += 1
print(*idx)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49221498提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys
input = sys.stdin.read
data = input().split()
index = 0
n = int(data[index])
index += 1
m = int(data[index])
index += 1
s = [int(i) for i in data[index:index+n]]
# n,m=map(int,input().split())
# s=list(map(int,input().split()))
hash_table = [-1] * m
idx = []
for i in s:
    base = 0
    while True:
        pos = (i % m + base ** 2) % m
        if hash_table[pos] in (-1, i):
            hash_table[pos] = i
            idx.append(pos)
            break
        pos = (i % m - base ** 2) % m
        if hash_table[pos] in (-1, i):
            hash_table[pos] = i
            idx.append(pos)
            break
        base += 1
print(*idx)
```

基本信息

#: 49221498
题目: 17975
提交人: 24n2400016635
内存: 3636kB
时间: 20ms
语言: Python3
提交时间: 2025-05-21 13:11:54

English 帮助 关于

M01258: Agri-Net

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

思路:

prim 算法，从点的角度出发，维护一个 heap，每次取出一个节点，并把这个节点相邻的所有节点和路径保存到 heap 中，有点相似与 dijkstra 的写法。这样贪心地每次取出一个到下一个节点的最短路加入到答案中。

代码:

```
from heapq import heappush,heappop

while True:
    try:
        n = int(input())
        g = []
        for _ in range(n):
            g.append(list(map(int, input().split())))
        visited = [False] * n
        heap = []
        heappush(heap, (0, 0)) # (weight, node)
        ans = 0
        while heap:
            d, u = heappop(heap)
            if visited[u]:
```

```
                continue
            visited[u] = True
            ans += d
            for v in range(n):
                if not visited[v]:
                    heappush(heap, (g[u][v], v))
        print(ans)
    except EOFError:
        break
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49215719提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from heapq import heappush,heappop

while True:
    try:
        n = int(input())
        g = []
        for _ in range(n):
            g.append(list(map(int, input().split())))
        visited = [False] * n
        heap = []
        heappush(heap, (0, 0)) # (weight, node)
        ans = 0
        while heap:
            d, u = heappop(heap)
            if visited[u]:
                continue
            visited[u] = True
            ans += d
            for v in range(n):
                if not visited[v]:
                    heappush(heap, (g[u][v], v))
        print(ans)
    except EOFError:
        break
```

基本信息

#: 49215719
题目: 01258
提交人: 24n2400016635
内存: 4276kB
时间: 38ms
语言: Python3
提交时间: 2025-05-20 16:20:16

English 帮助 关于

M3552.网络传送门旅游

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

思路：

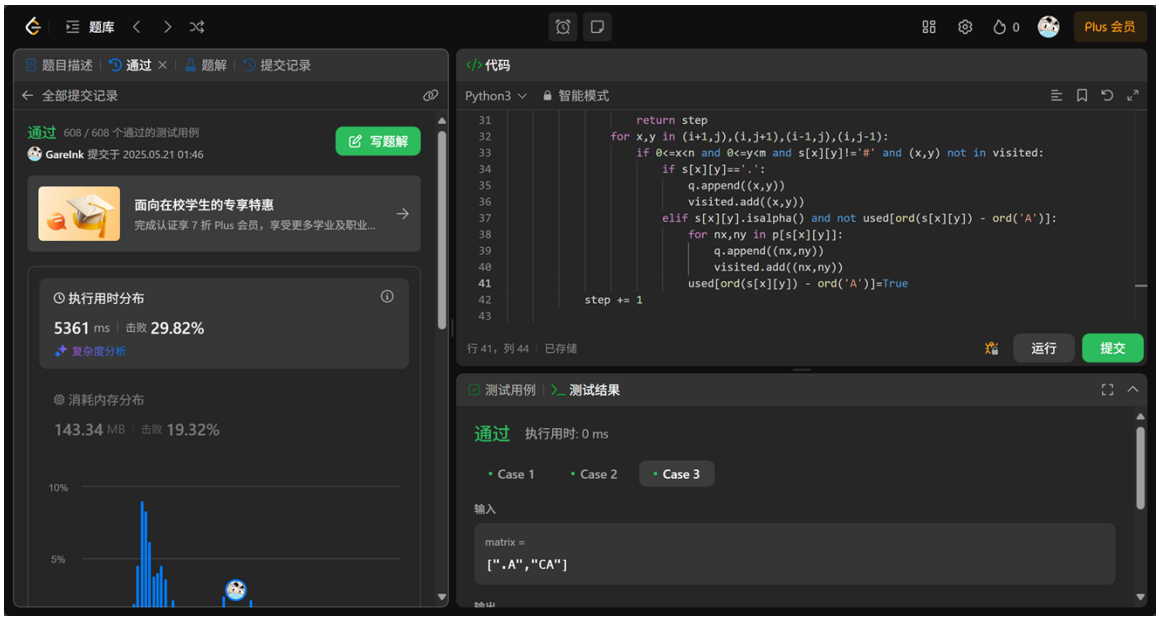
正常的 bfs 之外加入了传送门，一开始先把所有传送门的坐标放在一起，用 defaultdict 存；再用一个长为 26 的 used 布尔数组来判断当前字母传送门是否使用过。之后的 bfs 过程中加入了畸形的判断条件：考虑到 0，0 位置可能就是字母，在弹出元素之前先对即将弹出的坐标进行判断，如果是字母提前把所有字母加入 deque 中，方便进行 for 循环；在遍历接下来四个位置的时候，同样地如果遇到字母了也进行这样的操作。最后返回的是 step。感觉我的写法很奇怪，题解用的是 dist 最短路的二维数组，就没有这种麻烦了。我是想在每次 for 循环之后加一次 step 所以才搞这么麻烦的。

代码：

```
class Solution:
    def minMoves(self, s: List[str]) -> int:
        if s[-1][-1]=='#': return -1
```

```
        n,m=len(s),len(s[0])
        if n==m==1 and s[0][0]=='.': return 0

        used = [True] * 26 #alpha
        p = defaultdict(list) # A : [(i,j),(x,y)]
        for i in range(n):
            for j in range(m):
                if s[i][j].isalpha():
                    p[s[i][j]].append((i,j))
                    used[ord(s[i][j]) - ord('A')] = False

        step = 0
        i, j = 0, 0
        visited = set() #(x,y)
        visited.add((i,j))
        q = deque()
        q.append((i,j))

        while q:
            i,j=q[0]
            if s[i][j].isalpha() and not used[ord(s[i][j]) - ord('A')]:
                for nx,ny in p[s[i][j]]:
                    q.append((nx,ny))
                    visited.add((nx,ny))
                used[ord(s[i][j]) - ord('A')]=True
            for _ in range(len(q)):
                i,j = q.popleft()
                if (i,j)==(n-1,m-1):
                    return step
                for x,y in (i+1,j),(i,j+1),(i-1,j),(i,j-1):
                    if 0<=x<n and 0<=y<m and s[x][y]!='#' and (x,y) not in
visited:
                        if s[x][y]=='.':
                            q.append((x,y))
                            visited.add((x,y))
                        elif s[x][y].isalpha() and not used[ord(s[x][y]) -
ord('A')]:
                            for nx,ny in p[s[x][y]]:
                                q.append((nx,ny))
                                visited.add((nx,ny))
                            used[ord(s[x][y]) - ord('A')]=True
            step += 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/

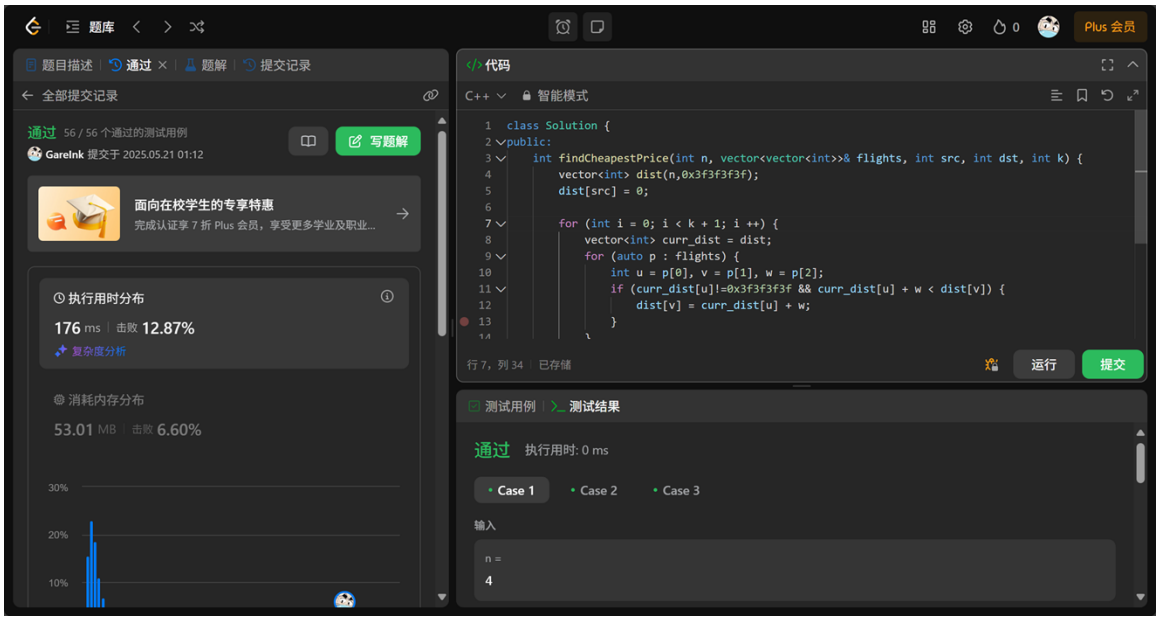
思路：

最多 k 站，就是最多 k+1 条边，用 bellman 松弛 k+1 次看看最后的终点的 dist 值是不是无穷大即可。

代码：

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

        for _ in range(k + 1):
            curr_dist = dist[:]
            for u, v, w in flights:
                if curr_dist[u]!=float('inf') and curr_dist[u] + w < dist[v]:
                    dist[v] = curr_dist[u] + w

        return dist[dst] if dist[dst]!=float('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：

看 tag 的 dijkstra 就 ac 了，但是自己读题目感觉没读懂到底要干嘛，读懂了之后又想不懂为啥要用最短路做，看了题解里面说的差分约束系统的解释后豁然开朗，相当于取两个节点之间路径的所有上界的最小值，那就是最短路了。好神奇啊

代码：

```
from collections import defaultdict,deque
from heapq import heappush,heappop

n, m = map(int, input().split())
g = defaultdict(list)
for _ in range(m):
    u, v, w = map(int, input().split())
    g[u].append((v,w))

heap = []
heappush(heap, (0,1))
dist = [float('inf')] * (n + 1)
dist[1] = 0
while True:
    d, u = heappop(heap)
    if u == n:
        print(d)
        break
    if d > dist[u]:
        continue

    for v, w in g[u]:
        if dist[u] + w < dist[v]:
```

```
            dist[v] = dist[u] + w
            heappush(heap, (dist[v], v))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49219931提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import defaultdict,deque
from heapq import heappush,heappop

n, m = map(int, input().split())
g = defaultdict(list)
for _ in range(m):
    u, v, w = map(int, input().split())
    g[u].append((v,w))

heap = []
heappush(heap, (0,1))
dist = [float('inf')] * (n + 1)
dist[1] = 0
while True:
    d, u = heappop(heap)
    if u == n:
        print(d)
        break
    if d > dist[u]:
        continue

    for v, w in g[u]:
        if dist[u] + w < dist[v]:
            dist[v] = dist[u] + w
            heappush(heap, (dist[v], v))
```

基本信息

#: 49219931
题目: 03424
提交人: 24n2400016635
内存: 26532kB
时间: 381ms
语言: Python3
提交时间: 2025-05-21 00:41:13

English 帮助 关于

M22508:最小奖金方案

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

思路：

这里有 dp 的意思。为了保证奖金最小，肯定是越往上一层，越+1，最底层是兜底的 100。所以建图的时候把有向图反过来，全部反向，再进行拓扑排序，不断寻找入度为 0 的节点加入 deque 中。同时，对于每一个遍历到的 neighbor 节点，都用 `ans[nei] = max(ans[nei], ans[node] + 1)`来不断更新，保证奖金的确是在递增的。

代码：

```
from collections import deque,defaultdict
n,m=map(int,input().split())
g=defaultdict(list)
indeg=[0]*n
for _ in range(m):
    u,v = map(int,input().split())
    g[v].append(u)
    indeg[u]+=1
ans=[100]*n

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

while q:
```

```
    for _ in range(len(q)):
        node = q.popleft()

        for nei in g[node]:
            indeg[nei]-=1
            if indeg[nei]==0:
                q.append(nei)
            ans[nei] = max(ans[nei],ans[node]+1)

print(sum(ans))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49166024提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque,defaultdict
n,m=map(int,input().split())
g=defaultdict(list)
indeg=[0]*n
for _ in range(m):
    u,v = map(int,input().split())
    g[v].append(u)
    indeg[u]+=1
ans=[100]*n

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

while q:
    for _ in range(len(q)):
        node = q.popleft()
        # print(node)
        for nei in g[node]:
            indeg[nei]-=1
            if indeg[nei]==0:
                q.append(nei)
            ans[nei] = max(ans[nei],ans[node]+1)
# print(layer)

print(sum(ans))
```

基本信息

#: 49166024
题目: 22508
提交人: 24n2400016635
内存: 3756kB
时间: 29ms
语言: Python3
提交时间: 2025-05-15 02:48:03

English 帮助 关于

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>

目前图的题模板性比较强，还在练习树上 dp 的题目，每日选做还在补齐。感觉图的各种算法的实现很大胆却又是正确的，创造这些算法的人太神了。