Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by ==胡景博 药学院==

说明：

1）请把每个题目解题思路（可选），源码 Python, 或者 C++（已经在 Codeforces/Openjudge 上 AC），截图（包含 Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用 word）。AC 或者没有 AC，都请标上每个题目大致花费时间。

2）提交时候先提交 pdf 文件，再把 md 或者 doc 文件上传到右侧“作业评论”。Canvas 需要有同学清晰头像、提交文件有 pdf、"作业评论" 区有上传的 md 或者 doc 附件。

3）如果不能在截止前提交作业，请写明原因。

编程环境

==（请改为同学的操作系统、编程环境等）==

操作系统：macOS Ventura 13.4.1 (c)

Python 编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

28170: 算鹰

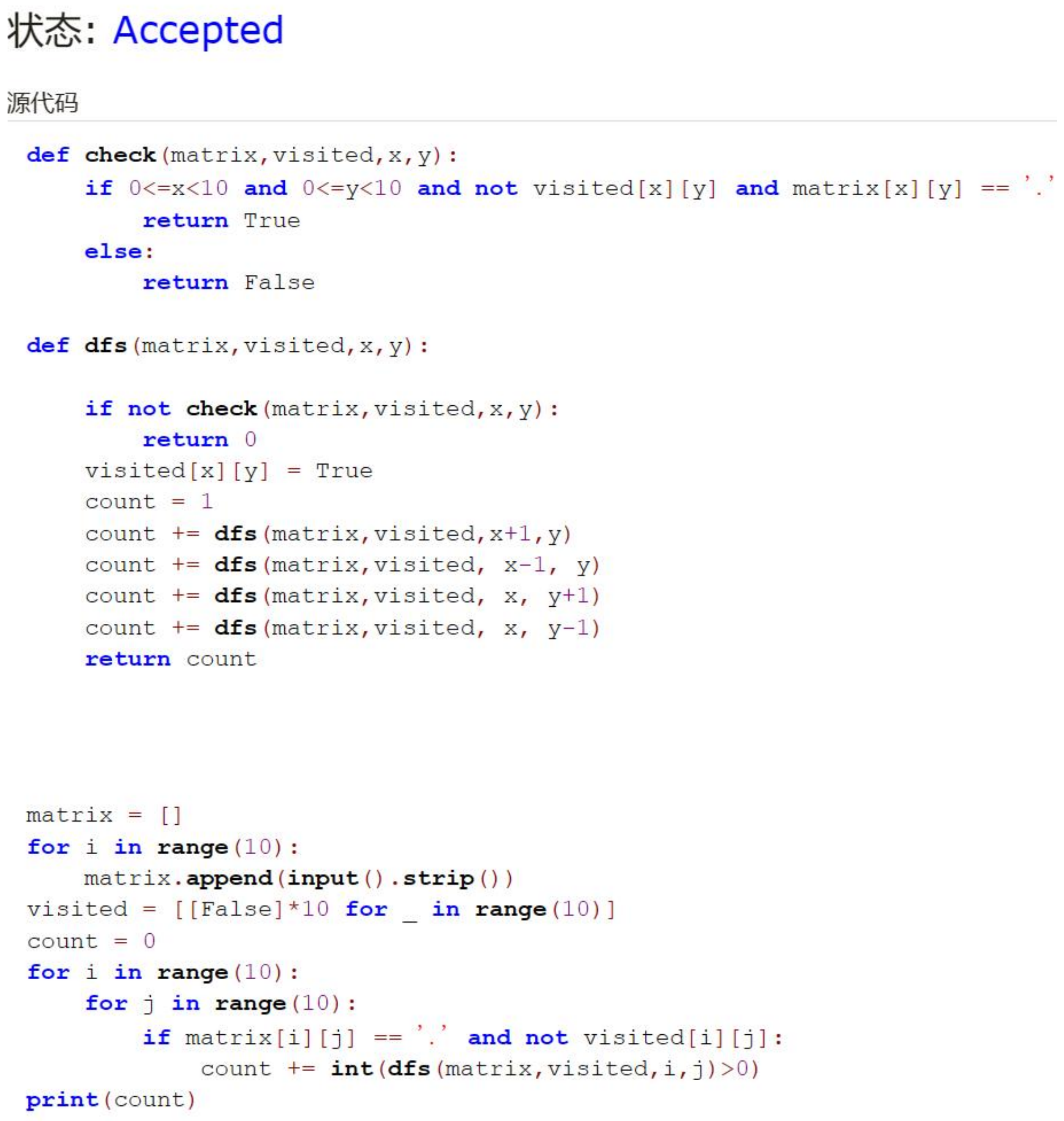
dfs, http://cs101.openjudge.cn/practice/28170/

思路： dfs,数联通区域。

代码

```
# def check(matrix,visited,x,y):
    if 0<=x<10 and 0<=y<10 and not visited[x][y] and matrix[x][y] == '.':
        return True
    else:
        return False
def dfs(matrix,visited,x,y):

    if not check(matrix,visited,x,y):
        return 0
    visited[x][y] = True
    count = 1
    count += dfs(matrix,visited,x+1,y)
    count += dfs(matrix,visited, x-1, y)
    count += dfs(matrix,visited, x, y+1)
    count += dfs(matrix,visited, x, y-1)
    return count




matrix = []for i in range(10):
    matrix.append(input().strip())
visited = [[False]*10 for _ in range(10)]
count = 0for i in range(10):
    for j in range(10):
        if matrix[i][j] == '.' and not visited[i][j]:
            count += int(dfs(matrix,visited,i,j)>0)print(count)
```

代码运行截图 ==（至少包含有"Accepted"）==



状态: Accepted

源代码

```python
def check(matrix,visited,x,y):
    if 0<=x<10 and 0<=y<10 and not visited[x][y] and matrix[x][y] == '.'
        return True
    else:
        return False

def dfs(matrix,visited,x,y):

    if not check(matrix,visited,x,y):
        return 0
    visited[x][y] = True
    count = 1
    count += dfs(matrix,visited,x+1,y)
    count += dfs(matrix,visited, x-1, y)
    count += dfs(matrix,visited, x, y+1)
    count += dfs(matrix,visited, x, y-1)
    return count



matrix = []
for i in range(10):
    matrix.append(input().strip())
visited = [[False]*10 for _ in range(10)]
count = 0
for i in range(10):
    for j in range(10):
        if matrix[i][j] == '.' and not visited[i][j]:
            count += int(dfs(matrix,visited,i,j)>0)
print(count)
```

02754: 八皇后

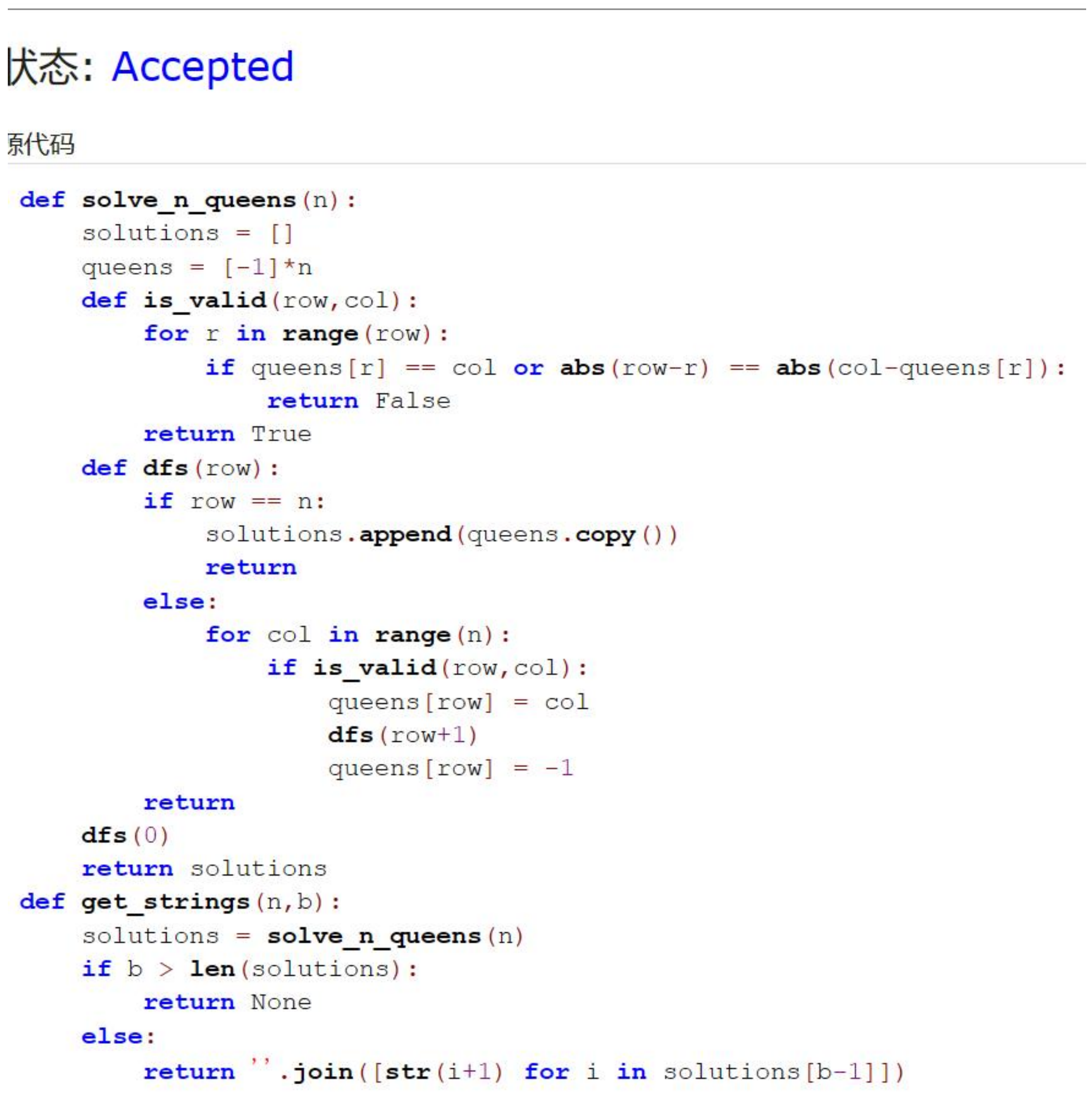
dfs, http://cs101.openjudge.cn/practice/02754/

思路： dfs

代码

```
# def solve_n_queens(n):
    solutions = []
    queens = [-1]*n
    def is_valid(row,col):
        for r in range(row):
            if queens[r] == col or abs(row-r) == abs(col-queens[r]):
                return False
        return True
    def dfs(row):
        if row == n:
            solutions.append(queens.copy())
            return
        else:
            for col in range(n):
                if is_valid(row,col):
                    queens[row] = col
                    dfs(row+1)
                    queens[row] = -1
        return
    dfs(0)
    return solutionsdef get_strings(n,b):
    solutions = solve_n_queens(n)
    if b > len(solutions):
        return None
    else:
        return ''.join([str(i+1) for i in solutions[b-1]])



T = int(input())
res=solve_n_queens(8)
res1=[]for line in res:
    newline = []
    for i in line:
        newline.append(int(i)+1)for i in range(T):
    b = int(input())
    print(get_strings(8,b))
```

代码运行截图 ==（至少包含有"Accepted"）==


状态: Accepted

源代码

```python
def solve_n_queens(n):
    solutions = []
    queens = [-1]*n
    def is_valid(row,col):
        for r in range(row):
            if queens[r] == col or abs(row-r) == abs(col-queens[r]):
                return False
        return True
    def dfs(row):
        if row == n:
            solutions.append(queens.copy())
            return
        else:
            for col in range(n):
                if is_valid(row,col):
                    queens[row] = col
                    dfs(row+1)
                    queens[row] = -1
        return
    dfs(0)
    return solutions
def get_strings(n,b):
    solutions = solve_n_queens(n)
    if b > len(solutions):
        return None
    else:
        return ''.join([str(i+1) for i in solutions[b-1]])
```

03151: Pots

bfs, http://cs101.openjudge.cn/practice/03151/

思路： bfs

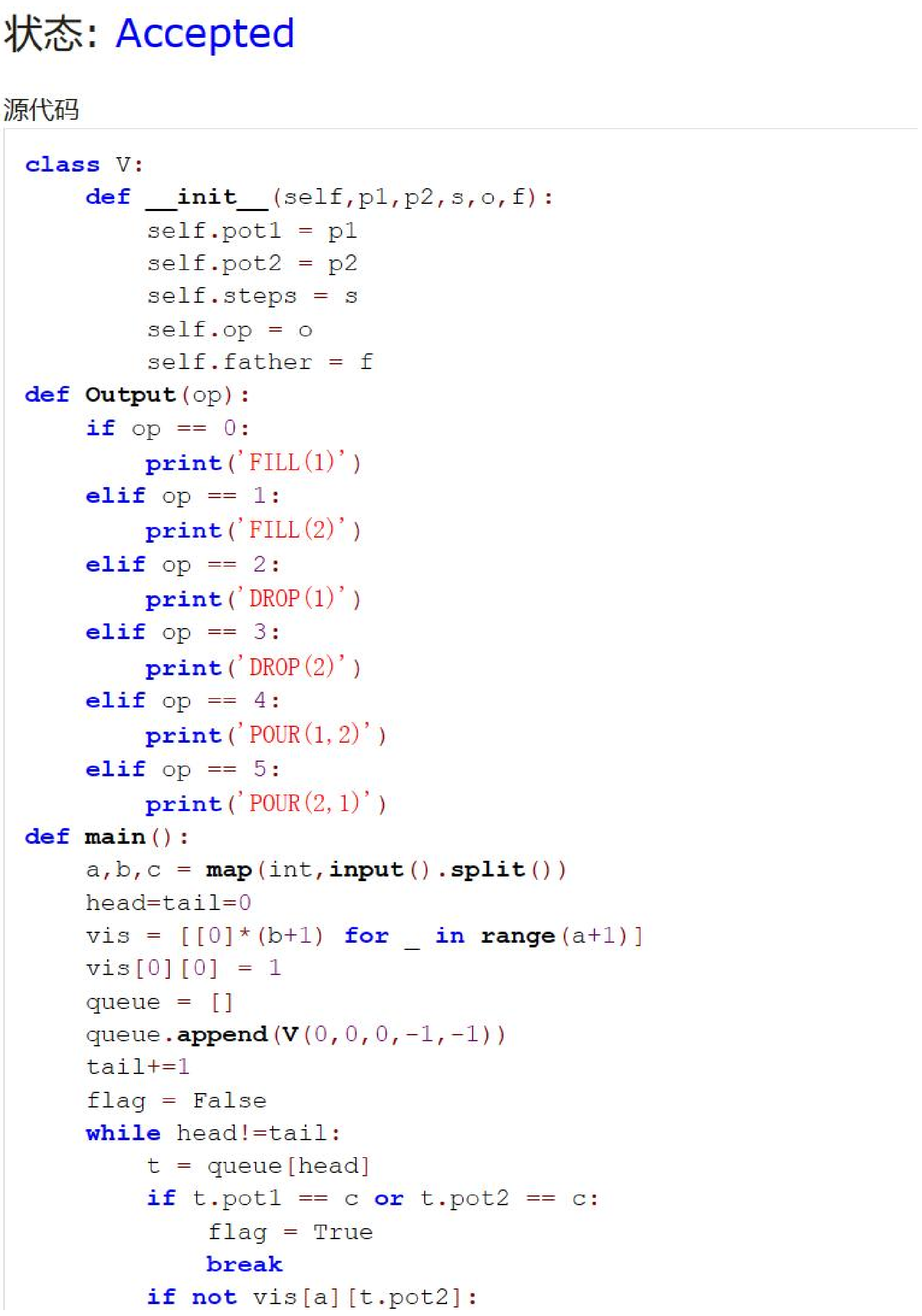
代码

```python
# class V:
    def __init__(self,p1,p2,s,o,f):
        self.pot1 = p1
        self.pot2 = p2
        self.steps = s
        self.op = o
        self.father = fdef Output(op):
```

```
if op == 0:
        print('FILL(1)')
elif op == 1:
        print('FILL(2)')
elif op == 2:
        print('DROP(1)')
elif op == 3:
        print('DROP(2)')
elif op == 4:
        print('POUR(1,2)')
elif op == 5:
        print('POUR(2,1)')def main():
a,b,c = map(int,input().split())
head=tail=0
vis = [[0]*(b+1) for _ in range(a+1)]
vis[0][0] = 1
queue = []
queue.append(V(0,0,0,-1,-1))
tail+=1
flag = False
while head!=tail:
        t = queue[head]
        if t.pot1 == c or t.pot2 == c:
                flag = True
                break
        if not vis[a][t.pot2]:
                vis[a][t.pot2] = 1
                queue.append(V(a,t.pot2,t.steps+1,0,head))
                tail+=1
        if not vis[t.pot1][b]:
                vis[t.pot1][b] = 1
                queue.append(V(t.pot1,b,t.steps+1,1,head))
                tail+=1
        if not vis[0][t.pot2]:
                vis[0][t.pot2] = 1
                queue.append(V(0,t.pot2,t.steps+1,2,head))
                tail+=1
        if not vis[t.pot1][0]:
                vis[t.pot1][0] = 1
                queue.append(V(t.pot1,0,t.steps+1,3,head))
                tail+=1
        sum = t.pot1 + t.pot2
        if sum>b:
                if not vis[sum-b][b]:
```

```
                vis[sum-b][b] = 1
                queue.append(V(sum-b,b,t.steps+1,4,head))
                tail+=1
        else:
            if not vis[0][sum]:
                vis[0][sum] = 1
                queue.append(V(0,sum, t.steps + 1, 4, head))
                tail+=1
        if sum>a:
            if not vis[a][sum-a]:
                vis[a][sum-a] = 1
                queue.append(V(a,sum-a,t.steps+1,5,head))
                tail+=1
        else:
            if not vis[sum][0]:
                vis[sum][0] = 1
                queue.append(V(sum,0,t.steps+1,5,head))
                tail+=1
        head+=1
    if not flag:
        print('impossible')

    else:
        print(queue[head].steps)
        stack = []
        node = queue[head]
        while node.father != -1:
            stack.append(node)
            node = queue[node.father]
        while stack:
            Output(stack.pop().op)
main()
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==


状态: Accepted

源代码

```
class V:
    def __init__(self,p1,p2,s,o,f):
        self.pot1 = p1
        self.pot2 = p2
        self.steps = s
        self.op = o
        self.father = f
def Output(op):
    if op == 0:
        print('FILL(1)')
    elif op == 1:
        print('FILL(2)')
    elif op == 2:
        print('DROP(1)')
    elif op == 3:
        print('DROP(2)')
    elif op == 4:
        print('POUR(1,2)')
    elif op == 5:
        print('POUR(2,1)')
def main():
    a,b,c = map(int,input().split())
    head=tail=0
    vis = [[0]*(b+1) for _ in range(a+1)]
    vis[0][0] = 1
    queue = []
    queue.append(V(0,0,0,-1,-1))
    tail+=1
    flag = False
    while head!=tail:
        t = queue[head]
        if t.pot1 == c or t.pot2 == c:
            flag = True
            break
        if not vis[a][t.pot2]:
```

05907: 二叉树的操作

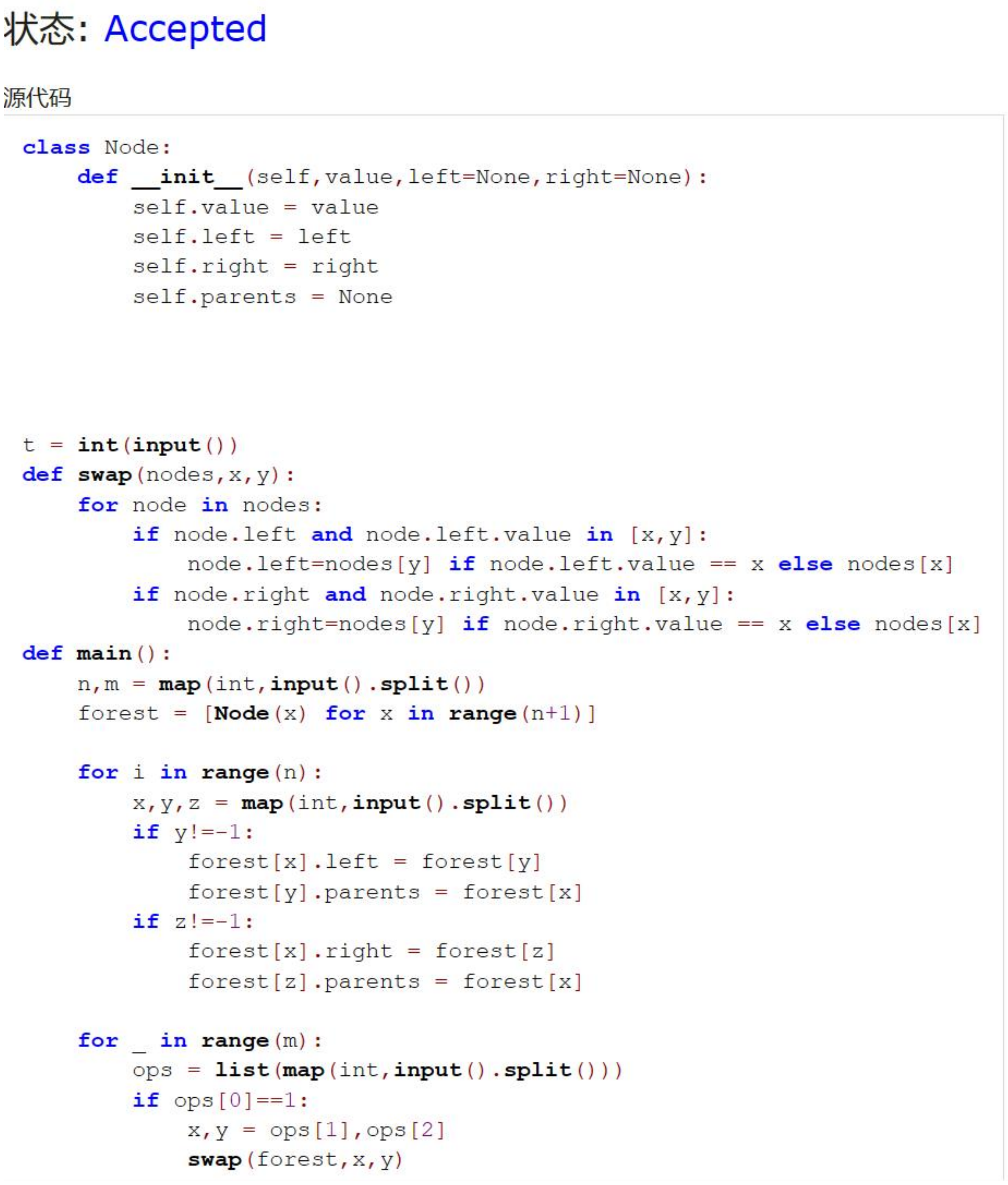
http://cs101.openjudge.cn/practice/05907/

思路： 建树

代码

```
# class Node:
    def __init__(self,value,left=None,right=None):
        self.value = value
        self.left = left
        self.right = right
        self.parents = None




t = int(input())def swap(nodes,x,y):
    for node in nodes:
        if node.left and node.left.value in [x,y]:
            node.left=nodes[y] if node.left.value == x else nodes[x]
        if node.right and node.right.value in [x,y]:
            node.right=nodes[y] if node.right.value == x else nodes[x]def main():
    n,m = map(int,input().split())
    forest = [Node(x) for x in range(n+1)]

    for i in range(n):
        x,y,z = map(int,input().split())
        if y!=-1:
            forest[x].left = forest[y]
            forest[y].parents = forest[x]
        if z!=-1:
            forest[x].right = forest[z]
            forest[z].parents = forest[x]

    for _ in range(m):
        ops = list(map(int,input().split()))
        if ops[0]==1:
            x,y = ops[1],ops[2]
            swap(forest,x,y)



        if ops[0]==2:
            index = ops[1]
            node = forest[index]
            while node and node.left:
                node = node.left
            print(node.value)for _ in range(t):
    main()
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==


状态: Accepted

源代码

```python
class Node:
    def __init__(self,value,left=None,right=None):
        self.value = value
        self.left = left
        self.right = right
        self.parents = None




t = int(input())
def swap(nodes,x,y):
    for node in nodes:
        if node.left and node.left.value in [x,y]:
            node.left=nodes[y] if node.left.value == x else nodes[x]
        if node.right and node.right.value in [x,y]:
            node.right=nodes[y] if node.right.value == x else nodes[x]
def main():
    n,m = map(int,input().split())
    forest = [Node(x) for x in range(n+1)]

    for i in range(n):
        x,y,z = map(int,input().split())
        if y!=-1:
            forest[x].left = forest[y]
            forest[y].parents = forest[x]
        if z!=-1:
            forest[x].right = forest[z]
            forest[z].parents = forest[x]

    for _ in range(m):
        ops = list(map(int,input().split()))
        if ops[0]==1:
            x,y = ops[1],ops[2]
            swap(forest,x,y)
```

18250：冰阔落 I

Disjoint set, http://cs101.openjudge.cn/practice/18250/

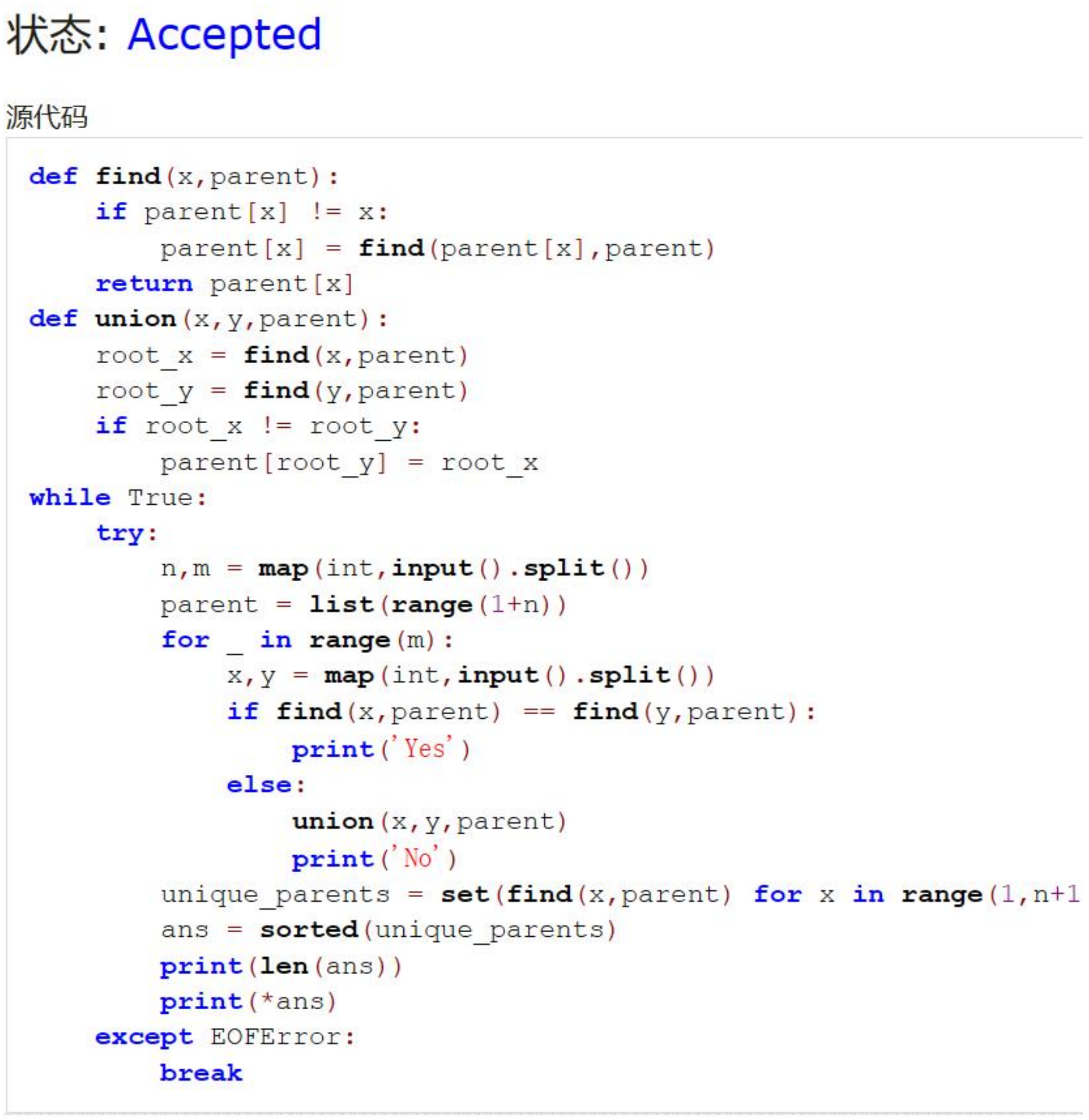
思路： 竟然是并查集。

代码

#

```
def find(x,parent):
    if parent[x] != x:
        parent[x] = find(parent[x],parent)
    return parent[x]def union(x,y,parent):
    root_x = find(x,parent)
    root_y = find(y,parent)
    if root_x != root_y:
        parent[root_y] = root_xwhile True:
    try:
        n,m = map(int,input().split())
        parent = list(range(1+n))
        for _ in range(m):
            x,y = map(int,input().split())
            if find(x,parent) == find(y,parent):
                print('Yes')
            else:
                union(x,y,parent)
                print('No')
        unique_parents = set(find(x,parent) for x in range(1,n+1))
        ans = sorted(unique_parents)
        print(len(ans))
        print(*ans)
    except EOFError:
        break
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

```python
def find(x,parent):
    if parent[x] != x:
        parent[x] = find(parent[x],parent)
    return parent[x]
def union(x,y,parent):
    root_x = find(x,parent)
    root_y = find(y,parent)
    if root_x != root_y:
        parent[root_y] = root_x
while True:
    try:
        n,m = map(int,input().split())
        parent = list(range(1+n))
        for _ in range(m):
            x,y = map(int,input().split())
            if find(x,parent) == find(y,parent):
                print('Yes')
            else:
                union(x,y,parent)
                print('No')
        unique_parents = set(find(x,parent) for x in range(1,n+1
        ans = sorted(unique_parents)
        print(len(ans))
        print(*ans)
    except EOFError:
        break
```

05443: 兔子与樱花

http://cs101.openjudge.cn/practice/05443/

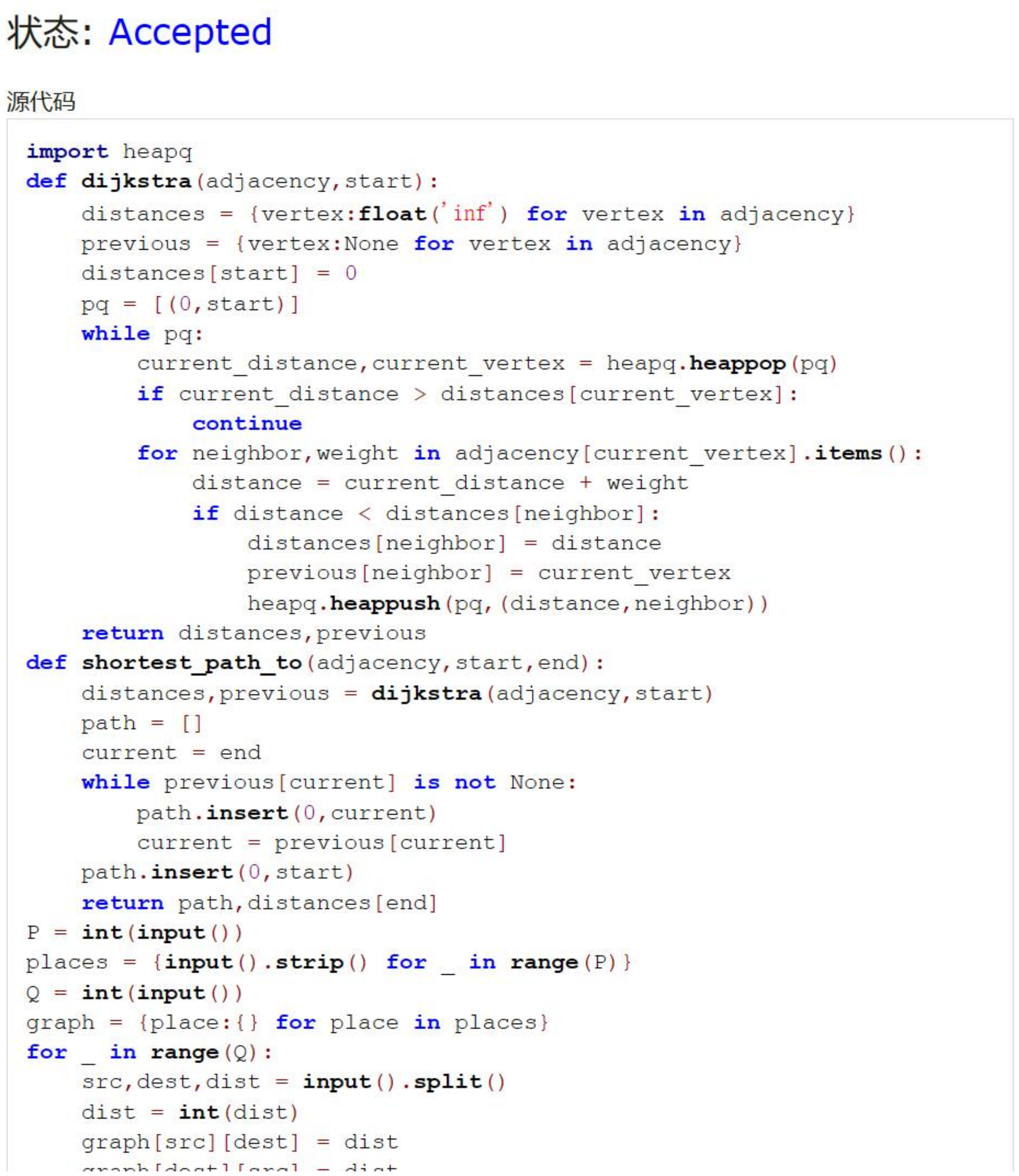
思路：

邻接链表的 Dijkstra 算法

代码

```
# import heapqdef dijkstra(adjacency,start):
    distances = {vertex:float('inf') for vertex in adjacency}
    previous = {vertex:None for vertex in adjacency}
    distances[start] = 0
    pq = [(0,start)]
    while pq:
        current_distance,current_vertex = heapq.heappop(pq)
        if current_distance > distances[current_vertex]:
            continue
        for neighbor,weight in adjacency[current_vertex].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous[neighbor] = current_vertex
                heapq.heappush(pq,(distance,neighbor))
    return distances,previousdef shortest_path_to(adjacency,start,end):
    distances,previous = dijkstra(adjacency,start)
    path = []
    current = end
    while previous[current] is not None:
        path.insert(0,current)
        current = previous[current]
    path.insert(0,start)
    return path,distances[end]
P = int(input())
places = {input().strip() for _ in range(P)}
Q = int(input())
graph = {place:{} for place in places}for _ in range(Q):
    src,dest,dist = input().split()
    dist = int(dist)
    graph[src][dest] = dist
    graph[dest][src] = dist
R = int(input())
requests = [input().split() for _ in range(R)]for start,end in requests:
    if start == end:
        print(start)
        continue
    path,total_dist = shortest_path_to(graph,start,end)
    output = ""
    for i in range(len(path)-1):
        output += f"{path[i]}->({graph[path[i]][path[i+1]]})->"
    output += f"{end}"
    print(output)
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==


状态: Accepted

源代码

```python
import heapq
def dijkstra(adjacency,start):
    distances = {vertex:float('inf') for vertex in adjacency}
    previous = {vertex:None for vertex in adjacency}
    distances[start] = 0
    pq = [(0,start)]
    while pq:
        current_distance,current_vertex = heapq.heappop(pq)
        if current_distance > distances[current_vertex]:
            continue
        for neighbor,weight in adjacency[current_vertex].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous[neighbor] = current_vertex
                heapq.heappush(pq, (distance,neighbor))
    return distances,previous
def shortest_path_to(adjacency,start,end):
    distances,previous = dijkstra(adjacency,start)
    path = []
    current = end
    while previous[current] is not None:
        path.insert(0,current)
        current = previous[current]
    path.insert(0,start)
    return path,distances[end]
P = int(input())
places = {input().strip() for _ in range(P)}
Q = int(input())
graph = {place:{} for place in places}
for _ in range(Q):
    src,dest,dist = input().split()
    dist = int(dist)
    graph[src][dest] = dist
    graph[dest][src] = dist
```

2. 学习总结和收获

==如果作业题目简单，有否额外练习题目，比如：OJ“2024spring 每日选做”、CF、LeetCode、洛谷等网站题目。==

第三题挺神奇的，没想到用也不会用图解决，照搬网上 C++代码改的； 熟悉了各种数据结构在实际问题中的用法。