Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by ~~天幂~~ 化学与分子工程学院

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

操作系统：Windows 11 23H2

Python编程环境：Visual Studio Code 1.86.2230.

1. 题目

28170: 算鹰

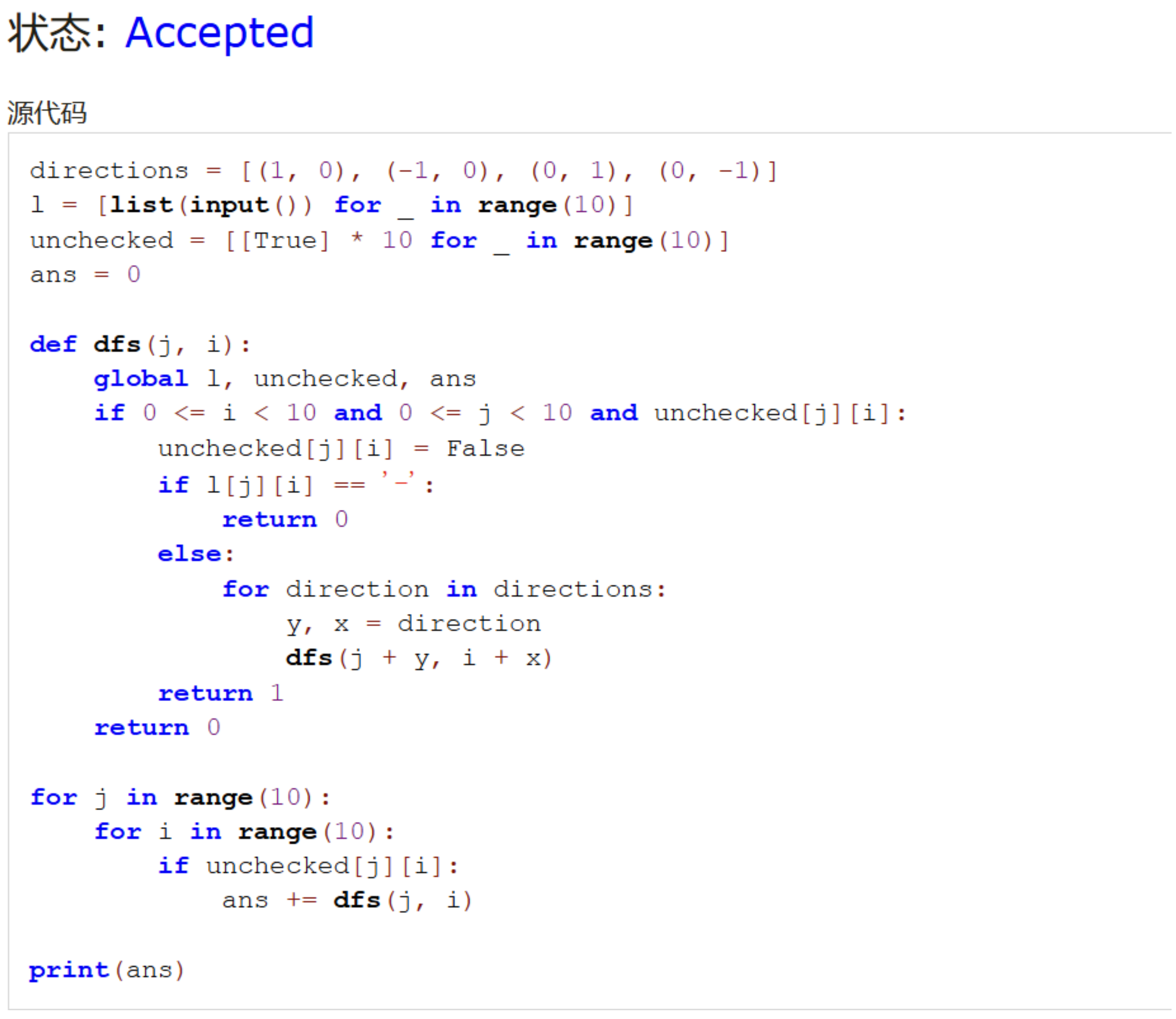
dfs, http://cs101.openjudge.cn/practice/28170/

思路：遍历+dfs读区域

代码

```python
directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
l = [list(input()) for _ in range(10)]
unchecked = [[True] * 10 for _ in range(10)]
ans = 0

def dfs(j, i):
    global l, unchecked, ans
    if 0 <= i < 10 and 0 <= j < 10 and unchecked[j][i]:
        unchecked[j][i] = False
        if l[j][i] == '-':
            return 0
        else:
            for direction in directions:
                y, x = direction
                dfs(j + y, i + x)
        return 1
    return 0

for j in range(10):
    for i in range(10):
        if unchecked[j][i]:
            ans += dfs(j, i)

print(ans)
```

代码运行截图



状态: Accepted

源代码

```python
directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
l = [list(input()) for _ in range(10)]
unchecked = [[True] * 10 for _ in range(10)]
ans = 0

def dfs(j, i):
    global l, unchecked, ans
    if 0 <= i < 10 and 0 <= j < 10 and unchecked[j][i]:
        unchecked[j][i] = False
        if l[j][i] == '-':
            return 0
        else:
            for direction in directions:
                y, x = direction
                dfs(j + y, i + x)
        return 1
    return 0

for j in range(10):
    for i in range(10):
        if unchecked[j][i]:
            ans += dfs(j, i)

print(ans)
```

02754: 八皇后

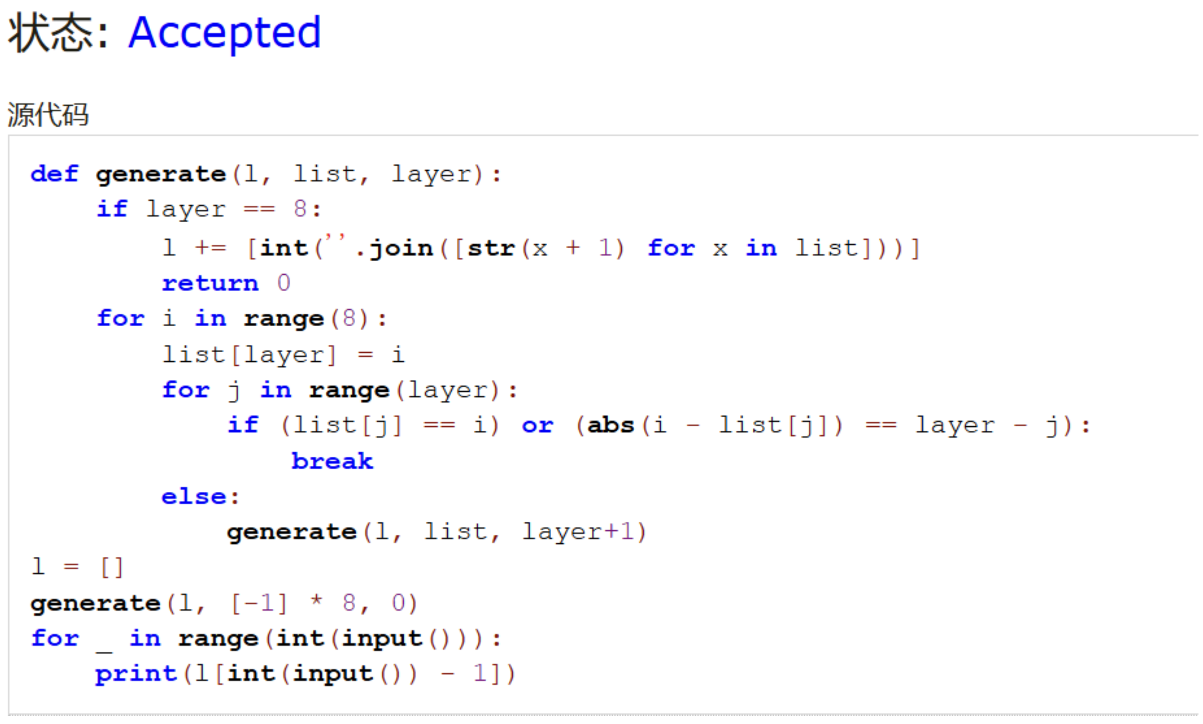
dfs, http://cs101.openjudge.cn/practice/02754/

思路：计概老代码

代码

```python
def generate(l, list, layer):
    if layer == 8:
        l += [int(''.join([str(x + 1) for x in list]))]
        return 0
    for i in range(8):
        list[layer] = i
        for j in range(layer):
            if (list[j] == i) or (abs(i - list[j]) == layer - j):
                break
        else:
            generate(l, list, layer+1)
l = []
generate(l, [-1] * 8, 0)
for _ in range(int(input())):
    print(l[int(input()) - 1])
```

代码运行截图



状态: Accepted

源代码

```python
def generate(l, list, layer):
    if layer == 8:
        l += [int(''.join([str(x + 1) for x in list]))]
        return 0
    for i in range(8):
        list[layer] = i
        for j in range(layer):
            if (list[j] == i) or (abs(i - list[j]) == layer - j):
                break
        else:
            generate(l, list, layer+1)
l = []
generate(l, [-1] * 8, 0)
for _ in range(int(input())):
    print(l[int(input()) - 1])
```

03151: Pots

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

思路：利用bfs保证最优解，用二维数组统计状态，额外存一个路径用来输出。

代码

```python
a, b, c =map(int, input().split())
unvisited = [[True] * (a + 1) for _ in range(b + 1)]
```

```python

def bfs(l, depth = 0):
    global a, b, c, unvisited
    nextl = []
    for traceablepos in l:
        x, y, string = traceablepos
        if 0 <= x <= a and 0 <= y <= b and unvisited[y][x]:
            if x == c or y == c:
                return str(depth) + string
            unvisited[y][x] = False
            if x < a:
                nextl.append((a, y, string + '\nFILL(1)'))
            if y < b:
                nextl.append((x, b, string + '\nFILL(2)'))
            if x > 0:
                nextl.append((0, y, string + '\nDROP(1)'))
                if y < b:
                    remain = y + x - b
                    if remain >= 0:
                        nextl.append((remain, b, string + '\nPOUR(1,2)'))
                    else:
                        nextl.append((0, x + y, string + '\nPOUR(1,2)'))
            if y > 0:
                nextl.append((x, 0, string + '\nDROP(2)'))
                if x < a:
                    remain = x + y - a
                    if remain >= 0:
                        nextl.append((a, remain, string + '\nPOUR(2,1)'))
                    else:
                        nextl.append((x + y, 0, string + '\nPOUR(2,1)'))
    if not nextl:
        return 'impossible'
    return bfs(nextl, depth + 1)

print(bfs([(0, 0, '')]))
```

代码运行截图

状态: Accepted

源代码

```
a, b, c =map(int, input().split())
unvisited = [[True] * (a + 1) for _ in range(b + 1)]

def bfs(l, depth = 0):
    global a, b, c, unvisited
    nextl = []
    for traceablepos in l:
        x, y, string = traceablepos
        if 0 <= x <= a and 0 <= y <= b and unvisited[y][x]:
            if x == c or y == c:
                return str(depth) + string
            unvisited[y][x] = False
            if x < a:
                nextl.append((a, y, string + '\nFILL(1)'))
            if y < b:
                nextl.append((x, b, string + '\nFILL(2)'))
            if x > 0:
                nextl.append((0, y, string + '\nDROP(1)'))
                if y < b:
                    remain = y + x - b
                    if remain >= 0:
                        nextl.append((remain, b, string + '\nPOUR(1,2)'))
                    else:
                        nextl.append((0, x + y, string + '\nPOUR(1,2)'))
            if y > 0:
                nextl.append((x, 0, string + '\nDROP(2)'))
                if x < a:
                    remain = x + y - a
                    if remain >= 0:
                        nextl.append((a, remain, string + '\nPOUR(2,1)'))
                    else:
                        nextl.append((x + y, 0, string + '\nPOUR(2,1)'))
    if not nextl:
        return 'impossible'
    return bfs(nextl, depth + 1)

print(bfs([(0, 0, '')]))
```

05907: 二叉树的操作

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

思路：需要构建能追溯parent的树，故需要一次initTree()

代码

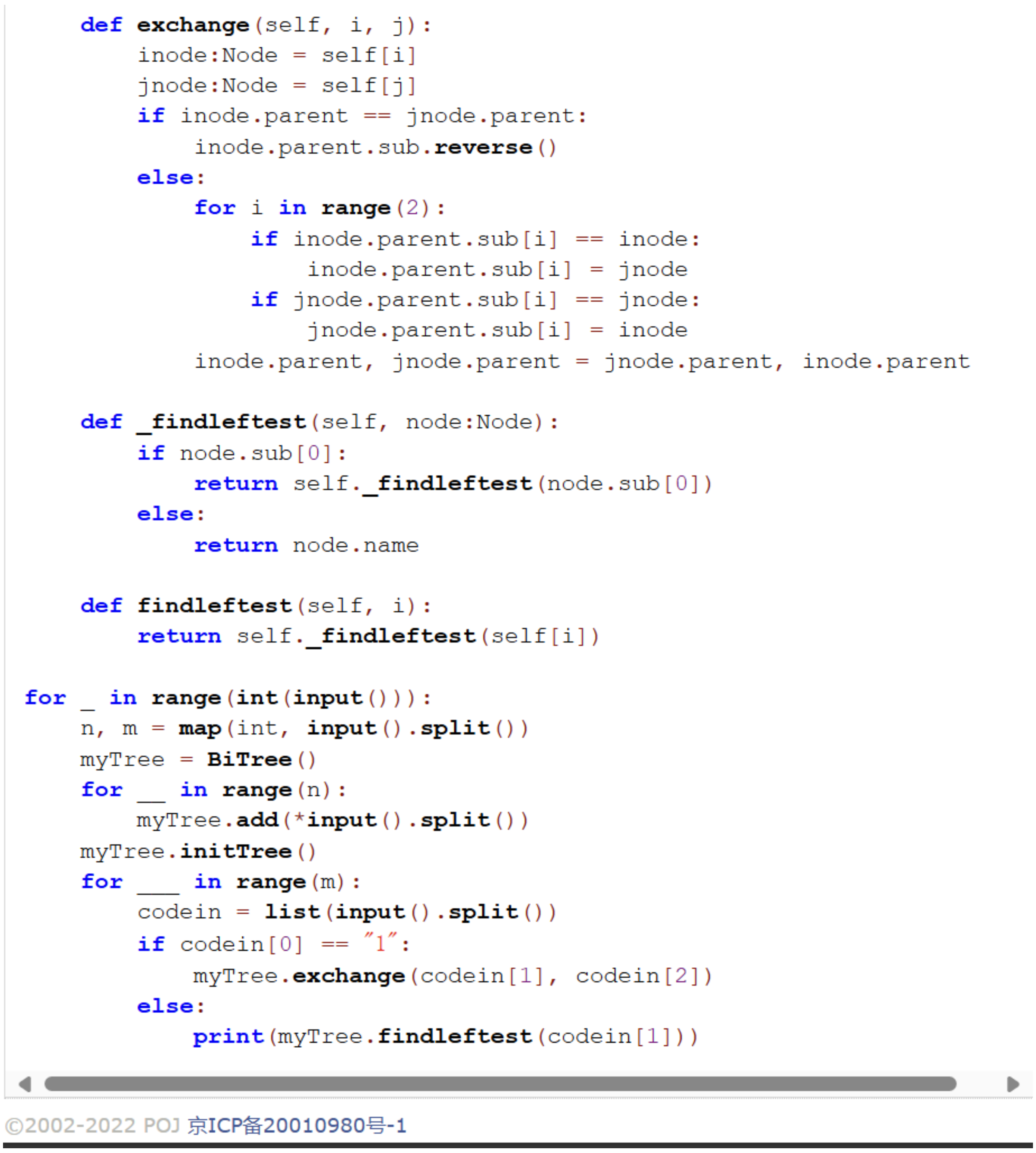
```python
from __future__ import annotations

class Node:
    _ID = 0
    NodeID:int
    name:str
    sub:list
    parent:Node

    def __init__(self, name, sub, parent=None):
        self.NodeID = Node._ID
        Node._ID += 1
        self.name = name
        self.sub = sub
        self.parent = parent

class BiTree(dict):
    root:Node
    def __init__(self):
        self.parent = dict()
        self.root = None

    def findParent(self, t):
        if t not in self.parent:
            return None
        return self.parent[t]

    def getOrCreate(self, nodename):
        if nodename == "-1":
            return False
        if nodename not in self:
            self[nodename] = Node(nodename, [False, False])
        return self[nodename]

    def add(self, t, l, r):
        if t not in self:
            neonode = Node(t, [self.getOrCreate(l), self.getOrCreate(r)])
            self[t] = neonode
        else:
            neonode = self[t]
            neonode.sub = [self.getOrCreate(l), self.getOrCreate(r)]
        if not self.root:
            self.root = neonode
        if l != "-1":
            self.parent[l] = neonode
        if r != "-1":
            self.parent[r] = neonode

    def initTree(self):
        for nodename in self:
            if nodename == '0':
                continue
            node:Node = self[nodename]
            node.parent = self.parent[nodename]

    def exchange(self, i, j):
        inode:Node = self[i]
        jnode:Node = self[j]
        if inode.parent == jnode.parent:
            inode.parent.sub.reverse()
        else:
            for i in range(2):
                if inode.parent.sub[i] == inode:
                    inode.parent.sub[i] = jnode
                if jnode.parent.sub[i] == jnode:
                    jnode.parent.sub[i] = inode
            inode.parent, jnode.parent = jnode.parent, inode.parent

    def _findleftest(self, node:Node):
        if node.sub[0]:
            return self._findleftest(node.sub[0])
        else:
            return node.name

    def findleftest(self, i):
        return self._findleftest(self[i])

for _ in range(int(input())):
    n, m = map(int, input().split())
    myTree = BiTree()
    for __ in range(n):
        myTree.add(*input().split())
    myTree.initTree()
    for ___ in range(m):
        codein = list(input().split())
        if codein[0] == "1":
            myTree.exchange(codein[1], codein[2])
        else:
            print(myTree.findleftest(codein[1]))
```

代码运行截图

状态: Accepted

源代码

```python
from __future__ import annotations

class Node:
    _ID = 0
    NodeID:int
    name:str
    sub:list
    parent:Node

    def __init__(self, name, sub, parent=None):
        self.NodeID = Node._ID
        Node._ID += 1
        self.name = name
        self.sub = sub
        self.parent = parent

class BiTree(dict):
    root:Node
    def __init__(self):
        self.parent = dict()
        self.root = None

    def findParent(self, t):
        if t not in self.parent:
            return None
        return self.parent[t]

    def getOrCreate(self, nodename):
        if nodename == "-1":
            return False
        if nodename not in self:
            self[nodename] = Node(nodename, [False, False])
        return self[nodename]

    def add(self, t, l, r):
        if t not in self:
            neonode = Node(t, [self.getOrCreate(l), self.getOrCreate(r)
            self[t] = neonode
        else:
            neonode = self[t]
            neonode.sub = [self.getOrCreate(l), self.getOrCreate(r)]
        if not self.root:
            self.root = neonode
        if l != "-1":
            self.parent[l] = neonode
        if r != "-1":
            self.parent[r] = neonode

    def initTree(self):
        for nodename in self:
            if nodename == '0':
                continue
            node:Node = self[nodename]
            node.parent = self.parent[nodename]
```



```python
    def exchange(self, i, j):
        inode:Node = self[i]
        jnode:Node = self[j]
        if inode.parent == jnode.parent:
            inode.parent.sub.reverse()
        else:
            for i in range(2):
                if inode.parent.sub[i] == inode:
                    inode.parent.sub[i] = jnode
                if jnode.parent.sub[i] == jnode:
                    jnode.parent.sub[i] = inode
            inode.parent, jnode.parent = jnode.parent, inode.parent

    def _findleftest(self, node:Node):
        if node.sub[0]:
            return self._findleftest(node.sub[0])
        else:
            return node.name

    def findleftest(self, i):
        return self._findleftest(self[i])

for _ in range(int(input())):
    n, m = map(int, input().split())
    myTree = BiTree()
    for __ in range(n):
        myTree.add(*input().split())
    myTree.initTree()
    for ___ in range(m):
        codein = list(input().split())
        if codein[0] == "1":
            myTree.exchange(codein[1], codein[2])
        else:
            print(myTree.findleftest(codein[1]))
```

18250: 冰阔落 I

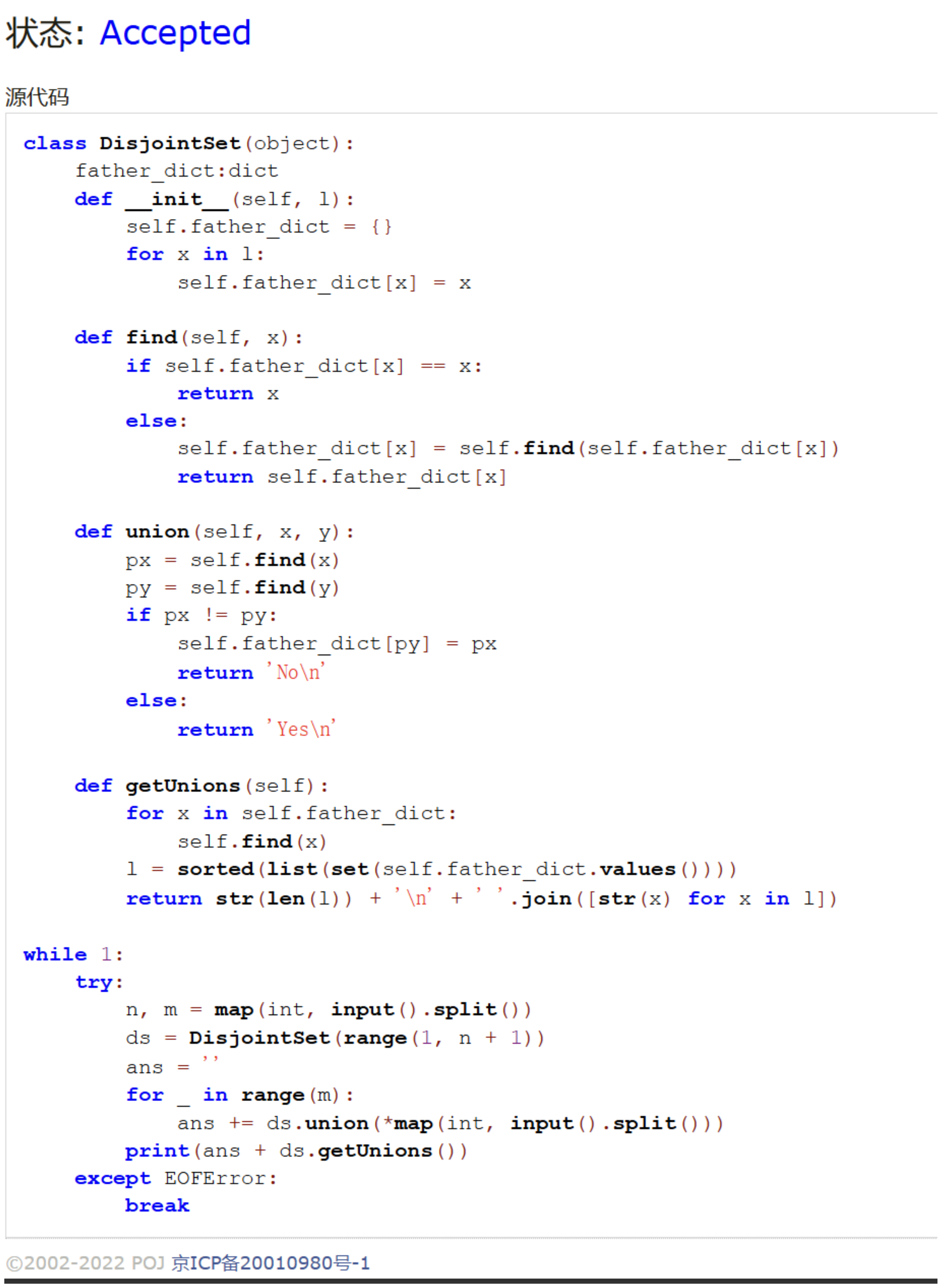
Disjoint set, http://cs101.openjudge.cn/practice/18250/

思路：基本的并查集。需要注意最后输出前需要转移一遍根节点。

代码

```python
class DisjointSet(object):
    father_dict:dict
    def __init__(self, l):
        self.father_dict = {}
        for x in l:
            self.father_dict[x] = x

    def find(self, x):
        if self.father_dict[x] == x:
            return x
        else:
            self.father_dict[x] = self.find(self.father_dict[x])
            return self.father_dict[x]

    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No\n'
        else:
            return 'Yes\n'

    def getUnions(self):
        for x in self.father_dict:
            self.find(x)
        l = sorted(list(set(self.father_dict.values())))
        return str(len(l)) + '\n' + ' '.join([str(x) for x in l])

while 1:
    try:
        n, m = map(int, input().split())
        ds = DisjointSet(range(1, n + 1))
        ans = ''
        for _ in range(m):
            ans += ds.union(*map(int, input().split()))
        print(ans + ds.getUnions())
    except EOFError:
        break
```

代码运行截图



状态: Accepted

源代码

```python
class DisjointSet(object):
    father_dict:dict
    def __init__(self, l):
        self.father_dict = {}
        for x in l:
            self.father_dict[x] = x

    def find(self, x):
        if self.father_dict[x] == x:
            return x
        else:
            self.father_dict[x] = self.find(self.father_dict[x])
            return self.father_dict[x]

    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No\n'
        else:
            return 'Yes\n'

    def getUnions(self):
        for x in self.father_dict:
            self.find(x)
        l = sorted(list(set(self.father_dict.values())))
        return str(len(l)) + '\n' + ' '.join([str(x) for x in l])

while 1:
    try:
        n, m = map(int, input().split())
        ds = DisjointSet(range(1, n + 1))
        ans = ''
        for _ in range(m):
            ans += ds.union(*map(int, input().split()))
        print(ans + ds.getUnions())
    except EOFError:
        break
```

05443: 兔子与樱花

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

思路：和走山路一个思路。

代码

```python
from heapq import *

inf = float('inf')

p = int(input())
distance = dict()
l = [input() for _ in range(p)]
for i in l:
    distance[i] = dict()
    for j in l:
        distance[i][j] = inf

q = int(input())
for _ in range(q):
    i, j, dis = input().split()
    distance[i][j] = distance[j][i] = int(dis)

r = int(input())

def to(traceablepos, target):
    global l, distance
    dis, pos, info = traceablepos
    return (dis + distance[pos][target], target, info + '->({})->{}'.format(distance[pos][target], target))

def bfs(origin, terminal):
    global distance

    nearest = dict()
    for i in l:
        nearest[i] = inf

    ways = [(0, origin, origin)]

    while ways:
        traceablepos = heappop(ways)
        dis, pos, info = traceablepos
        if pos == terminal:
            return info
        for target in l:
            if distance[pos][target] != inf and nearest[target] > dis + distance[pos][target]:
                nearest[target] = dis + distance[pos][target]
                heappush(ways, to(traceablepos, target))
    raise FileNotFoundError

for _ in range(r):
    print(bfs(*input().split()))
```

代码运行截图

状态: Accepted

源代码

```
from heapq import *

inf = float('inf')

p = int(input())
distance = dict()
l = [input() for _ in range(p)]
for i in l:
    distance[i] = dict()
    for j in l:
        distance[i][j] = inf

q = int(input())
for _ in range(q):
    i, j, dis = input().split()
    distance[i][j] = distance[j][i] = int(dis)

r = int(input())

def to(traceablepos, target):
    global l, distance
    dis, pos, info = traceablepos
    return (dis + distance[pos][target], target, info + '->({})->{}'.forma

def bfs(origin, terminal):
    global distance

    nearest = dict()
    for i in l:
        nearest[i] = inf

    ways = [(0, origin, origin)]

    while ways:
        traceablepos = heappop(ways)
        dis, pos, info = traceablepos
        if pos == terminal:
            return info
        for target in l:
            if distance[pos][target] != inf and nearest[target] > dis +
                nearest[target] = dis + distance[pos][target]
                heappush(ways, to(traceablepos, target))
    raise FileNotFoundError

for _ in range(r):
    print(bfs(*input().split()))
```

2. 学习总结和收获

这周的题做起来比较顺手，可能是因为前几周做熟了。感觉这些题目是非常不错的复习。