Assignment #A: 图论和树算

Updated 1600 GMT+8 Apr 28, 2024

2024 spring, Complied by 王申睿——物理学院

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 typorahttps://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/

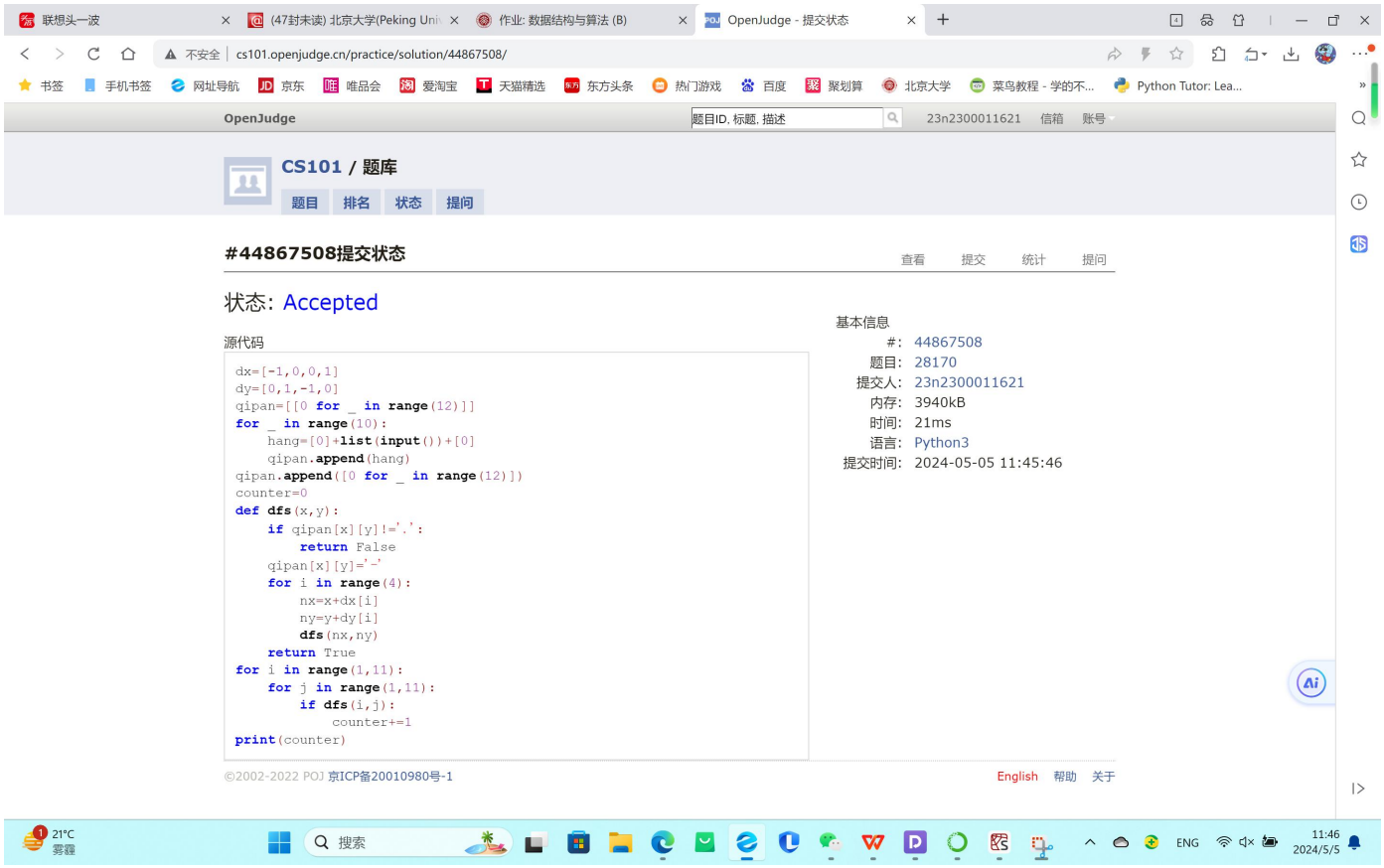
思路:

代码

```
dx=[-1,0,0,1]

dy=[0,1,-1,0]
```

```
qipan=[[0 for _ in range(12)]]

for _ in range(10):

    hang=[0]+list(input())+[0]

    qipan.append(hang)

qipan.append([0 for _ in range(12)])

counter=0

def dfs(x,y):

    if qipan[x][y]!='.':

        return False

    qipan[x][y]='-'

    for i in range(4):

        nx=x+dx[i]

        ny=y+dy[i]

        dfs(nx,ny)
```

```
    return True

for i in range(1,11):

    for j in range(1,11):

        if dfs(i,j):

            counter+=1

print(counter)
```



02754: 八皇后

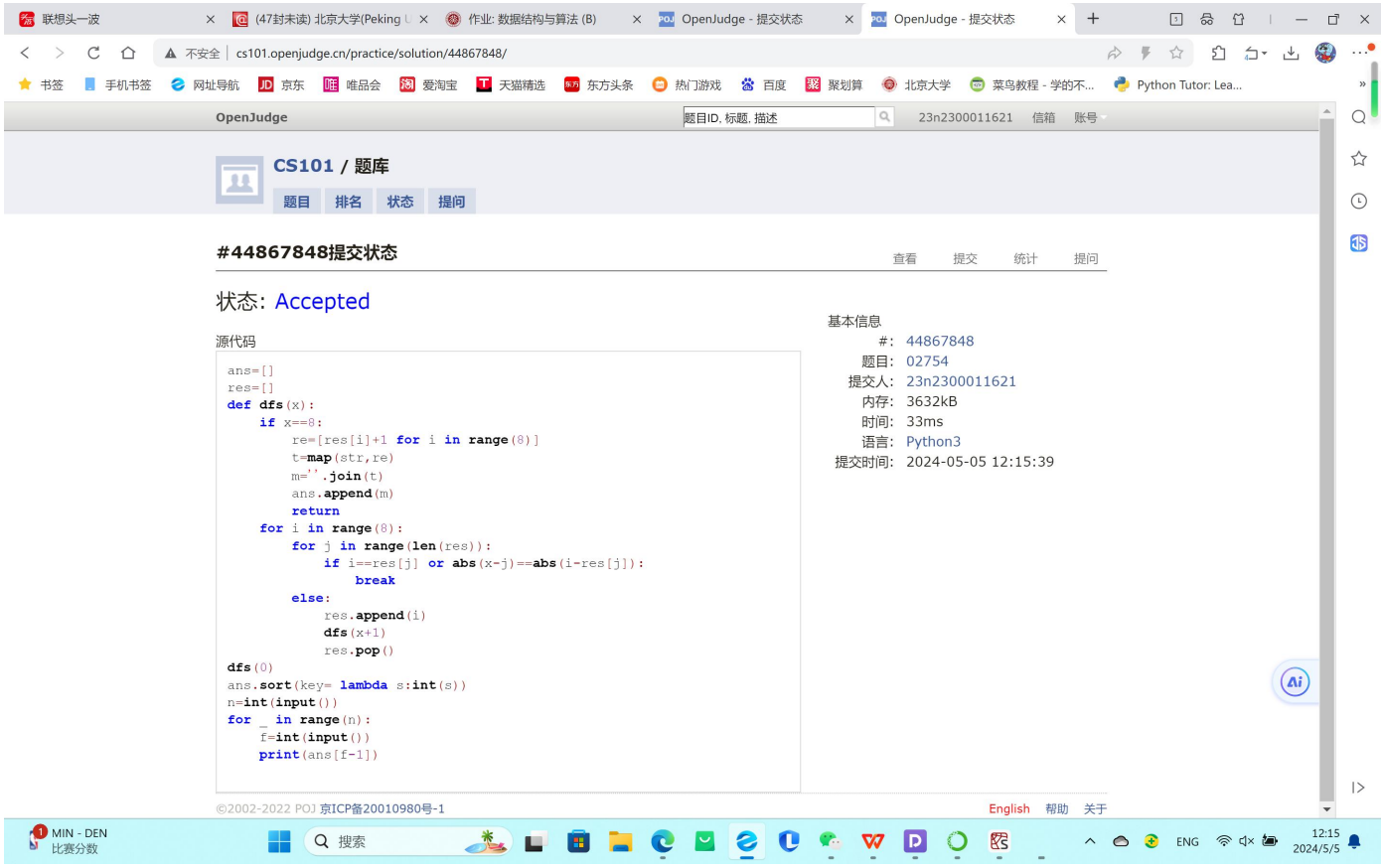
dfs, http://cs101.openjudge.cn/practice/02754/

思路:

代码

```
ans=[]

res=[]

def dfs(x):

    if x==8:

        re=[res[i]+1 for i in range(8)]

        t=map(str,re)

        m=''.join(t)

        ans.append(m)

        return

    for i in range(8):
```

```
        for j in range(len(res)):
            if i==res[j] or abs(x-j)==abs(i-res[j]):
                break
        else:
            res.append(i)
            dfs(x+1)
            res.pop()
dfs(0)
ans.sort(key= lambda s:int(s))
n=int(input())
for _ in range(n):
    f=int(input())
    print(ans[f-1])
```



03151: Pots

bfs, http://cs101.openjudge.cn/2024sp_routine/03151/

思路:

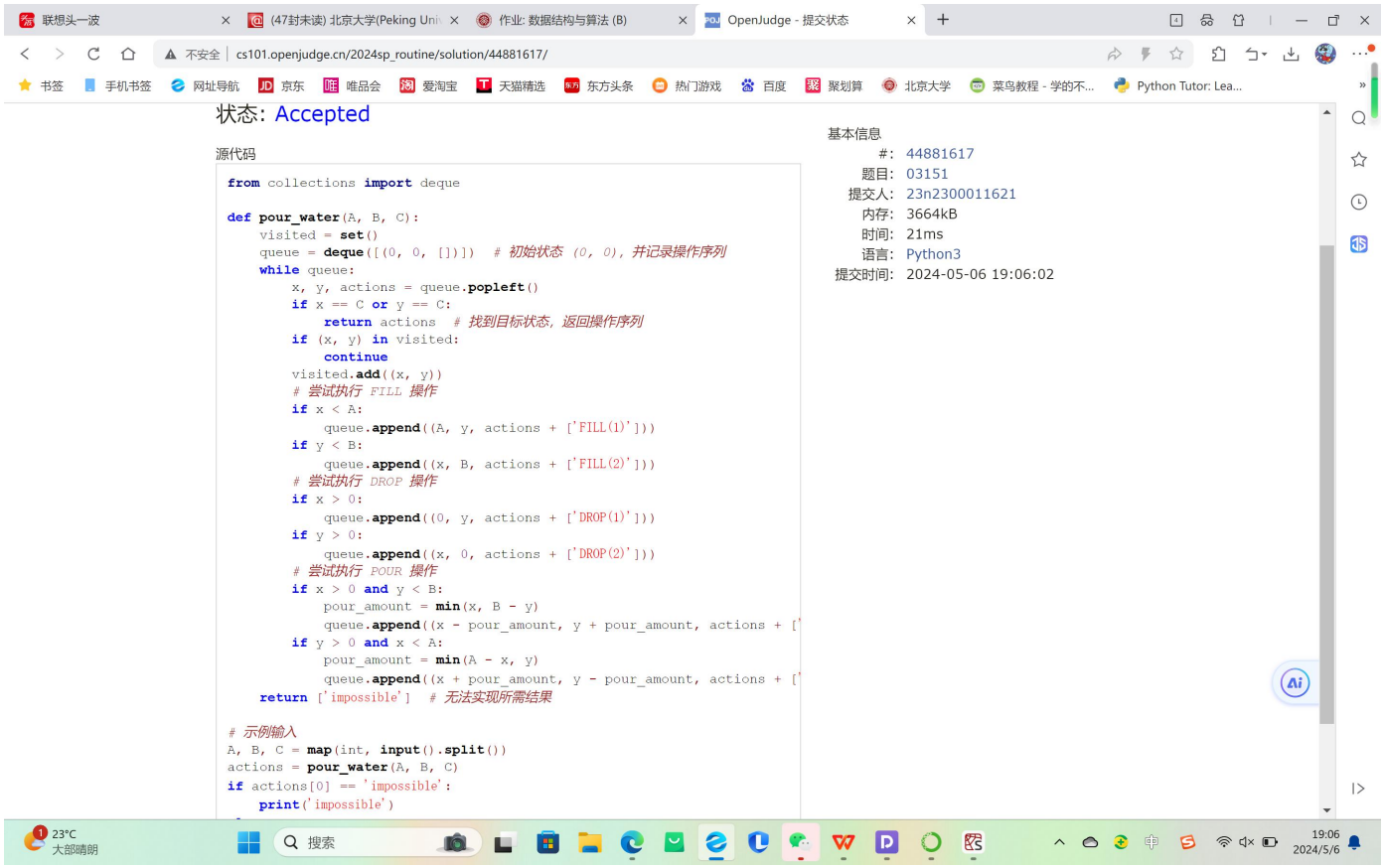
代码

```
from collections import deque



def pour_water(A, B, C):
```

```
visited = set()

queue = deque([(0, 0, [])])  # 初始状态 (0, 0)，并记录操作序列

while queue:

    x, y, actions = queue.popleft()

    if x == C or y == C:

        return actions  # 找到目标状态，返回操作序列

    if (x, y) in visited:

        continue

    visited.add((x, y))

    # 尝试执行 FILL 操作

    if x < A:

        queue.append((A, y, actions + ['FILL(1)']))

    if y < B:

        queue.append((x, B, actions + ['FILL(2)']))
```

```
    # 尝试执行 DROP 操作
    if x > 0:
        queue.append((0, y, actions + ['DROP(1)']))
    if y > 0:
        queue.append((x, 0, actions + ['DROP(2)']))
    # 尝试执行 POUR 操作
    if x > 0 and y < B:
        pour_amount = min(x, B - y)
        queue.append((x - pour_amount, y + pour_amount, actions + ['POUR(1,2)']))
    if y > 0 and x < A:
        pour_amount = min(A - x, y)
        queue.append((x + pour_amount, y - pour_amount, actions + ['POUR(2,1)']))
return ['impossible']  # 无法实现所需结果
```

```
# 示例输入

A, B, C = map(int, input().split())

actions = pour_water(A, B, C)

if actions[0] == 'impossible':

    print('impossible')

else:

    print(len(actions))

    for action in actions:

        print(action)
```



05907: 二叉树的操作

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

思路:

代码

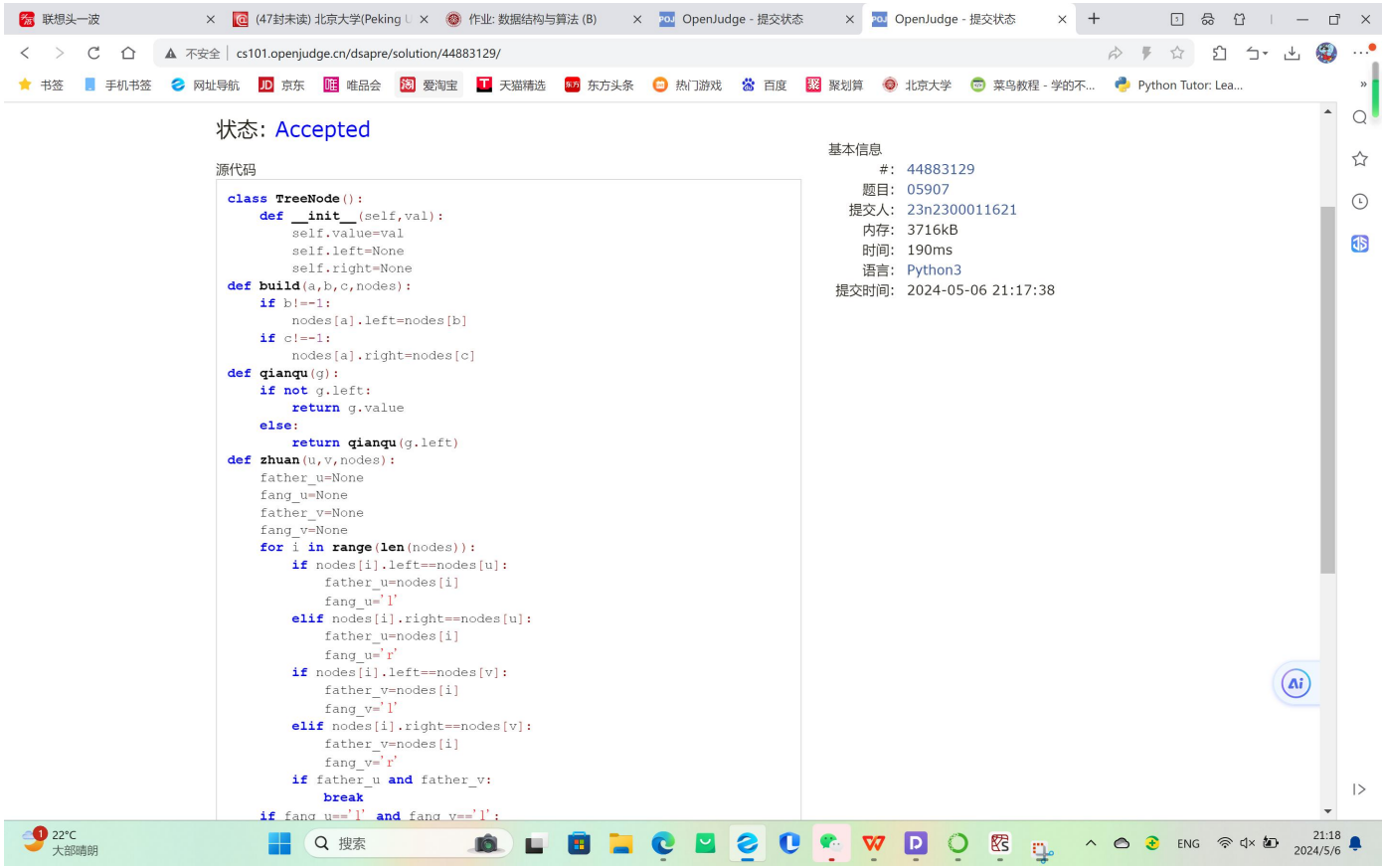
```
class TreeNode():
    def __init__(self,val):
        self.value=val
        self.left=None
        self.right=None
def build(a,b,c,nodes):
    if b!=-1:
        nodes[a].left=nodes[b]
    if c!=-1:
        nodes[a].right=nodes[c]
def qianqu(g):
    if not g.left:
        return g.value
```

```
    else:
        return qianqu(g.left)
def zhuan(u,v,nodes):
    father_u=None
    fang_u=None
    father_v=None
    fang_v=None
    for i in range(len(nodes)):
        if nodes[i].left==nodes[u]:
            father_u=nodes[i]
            fang_u='l'
        elif nodes[i].right==nodes[u]:
            father_u=nodes[i]
            fang_u='r'
```

```
    if nodes[i].left==nodes[v]:
        father_v=nodes[i]
        fang_v='l'
    elif nodes[i].right==nodes[v]:
        father_v=nodes[i]
        fang_v='r'
    if father_u and father_v:
        break
if fang_u=='l' and fang_v=='l':
    father_u.left,father_v.left=nodes[v],nodes[u]
elif fang_u=='r' and fang_v=='l':
    father_u.right,father_v.left=nodes[v],nodes[u]
elif fang_u=='l' and fang_v=='r':
    father_u.left,father_v.right=nodes[v],nodes[u]
```

```
    elif fang_u=='r' and fang_v=='r':
        father_u.right,father_v.right=nodes[v],nodes[u]




t=int(input())
for _ in range(t):
    n,m=map(int,input().split())
    nodes_1=[TreeNode(i) for i in range(n)]
    for _ in range(n):
        X,Y,Z=map(int,input().split())
        build(X,Y,Z,nodes_1)
    for _ in range(m):
        fa=list(map(int,input().split()))
        if fa[0]==1:
```

```
        zhuan(fa[1],fa[2],nodes_1)

    elif fa[0]==2:

        print(qianqu(nodes_1[fa[1]]))
```



18250: 冰阔落 I

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

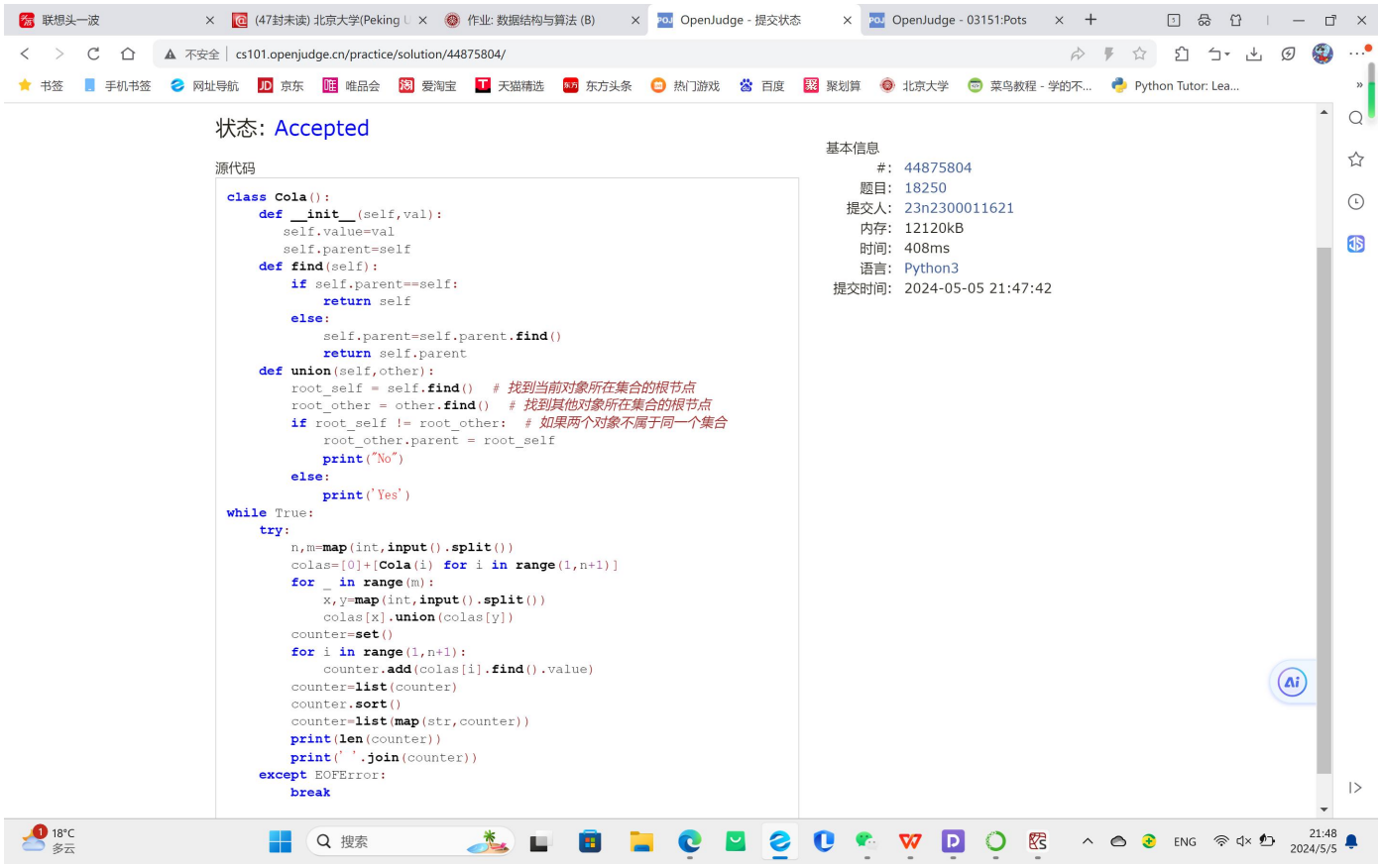
思路:

代码

```
class Cola():

    def __init__(self,val):

        self.value=val

        self.parent=self

    def find(self):

        if self.parent==self:

            return self

        else:

            self.parent=self.parent.find()

            return self.parent

    def union(self,other):

        root_self = self.find()  # 找到当前对象所在集合的根节点

        root_other = other.find()  # 找到其他对象所在集合的根节点

        if root_self != root_other:  # 如果两个对象不属于同一个集合
```

```
            root_other.parent = root_self

            print("No")

        else:

            print('Yes')

while True:

    try:

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

        colas=[0]+[Cola(i) for i in range(1,n+1)]

        for _ in range(m):

            x,y=map(int,input().split())

            colas[x].union(colas[y])

        counter=set()

        for i in range(1,n+1):

            counter.add(colas[i].find().value)
```

```
        counter=list(counter)

        counter.sort()

        counter=list(map(str,counter))

        print(len(counter))

        print(' '.join(counter))

    except EOFError:

        break
```



05443: 兔子与樱花

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

思路:

代码

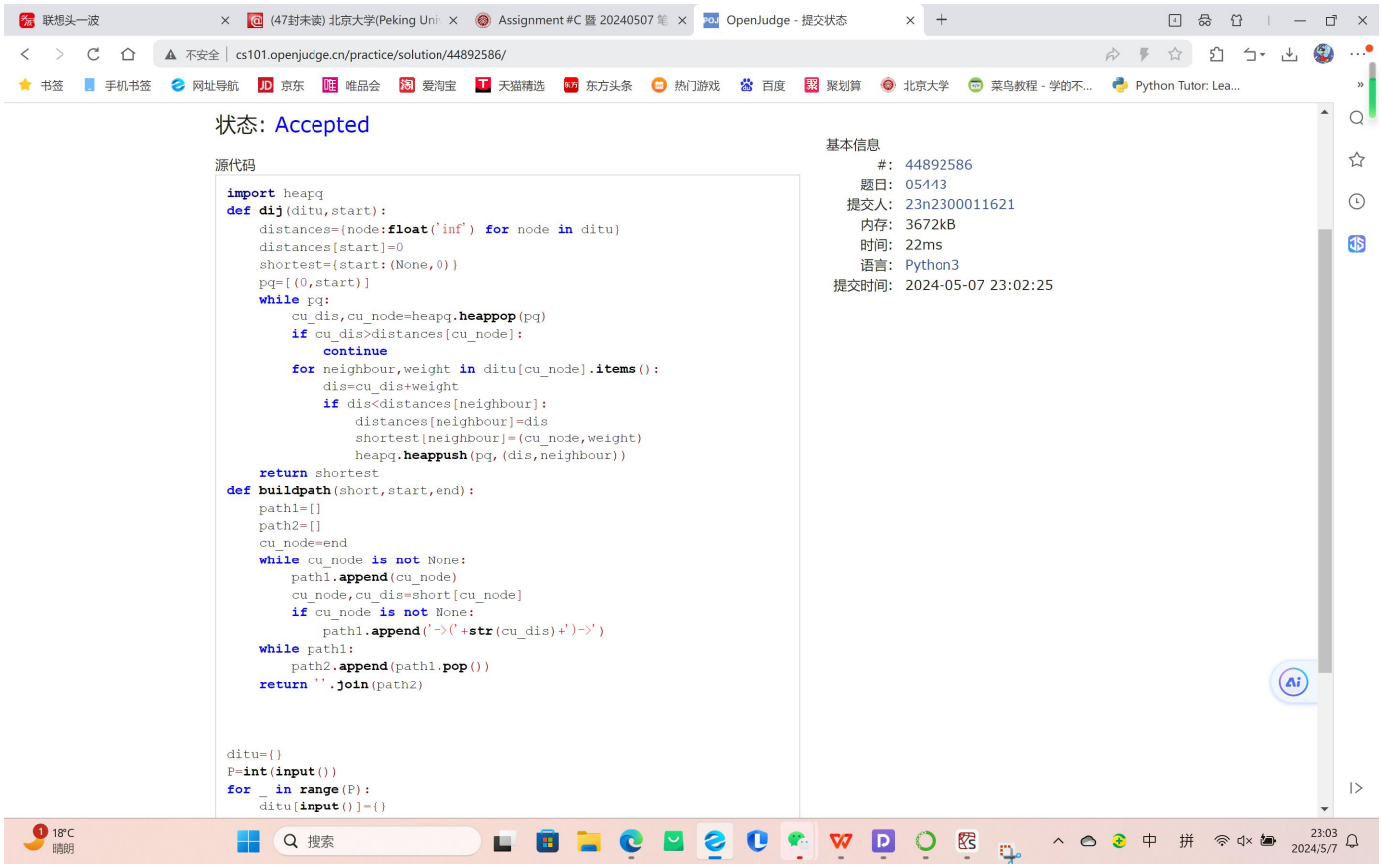
。

```
import heapq

def dij(ditu,start):
    distances={node:float('inf') for node in ditu}
    distances[start]=0
    shortest={start:(None,0)}
    pq=[(0,start)]
    while pq:
        cu_dis,cu_node=heapq.heappop(pq)
        if cu_dis>distances[cu_node]:
            continue
        for neighbour,weight in ditu[cu_node].items():
            dis=cu_dis+weight
            if dis<distances[neighbour]:
                distances[neighbour]=dis
                shortest[neighbour]=(cu_node,weight)
                heapq.heappush(pq,(dis,neighbour))
    return shortest
def buildpath(short,start,end):
```

```
    path1=[]
    path2=[]
    cu_node=end
    while cu_node is not None:
        path1.append(cu_node)
        cu_node,cu_dis=short[cu_node]
        if cu_node is not None:
            path1.append('->('+str(cu_dis)+')->')
    while path1:
        path2.append(path1.pop())
    return ''.join(path2)
```

```
ditu={}
P=int(input())
for _ in range(P):
    ditu[input()]={}
```

```
Q=int(input())

for _ in range(Q):

    fa=list(input().split())

    ditu[fa[0]][fa[1]]=int(fa[2])

    ditu[fa[1]][fa[0]]=int(fa[2])

R=int(input())

for _ in range(R):

    st,en=input().split()

    lujing=dij(ditu,st)

    print(buildpath(lujing,st,en))
```



2. 学习总结和收获

今天是周日，然而我才想起来assignmentB.....

算鹰给了我一种举一反三的快感：它和上学期的最大连通域一样，都是无需回溯的递归。

终于制服了八皇后，数算给我带来的提升还是很明显的。

这次作业的输出格式竟然成了个大问题，诸如“YES”和“Yes”的细节以后一定要引起警惕！

另外，整型对象的排序不能用字符串代替！字符串“23”和“123”是后者在前，因为字符串排序按从高到低的ASCII码！（被这种“小又不小”的问题干扰真的很让人抓狂，已列入cheatsheet保留项）

总结并查集优化的路径压缩模版：每逢递归调用，将父节点更新为递归调用的结果；合并两集合时，先用find函数将路径压缩完毕并将根节点赋值新变量，再进行比较。

BFS和Dijkstra一提到路径就歇菜了，用字典表示指针来保存路径的技巧还要多加练习。