Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by ==同学的姓名、院系==

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

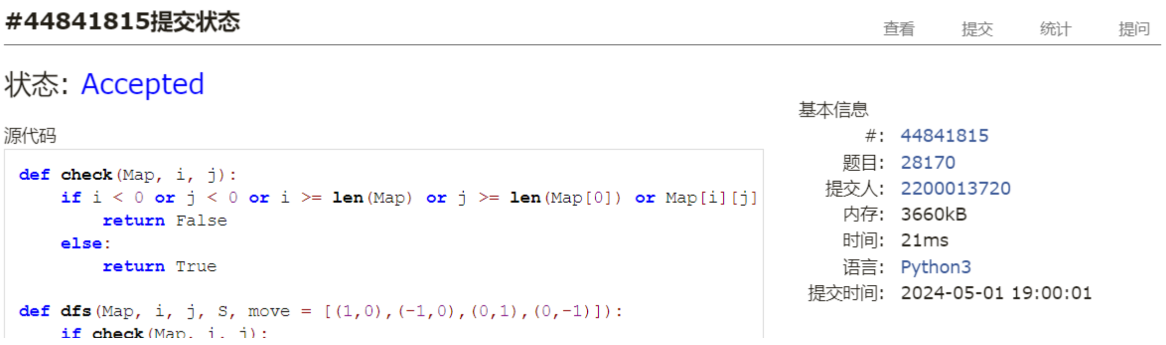
dfs, http://cs101.openjudge.cn/practice/28170/

思路：一开始写的时候知道跟最大连通域（或Lake counting）是一样的，但尝试开辟新的写代码习惯，最后惨败而归，警醒不要轻易尝试更改自己的写代码习惯（在没有造成太大影响的情况下）

代码

```python
# 
def check(Map, i, j):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]) or Map[i][j] == '-':
        return False
    else:
        return True

def dfs(Map, i, j, S, move = [(1,0),(-1,0),(0,1),(0,-1)]):
    if check(Map, i, j):
        if Map[i][j] == '.':
            S += 1
            Map[i][j] = '-'
            for m in move:
                S = dfs(Map, i + m[0], j + m[1], S)
    return S

Map = []
for _ in range(10):
    Map.append(list(input()))
result = 0
for i in range(10):
    for j in range(10):
        result += (dfs(Map, i, j, 0) > 0)
print(result)
```

代码运行截图 == （至少包含有"Accepted"） ==



#44841815提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def check(Map, i, j):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]) or Map[i][j]
        return False
    else:
        return True

def dfs(Map, i, j, S, move = [(1,0),(-1,0),(0,1),(0,-1)]):
    if check(Map, i, j):
```

基本信息

#:	44841815
题目:	28170
提交人:	2200013720
内存:	3660kB
时间:	21ms
语言:	Python3
提交时间:	2024-05-01 19:00:01

02754: 八皇后

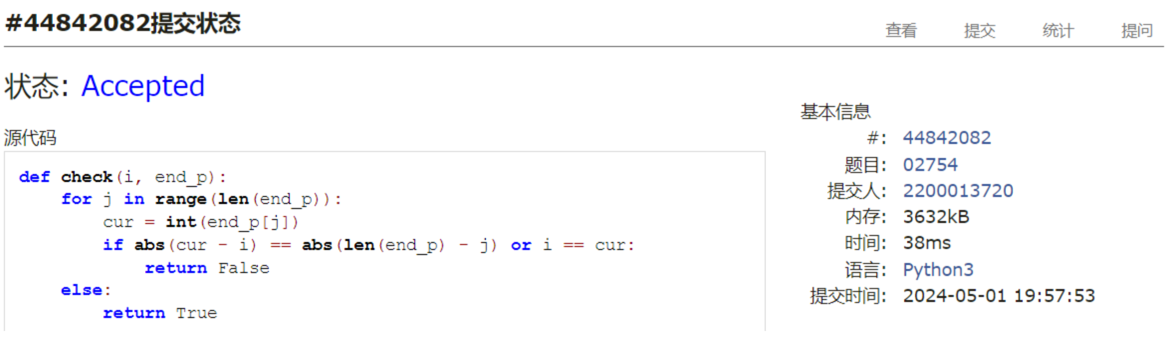
dfs, http://cs101.openjudge.cn/practice/02754/

思路：依据之前写的那段写了个简略版，把check看懂了就明白了，其实现在来看还蛮简单的）中途卡了一会是因为dfs利多加了个result的前置即dfs(n, end_p, result = [])导致所有递归的result被串一起了，最后直接在函数里定义result就好了，还是不太明白为什么会这样

代码

```
#
def check(i, end_p):
    for j in range(len(end_p)):
        cur = int(end_p[j])
        if abs(cur - i) == abs(len(end_p) - j) or i == cur:
            return False
    else:
        return True

def dfs(n, end_p):
    result = []
    if len(end_p) == n:
        result.append(int(end_p))
    else:
        for i in range(n):
            if check(i, end_p):
                result += dfs(n, end_p + str(i))
    return result

n = 8
result = dfs(n, '')
t = int(input())
for _ in range(t):
    print(result[int(input())-1] + 11111111)
```

代码运行截图 == （至少包含有"Accepted"） ==


#44842082提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def check(i, end_p):
    for j in range(len(end_p)):
        cur = int(end_p[j])
        if abs(cur - i) == abs(len(end_p) - j) or i == cur:
            return False
    else:
        return True
```

基本信息

#: 44842082
题目: 02754
提交人: 2200013720
内存: 3632kB
时间: 38ms
语言: Python3
提交时间: 2024-05-01 19:57:53

03151: Pots

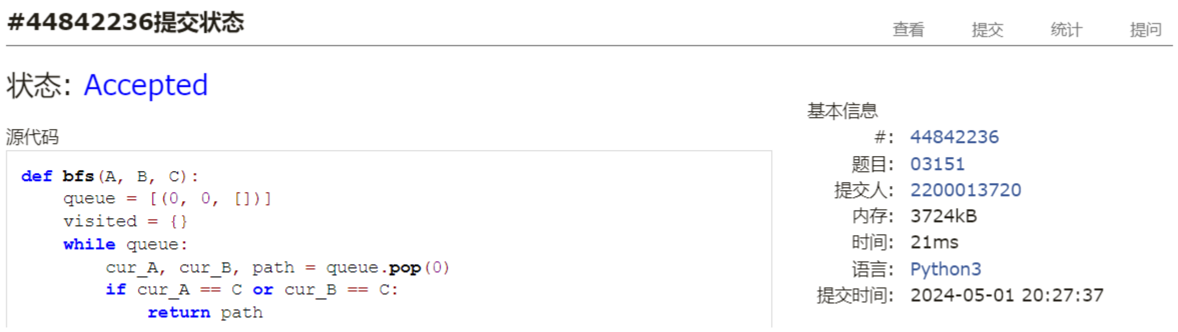
bfs, http://cs101.openjudge.cn/practice/03151/

思路：真是一场酣畅淋漓的bfs啊，其实不难，关键把visited有关的判断写好就行了，只不过写的时候忘了queue.pop(0)忘了写0导致想了半天，最后把0加上就AC了

代码

```
#
def bfs(A, B, C):
    queue = [(0, 0, [])]
    visited = {}
    while queue:
        cur_A, cur_B, path = queue.pop(0)
        if cur_A == C or cur_B == C:
            return path
        if (A, cur_B) not in visited:
            queue.append((A, cur_B, path + ['FILL(1)']))
            visited[(A, cur_B)] = 0
        if (cur_A, B) not in visited:
            queue.append((cur_A, B, path + ['FILL(2)']))
            visited[(cur_A, B)] = 0
        if (0, cur_B) not in visited:
            queue.append((0, cur_B, path + ['DROP(1)']))
            visited[(0, cur_B)] = 0
        if (cur_A, 0) not in visited:
            queue.append((cur_A, 0, path + ['DROP(2)']))
            visited[(cur_A, 0)] = 0
        if (min(A, cur_A + cur_B), max(0, cur_A + cur_B - A)) not in visited:
                queue.append((min(A, cur_A + cur_B), max(0, cur_A + cur_B - A), path
+ ['POUR(2,1)']))
                visited[(min(A, cur_A + cur_B), max(0, cur_A + cur_B - A))] = 0
            if (max(0, cur_A + cur_B - B), min(B, cur_A + cur_B)) not in visited:
                queue.append((max(0, cur_A + cur_B - B), min(B, cur_A + cur_B), path
+ ['POUR(1,2)']))
                visited[(max(0, cur_A + cur_B - B), min(B, cur_A + cur_B))] = 0

A, B, C = map(int,input().split())
result = bfs(A, B, C)
if result == None:
    print('impossible')
else:
    print(len(result))
    print('\n'.join(result))
```

代码运行截图 == （AC代码截图，至少包含有"Accepted"） ==


#44842236提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def bfs(A, B, C):
    queue = [(0, 0, [])]
    visited = {}
    while queue:
        cur_A, cur_B, path = queue.pop(0)
        if cur_A == C or cur_B == C:
            return path
```

基本信息

#:	44842236
题目:	03151
提交人:	2200013720
内存:	3724kB
时间:	21ms
语言:	Python3
提交时间:	2024-05-01 20:27:37

05907: 二叉树的操作

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

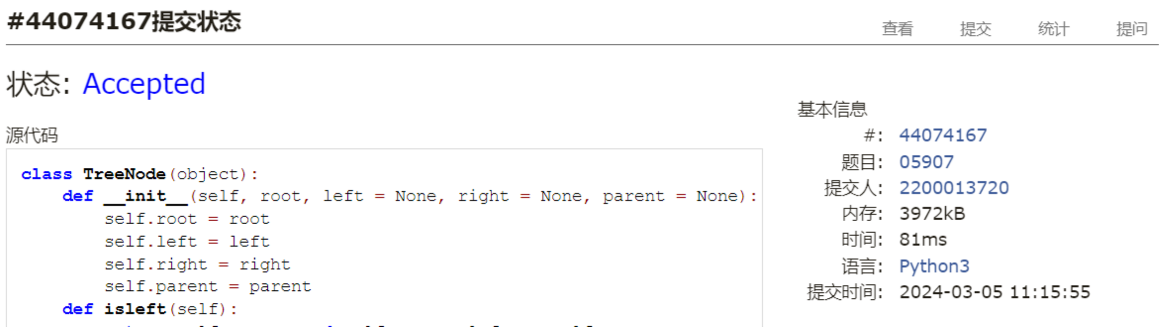
思路：大概看了下感觉困难在于交换节点Change函数的实现，查询挺简单的一直递归tree.left就行了。而且由于输入是按节点顺序来的，因此最开始可以之间先建一个全为空树节点的列表，然后随着输入更新就好了，这样也不会导致后续的输入污染了原来已经绑定的节点。把之前的AC代码搬过来了，这个Change果然比较复杂，不过也好理解。这道题由于输入的特殊性还是挺简单的

代码

```
#
class TreeNode(object):
    def __init__(self, root, left = None, right = None, parent = None):
        self.root = root
        self.left = left
        self.right = right
        self.parent = parent
    def isleft(self):
        return self.parent and self.parent.left == self
    def isright(self):
        return self.parent and self.parent.right == self
```

```python
def Change(tree1, tree2):
    newparent1 = tree1.parent
    newparent2 = tree2.parent
    if tree1.isleft() and tree2.isleft():
        tree1.parent = newparent2
        newparent2.left = tree1
        tree2.parent = newparent1
        newparent1.left = tree2
    elif tree1.isright() and tree2.isleft():
        tree1.parent = newparent2
        newparent2.left = tree1
        tree2.parent = newparent1
        newparent1.right = tree2
    elif tree1.isleft() and tree2.isright():
        tree1.parent = newparent2
        newparent2.right = tree1
        tree2.parent = newparent1
        newparent1.left = tree2
    elif tree1.isright() and tree2.isright():
        tree1.parent = newparent2
        newparent2.right = tree1
        tree2.parent = newparent1
        newparent1.right = tree2

def search(tree1):
    if tree1.left:
        return search(tree1.left)
    else:
        return tree1.root

t = int(input())
for _ in range(t):
    n,m = map(int,input().split())
    trees = [TreeNode(i) for i in range(n)]
    for i in range(n):
        root, left, right = map(int,input().split())
        if left != -1:
            trees[root].left = trees[left]
            trees[left].parent = trees[root]
        if right != -1:
            trees[root].right = trees[right]
            trees[right].parent = trees[root]
    for j in range(m):
        order = list(map(int,input().split()))
        if order[0] == 1:
            Change(trees[order[1]], trees[order[2]])
        elif order[0] == 2:
            print(search(trees[order[1]]))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44074167提交状态

状态: Accepted

源代码

```
class TreeNode(object):
    def __init__(self, root, left = None, right = None, parent = None):
        self.root = root
        self.left = left
        self.right = right
        self.parent = parent
    def isleft(self):
```

基本信息

#:	44074167
题目:	05907
提交人:	2200013720
内存:	3972kB
时间:	81ms
语言:	Python3
提交时间:	2024-03-05 11:15:55

18250: 冰阔落 I

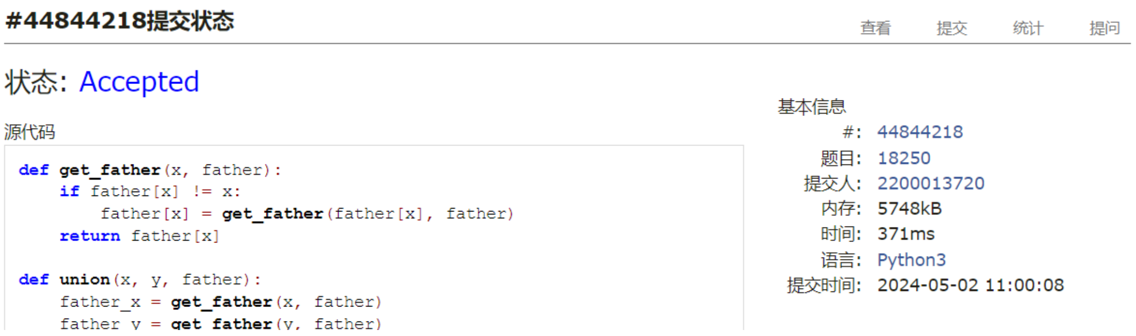
Disjoint set, http://cs101.openjudge.cn/practice/18250/

思路：并查集的思路，只是我一直以为字典内部已经排好序了所以没写result = sorted(coke)直接输出coke了，结果WA后想了半天，一看题解还得sorted一下，上网看了下发现set本身是无序的，但是输出时升序，所以我还是不太理解）

代码

```
#
def get_father(x, father):
    if father[x] != x:
        father[x] = get_father(father[x], father)
    return father[x]

def union(x, y, father):
    father_x = get_father(x, father)
    father_y = get_father(y, father)
    if not father_x == father_y:
        father[father_y] = father_x


while True:
    try:
        n,m = map(int,input().split())
        code = [i for i in range(n+1)]
        for _ in range(m):
            x,y = map(int,input().split())
            if get_father(x, code) == get_father(y, code):
                print('Yes')
            else:
                print('No')
                union(x, y, code)
        coke = set(get_father(i, code) for i in range(1,n+1))
        result = sorted(coke)
        print(len(result))
        print(*result)

    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44844218提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def get_father(x, father):
    if father[x] != x:
        father[x] = get_father(father[x], father)
    return father[x]

def union(x, y, father):
    father_x = get_father(x, father)
    father_y = get_father(y, father)
```

基本信息

#:	44844218
题目:	18250
提交人:	2200013720
内存:	5748kB
时间:	371ms
语言:	Python3
提交时间:	2024-05-02 11:00:08

05443: 兔子与樱花

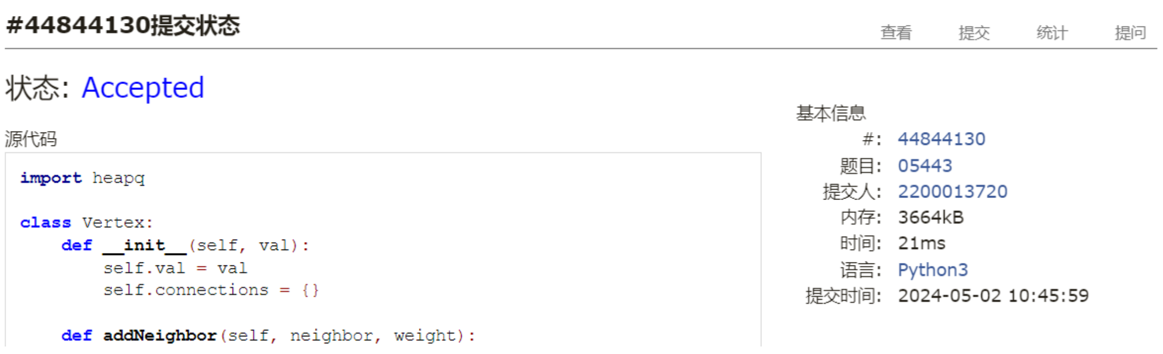
http://cs101.openjudge.cn/practice/05443/

思路：用上次作业的Dijkstra算法即可

代码

```python
#
import heapq

class Vertex:
    def __init__(self, val):
        self.val = val
        self.connections = {}

    def addNeighbor(self, neighbor, weight):
        self.connections[neighbor] = weight

class Graph:
    def __init__(self):
        self.ids = {}

    def addVertex(self, val):
        self.ids[val] = Vertex(val)

    def addEdge(self, fr, to, weight):
        if fr not in self.ids:
            self.addVertex(fr)
        if to not in self.ids:
            self.addVertex(to)
        self.ids[fr].addNeighbor(to, weight)

def find(graph, fr, to):
    visited = {fr:0}
    queue = [(0, [fr], fr)]
    heapq.heapify(queue)
    while queue:
        path_length, path, cur = heapq.heappop(queue)
        cur = graph.ids[cur]
        if cur.val == to:
            return path
        for c in cur.connections:
            new_length = cur.connections[c]
            if c not in visited or path_length + new_length < visited[c]:
                visited[c] = path_length + new_length
                heapq.heappush(queue, (path_length + new_length, path + ['->(' +
str(new_length) + ')->'+c], c))


Map = Graph()
for _ in range(int(input())):
    key = input()
    Map.addVertex(key)
for _ in range(int(input())):
    fr, to, weight = map(str,input().split())
    Map.addEdge(fr, to, int(weight))
    Map.addEdge(to, fr, int(weight))
for _ in range(int(input())):
    fr, to = map(str,input().split())
    result = find(Map, fr, to)
    print(''.join(result))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44844130提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

class Vertex:
    def __init__(self, val):
        self.val = val
        self.connections = {}

    def addNeighbor(self, neighbor, weight):
```

基本信息

#:	44844130
题目:	05443
提交人:	2200013720
内存:	3664kB
时间:	21ms
语言:	Python3
提交时间:	2024-05-02 10:45:59

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。==

感觉这次的作业就是对之前知识的回顾，把先前关于图、树和bfs、dfs等算法了解清楚就行了，并查集find和union两个函数一定义其实也就差不多了，确实比较简单（毕竟已经接触过了）。

五一还有好多事啊啊啊啊怎么感觉群里的大伙都在旅游谈恋爱啊啊啊啊啊