Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by 潘子轩、信科

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：最开始把题意理解错了，看了眼群才知道是数联通数

代码

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

m = [[0 for __ in range(10)] for _ in range(10)]
ans = 0
for i in range(10):
    s = input()
    for j in range(10):
        if s[j] == '.':
            m[i][j] = True
        else:
            m[i][j] = False


def dfs(x, y):
    m[x][y] = False
    for ii in range(4):
        x1 = x + dx[ii]
        y1 = y + dy[ii]
        if 0 <= x1 < 10 and 0 <= y1 < 10 and m[x1][y1]:
            dfs(x1, y1)
```

```
    for i in range(10):
        for j in range(10):
            if m[i][j]:
                ans += 1
                dfs(i, j)
    print(ans)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

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

m = [[0 for __ in range(10)] for _ in range(10)]
ans = 0
for i in range(10):
    s = input()
    for j in range(10):
        if s[j] == '.':
            m[i][j] = True
        else:
            m[i][j] = False


def dfs(x, y):
    m[x][y] = False
    for ii in range(4):
        x1 = x + dx[ii]
        y1 = y + dy[ii]
        if 0 <= x1 < 10 and 0 <= y1 < 10 and m[x1][y1]:
            dfs(x1, y1)


for i in range(10):
    for j in range(10):
        if m[i][j]:
            ans += 1
            dfs(i, j)
print(ans)
```

02754: 八皇后

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

思路：排列组合库真的很好用

代码

```
import itertools

p = [i for i in range(8)]
ans = []
for a in itertools.permutations(p, 8):
    found = True
    for b in itertools.permutations(p, 2):
        if b[0] - b[1] == a[b[0]] - a[b[1]] or b[0] - b[1] == a[b[1]]
- a[b[0]]:
            found = False
            break
    if found:
        ans.append(a)

n = int(input())
for _ in range(n):
    od = int(input())
    print(''.join(map(str, [x + 1 for x in ans[od - 1]])))
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
import itertools

p = [i for i in range(8)]
ans = []
for a in itertools.permutations(p, 8):
    found = True
    for b in itertools.permutations(p, 2):
        if b[0] - b[1] == a[b[0]] - a[b[1]] or b[0] - b[1] == a[b[1]] -
            found = False
            break
    if found:
        ans.append(a)

n = int(input())
for _ in range(n):
    od = int(input())
    print(''.join(map(str, [x + 1 for x in ans[od - 1]])))
```

03151: Pots

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

思路：bfs

代码

```
from collections import deque


class Node:
    def __init__(self, _a, _b, _pos, _father):
```

```
        self.a = _a
        self.b = _b
        self.father = _father
        self.pos = _pos
        self.searched = False


a0, b0, c = map(int, input().split())
root = Node(0, 0, 0, None)
q = deque()
q.append(root)
found = False
searched = []
while q:
    current_node = q[0]

    if (current_node.a, current_node.b) in searched:
        q.popleft()
        continue

    if current_node.a == c or current_node.b == c:
        found = True
        ans_list = []
        while current_node.father:
            ans_list.append(current_node.pos)
            current_node = current_node.father
        print(len(ans_list))
        for item in ans_list[::-1]:
            if item == 0:
                print('FILL(1)')
            elif item == 1:
                print('FILL(2)')
            elif item == 2:
                print('DROP(1)')
            elif item == 3:
                print('DROP(2)')
```

```
            elif item == 4:
                print('POUR(1,2)')
            elif item == 5:
                print('POUR(2,1)')
        break

    searched.append((current_node.a, current_node.b))
    q.popleft()
    q.append(Node(a0, current_node.b, 0, current_node))
    q.append(Node(current_node.a, b0, 1, current_node))
    q.append(Node(0, current_node.b, 2, current_node))
    q.append(Node(current_node.a, 0, 3, current_node))
    if current_node.a > b0 - current_node.b:
        q.append(Node(current_node.a - (b0 - current_node.b), b0, 4,
current_node))
    else:
        q.append(Node(0, current_node.b + current_node.a, 4,
current_node))
    if current_node.b > a0 - current_node.a:
        q.append(Node(a0, current_node.b - (a0 - current_node.a), 5,
current_node))
    else:
        q.append(Node(current_node.a + current_node.b, 0, 5,
current_node))

if not found:
    print('impossible')
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
from collections import deque


class Node:
    def __init__(self, _a, _b, _pos, _father):
        self.a = _a
        self.b = _b
        self.father = _father
        self.pos = _pos
        self.searched = False
```

05907: 二叉树的操作

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

思路：要考虑两个节点的父节点相同的情况

代码

```
class Node:
    def __init__(self, _v):
        self.v = _v
        self.lf = None
        self.rt = None
        self.p = None


t = int(input())
for _ in range(t):
```

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

tree_list = {}
for __ in range(n):
    s, l, r = map(int, input().split())
    tree_list[s] = (l, r)
root_num = 0
root = Node(root_num)
node_list = {root_num: root}


def build_tree(node):
    if tree_list[node.v][0] != -1:
        new_node = Node(tree_list[node.v][0])
        new_node.p = node
        node_list[tree_list[node.v][0]] = new_node
        build_tree(new_node)
        node.lf = new_node
    if tree_list[node.v][1] != -1:
        new_node = Node(tree_list[node.v][1])
        new_node.p = node
        node_list[tree_list[node.v][1]] = new_node
        build_tree(new_node)
        node.rt = new_node


build_tree(root)

for __ in range(m):
    sign_list = list(map(int, input().split()))
    if sign_list[0] == 1:
        node1 = node_list[sign_list[1]]
        node2 = node_list[sign_list[2]]
        if node1.p == node2.p:
            node1.p.lf, node1.p.rt = node1.p.rt, node1.p.lf
        else:
```

```
                if node2 == node2.p.lf:
                    node2.p.lf = node1
                elif node2 == node2.p.rt:
                    node2.p.rt = node1
                if node1 == node1.p.lf:
                    node1.p.lf = node2
                elif node1 == node1.p.rt:
                    node1.p.rt = node2
                node2.p, node1.p = node1.p, node2.p
            elif sign_list[0] == 2:
                current_node = node_list[sign_list[1]]
                while current_node.lf:
                    current_node = current_node.lf
                print(current_node.v)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
class Node:
    def __init__(self, _v):
        self.v = _v
        self.lf = None
        self.rt = None
        self.p = None


t = int(input())
for _ in range(t):
    n, m = map(int, input().split())

    tree_list = {}
    for __ in range(n):
        s, l, r = map(int, input().split())
        tree_list[s] = (l, r)
    root_num = 0
    root = Node(root_num)
    node_list = {root_num: root}
```

18250: 冰阔落 I

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

思路：并查集

代码

```
class UnionFind:
```

```
    def __init__(self, _n):
        self.f = list(range(0, _n + 1))

    def find(self, num):
        if self.f[num] != num:
            self.f[num] = self.find(self.f[num])
        return self.f[num]

    def union(self, a, b):
        root_a = self.find(a)
        root_b = self.find(b)
        self.f[root_b] = root_a


while True:
    try:
        n, m = map(int, input().split())
        uf = UnionFind(n)
        for _ in range(m):
            x, y = map(int, input().split())
            if uf.find(x) == uf.find(y):
                print('Yes')
            else:
                uf.union(x, y)
                print('No')
        ans = [i for i in range(1, n+1) if uf.f[i] == i]
        print(len(ans))
        print(' '.join(map(str, 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

```
class UnionFind:
    def __init__(self, _n):
        self.f = list(range(0, _n + 1))

    def find(self, num):
        if self.f[num] != num:
            self.f[num] = self.find(self.f[num])
        return self.f[num]

    def union(self, a, b):
        root_a = self.find(a)
        root_b = self.find(b)
        self.f[root_b] = root_a
```

05443: 兔子与樱花

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

思路：Floyd算法

代码

```
import sys

p = int(input())
place_dic = {}
id_dic = {}
for i in range(p):
    s = input()
    place_dic[s] = i
```

```
    id_dic[i] = s
neighbour_matrix = [[sys.maxsize for __ in range(p)] for _ in
range(p)]
pre_matrix = [[-1 for __ in range(p)] for _ in range(p)]
for i in range(p):
    neighbour_matrix[i][i] = 0
for _ in range(int(input())):
    place_1, place_2, weight = input().split()
    weight = int(weight)
    id_1, id_2 = place_dic[place_1], place_dic[place_2]
    neighbour_matrix[id_1][id_2] = weight
    neighbour_matrix[id_2][id_1] = weight
    pre_matrix[id_1][id_2] = id_1
    pre_matrix[id_2][id_1] = id_2
original_matrix = [[0 for __ in range(p)] for _ in range(p)]
for i in range(p):
    for j in range(p):
        original_matrix[i][j] = neighbour_matrix[i][j]

for mid_point in range(p):
    for i in range(p):
        for j in range(p):
            if neighbour_matrix[i][mid_point] +
neighbour_matrix[mid_point][j] < neighbour_matrix[i][j]:
                neighbour_matrix[i][j] = neighbour_matrix[i]
[mid_point] + neighbour_matrix[mid_point][j]
                pre_matrix[i][j] = pre_matrix[mid_point][j]

r = int(input())
for _ in range(r):
    from_place, to_place = input().split()
    from_id, to_id = place_dic[from_place], place_dic[to_place]
    path = [to_id]
    while to_id != from_id:
        to_id = pre_matrix[from_id][to_id]
        path.append(to_id)
```

```
        for i in range(len(path) - 1, 0, -1):
            print(f"{id_dic[path[i]]}->({original_matrix[path[i]][path[i - 1]]})->", end='')
        print(to_place)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
import sys

p = int(input())
place_dic = {}
id_dic = {}
for i in range(p):
    s = input()
    place_dic[s] = i
    id_dic[i] = s
neighbour_matrix = [[sys.maxsize for __ in range(p)] for _ in range(p)]
pre_matrix = [[-1 for __ in range(p)] for _ in range(p)]
for i in range(p):
    neighbour_matrix[i][i] = 0
for _ in range(int(input())):
    place_1, place_2, weight = input().split()
    weight = int(weight)
    id_1, id_2 = place_dic[place_1], place_dic[place_2]
    neighbour_matrix[id_1][id_2] = weight
    neighbour_matrix[id_2][id_1] = weight
    pre_matrix[id_1][id_2] = id_1
    pre_matrix[id_2][id_1] = id_2
original_matrix = [[0 for __ in range(p)] for _ in range(p)]
for i in range(p):
    for j in range(p):
        original_matrix[i][j] = neighbour_matrix[i][j]
```

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。

这次作业主要涉及图算法，练习了dfs和bfs，同时学习了几种最短路径的算法。