

Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Compiled by 天幕 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

```
1  directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
2  l = [list(input()) for _ in range(10)]
3  unchecked = [[True] * 10 for _ in range(10)]
4  ans = 0
5
6  def dfs(j, i):
7      global l, unchecked, ans
8      if 0 <= i < 10 and 0 <= j < 10 and unchecked[j][i]:
9          unchecked[j][i] = False
10         if l[j][i] == '-':
11             return 0
12         else:
13             for direction in directions:
14                 y, x = direction
15                 dfs(j + y, i + x)
```

```

16         return 1
17     return 0
18
19 for j in range(10):
20     for i in range(10):
21         if unchecked[j][i]:
22             ans += dfs(j, i)
23
24 print(ans)

```

代码运行截图

状态: Accepted

源代码

```

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

```

1 def generate(l, list, layer):
2     if layer == 8:
3         l += [int(''.join([str(x + 1) for x in list]))]
4         return 0
5     for i in range(8):
6         list[layer] = i
7         for j in range(layer):
8             if (list[j] == i) or (abs(i - list[j]) == layer - j):
9                 break
10        else:
11            generate(l, list, layer+1)
12    l = []
13    generate(l, [-1] * 8, 0)
14    for _ in range(int(input())):
15        print(l[int(input()) - 1])

```

代码运行截图

状态: Accepted

源代码

```

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

```

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

```

```

3
4 def bfs(l, depth = 0):
5     global a, b, c, unvisited
6     nextl = []
7     for traceablepos in l:
8         x, y, string = traceablepos
9         if 0 <= x <= a and 0 <= y <= b and unvisited[y][x]:
10             if x == c or y == c:
11                 return str(depth) + string
12             unvisited[y][x] = False
13             if x < a:
14                 nextl.append((a, y, string + '\nFILL(1)'))
15             if y < b:
16                 nextl.append((x, b, string + '\nFILL(2)'))
17             if x > 0:
18                 nextl.append((0, y, string + '\nDROP(1)'))
19                 if y < b:
20                     remain = y + x - b
21                     if remain >= 0:
22                         nextl.append((remain, b, string + '\nPOUR(1,2)'))
23                     else:
24                         nextl.append((0, x + y, string + '\nPOUR(1,2)'))
25             if y > 0:
26                 nextl.append((x, 0, string + '\nDROP(2)'))
27                 if x < a:
28                     remain = x + y - a
29                     if remain >= 0:
30                         nextl.append((a, remain, string + '\nPOUR(2,1)'))
31                     else:
32                         nextl.append((x + y, 0, string + '\nPOUR(2,1)'))
33         if not nextl:
34             return 'impossible'
35         return bfs(nextl, depth + 1)
36
37 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

```

1  from __future__ import annotations
2
3  class Node:
4      _ID = 0
5      NodeID:int
6      name:str
7      sub:list
8      parent:Node
9
10     def __init__(self, name, sub, parent=None):
11         self.NodeID = Node._ID
12         Node._ID += 1
13         self.name = name
14         self.sub = sub
15         self.parent = parent
16
17     class BiTree(dict):
18         root:Node
19         def __init__(self):
20             self.parent = dict()
21             self.root = None
22
23         def findParent(self, t):
24             if t not in self.parent:
25                 return None
26             return self.parent[t]
27
28         def getOrCreate(self, nodename):
29             if nodename == "-1":
30                 return False
31             if nodename not in self:
32                 self[nodename] = Node(nodename, [False, False])
33             return self[nodename]
34
35         def add(self, t, l, r):
36             if t not in self:
37                 neonode = Node(t, [self.getOrCreate(l), self.getOrCreate(r)])
38                 self[t] = neonode
39             else:
40                 neonode = self[t]
41                 neonode.sub = [self.getOrCreate(l), self.getOrCreate(r)]
42             if not self.root:
43                 self.root = neonode
44             if l != "-1":
45                 self.parent[l] = neonode
46             if r != "-1":
47                 self.parent[r] = neonode
48
49         def initTree(self):
50             for nodename in self:
51                 if nodename == '0':
52                     continue
53                 node:Node = self[nodename]
54                 node.parent = self.parent[nodename]
55

```

```

56     def exchange(self, i, j):
57         inode:Node = self[i]
58         jnode:Node = self[j]
59         if inode.parent == jnode.parent:
60             inode.parent.sub.reverse()
61         else:
62             for i in range(2):
63                 if inode.parent.sub[i] == inode:
64                     inode.parent.sub[i] = jnode
65                 if jnode.parent.sub[i] == jnode:
66                     jnode.parent.sub[i] = inode
67             inode.parent, jnode.parent = jnode.parent, inode.parent
68
69     def _findlefttest(self, node:Node):
70         if node.sub[0]:
71             return self._findlefttest(node.sub[0])
72         else:
73             return node.name
74
75     def findlefttest(self, i):
76         return self._findlefttest(self[i])
77
78 for _ in range(int(input())):
79     n, m = map(int, input().split())
80     myTree = BiTree()
81     for __ in range(n):
82         myTree.add(*input().split())
83     myTree.initTree()
84     for ___ in range(m):
85         codein = list(input().split())
86         if codein[0] == "1":
87             myTree.exchange(codein[1], codein[2])
88         else:
89             print(myTree.findlefttest(codein[1]))

```

代码运行截图

状态: Accepted

源代码

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

def findlefttest(self, i):
    return self._findlefttest(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.findlefttest(codein[1]))

```

©2002-2022 POJ 京ICP备20010980号-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

```

1 class DisjointSet(object):
2     father_dict:dict
3     def __init__(self, l):

```

```

4         self.father_dict = {}
5         for x in l:
6             self.father_dict[x] = x
7
8     def find(self, x):
9         if self.father_dict[x] == x:
10             return x
11         else:
12             self.father_dict[x] = self.find(self.father_dict[x])
13             return self.father_dict[x]
14
15     def union(self, x, y):
16         px = self.find(x)
17         py = self.find(y)
18         if px != py:
19             self.father_dict[py] = px
20             return 'No\n'
21         else:
22             return 'Yes\n'
23
24     def getUnions(self):
25         for x in self.father_dict:
26             self.find(x)
27         l = sorted(list(set(self.father_dict.values())))
28         return str(len(l)) + '\n' + ' '.join([str(x) for x in l])
29
30 while 1:
31     try:
32         n, m = map(int, input().split())
33         ds = DisjointSet(range(1, n + 1))
34         ans = ''
35         for _ in range(m):
36             ans += ds.union(*map(int, input().split()))
37         print(ans + ds.getUnions())
38     except EOFError:
39         break

```

代码运行截图

状态: Accepted

源代码

```
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
```

©2002-2022 POJ 京ICP备20010980号-1

05443: 兔子与樱花

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

思路：和走山路一个思路。

代码

```
1  from heapq import *
2
3  inf = float('inf')
4
5  p = int(input())
6  distance = dict()
7  l = [input() for _ in range(p)]
8  for i in l:
9      distance[i] = dict()
10     for j in l:
11         distance[i][j] = inf
12
13  q = int(input())
14  for _ in range(q):
15      i, j, dis = input().split()
16      distance[i][j] = distance[j][i] = int(dis)
17
18  r = int(input())
19
20  def to(traceablepos, target):
21      global l, distance
22      dis, pos, info = traceablepos
23      return (dis + distance[pos][target], target, info + '->({})->
24      {}'.format(distance[pos][target], target))
25
26  def bfs(origin, terminal):
27      global distance
28
29      nearest = dict()
30      for i in l:
31          nearest[i] = inf
32
33      ways = [(0, origin, origin)]
34
35      while ways:
36          traceablepos = heappop(ways)
37          dis, pos, info = traceablepos
38          if pos == terminal:
39              return info
40          for target in l:
41              if distance[pos][target] != inf and nearest[target] > dis +
42              distance[pos][target]:
43                  nearest[target] = dis + distance[pos][target]
44                  heappush(ways, to(traceablepos, target))
45          raise FileNotFoundError
46
47  for _ in range(r):
48      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 + '->({})->{}'.format(pos, 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()))
```

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。