

Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Compiled by 武昱达 23工

编程环境

操作系统: Win 11

Python编程环境: PyCharm

1. 题目

28170: 算鹰

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

思路:

代码

```
1 visited=[[False for _ in range(10)] for _ in range(10)]
2 board=[list(input()) for _ in range(10)]
3 # 这里dfs函数的功能是探出所有连通区域，并打上标记
4 steps=[(0,1),(0,-1),(1,0),(-1,0)]
5 def dfs(x,y):
6     visited[x][y]=True
7     for dx,dy in steps:
8         nx,ny=x+dx,y+dy
9         if 0<=nx<10 and 0<=ny<10 and not visited[nx][ny] and board[nx]
[ny]=='.':
10             dfs(nx,ny)
11 cnt=0
12 for x in range(10):
13     for y in range(10):
14         if board[x][y]=='-':
15             visited[x][y]=True
16 for x in range(10):
17     for y in range(10):
18         if board[x][y]=='.' and not visited[x][y]:
19             cnt+=1
20             dfs(x,y)
21 print(cnt)
```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
visited=[[False for _ in range(10)] for _ in range(10)]
board=[list(input()) for _ in range(10)]
# 这里dfs函数的功能是探出所有连通区域,并打上标记
steps=[(0,1),(0,-1),(1,0),(-1,0)]
def dfs(x,y):
    visited[x][y]=True
    for dx,dy in steps:
        nx,ny=x+dx,y+dy
        if 0<=nx<10 and 0<=ny<10 and not visited[nx][ny] and board[nx][ny]!='-':
            dfs(nx,ny)

cnt=0
for x in range(10):
    for y in range(10):
        if board[x][y]=='-':
            visited[x][y]=True
for x in range(10):
    for y in range(10):
        if board[x][y]=='.' and not visited[x][y]:
            cnt+=1
            dfs(x,y)
print(cnt)
```

基本信息

#: 44866557
题目: 28170
提交人: 23n2300011119 (武)
内存: 3732kB
时间: 21ms
语言: Python3
提交时间: 2024-05-05 10:51:14

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

02754: 八皇后

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

思路:

代码

```
1 cols=[i for i in range(8)]
2 res=[]
3 def Queen(path,choices,main_diag,vice_diag):
4     #退出条件
5     if len(path)==8:
6         temp=[str(j+1) for j in path]
7         res.append("".join(temp))
8         return
9     #下一步操作
10    for j in choices:
11        #剪枝操作
12        if j in path or j+len(path) in vice_diag or j-len(path) in
main_diag:
13            continue
14        new_path = path + [j]
15        new_main_diag = main_diag.copy()
16        new_vice_diag = vice_diag.copy()
17        new_main_diag.add(j - len(path))
18        new_vice_diag.add(j + len(path))
19        #下一层递归
20        Queen(new_path, choices, new_main_diag, new_vice_diag)
21 main_diags=set()
22 vice_diags=set()
23 #直接调用函数,没有返回值
24 Queen([],cols,main_diags,vice_diags)
25 lst=[]
26 t=int(input())
27 for _ in range(t):
```

```

1 from collections import deque
2 def bfs(a, b, c):
3     queue = deque([(0, 0, [])])
4     visited = set()
5     while queue:
6         x, y, steps = queue.popleft()
7         if x == c or y == c:
8             return steps
9         operations = [(a, y, 'FILL(1)'), (x, b, 'FILL(2)'), (0, y, 'DROP(1)'),
10                      (x, 0, 'DROP(2)'),
11                      (max(0, x - (b - y)), min(b, y + x), 'POUR(1,2)'),
12                      (min(a, x + y), max(0, y - (a - x)), 'POUR(2,1)')]
13         for nx, ny, op in operations:
14             if (nx, ny) not in visited:
15                 visited.add((nx, ny))
16                 queue.append((nx, ny, steps + [op]))

```

```

16         return None
17
18     A, B, C = map(int, input().strip().split())
19     result = bfs(A, B, C)
20     if result is None:
21         print("impossible")
22     else:
23         print(len(result))
24         for step in result:
25             print(step)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44867886提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44867886

题目: 03151

提交人: 23n2300011119 (武)

内存: 3684kB

时间: 22ms

语言: Python3

提交时间: 2024-05-05 12:20:45

源代码

```

from collections import deque
def bfs(a, b, c):
    queue = deque([(0, 0, [])])
    visited = set()
    while queue:
        x, y, steps = queue.popleft()
        if x == c or y == c:
            return steps
        operations = [(a, y, 'FILL(1)'), (x, b, 'FILL(2)'), (0, y, 'DROP(1)'),
                     (max(0, x - (b - y)), min(b, y + x), 'POUR(1,2)'),
                     (min(a, x + y), max(0, y - (a - x)), 'POUR(2,1)')]
        for nx, ny, op in operations:
            if (nx, ny) not in visited:
                visited.add((nx, ny))
                queue.append((nx, ny, steps + [op]))
    return None

A, B, C = map(int, input().strip().split())
result = bfs(A, B, C)
if result is None:
    print("impossible")
else:
    print(len(result))
    for step in result:
        print(step)

```

05907: 二叉树的操作

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

思路:

代码

```

1 class TreeNode:
2     def __init__(self, val):
3         self.val = val
4         self.left = None
5         self.right = None
6         self.parent = None
7
8     def is_left_to(self, parent):
9         return parent.left == self
10
11     def _exchange_parent(self, other):
12         self.parent, other.parent = other.parent, self.parent
13         return other.parent, self.parent
14
15     def exchange(self, other):
16         parent_1, parent_2 = self._exchange_parent(other)

```

```

17     flag_1, flag_2=None, None
18
19     if self.is_left_to(parent_1): flag_1=True
20     else: flag_1=False
21
22     if other.is_left_to(parent_2): flag_2=True
23     else: flag_2=False
24
25     if flag_1: parent_1.left = other
26     else: parent_1.right=other
27
28     if flag_2: parent_2.left=self
29     else: parent_2.right=self
30
31     def find_left_most(self):
32         if self.left==None: return self
33         return self.left.find_left_most()
34
35 for _ in range(t:=int(input())):
36     n,m=map(int, input().split())
37     tree={}
38
39     for _ in range(n):
40         val, left, right=map(int, input().split())
41         if val not in tree: tree[val]=TreeNode(val)
42         if left not in tree: tree[left]=TreeNode(left)
43         if right not in tree: tree[right]=TreeNode(right)
44         if left!=-1:
45             tree[val].left=tree[left]
46             tree[left].parent=tree[val]
47         if right!=-1:
48             tree[val].right=tree[right]
49             tree[right].parent=tree[val]
50
51     for _ in range(m):
52         if (raw:=input())[0]=='1':
53             type,x,y=map(int, raw.split())
54             tree[x].exchange(tree[y])
55         else:
56             type, val=map(int, raw.split())
57             print(tree[val].find_left_most().val)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

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

    def is_left_to(self, parent):
        return parent.left==self

    def _exchange_parent(self, other):
        self.parent, other.parent=other.parent, self.parent
        return other.parent, self.parent

    def exchange(self, other):
        parent_1, parent_2=self._exchange_parent(other)
        flag_1, flag_2=None, None

        if self.is_left_to(parent_1): flag_1=True
        else: flag_1=False

        if other.is_left_to(parent_2): flag_2=True
        else: flag_2=False

        if flag_1: parent_1.left = other
        else: parent_1.right=other

        if flag_2: parent_2.left=self
        else: parent_2.right=self

    def find_left_most(self):
        if self.left==None: return self
        return self.left.find_left_most()
```

基本信息

#: 44865971
题目: 05907
提交人: 23n2300011119 (武)
内存: 3992kB
时间: 82ms
语言: Python3
提交时间: 2024-05-05 10:19:10

18250: 冰阔落 I

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

思路:

不能按秩合并。在报满杯编号时会出错。

代码

```
1 class DisjointSet:
2     # 用index作为每个元素的储存位置。
3     def __init__(self, n):
4         self.parent=[i for i in range(n+1)]
5
6     def find(self, x): # find方法的作用是寻找元素x的代表元素
7         if self.parent[x]!=x:
8             # 注意，在递归地寻找父元素时，每一步操作并不浪费。
9             # 我们递归地把跨越两层的路径压缩成跨越1层路径，这样能有效减少后续递归层数。
10            self.parent[x] = self.find(self.parent[x])
11            return self.parent[x]
12
13     def union(self, x, y):
14         root_x=self.find(x)
15         root_y=self.find(y)
16         if root_x==root_y:
17             return
18         self.parent[root_y]=root_x
19
20
21 while True:
22     try:
23         n,m=map(int,input().split())
```

```

24     djs=DisjointSet(n)
25     for _ in range(m):
26         a,b=map(int,input().split())
27         if djs.find(a)!=djs.find(b):
28             djs.union(a,b)
29             print('No')
30         else:print('Yes')
31     cnt,stack=0,[]
32     for i in range(1,n+1):
33         if djs.parent[i]==i:
34             stack.append(i)
35             cnt+=1
36     print(cnt)
37     print(*stack)
38 except: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 DisjointSet:
    # 用index作为每个元素的储存位置。
    def __init__(self, n):
        self.parent=[i for i in range(n+1)]

    def find(self, x): # find方法的作用是寻找元素x的代表元素
        if self.parent[x]!=x:
            # 注意,在递归地寻找父元素时,每一步操作并不浪费。
            # 我们递归地把跨越两层的路径压缩成跨越1层路径,这样能有效减少后续递归层
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]

    def union(self,x,y):
        root_x=self.find(x)
        root_y=self.find(y)
        if root_x==root_y:
            return
        self.parent[root_y]=root_x

while True:
    try:
        n,m=map(int,input().split())
        djs=DisjointSet(n)
        for _ in range(m):
            a,b=map(int,input().split())
            if djs.find(a)!=djs.find(b):
                djs.union(a,b)
                print('No')
            else:print('Yes')
        cnt,stack=0,[]
        for i in range(1,n+1):
            if djs.parent[i]==i:

```

基本信息

#: 44873351
 题目: 18250
 提交人: 23n2300011119 (武)
 内存: 6012kB
 时间: 399ms
 语言: Python3
 提交时间: 2024-05-05 19:01:51

05443: 兔子与樱花

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

思路:

代码

```

1 import heapq
2 def dijkstra(adjacency, start):
3     distances = {vertex: float('infinity') for vertex in adjacency}
4     previous = {vertex: None for vertex in adjacency}
5     distances[start] = 0
6     pq = [(0, start)]
7
8     while pq:

```

```

9         current_distance, current_vertex = heapq.heappop(pq)
10        if current_distance > distances[current_vertex]:
11            continue
12
13        for neighbor, weight in adjacency[current_vertex].items():
14            distance = current_distance + weight
15            if distance < distances[neighbor]:
16                distances[neighbor] = distance
17                previous[neighbor] = current_vertex
18                heapq.heappush(pq, (distance, neighbor))
19
20        return distances, previous
21
22    def shortest_path_to(adjacency, start, end):
23        distances, previous = dijkstra(adjacency, start)
24        path = []
25        current = end
26        while previous[current] is not None:
27            path.insert(0, current)
28            current = previous[current]
29        path.insert(0, start)
30        return path, distances[end]
31
32    P = int(input())
33    places = {input().strip() for _ in range(P)}
34
35    Q = int(input())
36    graph = {place: {} for place in places}
37    for _ in range(Q):
38        src, dest, dist = input().split()
39        dist = int(dist)
40        graph[src][dest] = dist
41        graph[dest][src] = dist
42
43    R = int(input())
44    requests = [input().split() for _ in range(R)]
45
46    for start, end in requests:
47        if start == end:
48            print(start)
49            continue
50
51        path, total_dist = shortest_path_to(graph, start, end)
52        output = ""
53        for i in range(len(path) - 1):
54            output += f"{path[i]}->({graph[path[i]][path[i+1]]})->"
55        output += f"{end}"
56        print(output)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```
import heapq

def dijkstra(adjacency, start):
    distances = {vertex: float('infinity') for vertex in adjacency}
    previous = {vertex: None for vertex in adjacency}
    distances[start] = 0
    pq = [(0, start)]

    while pq:
        current_distance, current_vertex = heapq.heappop(pq)
        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in adjacency[current_vertex].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous[neighbor] = current_vertex
                heapq.heappush(pq, (distance, neighbor))

    return distances, previous

def shortest_path_to(adjacency, start, end):
    distances, previous = dijkstra(adjacency, start)
    path = []
    current = end
    while previous[current] is not None:
        path.insert(0, current)
        current = previous[current]
    path.insert(0, start)
```

基本信息

#: 44872951
题目: 05443
提交人: 23n2300011119 (武)
内存: 3688kB
时间: 20ms
语言: Python3
提交时间: 2024-05-05 18:27:42

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。

时间长不写手生，写了小一天才写完。

题目中规中矩，但是又发现了两个问题：

一是抽象数据结构处理得不好，地址以及浅拷贝深拷贝等问题常常搞得很混乱。

二是写代码习惯问题。除了一些经典的结构，尽量避免使用类，因为类是面对一个确定需求来写的，属于一劳永逸型，但是做题要追求敲代码的速度，写类的时间成本太高，而且会遇到问题一中描述的情况。

(一道题写了一百多行血的教训)