Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied 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

```
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)
```

代码运行截图 （至少包含有"Accepted"）

#44866557提交状态

查看 提交 统计 提问

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][n
            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

English 帮助 关于

02754: 八皇后

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

思路:

代码

```
cols=[i for i in range(8)]
res=[]
def Queen(path,choices,main_diag,vice_diag):
    #退出条件
    if len(path)==8:
        temp=[str(j+1) for j in path]
        res.append("".join(temp))
        return
    #下一步操作
    for j in choices:
        #剪枝操作
        if j in path or j+len(path) in vice_diag or j-len(path) in
main_diag:
            continue
        new_path = path + [j]
        new_main_diag = main_diag.copy()
        new_vice_diag = vice_diag.copy()
        new_main_diag.add(j - len(path))
        new_vice_diag.add(j + len(path))
        #下一层递归
        Queen(new_path, choices, new_main_diag, new_vice_diag)
main_diags=set()
vice_diags=set()
#直接调用函数，没有返回值
Queen([],cols,main_diags,vice_diags)
lst=[]
t=int(input())
for _ in range(t):
```

```
    lst.append(res[int(input())-1])
for i in lst:
    print(i)
```

代码运行截图 （至少包含有"Accepted"）

#44821875提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
cols=[i for i in range(8)]
res=[]
def Queen(path,choices,main_diag,vice_diag):
    #退出条件
    if len(path)==8:
        temp=[str(j+1) for j in path]
        res.append("".join(temp))
        return
    #下一步操作
    for j in choices:
        #剪枝操作
        if j in path or j+len(path) in vice_diag or j-len(path) in main_
            continue
        new_path = path + [j]
        new_main_diag = main_diag.copy()
        new_vice_diag = vice_diag.copy()
        new_main_diag.add(j - len(path))
        new_vice_diag.add(j + len(path))
        #下一层递归
        Queen(new_path, choices, new_main_diag, new_vice_diag)
main_diags=set()
vice_diags=set()
#直接调用函数，没有返回值
Queen([],cols,main_diags,vice_diags)
lst=[]
t=int(input())
for _ in range(t):
    lst.append(res[int(input())-1])
for i in lst:
    print(i)
```

基本信息

#: 44821875
题目: 02754
提交人: 23n2300011119 (武)
内存: 3640kB
时间: 33ms
语言: Python3
提交时间: 2024-04-28 11:54:03

03151: Pots

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

思路:

对于一般题目，能不要用类来模拟就不要用，地址等会异常抽象和麻烦。

类在一些特定的结构下，如树，图等比较好用。

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)'),
(x, 0, 'DROP(2)'),
                      (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)
```

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

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)
```

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

05907: 二叉树的操作

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

思路:

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()

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

    for _ in range(n):
        val,left,right=map(int,input().split())
        if val not in tree:tree[val]=TreeNode(val)
        if left not in tree:tree[left]=TreeNode(left)
        if right not in tree:tree[right]=TreeNode(right)
        if left!=-1:
            tree[val].left=tree[left]
            tree[left].parent=tree[val]
        if right!=-1:
            tree[val].right=tree[right]
            tree[right].parent=tree[val]

    for _ in range(m):
        if (raw:=input())[0]=='1':
            type,x,y=map(int,raw.split())
            tree[x].exchange(tree[y])
        else:
            type,val=map(int,raw.split())
            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:

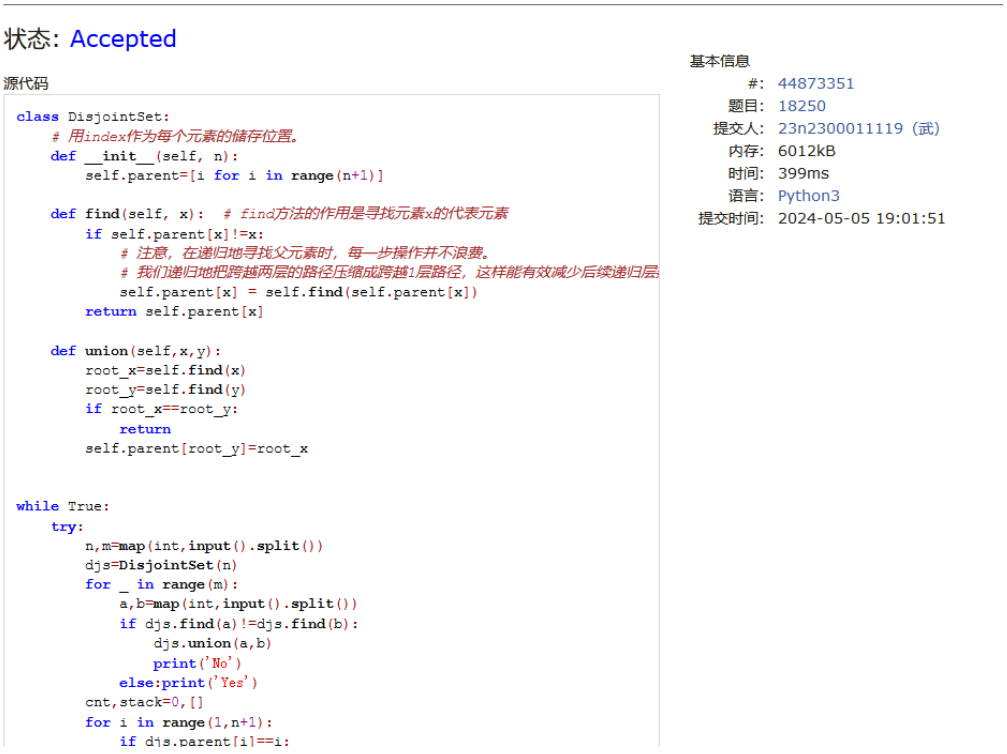
不能按秩合并。在报满杯编号时会出错。

代码

```
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数。
            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:
                stack.append(i)
                cnt+=1
        print(cnt)
        print(*stack)
    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

```
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)
    return path, distances[end]

P = int(input())
places = {input().strip() for _ in range(P)}

Q = int(input())
graph = {place: {} for place in places}
for _ in range(Q):
    src, dest, dist = input().split()
    dist = int(dist)
    graph[src][dest] = dist
    graph[dest][src] = dist

R = int(input())
requests = [input().split() for _ in range(R)]

for start, end in requests:
    if start == end:
        print(start)
        continue

    path, total_dist = shortest_path_to(graph, start, end)
    output = ""
    for i in range(len(path) - 1):
        output += f"{path[i]}->({graph[path[i]][path[i+1]]})->"
    output += f"{end}"
    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

```python
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）