Assignment #B: 图论和树算

Updated 2018 GMT+8 Apr 28, 2024

2024 spring, Complied by 夏天、生命科学学院

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 10 家庭版

Python编程环境：Spyder （python 3.11）

1. 题目

28170: 算鹰

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

思路：模板题，套最大连通域面积的dfs代码，并改成求连通域的个数即可

代码

```
board=[input()for _ in range(10)]
visited=[[False]*10 for _ in range(10)]
def dfs(x,y,board,visited):
    if x<0 or x>=10 or y<0 or y>=10 or board[x][y]=='-' or
visited[x][y]:
        return
    visited[x][y]=True
    for dx,dy in [(1,0),(-1,0),(0,1),(0,-1)]:
        nx,ny=dx+x,dy+y
        dfs(nx,ny,board,visited)
    return
ans=0
for x in range(10):
    for y in range(10):
        if not visited[x][y] and board[x][y]=='.':
            dfs(x,y,board,visited)
            ans+=1
print(ans)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
board=[input()for _ in range(10)]
visited=[[False]*10 for _ in range(10)]
def dfs(x,y,board,visited):
    if x<0 or x>=10 or y<0 or y>=10 or board[x][y]=='-' or visited[x][y]
        return
    visited[x][y]=True
    for dx,dy in [(1,0),(-1,0),(0,1),(0,-1)]:
        nx,ny=dx+x,dy+y
        dfs(nx,ny,board,visited)
    return
ans=0
for x in range(10):
    for y in range(10):
        if not visited[x][y] and board[x][y]=='.':
            dfs(x,y,board,visited)
            ans+=1
print(ans)
```

基本信息

```
#: 44854545
题目: 28170
提交人: 23n2300012289
内存: 3656kB
时间: 21ms
语言: Python3
提交时间: 2024-05-04 09:52:04
```

02754: 八皇后

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

思路：用集合/列表记录不能放置皇后的列即可

代码

```
ans=[]
def dfs(result='',i=0,selected_row=[],diagnol_1=set(),diagnol_2=set()):
    if i==8:
        ans.append(result)
        return
    for j in range(1,9):
        if j not in selected_row and i+j not in diagnol_1 and i-j not in diagnol_2:
            dfs(result+str(j),i+1,selected_row+[j],diagnol_1|{i+j},diagnol_2|{i-j})
dfs()
for _ in range(int(input())):
    index=int(input())
    print(ans[index-1])
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
ans=[]
def dfs(result='',i=0,selected_row=[],diagnol_1=set(),diagnol_2=set()):
    if i==8:
        ans.append(result)
        return
    for j in range(1,9):
        if j not in selected_row and i+j not in diagnol_1 and i-j not in i
            dfs(result+str(j),i+1,selected_row+[j],diagnol_1|{i+j},diagn
dfs()
for _ in range(int(input())):
    index=int(input())
    print(ans[index-1])
```

基本信息

#:	44854766
题目:	02754
提交人:	23n2300012289
内存:	3628kB
时间:	23ms
语言:	Python3
提交时间:	2024-05-04 10:12:34

03151: Pots

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

思路：让我想起来了小学的奥数题，抽象成bfs的话就是把A,B当前水量的二元数组看成坐标，fill、drop、pour对应六种移动方式，最终目标是移动到横坐标或纵坐标为C的点。

代码

```
from collections import deque
A,B,C=map(int,input().split())
def bfs(A,B,C):
    start=(0,0,[])
    queue=deque([start])
    visited=set()
    visited.add((0,0))
    while queue:
        a,b,ans=queue.popleft()
        if a==C or b==C:
            return ans
        operations=[(A,b,ans+['FILL(1)']),(a,B,ans+['FILL(2)']),\
                    (0,b,ans+['DROP(1)']),(a,0,ans+['DROP(2)']),\
                    (max(0,a+b-B),min(a+b,B),ans+['POUR(1,2)']),\
                    (min(A,a+b),max(0,a+b-A),ans+['POUR(2,1)'])]
        for operation in operations:
            new_state=(operation[0],operation[1])
            if new_state not in visited:
                visited.add(new_state)
                queue.append(operation)
    return 'impossible'
ans=bfs(A,B,C)
if ans=='impossible':
    print(ans)
else:
    print(len(ans))
    for _ in ans:
        print(_)
```

状态: Accepted

源代码

```
from collections import deque
A,B,C=map(int,input().split())
def bfs(A,B,C):
    start=(0,0,[])
    queue=deque([start])
    visited=set()
    visited.add((0,0))
    while queue:
        a,b,ans=queue.popleft()
        if a==C or b==C:
            return ans
        operations=[(A,b,ans+['FILL(1)']),(a,B,ans+['FILL(2)']),\
                    (0,b,ans+['DROP(1)']),(a,0,ans+['DROP(2)']),\
                    (max(0,a+b-B),min(a+b,B),ans+['POUR(1,2)']),\
                    (min(A,a+b),max(0,a+b-A),ans+['POUR(2,1)'])]
        for operation in operations:
            new_state=(operation[0],operation[1])
            if new_state not in visited:
                visited.add(new_state)
                queue.append(operation)
    return 'impossible'
ans=bfs(A,B,C)
if ans=='impossible':
    print(ans)
else:
    print(len(ans))
    for _ in ans:
        print(_)
```

基本信息

#:	44855401
题目:	03151
提交人:	23n2300012289
内存:	3692kB
时间:	21ms
语言:	Python3
提交时间:	2024-05-04 11:00:48

代码运行截图 （AC代码截图，至少包含有"Accepted"）

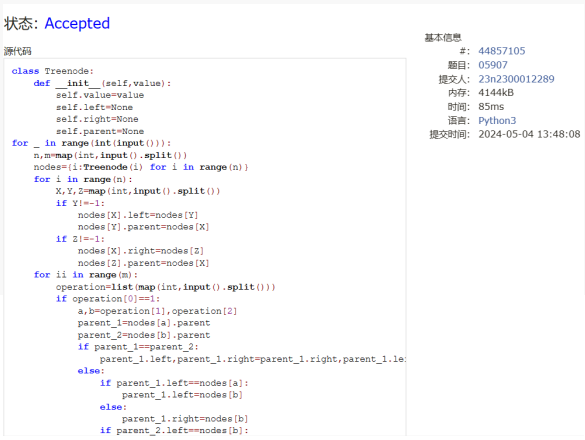
05907: 二叉树的操作

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

思路：正常的建树，关于交换节点需要找双亲节点，于是treenode类初始化时多加一个parent，以便判断是左孩子还是右孩子

```
class Treenode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
        self.parent=None
for _ in range(int(input())):
    n,m=map(int,input().split())
    nodes={i:Treenode(i) for i in range(n)}
    for i in range(n):
        X,Y,Z=map(int,input().split())
        if Y!=-1:
            nodes[X].left=nodes[Y]
            nodes[Y].parent=nodes[X]
        if Z!=-1:
            nodes[X].right=nodes[Z]
            nodes[Z].parent=nodes[X]
    for ii in range(m):
        operation=list(map(int,input().split()))
        if operation[0]==1:
            a,b=operation[1],operation[2]
            parent_1=nodes[a].parent
            parent_2=nodes[b].parent
            if parent_1==parent_2:
                parent_1.left,parent_1.right=parent_1.right,parent_1.left
            else:
                if parent_1.left==nodes[a]:
                    parent_1.left=nodes[b]
                else:
                    parent_1.right=nodes[b]
                if parent_2.left==nodes[b]:
                    parent_2.left=nodes[a]
                else:
                    parent_2.right=nodes[a]
                nodes[b].parent,nodes[a].parent=parent_1,parent_2
        else:
            c=operation[1]
            node=nodes[c]
            while node.left:
                node=node.left
            print(node.value)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



18250: 冰阔落 I

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

思路：用路径压缩减少递归深度，避免爆栈导致RE

代码

```
def find(x):
    if parents[x]!=x:
        parents[x]=find(parents[x])
    return parents[x]
def union(x,y):
    parents[find(y)]=find(x)
while True:
    try:
        n,m=map(int,input().split())
        parents=[i for i in range(n+1)]
        for _ in range(m):
            x,y=map(int,input().split())
            if find(x)==find(y):
                print('Yes')
            else:
                print('No')
                union(x, y)
        res=set(find(x) for x in range(1,n+1))
        num=len(res)
        print(num)
        print(' '.join(map(str,sorted(res))))
    except EOFError:
        break
```

代码运行截图
（AC代码截图，
至少包含有"Accepted"）

05443: 兔子与樱花

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

思路：用heapq写bfs，第一个变量记为总距离，以保证取出的元素一定是距离最短的

代码

```
import heapq
graph={}
P=int(input())
for i in range(P):
    graph[input()]={}
Q=int(input())
for ii in range(Q):
    place_1,place_2,distance=input().split()
    graph[place_1][place_2]=int(distance)
    graph[place_2][place_1]=int(distance)
R=int(input())
for iii in range(R):
    start,end=input().split()
    heap=[]
    heapq.heappush(heap,(0,start,[start]))
    visited=set()
    while heap:
        total_distance,place,way=heapq.heappop(heap)
        if place==end:
            break
        visited.add(place)
        for next_place in graph[place]:
            heapq.heappush(heap,(total_distance+graph[place][next
_place],next_place,\
                                 way+['('+format(str(graph[place]
[next_place]))+')']+[next_place]))
    print('->'.join(way))
```

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
graph={}
P=int(input())
for i in range(P):
    graph[input()]={}
Q=int(input())
for ii in range(Q):
    place_1,place_2,distance=input().split()
    graph[place_1][place_2]=int(distance)
    graph[place_2][place_1]=int(distance)
R=int(input())
for iii in range(R):
    start,end=input().split()
    heap=[]
    heapq.heappush(heap,(0,start,[start]))
    visited=set()
    while heap:
        total_distance,place,way=heapq.heappop(heap)
        if place==end:
            break
        visited.add(place)
        for next_place in graph[place]:
            heapq.heappush(heap,(total_distance+graph[place][next_place
                                 way+['('+format(str(graph[place][next_p
    print('->'.join(way))
```

基本信息

#:	44857660
题目:	05443
提交人:	23n2300012289
内存:	3632kB
时间:	24ms
语言:	Python3
提交时间:	2024-05-04 14:31:55

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。

想趁着五一假期多刷些题目，奈何有更重要的事情去忙，连这周作业都是今天挤了些时间才写的，好在难度不大，除了二叉树的操作那题犯懒复制重复部分的代码结果多了个‘=’导致TLE之外其他都是很常规的dfs/bfs/树算/并查集的题目，下次得更细心一点！

可能第13周之后才能有更多时间大量刷题和整理cheat sheet了，希望还来得及（双手合十）