

# Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Compiled by 同学的姓名、院系

## 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:

对每个位置进行dfs, 根据dfs的次数确定连通数

代码

```
def get_directions(i,j):
    a=[[i-1,j],[i+1,j],[i,j-1],[i,j+1]]
    ans=[]
    for s in a:
        if s[0]>=0 and s[1]>=0 and s[0]<=9 and s[1]<=9:
            ans.append(s)
    return ans

def dfs(i,j):
    if mapmatrix[i][j]=='-':
        return False
```

```
mapmatrix[i][j]='-'
for direction in get_directions(i,j):
    nx=direction[0]
    ny=direction[1]
    dfs(nx,ny)
return True

mapmatrix=[]
for _ in range(10):
    ipt=input()
    iptrow=[x for x in ipt]
    mapmatrix.append(iptrow)
cnt=0
for i in range(10):
    for j in range(10):
        if dfs(i,j):
            cnt+=1
print(cnt)
```

代码运行截图 (至少包含有"Accepted")

## #44851143提交状态

[查看](#) [报错](#)

状态: **Accepted**

源代码

```
def get_directions(i,j):
    a=[[i-1,j],[i+1,j],[i,j-1],[i,j+1]]
    ans=[]
    for s in a:
        if s[0]>=0 and s[1]>=0 and s[0]<=9 and s[1]<=9:
            ans.append(s)
    return ans

def dfs(i,j):
    if mapmatrix[i][j]=='-':
        return False
    mapmatrix[i][j]='-'
    for direction in get_directions(i,j):
        nx=direction[0]
        ny=direction[1]
        dfs(nx,ny)
    return True

mapmatrix=[]
for _ in range(10):
    ipt=input()
    iptrow=[x for x in ipt]
    mapmatrix.append(iptrow)
cnt=0
for i in range(10):
    for j in range(10):
        if dfs(i,j):
            cnt+=1
print(cnt)
```

基本信息

#: 4485114  
题目: 28170  
提交人: zzk  
内存: 3868kB  
时间: 21ms  
语言: Python3  
提交时间: 2024-05

©2002-2022 POJ 京ICP备20010980号-1

## 02754: 八皇后

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

思路:

抄的答案，首先定义了检查位置是否合理的函数，然后定义了寻找解的dfs，通过回溯找到合适的走法，然后输出

代码

```
def is_safe(board, row, col):
    # 检查当前位置是否安全
    # 检查同一列是否有皇后
    for i in range(row):
        if board[i] == col:
            return False
```

```

# 检查左上方是否有皇后
i = row - 1
j = col - 1
while i >= 0 and j >= 0:
    if board[i] == j:
        return False
    i -= 1
    j -= 1
# 检查右上方是否有皇后
i = row - 1
j = col + 1
while i >= 0 and j < 8:
    if board[i] == j:
        return False
    i -= 1
    j += 1
return True

def queen_dfs(board, row):
    if row == 8:
        # 找到第b个解，将解存储到result列表中
        ans.append(''.join([str(x+1) for x in board]))
        return
    for col in range(8):
        if is_safe(board, row, col):
            # 当前位置安全，放置皇后
            board[row] = col
            # 继续递归放置下一行的皇后
            queen_dfs(board, row + 1)
            # 回溯，撤销当前位置的皇后
            board[row] = 0

ans = []
queen_dfs([None]*8, 0)
for _ in range(int(input())):
    print(ans[int(input()) - 1])

```

代码运行截图 (至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述

 **CS101 / 题库**

题目 排名 状态 提问

#44851645提交状态

查看

状态: Accepted

源代码

```
def is_safe(board, row, col):  
    # 检查当前位置是否安全  
    # 检查同一列是否有皇后  
    for i in range(row):  
        if board[i] == col:  
            return False  
    # 检查左上方是否有皇后  
    i = row - 1  
    j = col - 1  
    while i >= 0 and j >= 0:  
        if board[i] == j:  
            return False  
        i -= 1  
        j -= 1  
    # 检查右上方是否有皇后  
    i = row - 1  
    j = col + 1  
    while i >= 0 and j < 8:  
        if board[i] == j:  
            return False  
        i -= 1  
        j += 1  
    return True  
  
def queen_dfs(board, row):  
    if row == 8:  
        # 找到第b个解, 将解存储到result列表中  
        ans.append(''.join([str(x+1) for x in board]))  
        return  
    for col in range(8):  
        if is_safe(board, row, col):  
            # 当前位置安全, 放置皇后
```

基本信息

#: 44851

题目: 02754

提交人: zzk

内存: 3664k

时间: 31ms

语言: Python

提交时间: 2024-

cs101.openjudge.cn/practice/solution/448516...

## 03151: Pots

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

思路:

分块运行, 一个标准的bfs找到最终输出, 同时记录每个节点的路径, 一个find将字典转化成目标输出的路径, 一个option将路径转化成操作表示

代码

```
def option(pre,post):  
    la,lb=pre  
    na,nb=post  
    if la>na and lb<nb:  
        return 'POUR(1,2)'  
    elif la<na and lb>nb:
```

```

        return 'POUR(2,1)'
    elif la!=A and na==A:
        return 'FILL(1)'
    elif la!=0 and na==0:
        return 'DROP(1)'
    elif lb!=B and nb==B:
        return 'FILL(2)'
    elif lb!=0 and nb==0:
        return 'DROP(2)'

def find(ans):
    way=[]
    if ans==(0,0):
        return way
    else :
        way.append(pathdct[ans])
        way.extend(find(pathdct[ans]))
        return way

A,B,C=map(int,input().split())

queue=[]
pathdct={}
def get_directions(a,b):
    directions=[]
    if a!=0:
        directions.append((0,b))
    if b!=0:
        directions.append((a,0))
    if a!=A:
        directions.append((A,b))
    if b!=B:
        directions.append((a,B))
    db=B-b
    da=A-a
    if db>a:
        directions.append((0,a+b))
    else :
        directions.append((a-db,B))
    if da>b:
        directions.append((a+b,0))
    else :
        directions.append((A,b-da))
    return directions

def bfs(a,b,c):
    if a==c or b==c:
        return (a,b)
    extd=get_directions(a,b)
    for direction in extd:
        queue.append(direction+(a,b))
    while queue:
        na,nb,la,lb=queue.pop(0)
        if (na,nb) not in pathdct:
            pathdct[(na,nb)]=(la,lb)

```

```

        return bfs(na,nb,c)
    return 'impossible'

final=bfs(0,0,C)
if final=='impossible':
    print(final)
else :
    path=[final]+find(final)
    path.reverse()
#    print(path)
    print(len(path)-1)
    for i in range(len(path)-1):
        pre=path[i]
        post=path[i+1]
#        print(pre,'to',post)
        print(option(pre,post))

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```

def option(pre,post):
    la,lb=pre
    na,nb=post
    if la>na and lb<nb:
        return 'POUR(1,2)'
    elif la<na and lb>nb:
        return 'POUR(2,1)'
    elif la!=A and na==A:
        return 'FILL(1)'
    elif la!=0 and na==0:
        return 'DROP(1)'
    elif lb!=B and nb==B:
        return 'FILL(2)'
    elif lb!=0 and nb==0:
        return 'DROP(2)'

def find(ans):
    way=[]
    if ans==(0,0):
        return way
    else :
        way.append(pathdct[ans])
        way.extend(find(pathdct[ans]))
    return way

A,B,C=map(int,input().split())

queue=[]
pathdct={}
def get_directions(a,b):
    directions=[]
    if a!=0:
        directions.append((0,b))
    if b!=0:
        directions.append((a,0))
    if a!=A:
        directions.append((A,b))
    if b!=B:
        directions.append((a,B))
    db=B-b
    da=A-a

```

基本信息

#: 44880

题目: 03151

提交人: zzk

内存: 3752k

时间: 19ms

语言: Python

提交时间: 2024-

## 05907: 二叉树的操作

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

思路：

三个字典，分别记录父左右，交换就是改动，查询就是顺着左找

代码

```
def change(x, y):
    if parentdct[x] == parentdct[y]:
        p = parentdct[x]
        if lcdct[p]==x:
            lcdct[p]=y
            rcdct[p]=x
        else:
            lcdct[p]=x
            rcdct[p]=y
    else:
        px = parentdct[x]
        py = parentdct[y]
        if px in lcdct:
            if lcdct[px] == x:
                lcdct[px] = y
            else:
                rcdct[px] = y
        if py in lcdct:
            if lcdct[py] == y:
                lcdct[py] = x
            else:
                rcdct[py] = x
        parentdct[x] = py
        parentdct[y] = px

def find(x):
    if x not in lcdct:
        return x
    return find(lcdct[x])

t = int(input())
anslst = []
for _ in range(t):
    n, m = map(int, input().split())
    parentdct = {}
    lcdct = {}
    rcdct = {}

    for i in range(n):
        x, y, z = map(int, input().split())
        if y != -1:
```



```

        lcdct[X] = Y
        parentdct[Y] = X
    if Z != -1:
        rcdct[X] = Z
        parentdct[Z] = X

for j in range(m):
    opt = [int(x) for x in input().split()]
    if opt[0] == 1:
        change(opt[1], opt[2])
    else:
        check = opt[1]
        ans1st.append(find(check))

for x in ans1st:
    print(x)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

## #44878332提交状态

[查看](#)

状态: **Accepted**

源代码

```

def change(x, y):
    if parentdct[x] == parentdct[y]:
        p = parentdct[x]
        if lcdct[p] == x:
            lcdct[p] = y
            rcdct[p] = x
        else:
            lcdct[p] = x
            rcdct[p] = y
    else:
        px = parentdct[x]
        py = parentdct[y]
        if px in lcdct:
            if lcdct[px] == x:
                lcdct[px] = y
            else:
                rcdct[px] = y
        if py in lcdct:
            if lcdct[py] == y:
                lcdct[py] = x
            else:
                rcdct[py] = x
        parentdct[x] = py
        parentdct[y] = px

def find(x):
    if x not in lcdct:
        return x
    return find(lcdct[x])

t = int(input())
ans1st = []
for _ in range(t):
    n, m = map(int, input().split())
    parentdct = {}
    lcdct = {}
    rcdct = {}

```

基本信息

#: 44878332  
 题目: 05907  
 提交人: zxk  
 内存: 3796k  
 时间: 62ms  
 语言: Python3  
 提交时间: 2024-

## 18250: 冰阔落 I

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

思路：

并查集

代码

```
def find(x):
    if bottles[x]!=x:
        bottles[x]=find(bottles[x])
    return bottles[x]
def union(x,y):
    rx=find(x)
    ry=find(y)
    if rx!=ry:
        bottles[ry]=rx
while True :
    try:
        n,m=map(int,input().split())
        bottles=list(range(n+1))
        for _ in range(m):
            a,b=map(int,input().split())
            if find(a)==find(b):
                print('Yes')
            else :
                print('No')
                union(a,b)

        not_empty=set(find(x) for x in range(1,n+1))
        ans=sorted(not_empty)
        print(len(ans))
        print(*ans)
    except EOFError:
        break
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

## #44873655提交状态

[查看](#) [报错](#)

状态: **Accepted**

源代码

```
def find(x):
    if bottles[x] != x:
        bottles[x] = find(bottles[x])
    return bottles[x]
def union(x, y):
    rx = find(x)
    ry = find(y)
    if rx != ry:
        bottles[ry] = rx
while True:
    try:
        n, m = map(int, input().split())
        bottles = list(range(n + 1))
        for _ in range(m):
            a, b = map(int, input().split())
            if find(a) == find(b):
                print('Yes')
            else:
                print('No')
                union(a, b)

        not_empty = set(find(x) for x in range(1, n + 1))
        ans = sorted(not_empty)
        print(len(ans))
        print(*ans)
    except EOFError:
        break
```

基本信息

#: 44873655  
题目: 18250  
提交人: zzk  
内存: 5488kB  
时间: 364ms  
语言: Python3  
提交时间: 2024-05

©2002-2022 POJ 京ICP备20010980号-1

## 05443: 兔子与樱花

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

思路:

dijkstra

代码

```
class Vertex:
    def __init__(self, v):
        self.value = v
        self.connectedto = {}

class Graph:
    def __init__(self):
```

```

self.vertexes={}

def add_vertex(self,s):
    self.vertexes[s]=Vertex(s)

def add_edge(self,s1,s2,w):
    if s1 not in self.vertexes:
        self.vertexes[s1]=Vertex(s1)
    if s2 not in self.vertexes:
        self.vertexes[s2]=Vertex(s2)
    v1=self.vertexes[s1]
    v2=self.vertexes[s2]
    v1.connectedto[s2]=w
    v2.connectedto[s1]=w
    self.vertexes[s1]=v1
    self.vertexes[s2]=v2

def dijkstra(v1,v2):
    sheet={}
    not_visited=set(map_graph.vertexes.keys())
    path={}
    for ver in map_graph.vertexes:
        sheet[ver]=99999999
    sheet[v1]=0
    nearest=v1
    while not_visited:
        nearest=None
        for vtx in not_visited:
            if not nearest or sheet[vtx]<sheet[nearest]:
                nearest=vtx
        not_visited.discard(nearest)
        neighbours=map_graph.vertexes[nearest].connectedto.keys()
        for neighbour in neighbours:
            n1=sheet[nearest]+map_graph.vertexes[nearest].connectedto[neighbour]
            if n1<sheet[neighbour]:
                sheet[neighbour]=n1
                path[neighbour]=(nearest,map_graph.vertexes[nearest].connectedto[neighbour])
        way=[v2]
        while v2 in path:
            lastver,w=path[v2]
            way.append(f'({w})')
            way.append(lastver)
            v2=lastver
        way.reverse()
        return '->'.join(way)

vertexes=[]
P=int(input())
for _ in range(P):
    vertexes.append(str(input()))
#输入节点

edges=[]
Q=int(input())
for _ in range(Q):

```

```
v1,v2,weight=map(str,input().split())
w=int(weight)
edges.append((v1,v2,w))
#输入边

route=[]
R=int(input())
for _ in range(R):
    v1,v2=input().split()
    route.append((v1,v2))
#输入待求路线起止点

map_graph=Graph()
for v in vertexes:
    map_graph.add_vertex(v)
for e in edges:
    v1,v2,w=e
    map_graph.add_edge(v1,v2,w)
for begin,end in route:
    print(dijkstra(begin,end))
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述

 **CS101 / 题库**

题目 排名 状态 提问

#44887693提交状态

查看

状态: Accepted

源代码

```
class Vertex:
    def __init__(self,v):
        self.value=v
        self.connectedto={}

class Graph:
    def __init__(self):
        self.vertexes={}

    def add_vertex(self,s):
        self.vertexes[s]=Vertex(s)

    def add_edge(self,s1,s2,w):
        if s1 not in self.vertexes:
            self.vertexes[s1]=Vertex(s1)
        if s2 not in self.vertexes:
            self.vertexes[s2]=Vertex(s2)
        V1=self.vertexes[s1]
        V2=self.vertexes[s2]
        V1.connectedto[s2]=w
        V2.connectedto[s1]=w
        self.vertexes[s1]=V1
        self.vertexes[s2]=V2

    def dijkstra(v1,v2):
        sheet={}
        not_visited=set(map_graph.vertexes.keys())
        path={}
        for ver in map_graph.vertexes:
            sheet[ver]=99999999
        sheet[v1]=0
        nearest=v1
        while not_visited:
```

基本信息

#: 44887

题目: 05443

提交人: zzk

内存: 3720k

时间: 20ms

语言: Python

提交时间: 2024-

## 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。

1.算鹰：好抽象的题目，使我理解错误，不是中心+四方算一个，而是连通区域，需要dfs，网上找到了一个更简单的方法，dfs的过程中对原图进行修改，这样就不用再单独记录每个节点是否访问过了，非常巧妙，但是这种整活题目的语义不明真的让人很费解（

2.八皇后：自己尝试了半天，但是还是没任何思路，于是翻了答案；之前对所谓的回溯不了解，其实和dfs类似，就是顺着往下走，走不通了就退一步找别的路走，这个题的区别是走通了也需要退回去接着走，体现在了递归调用dfs后均进行了回溯，并不停止

5.冰阔落：并查集并不熟悉，开始的时候是没有用并查集，之后反复改动都是wa，应当是并集时没有把根覆盖好，并查集一般都用列表，感觉有点怪，容易把所谓的原始编号和杯子的编号弄混，看来对并查集还是没弄懂

4.二叉树的操作：一开始的思路就是用字典和列表，但是代码实现卡了很久，莫名其妙就会出错，原来是树形没考虑到，主要是交换时候查询是左节点还是右节点的时候卡在了语法上，老师改动了之后才AC，之前用的列表表示子节点，应该也是这个问题，或许可以保留-1标志为叶节点，遇到-1表示到底了，但是判断也挺麻烦的

3.pots: bfs很不熟悉，认识到了很吊诡的一件事就是bfs本身就是最短路径了，至于具体为什么它就是最短的我不理解（，先是尝试了bfs然后比较大小选最小的，写着写着发现写成dfs了貌似。。。既然比较每个走法的长度岂不是需要先走一遍。。。后边意识到这一点于是放弃比较的想法，直接bfs，后又在语言实现上卡半天，对于我这种脑子不好使的就干脆多写几个函数，分部分写代码，输入什么输出什么，把问题转化一下，虽然麻烦但是AC了

6.兔子与樱花：竟然一遍过了！没做的时候看群里问来问去，还有新的什么算法，但是dijkstra也不错啊...或许是数据量不大吧，感觉自己手搓vertex、graph、dijkstra的能力得到了很大的锻炼（，对dijkstra的理解也更深了，虽然还是没太理解为啥这么走就是最短的

终于又能AC6，花了很长的时间做作业，五一结束但是作业压得太多了/(T o T)/~~