Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 余汶青 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：版本　Windows 11 家庭中文版
版本　22H2
安装日期 2023/7/18
操作系统版本 22621.2283
序列号　5CD323PJKL
体验　Windows Feature Experience Pack 1000.22662.1000.0

Python编程环境：* Spyder version: 5.4.3 (conda)

- Python version: 3.11.4 64-bit
- Qt version: 5.15.2
- PyQt5 version: 5.15.7
- Operating System: Windows 10

1. 题目

04081: 树的转换

http://cs101.openjudge.cn/dsapre/04081/

思路：

代码

```
ans1=0
ans2=0
class tree():
    def __init__(self,key):
```

```
        self.father=None
        self.son=[]
        self.dep=key

class bitree():
    def __init__(self,dep):
        self.father=None
        self.left=None
        self.right=None
        self.dep=dep

def build(root,s):
    global ans1
    if len(s)==0:
        return root
    if s[0]=='u':
        return build(root.father, s[1:])
    if s[0]=='d':
        node=tree(root.dep+1)
        root.son.append(node)
        node.father=root
        ans1=max(ans1,node.dep)
        return build(node,s[1:])

def transfer(old,new):
    global ans2
    if old.father!=None:
        node=bitree(new.dep+1)
        ans2=max(ans2,node.dep)
        if old.father.son[0]!=old:
            new.right=node
        else:
            new.left=node
        if len(old.son)!=0:
            transfer(old.son[0], node)
        if old!=old.father.son[-1]:
            ind=old.father.son.index(old)
            transfer(old.father.son[ind+1], node)
    else:
        transfer(old.son[0],new)

s=input()
root=tree(0)
root=build(root,s)
root2=bitree(0)
transfer(root,root2)
print("%d => %d"%(ans1,ans2))
```

代码运行截图

#44752724提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
ans1=0
ans2=0
class tree():
    def __init__(self,key):
        self.father=None
        self.son=[]
        self.dep=key
```

基本信息

#: 44752724
题目: 04081
提交人: 23n2300012265
内存: 3724kB
时间: 29ms
语言: Python3
提交时间: 2024-04-22 17:04:33

08581: 扩展二叉树

http://cs101.openjudge.cn/dsapre/08581/

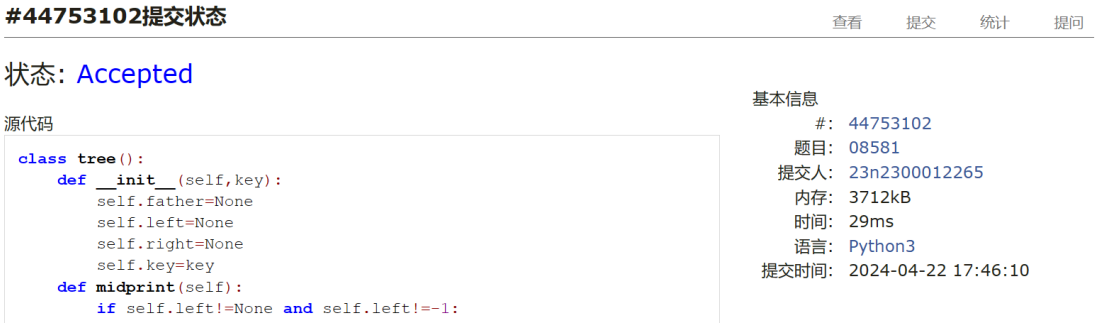
思路:

代码

```
class tree():
    def __init__(self,key):
        self.father=None
        self.left=None
        self.right=None
        self.key=key
    def midprint(self):
        if self.left!=None and self.left!=-1:
            self.left.midprint()
        print(self.key,end='')
        if self.right!=None and self.right!=-1:
            self.right.midprint()
    def postprint(self):
        if self.left!=None and self.left!=-1:
            self.left.postprint()
        if self.right!=None and self.right!=-1:
            self.right.postprint()
        print(self.key,end='')

def build(root,s):
    #print(s[0])
    if len(s)==0:
        return
    if s[0]!='.':
        node=tree(s[0])
        if root.left==None:
            root.left=node
            node.father=root
            build(root.left,s[1:])
        elif root.right==None:
            root.right=node
```

```
            node.father=root
            build(root.right,s[1:])
        else:
            build(root.father,s)
    else:
        if root.left==None:
            root.left=-1
            build(root,s[1:])
        elif root.right==None:
            root.right=-1
            build(root.father,s[1:])
        else:
            build(root.father,s)
    return root


s=input()
root=tree(s[0])
root=build(root,s[1:])
root.midprint()
print()
root.postprint()
```

代码运行截图



22067: 快速堆猪

http://cs101.openjudge.cn/practice/22067/

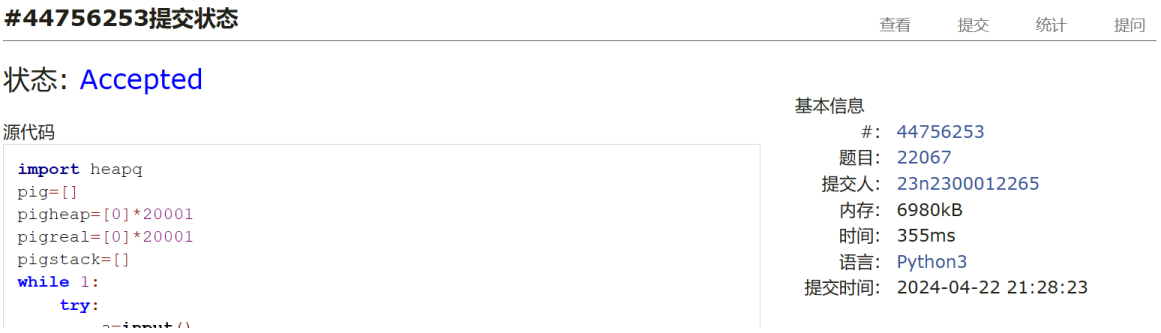
思路:

代码

```
import heapq
pig=[]
pigheap=[0]*20001
pigreal=[0]*20001
pigstack=[]
while 1:
    try:
```

```
        a=input()
        if a[1]=='o':
            if len(pigstack)>0:
                pigreal[pigstack.pop()]-=1
        if a[1]=='i':
            if len(pigstack)>0:
                b=heapq.heappop(pig)
                while pigheap[b]!=pigreal[b]:
                    pigheap[b]-=1
                    b=heapq.heappop(pig)
                print(b)
                heapq.heappush(pig,b)
        if a[1]=='u':
            w=int(a[5:])
            heapq.heappush(pig,w)
            pigheap[w]+=1
            pigreal[w]+=1
            pigstack.append(w)
    except:
        break
```

代码运行截图



04123: 马走日

dfs, http://cs101.openjudge.cn/practice/04123

思路:

代码

```
class Vertex:
    def __init__(self,num):
        self.key=num
        self.connectedTo={}
```

```
        self.color="white"
    def add_neighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def get_neightbors(self):
        return self.connectedTo.keys()

class Graph:
    def __init__(self):
        self.vertices={}
        self.num_vertices=0
    def add_vertex(self,key):
        self.num_vertices+=1
        new_ertex=Vertex(key)
        self.vertices[key]=new_ertex
        return new_ertex
    def get_vertex(self,n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None
    def add_edge(self,f,t,cost=0):
        if f not in self.vertices:
            nv=self.add_vertex(f)
        if t not in self.vertices:
            nv=self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t],cost)
    def getVertces(self):
        return list(self.vertices.keys())

def pos_to_node_id(x,y,bdsize):
    return x*bdsize+y

def gen_legal_moves(row,col,board_sizex,board_sizey):
    new_moves=[]
    move_offsets=[
        (-1,-2),
        (-1,2),
        (-2,-1),
        (-2,1),
        (1,-2),
        (1,2),
        (2,-1),
        (2,1),
        ]
    for r_off,c_off in move_offsets:
        if (
            0<=row+r_off<board_sizex
            and 0<=col+c_off<board_sizey
                ):
            new_moves.append((row+r_off,col+c_off))
    return new_moves


def knight_graph(board_sizex,board_sizey):
    kt_graph=Graph()
    for row in range(board_sizex):
```

```
        for col in range(board_sizey):
            node_id=pos_to_node_id(row,col,board_sizey)
            new_positions=gen_legal_moves(row,col,board_sizex,board_sizey)
            for row2,col2 in new_positions:
                other_node_id=pos_to_node_id(row2, col2,board_sizey)
                kt_graph.add_edge(node_id, other_node_id)
    return kt_graph

def knight_tour(n,path,u,limit):
    u.color="gray"
    path.append(u)
    ans=0
    if n<limit:
        neighbors=ordered_by_avail(u)
        for nbr in neighbors:
            if nbr.color=="white":
                a=knight_tour(n+1, path, nbr, limit)
                if a:
                    ans+=a
        else:
            u.color="white"
        u.color="white"
        return ans
    else:
        u.color="white"
        return  ans+1

def ordered_by_avail(n):
    res_list=[]
    for v in n.get_neightbors():
        if v.color=="white":
            c=0
            for w in v.get_neightbors():
                if w.color=="white":
                    c+=1
            res_list.append((c,v))
    res_list.sort(key=lambda x:x[0])
    return [y[1] for y in res_list]


t=int(input())
for _ in range(t):
    n,m,x,y=map(int,input().split())
    g=knight_graph(n,m)
    start_vertex=g.get_vertex(pos_to_node_id(x,y,m))
    if start_vertex is None:
        print(0)
        exit(0)
    tour_path=[]
    done=knight_tour(0,tour_path,start_vertex,n*m-1)
    if done:
        print(done)
    else:
        print(0)
```

代码运行截图

#44761259提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class Vertex:
    def __init__(self,num):
        self.key=num
        self.connectedTo={}
        self.color="white"
    def add_neighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def get_neightbors(self):
        return self.connectedTo.keys()
```

基本信息

#: 44761259
题目: 04123
提交人: 23n2300012265
内存: 9032kB
时间: 3107ms
语言: Python3
提交时间: 2024-04-23 11:52:12

28046: 词梯

bfs, http://cs101.openjudge.cn/practice/28046/

思路:

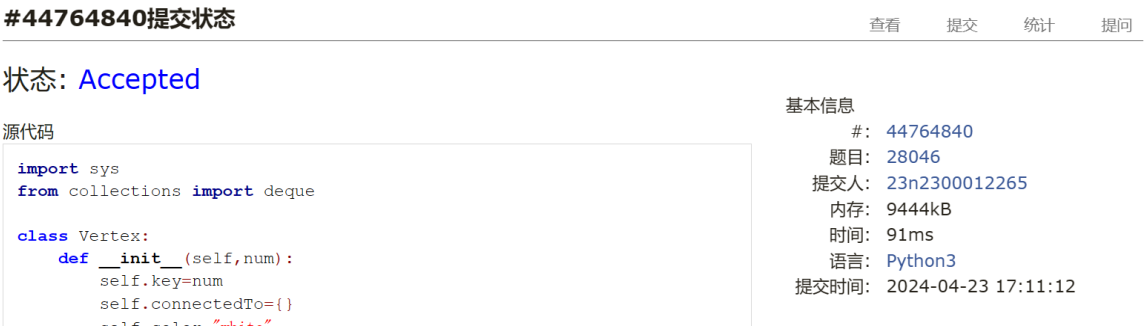
代码

```
import sys
from collections import deque

class Vertex:
    def __init__(self,num):
        self.key=num
        self.connectedTo={}
        self.color="white"
        self.distance=99999999
        self.previous=None

    def add_neighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def get_neightbors(self):
        return self.connectedTo.keys()

class Graph:
    def __init__(self):
        self.vertices={}
        self.num_vertices=0
    def add_vertex(self,key):
        self.num_vertices+=1
        new_ertex=Vertex(key)
        self.vertices[key]=new_ertex
        return new_ertex
    def get_vertex(self,n):
```

```
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None
    def add_edge(self,f,t,cost=0):
        if f not in self.vertices:
            nv=self.add_vertex(f)
        if t not in self.vertices:
            nv=self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t],cost)
    def getVertces(self):
        return list(self.vertices.keys())

def bfs(start):
    start.distance=0
    start.previous=None
    vert_queue=deque()
    vert_queue.append(start)
    while len(vert_queue)>0:
        current=vert_queue.popleft()
        for neighbor in current.get_neightbors():
            if neighbor.color=="white":
                neighbor.color="gray"
                neighbor.distance=current.distance+1
                neighbor.previous=current
                vert_queue.append(neighbor)
        current.color="black"

def traverse(start):
    ans=[]
    current=start
    while current.previous:
        #print(current.key)
        ans.append(current.key)
        current=current.previous
    ans.append(current.key)
    return ans

n=int(input())
buckets={}
the_graph=Graph()
for _ in range(n):
    s=input()
    s=s.strip()
    for i,_ in enumerate(s):
        bucket = f"{s[:i]}_{s[i+1:]}"
        buckets.setdefault(bucket,set()).add(s)
for similar in buckets.values():
    for word1 in similar:
        for word2 in similar-{word1}:
            #print(word1,word2)
            the_graph.add_edge(word1, word2)
start,end=input().split()
if start not in the_graph.getVertces() or end not in the_graph.getVertces():
    print("NO")
    sys.exit()
```

```
bfs(the_graph.get_vertex(start))
ans=traverse(the_graph.get_vertex(end))
if len(ans)==1:
    print("NO")
    sys.exit()
print(*ans[::-1])
```

代码运行截图



28050: 骑士周游

dfs, http://cs101.openjudge.cn/practice/28050/

思路:

代码

```
class Vertex:
    def __init__(self,num):
        self.key=num
        self.connectedTo={}
        self.color="white"
    def add_neighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def get_neightbors(self):
        return self.connectedTo.keys()

class Graph:
    def __init__(self):
        self.vertices={}
        self.num_vertices=0
    def add_vertex(self,key):
        self.num_vertices+=1
        new_ertex=Vertex(key)
        self.vertices[key]=new_ertex
        return new_ertex
    def get_vertex(self,n):
        if n in self.vertices:
            return self.vertices[n]
```

```
        else:
            return None
    def add_edge(self,f,t,cost=0):
        if f not in self.vertices:
            nv=self.add_vertex(f)
        if t not in self.vertices:
            nv=self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t],cost)
    def getVertces(self):
        return list(self.vertices.keys())

def pos_to_node_id(x,y,bdsize):
    return x*bdsize+y

def gen_legal_moves(row,col,board_size):
    new_moves=[]
    move_offsets=[
        (-1,-2),
        (-1,2),
        (-2,-1),
        (-2,1),
        (1,-2),
        (1,2),
        (2,-1),
        (2,1),
        ]
    for r_off,c_off in move_offsets:
        if (
            0<=row+r_off<board_size
            and 0<=col+c_off<board_size
                ):
            new_moves.append((row+r_off,col+c_off))
    return new_moves


def knight_graph(board_size):
    kt_graph=Graph()
    for row in range(board_size):
        for col in range(board_size):
            node_id=pos_to_node_id(row,col,board_size)
            new_positions=gen_legal_moves(row,col,board_size)
            for row2,col2 in new_positions:
                other_node_id=pos_to_node_id(row2, col2, board_size)
                kt_graph.add_edge(node_id, other_node_id)
    return kt_graph

def knight_tour(n,path,u,limit):
    u.color="gray"
    path.append(u)
    if n<limit:
        neighbors=ordered_by_avail(u)
        for nbr in neighbors:
            if nbr.color=="white" and \
                knight_tour(n+1, path, nbr, limit):
                    return True
        else:
```

```
            path.pop()
            u.color="white"
            return False
    else:
        return  True

def ordered_by_avail(n):
    res_list=[]
    for v in n.get_neightbors():
        if v.color=="white":
            c=0
            for w in v.get_neightbors():
                if w.color=="white":
                    c+=1
            res_list.append((c,v))
    res_list.sort(key=lambda x:x[0])
    return [y[1] for y in res_list]

bdsize=int(input())
*start_pos,=map(int,input().split())
g=knight_graph(bdsize)
start_vertex=g.get_vertex(pos_to_node_id(start_pos[0],start_pos[1],bdsize))
if start_vertex is None:
    print("fail")
    exit(0)
tour_path=[]
done=knight_tour(0,tour_path,start_vertex,bdsize*bdsize-1)
if done:
    print("success")
else:
    print("fail")
```

代码运行截图

2. 学习总结和收获

做树的题目越来越熟练了

这次作业新学习了图的遍历，复习了dfs和bfs的写法，感觉最重要的是想好怎么建图，然后只需要套模板就好了

新学到了集合的使用