Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied 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

04081: 树的转换

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

思路：

原始树简单，转换后的树：通过栈记录每次下降之后的高度，如果遇到上升，就弹出栈顶的一个高度，之后再下降就从这个高度开始下降

代码

```
def H1(alst):
    h=0
    hm=0
    for x in alst:
        if x=='d':
            h+=1
        else:
            h-=1
        if h>hm:
            hm=h
```

```
    return hm

def H2(alst):
    h=0
    hm=0
    stack=[]
    for x in alst:
        if x=='d':
            h+=1
            stack.append(h)
        else:
            h=stack.pop()
        if h>=hm:
            hm=h
    return hm

ipt=input()
ipt_str=[x for x in ipt]
print(f'{H1(ipt_str)} => {H2(ipt_str)}')
```

代码运行截图 （至少包含有"Accepted"）

08581: 扩展二叉树

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

思路:

先根据扩展二叉树的前序建树，之后中序后序遍历

代码

```
class node:
    def __init__(self,v):
        self.value=v
        self.right=None
        self.left=None

    def inorder(self):
```

```
        ans=[]
        if self.left:
            ans.extend(self.left.inorder())
        ans.extend([self.value])
        if self.right:
            ans.extend(self.right.inorder())
        return ans

    def postorder(self):
        ans=[]
        if self.left:
            ans.extend(self.left.postorder())
        if self.right:
            ans.extend(self.right.postorder())
        ans.extend([self.value])
        return ans

def build_tree(alst):
    if not alst:
        return None
    value=alst.pop(0)
    if value=='.':
        return None
    root=node(value)
    root.left=build_tree(alst)
    root.right=build_tree(alst)
    return root

ipt=input()
preorder=[x for x in ipt]
T=build_tree(preorder)
print(''.join(T.inorder()))
print(''.join(T.postorder()))
```

代码运行截图 （至少包含有"Accepted"）


```python
class node:
    def __init__(self,v):
        self.value=v
        self.right=None
        self.left=None

    def inorder(self):
        ans=[]
        if self.left:
            ans.extend(self.left.inorder())
        ans.extend([self.value])
        if self.right:
            ans.extend(self.right.inorder())
        return ans

    def postorder(self):
        ans=[]
        if self.left:
            ans.extend(self.left.postorder())
        if self.right:
            ans.extend(self.right.postorder())
        ans.extend([self.value])
        return ans

def build_tree(alst):
    if not alst:
        return None
    value=alst.pop(0)
    if value=='.':
        return None
    root=node(value)
    root.left=build_tree(alst)
    root.right=build_tree(alst)
    return root

ipt=input()
preorder=[x for x in ipt]
T=build_tree(preorder)
print(''.join(T.inorder()))
print(''.join(T.postorder()))
```

22067: 快速堆猪

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

思路:

做了一个堆找最小值，一个栈记录顺序，一个字典记录堆里每个重量的猪有多少个，如果push就分别添加元素，字典+1，如果pop就在字典-1，栈pop，如果min就顺着堆查找，直到字典里这个重量的猪数量大于0

代码

```python
import heapq

pigs=[]
anslst=[]
pigdct={}
```

```
pigstack=[]
while True:
    try:
        opt=[str(x) for x in input().split()]
        if len(opt)==2:
            s=int(opt[1])
            heapq.heappush(pigs,s)
            pigstack.append(s)
            if s in pigdct.keys():
                pigdct[s]+=1
            else:
                pigdct[s]=1
        else:
            if opt[0]=='pop':
                if pigstack:
                    ot=pigstack.pop()
                    if ot in pigdct.keys():
                        pigdct[ot]-=1
            if opt[0]=='min':
                if pigstack:
                    while pigs:
                        t=pigs[0]
                        if pigdct[t]>0:
                            anslst.append(t)
                            break
                        else:
                            heapq.heappop(pigs)
#         print('pig heap',pigs)
#         print('pig dict',pigdct)
#         print('pig stack',pigstack)
#         print('answer list',anslst)
    except EOFError:
        break

for x in anslst:
    print(x)
```

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

pigs=[]
anslst=[]
pigdct={}
pigstack=[]
while True:
    try:
        opt=[str(x) for x in input().split()]
        if len(opt)==2:
            s=int(opt[1])
            heapq.heappush(pigs,s)
            pigstack.append(s)
            if s in pigdct.keys():
                pigdct[s]+=1
            else:
                pigdct[s]=1
        else:
            if opt[0]=='pop':
                if pigstack:
                    ot=pigstack.pop()
                    if ot in pigdct.keys():
                        pigdct[ot]-=1
            if opt[0]=='min':
                if pigstack:
                    while pigs:
                        t=pigs[0]
                        if pigdct[t]>0:
                            anslst.append(t)
                            break
                        else:
                            heapq.heappop(pigs)
#         print('pig heap',pigs)
#         print('pig dict',pigdct)
#         print('pig stack',pigstack)
#         print('answer list',anslst)
    except EOFError:
        break

for x in anslst:
    print(x)
```

基本信息

#: 44762
题目: 22067
提交人: zxk
内存: 7132k
时间: 275m
语言: Pytho
提交时间: 2024-

04123: 马走日

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

思路:

代码

```
#
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

28046: 词梯

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

思路：

我服了，从晚上11点半做到凌晨三点，一遍一遍改，各种问题；遂改不动，放弃了；

之后重新看了一遍，把测试数据放进去，发现是有一个NO写的No.....

！@#￥%.....&（）（&.....%￥#@#￥%.....&！@#￥%..... ￥#@%.....&（&*.....&%..... ￥%#

代码

```
class Graph:
    def __init__(self):
        self.vertices = {}
        self.numVertices = 0

    def addVertex(self, key):
        self.numVertices = self.numVertices + 1
        newVertex = Vertex(key)
        self.vertices[key] = newVertex
        return newVertex

    def __contains__(self, n):
        return n in self.vertices

    def addEdge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.addVertex(f)
        if t not in self.vertices:
            nv = self.addVertex(t)
        self.vertices[f].addNeighbor(self.vertices[t], cost)

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

    # def __lt__(self,o):
    #     return self.id < o.id

    def addNeighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def getConnections(self):
        return self.connectedTo.keys()

def buildgraph(data):
    buckets={}
    while data:
```

```
        ver=data.pop(0)
        for i in range(4):
            bkt=ver[:i]+'_'+ver[i+1:]
            if bkt.lower() in buckets.keys():
                buckets[bkt.lower()].append(ver)
            else:
                buckets[bkt.lower()]=[ver]

    g=Graph()
    for alst in buckets.values():
        for i in range(len(alst)):
            for j in range(i+1,len(alst)):
                g.addEdge(alst[i],alst[j])
                g.addEdge(alst[j],alst[i])
    return g

def bfs(graph,initial,end):
    visited=set()
    queue=[initial]
    bfstree={}
    while queue:
        node=queue.pop(0)
        if node not in graph.vertices:
            return 'NO'
#         print(f'current node is {node}')
        if node not in visited:
#             print(f'{node} not visited')
            visited.add(node)
#             print(f'current visited: {visited}')
            neighbours=[]
            current_node=graph.vertices[node]
            for neighbour in current_node.getConnections():
                neighbours.append(neighbour.id)
#             print(f'{node} has neighbours {neighbours}')
            for x in neighbours:
                if x not in visited and x not in queue:
                    bfstree[x]=node
                    queue.append(x)
#                 if x==end:
#                     queue.pop()
#                     break
#             print(f'current queue: {queue}')
#         else:
#             print(f'{node} is visited, do nothing')
#     print(bfstree)

    path=[end]
    while True:
        if path[-1]==begin:
            break
        if path[-1] not in bfstree.keys():
            return 'NO'
        else:
#             print(f'go through {bfstree[path[-1]]}')
            path.append(bfstree[path[-1]])
    path.reverse()
```

```
    return ' '.join(path)

n=int(input())
data=[]
for _ in range(n):
    data.append(input())
belst=[str(x) for x in input().split()]
begin,endd=belst[0],belst[1]
graph=buildgraph(data)
print(bfs(graph,begin,endd))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

题目 排名 状态 提问

#44762212提交状态 查看

状态: Accepted

源代码

```
class Graph:
    def __init__(self):
        self.vertices = {}
        self.numVertices = 0

    def addVertex(self, key):
        self.numVertices = self.numVertices + 1
        newVertex = Vertex(key)
        self.vertices[key] = newVertex
        return newVertex

    def __contains__(self, n):
        return n in self.vertices

    def addEdge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.addVertex(f)
        if t not in self.vertices:
            nv = self.addVertex(t)
        self.vertices[f].addNeighbor(self.vertices[t], cost)

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

    # def __lt__(self,o):
    #     return self.id < o.id

    def addNeighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def getConnections(self):
        return self.connectedTo.keys()

def buildgraph(data):
    buckets={}
```

基本信息

#: 44762
题目: 28046
提交人: zxk
内存: 8100k
时间: 206m
语言: Pytho
提交时间: 2024-

28050: 骑士周游

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

思路：

代码

```
#
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

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.树的转换：无思路，遂抄答案；感觉关键在于如何记录树的高度，u不一定上升，但是一定是从栈顶元素的高度向下继续行进

2.扩展二叉树：卡在了建树上，感觉之前学的树都忘干净了（，感觉关键在于用一个整体的列表建树，递归的操作时递归取表中的元素，这样递归可以保证每一层建树的结构

3.快速堆猪：开始的时候感觉数据量大不可能是简单的排序或者查找，后来发现有人好像也过了？换了个思路，用了各种数据结构...倒是记录的还不错，思路也没错；竟然卡在了int和str的格式区别上....当时还想是不是加一个把重量变成int，结果写到后边就给忘了.....数理课上写也写不专心（

4.词梯问题：用图、桶顺着课件的思路和代码做了一遍，发现其实没有那么好实现，后边的bfs卡了半天关于Vertex类和Graph类的用法，还去他那个包里的源代码看了半天才看懂怎么用；发现OJ竟然没有pythonds，于是从源代码直接复制了一堆下来（，将bfs写完后对于如何记录路径又写了半天，思路是构建一个bfs树，通过字典记录它的路径；最后依然不对，群里一直说大小写的问题，但是我不理解为什么大小写会有影响；于是测试数据放进去发现是NO打成了No...

这周太忙了，没有一点时间，各种作业和考试还有运动会的工作，稀里糊涂的，再加上这次作业的难度挺大的，就很崩溃

马走日和骑士周游没时间做了，先交上四个AC，等忙完这个星期的考试再补，下次一并上传