Assignment #6: "树"算：Huffman,BinHeap,BST,AVL,DisjointSet

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by ==狄晨阳 生命科学学院==

说明：

1）这次作业内容不简单，耗时长的话直接参考题解。

2）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。

3）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。

4）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：Windows11

Python编程环境：Spyder IDE 5.4.3

C/C++编程环境：无

1. 题目

22275: 二叉搜索树的遍历

http://cs101.openjudge.cn/practice/22275/

思路：注意到节点的数字大小顺序即中序遍历的顺序，然后就和上次作业前中推后一样了

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 15:00:55 2024

@author: 20311
"""

n=int(input())
```

```python
qx=list(map(int,input().split()))
zx=[x for x in range(1,n+1)]

def build(qx,zx):
    if not qx or not zx:
        return []

    root=qx[0]
    d=zx.index(root)

    qx_left=qx[1:d+1]
    qx_right=qx[d+1:]

    zx_left=zx[:d]
    zx_right=zx[d+1:]

    hx=build(qx_left,zx_left)
    hx.extend(build(qx_right,zx_right))
    hx.append(root)

    return hx

hx=build(qx,zx)
print(' '.join(map(str,hx)))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 15:00:55 2024

@author: 20311
"""

n=int(input())
qx=list(map(int,input().split()))
zx=[x for x in range(1,n+1)]

def build(qx,zx):
    if not qx or not zx:
        return []

    root=qx[0]
    d=zx.index(root)

    qx_left=qx[1:d+1]
    qx_right=qx[d+1:]

    zx_left=zx[:d]
    zx_right=zx[d+1:]

    hx=build(qx_left,zx_left)
    hx.extend(build(qx_right,zx_right))
    hx.append(root)

    return hx

hx=build(qx,zx)
print(' '.join(map(str,hx)))
```

基本信息

#:	44431608
题目:	22275
提交人:	23n2300012138(yukino)
内存:	3964kB
时间:	26ms
语言:	Python3
提交时间:	2024-03-28 15:14:41

05455: 二叉搜索树的层次遍历

http://cs101.openjudge.cn/practice/05455/

思路：依旧是中序遍历建成的树就是大小顺序，只要小于当前节点的放左儿子，大于的放右儿子即可

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 15:19:47 2024

@author: 20311
"""

class Tree:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None

def build(node,value):
    if node is None:
        return Tree(value)
    if value<node.value:
        node.left=build(node.left,value)
    elif value>node.value:
        node.right=build(node.right,value)
    return node

def lever_order_traverse(root):
    stack=[root]
    ans=[]

    while stack:
        node=stack.pop(0)
        ans.append(node.value)
        if node.left:
            stack.append(node.left)
        if node.right:
            stack.append(node.right)

    return ans

nums=list(map(int,input().split()))
nums=list(dict.fromkeys(nums))
root=None
for num in nums:
    root=build(root,num)
ans=lever_order_traverse(root)
print(' '.join(map(str,ans)))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态：Accepted

基本信息

#：44432068
题目：05455
提交人：23n2300012138(yukino)
内存：3672kB
时间：25ms
语言：Python3
提交时间：2024-03-28 15:57:55

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 15:19:47 2024

@author: 20311
"""

class Tree:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None

def build(node,value):
    if node is None:
        return Tree(value)
    if value<node.value:
        node.left=build(node.left,value)
    elif value>node.value:
        node.right=build(node.right,value)
    return node

def lever_order_traverse(root):
    stack=[root]
    ans=[]

    while stack:
        node=stack.pop(0)
        ans.append(node.value)
        if node.left:
            stack.append(node.left)
        if node.right:
            stack.append(node.right)

    return ans

nums=list(map(int,input().split()))
nums=list(dict.fromkeys(nums))
root=None
for num in nums:
    root=build(root,num)
ans=lever_order_traverse(root)
print(' '.join(map(str,ans)))
```

04078: 实现堆结构

http://cs101.openjudge.cn/practice/04078/

练习自己写个BinHeap。当然机考时候，如果遇到这样题目，直接import heapq。手搓栈、队列、堆、AVL等，考试前需要搓个遍。

思路：练习了一下手搓堆

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 16:20:42 2024

@author: 20311
"""
```

```python
class Heap:
    def __init__(self,array=[],less=lambda x,y:x<y):
        self.a=array[:]
        self.size=len(array)
        self.less=less

    def push(self,i):
        self.size+=1
        self.a.append(i)
        self.up(self.size-1)

    def pop(self):
        temp=self.a[0]
        self.a[0]=self.a[-1]
        self.a.pop()
        self.size-=1
        self.down(0)
        return temp

    def up(self,i):
        if i==0:
            return
        f=(i-1)//2

        if self.less(self.a[i],self.a[f]):
            self.a[i],self.a[f]=self.a[f],self.a[i]
            self.up(f)

    def down(self,i):
        if i*2+1>=self.size:
            return
        l,r=i*2+1,i*2+2
        if r>=self.size or self.less(self.a[l],self.a[r]):
            s=l
        else:
            s=r

        if self.less(self.a[s],self.a[i]):
            self.a[i],self.a[s]=self.a[s],self.a[i]
            self.down(s)

n=int(input())
heap=None
for _ in range(n):
    s=input()
    if s[0]=='1':
        if heap is None:
            heap=Heap([int(s[2:])])
        else:
            heap.push(int(s[2:]))
    if s[0]=='2':
        print(heap.pop())
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态：Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 16:20:42 2024

@author: 20311
"""

class Heap:
    def __init__(self,array=[],less=lambda x,y:x<y):
        self.a=array[:]
        self.size=len(array)
        self.less=less

    def push(self,i):
        self.size+=1
        self.a.append(i)
        self.up(self.size-1)

    def pop(self):
        temp=self.a[0]
        self.a[0]=self.a[-1]
        self.a.pop()
        self.size-=1
        self.down(0)
        return temp

    def up(self,i):
        if i==0:
            return
        f=(i-1)//2

        if self.less(self.a[i],self.a[f]):
            self.a[i],self.a[f]=self.a[f],self.a[i]
            self.up(f)

    def down(self,i):
        if i*2+1>=self.size:
            return
        l,r=i*2+1,i*2+2
        if r>=self.size or self.less(self.a[l],self.a[r]):
            s=l
        else:
            s=r

        if self.less(self.a[s],self.a[i]):
            self.a[i],self.a[s]=self.a[s],self.a[i]
            self.down(s)

n=int(input())
heap=None
for _ in range(n):
    s=input()
    if s[0]=='1':
        if heap is None:
            heap=Heap([int(s[2:])])
        else:
            heap.push(int(s[2:]))
    if s[0]=='2':
        print(heap.pop())
```

基本信息

#：	44433101
题目：	04078
提交人：	23n2300012138(yukino)
内存：	4152kB
时间：	714ms
语言：	Python3
提交时间：	2024-03-28 16:51:41

22161: 哈夫曼编码树

http://cs101.openjudge.cn/practice/22161/

思路：引入堆与树，用堆来实现排序，用函数合并，建树后遍历找出对应的编码与解码

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 18:57:20 2024

@author: 20311
"""
def change(x):
    x=x.value
    y=x[0].copy()
    y.sort()
    return ord(y[0])/1000+x[1]

def less(x,y):
    if change(x)<change(y):
        return True
    else:
        return False

def build():
    global tree
    while tree.size>1:
        x=tree.pop()
        y=tree.pop()
        f=combine(x,y)
        if less(x,y):
            f.left=x
            f.right=y
        else:
            f.right=x
            f.left=y
        tree.push(f)
    return tree

def combine(x,y):
    if less(x,y):
        b=x.value[0]+y.value[0]
    else:
        b=y.value[0]+x.value[0]
    a=x.value[1]+y.value[1]
    return Tree([b,a])

def traverse(node,i):
    global dic
    if not node.left and not node.right:
        dic[i]=node.value[0][0]
        return

    a=i+'0'
    b=i+'1'
    if node.left:
        traverse(node.left,a)

    if node.right:
        traverse(node.right,b)
```

```
class Heap:
    def __init__(self,array=[],less=lambda x,y:less(x,y)):
        self.a=array[:]
        self.size=len(array)
        self.less=less

    def push(self,i):
        self.size+=1
        self.a.append(i)
        self.up(self.size-1)

    def pop(self):
        temp=self.a[0]
        self.a[0]=self.a[-1]
        self.a.pop()
        self.size-=1
        self.down(0)
        return temp

    def up(self,i):
        if i==0:
            return
        f=(i-1)//2

        if self.less(self.a[i],self.a[f]):
            self.a[i],self.a[f]=self.a[f],self.a[i]
            self.up(f)

    def down(self,i):
        if i*2+1>=self.size:
            return
        l,r=i*2+1,i*2+2
        if r>=self.size or self.less(self.a[l],self.a[r]):
            s=l
        else:
            s=r

        if self.less(self.a[s],self.a[i]):
            self.a[i],self.a[s]=self.a[s],self.a[i]
            self.down(s)

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

n=int(input())
tree=[]
for _ in range(n):
    a,b=input().split()
    b=int(b)
    if not tree:
        tree.append(Tree([[a],b]))
        tree=Heap(tree)
    else:
```

```
        tree.push(Tree([[a],b]))

hoffman_tree=build()
hoffman_tree=hoffman_tree.pop()
dic={}
traverse(hoffman_tree.left,'0')
traverse(hoffman_tree.right,'1')
Dic={}
for d in dic:
    Dic[dic[d]]=d

while True:
    try:
        s=input()
        ans=[]
        tem=''
        for ss in s:
            tem+=ss
            if tem in dic:
                ans.append(dic[tem])
                tem=''
            elif tem in Dic:
                ans.append(Dic[tem])
                tem=''
        print(''.join(ans))
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44440024提交状态

状态: Accepted

基本信息
#: 44440024
题目: 22161
提交人: 23n2300012138(yukino)
内存: 3896kB
时间: 25ms
语言: Python3
提交时间: 2024-03-28 23:20:06

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 18:57:20 2024

@author: 20311
"""
def change(x):
    x=x.value
    y=x[0].copy()
    y.sort()
    return ord(y[0])/1000+x[1]

def less(x,y):
    if change(x)<change(y):
        return True
    else:
        return False

def build():
    global tree
    while tree.size>1:
        x=tree.pop()
        y=tree.pop()
        f=combine(x,y)
        if less(x,y):
            f.left=x
            f.right=y
        else:
            f.right=x
            f.left=y
        tree.push(f)
    return tree

def combine(x,y):
    if less(x,y):
        b=x.value[0]+y.value[0]
    else:
        b=y.value[0]+x.value[0]
    a=x.value[1]+y.value[1]
    return Tree([b,a])

def traverse(node,i):
    global dic
    if not node.left and not node.right:
        dic[i]=node.value[0][0]
        return

    a=i+'0'
    b=i+'1'
    if node.left:
        traverse(node.left,a)

    if node.right:
        traverse(node.right,b)


class Heap:
    def __init__(self,array=[],less=lambda x,y:less(x,y)):
        self.a=array[:]
        self.size=len(array)
        self.less=less

    def push(self,i):
        self.size+=1
        self.a.append(i)
        self.up(self.size-1)

    def pop(self):
        temp=self.a[0]
        self.a[0]=self.a[-1]
        self.a.pop()
        self.size-=1
        self.down(0)
        return temp

    def up(self,i):
        if i==0:
            return
        f=(i-1)//2

        if self.less(self.a[i],self.a[f]):
            self.a[i],self.a[f]=self.a[f],self.a[i]
            self.up(f)

    def down(self,i):
        if i*2+1>=self.size:
            return
        l,r=i*2+1,i*2+2
        if r>=self.size or self.less(self.a[l],self.a[r]):
            s=l
        else:
            s=r

        if self.less(self.a[s],self.a[i]):
            self.a[i],self.a[s]=self.a[s],self.a[i]
            self.down(s)

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

n=int(input())
tree=[]
for _ in range(n):
    a,b=input().split()
    b=int(b)
    if not tree:
        tree.append(Tree([[a],b]))
        tree=Heap(tree)
    else:
        tree.push(Tree([[a],b]))

hoffman_tree=build()
hoffman_tree=hoffman_tree.pop()
dic={}
traverse(hoffman_tree.left,'0')
traverse(hoffman_tree.right,'1')
Dic={}
for d in dic:
    Dic[dic[d]]=d

while True:
    try:
        s=input()
        ans=[]
        tem=''
        for ss in s:
            tem+=ss
            if tem in dic:
                ans.append(dic[tem])
                tem=''
            elif tem in Dic:
                ans.append(Dic[tem])
                tem=''
        print(''.join(ans))
    except EOFError:
        break
```

晴问9.5: 平衡二叉树的建立

https://sunnywhy.com/sfbj/9/5/359

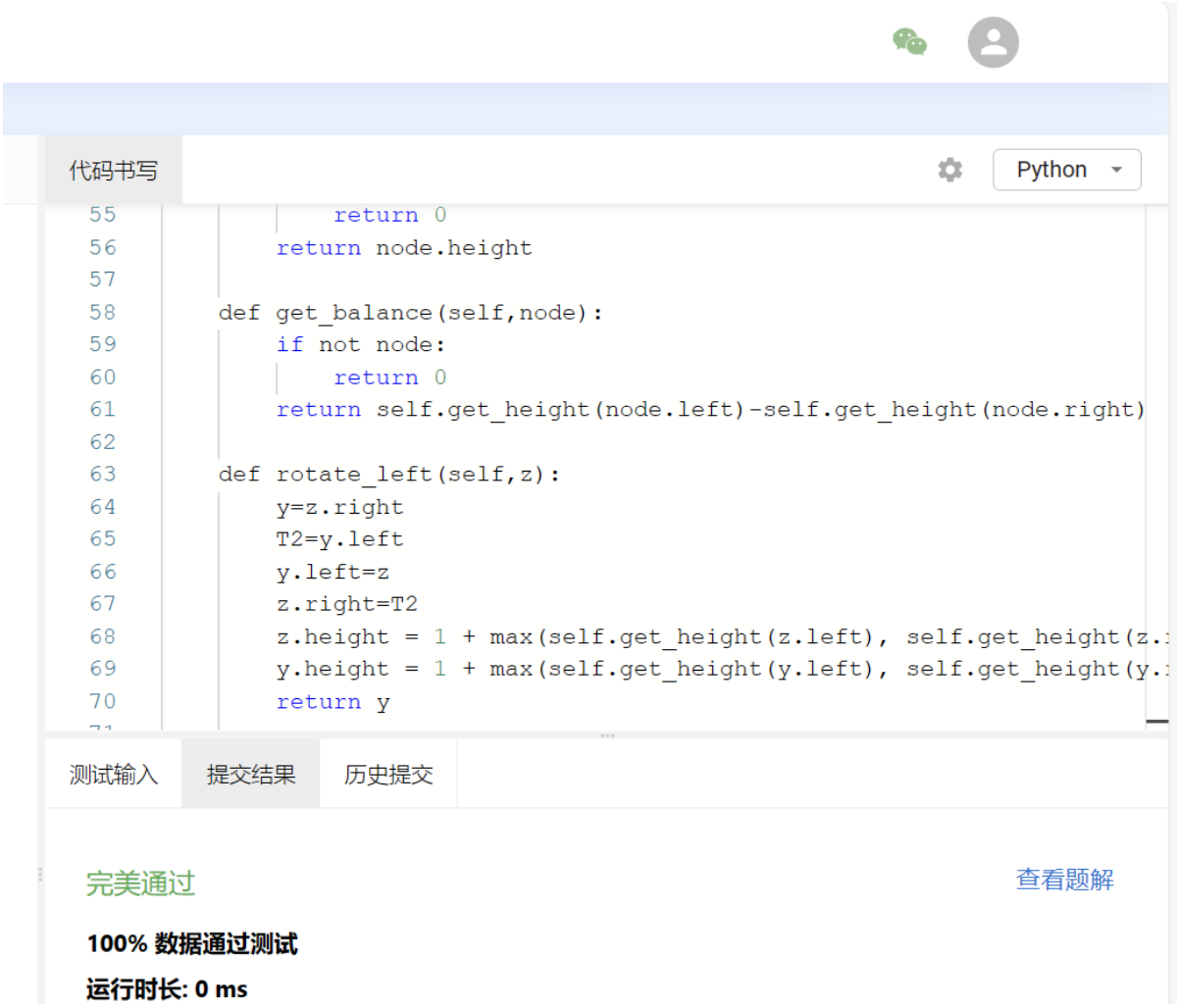
思路：一开始自己写了一个，直接在node中进行操作，好像会导致指向同一个目标而递归出现死循环，后来看了题解，又添加了一个avl类，在内部使用函数处理，用更好的写法避免引入反向的指向导致死循环

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Sat Mar 30 21:17:48 2024

@author: 20311
"""

class Node:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
        self.height=1

class Avl:
    def __init__(self):
        self.root=None

    def insert(self,value):
        if not self.root:
            self.root=Node(value)
        else:
            self.root=self._insert(value,self.root)

    def _insert(self,value,node):
        if not node:
            return Node(value)
        elif value<node.value:
            node.left=self._insert(value, node.left)
        else:
            node.right=self._insert(value, node.right)

        node.height=1+max(self.get_height(node.left),self.get_height(node.right))

        balance=self.get_balance(node)

        if balance>1:
            if value < node.left.value:
                return self.rotate_right(node)
            else:
```

```
                node.left=self.rotate_left(node.left)
                return self.rotate_right(node)

        if balance<-1:
            if value > node.right.value:
                return self.rotate_left(node)
            else:
                node.right=self.rotate_right(node.right)
                return self.rotate_left(node)

        return node

    def get_height(self,node):
        if not node:
            return 0
        return node.height

    def get_balance(self,node):
        if not node:
            return 0
        return self.get_height(node.left)-self.get_height(node.right)

    def rotate_left(self,z):
        y=z.right
        T2=y.left
        y.left=z
        z.right=T2
        z.height = 1 + max(self.get_height(z.left), self.get_height(z.right))
        y.height = 1 + max(self.get_height(y.left), self.get_height(y.right))
        return y

    def rotate_right(self,y):
        x=y.left
        T2=x.right
        x.right=y
        y.left=T2
        y.height = 1 + max(self.get_height(y.left), self.get_height(y.right))
        x.height = 1 + max(self.get_height(x.left), self.get_height(x.right))
        return x

    def traverse(self):
        return self._traverse(self.root)

    def _traverse(self,node):
        if not node:
            return []
        return [node.value]+self._traverse(node.left)+self._traverse(node.right)

n=int(input())
nums=list(map(int,input().strip().split()))

avl=Avl()
for num in nums:
    avl.insert(num)
print(' '.join(map(str,avl.traverse())))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



02524: 宗教信仰

http://cs101.openjudge.cn/practice/02524/

思路：采用字典来指向相同的信仰者，如果第一次出现即信仰编号与自身相等则加一

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Sat Mar 30 23:56:14 2024

@author: 20311
"""

def combine(a,b):
    global faiths
    f1=father(a)
```

```
    f2=father(b)
    if f1==f2:
        return
    else:
        faiths[f1]=f2

def father(a):
    global faiths
    if faiths[a]!=a:
        return father(faiths[a])
    return a

t=0
while True:
    ans=0
    t+=1
    n,m=map(int,input().strip().split())
    faiths={x:x for x in range(1,n+1)}
    if n==m==0:
        break

    for _ in range(m):
        a,b=map(int,input().strip().split())
        combine(a,b)

    for x in faiths:
        if faiths[x]==x:
            ans+=1

    print('Case {}: {}'.format(t,ans))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

基本信息

#: 44474429
题目: 02524
提交人: 23n2300012138(yukino)
内存: 11316kB
时间: 1620ms
语言: Python3
提交时间: 2024-03-31 00:05:54

源代码

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 30 23:56:14 2024

@author: 20311
"""

def combine(a,b):
    global faiths
    f1=father(a)
    f2=father(b)
    if f1==f2:
        return
    else:
        faiths[f1]=f2

def father(a):
    global faiths
    if faiths[a]!=a:
        return father(faiths[a])
    return a

t=0
while True:
    ans=0
    t+=1
    n,m=map(int,input().strip().split())
    faiths={x:x for x in range(1,n+1)}
    if n==m==0:
        break

    for _ in range(m):
        a,b=map(int,input().strip().split())
        combine(a,b)

    for x in faiths:
        if faiths[x]==x:
            ans+=1

    print('Case {}: {}'.format(t,ans))
```

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。==

有

本次题目还是比较困难的，有几道题目需要看题解的思路才能好写一点，更加熟练了类的运用，对调用函数等也有了更深的理解，同时对树的认知也更深了，还学习了二叉搜索树、平衡二叉树等概念