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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by 余汶青 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

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

22275: 二叉搜索树的遍历

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

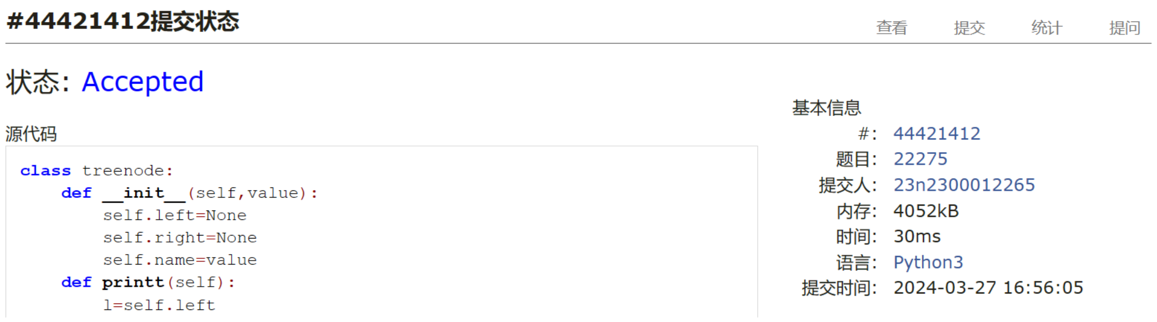
思路：

代码

```
class treenode:
```

```
class treenode:
    def __init__(self,value):
        self.left=None
        self.right=None
        self.name=value
    def printt(self):
        l=self.left
        r=self.right
        if l:
            l.printt()
        if r:
            r.printt()
        print(self.name,end=' ')

def build(pre):
    if len(pre)==0:
        return
    if len(pre)==1:
        node=treenode(pre[0])
        return node
    root=treenode(pre[0])
    k=0
    for i in range(len(pre)):
        if pre[i]>root.name:
            k=i
            break
    if k!=0:
        root.right=build(pre[k:])
        root.left=build(pre[1:k])
    else:
        root.left=build(pre[1:])
    return root

n=int(input())
pre=[int(i) for i in input().split()]
a=build(pre)
a.printt()
```

代码运行截图



#44421412提交状态

状态: Accepted

源代码

```
class treenode:
    def __init__(self,value):
        self.left=None
        self.right=None
        self.name=value
    def printt(self):
        l=self.left
```

基本信息

#:	44421412
题目:	22275
提交人:	23n2300012265
内存:	4052kB
时间:	30ms
语言:	Python3
提交时间:	2024-03-27 16:56:05

05455: 二叉搜索树的层次遍历

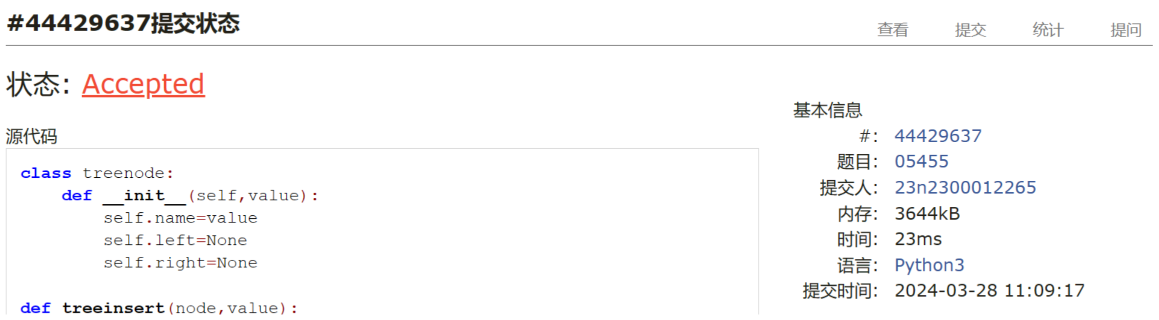
http://cs101.openjudge.cn/practice/05455/

思路:

代码

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

def treeinsert(node,value):
    if node==None:
        return treenode(value)
    if value<node.name:
        node.left=treeinsert(node.left, value)
    if value>node.name:
        node.right=treeinsert(node.right, value)
    return node

def level_order_traversal(node):
    queue=[node]
    traversal=[]
    while queue:
        a=queue.pop(0)
        traversal.append(a.name)
        if a.left:
            queue.append(a.left)
        if a.right:
            queue.append(a.right)
    return traversal


s=[int(i) for i in input().split()]
root=None
for i in range(len(s)):
    root=treeinsert(root, s[i])
traversal=level_order_traversal(root)
print(*traversal,sep=' ')
```

代码运行截图



#44429637提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class treenode:
    def __init__(self,value):
        self.name=value
        self.left=None
        self.right=None

def treeinsert(node,value):
```

基本信息

#:	44429637
题目:	05455
提交人:	23n2300012265
内存:	3644kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-28 11:09:17

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。

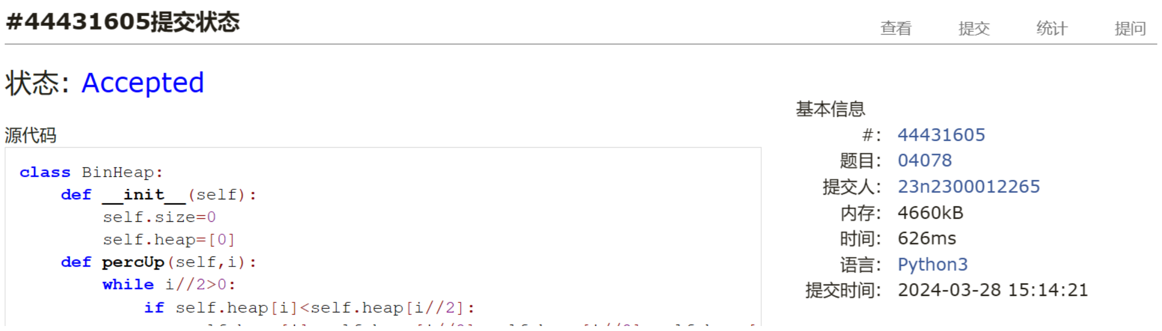
思路:

代码

```
class BinHeap:
    def __init__(self):
        self.size=0
        self.heap=[0]
    def percUp(self,i):
        while i//2>0:
            if self.heap[i]<self.heap[i//2]:
                self.heap[i],self.heap[i//2]=self.heap[i//2],self.heap[i]
            i//=2
    def insert(self,value):
        self.size+=1
        self.heap.append(value)
        self.percUp(self.size)
    def percDown(self,i):
        while i*2<=self.size:
            mc=self.minchild(i)
            if self.heap[i]>self.heap[mc]:
                self.heap[i],self.heap[mc]=self.heap[mc],self.heap[i]
            i=mc
    def minchild(self,i):
        if i*2+1>self.size:
            return i*2
        if self.heap[i*2]<self.heap[i*2+1]:
            return i*2
        else:
            return i*2+1
    def delmin(self):
        a=self.heap[1]
        self.heap[1]=self.heap[-1]
        self.heap.pop()
        self.size-=1
```

```
            self.percDown(1)
            return a

n=int(input())
a=BinHeap()
for i in range(n):
    s=input()
    if s[0]=='1':
        b=int(s[2:])
        a.insert(b)
    if s[0]=='2':
        print(a.delmin())
```

代码运行截图


#44431605提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class BinHeap:
    def __init__(self):
        self.size=0
        self.heap=[0]
    def percUp(self,i):
        while i//2>0:
            if self.heap[i]<self.heap[i//2]:
```

基本信息

#:	44431605
题目:	04078
提交人:	23n2300012265
内存:	4660kB
时间:	626ms
语言:	Python3
提交时间:	2024-03-28 15:14:21

22161: 哈夫曼编码树

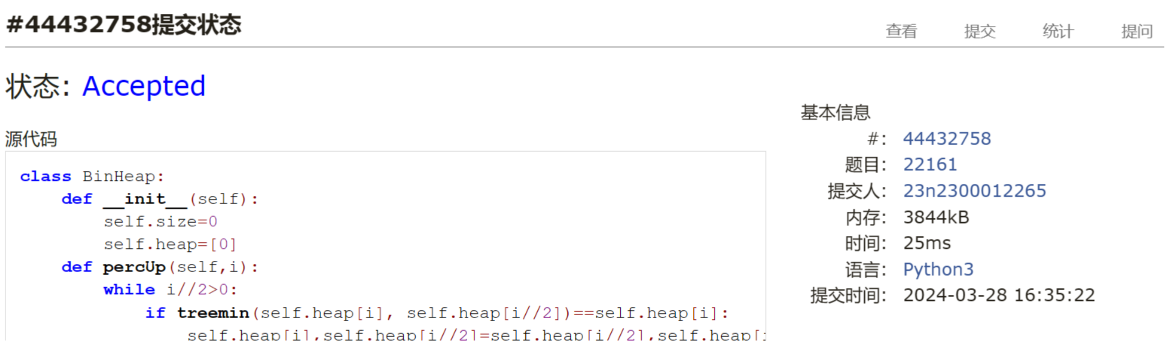
http://cs101.openjudge.cn/practice/22161/

思路:

代码

```
class BinHeap:
    def __init__(self):
        self.size=0
        self.heap=[0]
    def percUp(self,i):
        while i//2>0:
            if treemin(self.heap[i], self.heap[i//2])==self.heap[i]:
                self.heap[i],self.heap[i//2]=self.heap[i//2],self.heap[i]
            i//=2
    def insert(self,value):
        self.size+=1
        self.heap.append(value)
        self.percUp(self.size)
    def percDown(self,i):
        while i*2<=self.size:
            mc=self.minchild(i)
            if treemin(self.heap[i], self.heap[mc])==self.heap[mc]:
                self.heap[i],self.heap[mc]=self.heap[mc],self.heap[i]
            i=mc
    def minchild(self,i):
        if i*2+1>self.size:
            return i*2
        if treemin(self.heap[i*2], self.heap[i*2+1])==self.heap[i*2]:
            return i*2
        else:
            return i*2+1
    def delmin(self):
        a=self.heap[1]
        self.heap[1]=self.heap[-1]
        self.heap.pop()
        self.size-=1
        self.percDown(1)
        return a
    def buildheap(self,alist):
        i=len(alist)//2
        self.size=len(alist)
        self.heap=[0]+alist[:]
        while i>0:
            self.percDown(i)
            i-=1


class treenode:
    def __init__(self,value,name):
        self.left=None
        self.right=None
        self.father=None
        self.value=value
        self.name=name
def treemin(a,b):
    if a.value<b.value:
        return a
    if a.value==b.value:
        if a.name<b.name:
            return a
    return b

def buildtree(node):
    while node.size>1:
        a=node.delmin()
        b=node.delmin()
        c=treenode(a.value+b.value,None)
        a.father=c
        b.father=c
        c.left=a
        c.right=b
        node.insert(c)
    return node.delmin()

def findchr(root,a):
    ans=[]
    node=root
    for i in range(len(a)):
        if a[i]=='0':
            node=node.left
        else:
            node=node.right
        if node.name!=None:
            ans.append(node.name)
            node=root
    return ans

def findord(c,node,a):
    ans=[]
    for i in range(len(a)):
        for j in node:
            if j.name==a[i]:
                q=[]
                while j!=c:
                    root=j.father
                    if root.left==j:
                        q.append(0)
                    else:
                        q.append(1)
                    j=root
                q.reverse()
                ans.extend(q)
    return ans

n=int(input())
nodee=[]
heap=BinHeap()
for i in range(n):
    a,b=input().split()
    nodee.append(treenode(int(b),a))
heap.buildheap(nodee)
root=buildtree(heap)
#print(root.right.value,root.right.father.value)
while True:
    try:
        a=input()
        if a[0]=='0' or a[0]=='1':
            print(*findchr(root,a),sep='')
        else:
            print(*findord(root,nodee,a),sep='')
    except:
        break
```

代码运行截图


#44432758提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class BinHeap:
    def __init__(self):
        self.size=0
        self.heap=[0]
    def percUp(self,i):
        while i//2>0:
            if treemin(self.heap[i], self.heap[i//2])==self.heap[i]:
                self.heap[i],self.heap[i//2]=self.heap[i//2],self.heap[:
```

基本信息

#:	44432758
题目:	22161
提交人:	23n2300012265
内存:	3844kB
时间:	25ms
语言:	Python3
提交时间:	2024-03-28 16:35:22

晴问9.5: 平衡二叉树的建立

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

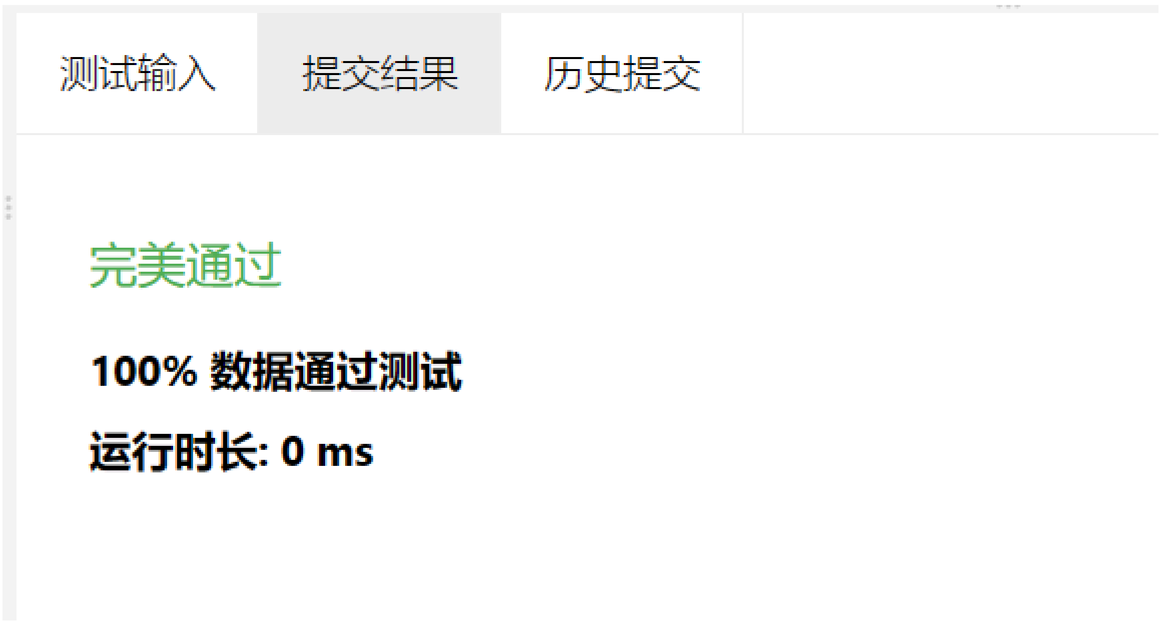
思路:

因为把elif写成if，找了一个半小时bug，想死

代码

```
ans=[]
class treenode:
    def __init__(self,value):
        self.value=value
        self.father=None
        self.rightleaf=None
        self.leftleaf=None
        self.balancefactor=0
    def preprint(self):
        global ans
        ans.append(self.value)
        if self.leftleaf!=None:
            self.leftleaf.preprint()
        if self.rightleaf!=None:
            self.rightleaf.preprint()

class AVLtree:
    def __init__(self):
        self.size=0
        self.root=None
    def put(self,node):
        if self.root!=None:
            self._put(node,self.root)
        else:
            self.root=node
        self.size+=1
    def _put(self,node,root):
        if node.value<root.value:
            if root.leftleaf!=None:
                self._put(node, root.leftleaf)
            else:
                root.leftleaf=node
                node.father=root
                self.updatebalance(node)
        else:
            if root.rightleaf!=None:
                self._put(node, root.rightleaf)
            else:
                root.rightleaf=node
                node.father=root
                self.updatebalance(node)
    def updatebalance(self,node):
        if node.balancefactor<-1 or node.balancefactor>1:
            self.rebalance(node)
            return
        if node.father!=None:
            if node.father.leftleaf==node:
                node.father.balancefactor+=1
            else:
                node.father.balancefactor-=1
            if node.father.balancefactor!=0:
                self.updatebalance(node.father)
    def rebalance(self,node):
        if node.balancefactor<0:
            if node.rightleaf.balancefactor>0:
                self.rotateright(node.rightleaf)
                self.rotateleft(node)
            else:
                self.rotateleft(node)
        elif node.balancefactor>0:
            if node.leftleaf.balancefactor<0:
                self.rotateleft(node.leftleaf)
                self.rotateright(node)
            else:
                self.rotateright(node)

    def rotateleft(self,node):
        root=node.rightleaf
        node.rightleaf=root.leftleaf
        if root.leftleaf!=None:
            root.leftleaf.father=node
        root.father=node.father
        if node==self.root:
            self.root=root
        else:
            if node==node.father.leftleaf:
                node.father.leftleaf=root
            else:
                node.father.rightleaf=root
        root.leftleaf=node
        node.father=root
        node.balancefactor=node.balancefactor+1-min(root.balancefactor,0)
        root.balancefactor=root.balancefactor+1+max(node.balancefactor,0)
    def rotateright(self,node):
        root=node.leftleaf
        node.leftleaf=root.rightleaf
        if root.rightleaf!=None:
            root.rightleaf.father=node
        root.father=node.father
        if node==self.root:
            self.root=root
        else:
            if node==node.father.leftleaf:
                node.father.leftleaf=root
            else:
                node.father.rightleaf=root
        root.rightleaf=node
        node.father=root
        node.balancefactor=node.balancefactor-1-max(root.balancefactor,0)
        root.balancefactor=root.balancefactor-1+min(node.balancefactor,0)


n=int(input())
s=[int(i) for i in input().split()]
tree=AVLtree()
for i in s:
    node=treenode(i)
    tree.put(node)
tree.root.preprint()
print(*ans,sep=' ')
```

代码运行截图



完美通过

100% 数据通过测试

运行时长: 0 ms

02524: 宗教信仰

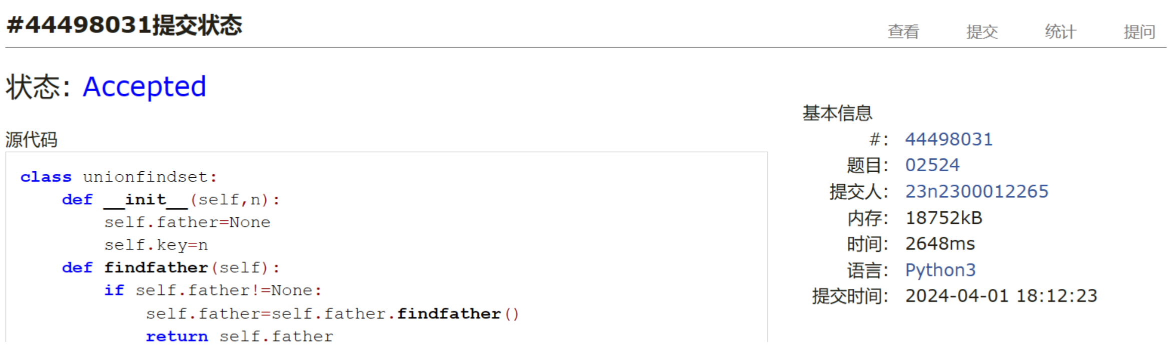
http://cs101.openjudge.cn/practice/02524/

思路:

代码

```python
class unionfindset:
    def __init__(self,n):
        self.father=None
        self.key=n
    def findfather(self):
        if self.father!=None:
            self.father=self.father.findfather()
            return self.father
        else:
            return self
def union(node1,node2):
    father1=node1.findfather()
    father2=node2.findfather()
    if father1!=father2:
        father1.father=father2
n,m=map(int,input().split())
ans=1
while n!=0:
    tree=[]
    for i in range(1,n+1):
        tree.append(unionfindset(i))
    for _ in range(m):
        a,b=map(int,input().split())
        union(tree[a-1],tree[b-1])
    father=[]
    for i in tree:
        a=i.findfather()
        if a not in father:
            father.append(a)
    print("Case %d: %d"%(ans,len(father)))
    ans+=1
    n,m=map(int,input().split())
```

代码运行截图


#44498031提交状态

状态: Accepted

源代码

```python
class unionfindset:
    def __init__(self,n):
        self.father=None
        self.key=n
    def findfather(self):
        if self.father!=None:
            self.father=self.father.findfather()
            return self.father
```

基本信息

#:	44498031
题目:	02524
提交人:	23n2300012265
内存:	18752kB
时间:	2648ms
语言:	Python3
提交时间:	2024-04-01 18:12:23

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。==

树终于结束了，这次作业让我手搓了无比长无比复杂的各种树，累死了累死了累死了

当然，对树的理解也有了一点提升

终于，可以，开始，学，图 ，了!