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

Updated 2214 GMT+8 March 24, 2024

2024 spring, Complied by <mark>王申睿——物理学院</mark>

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 typorahttps://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: 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)

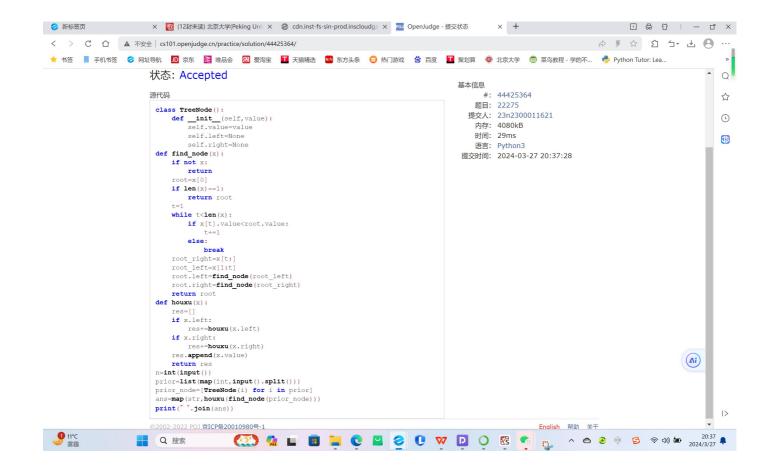
1. 题目

22275: 二叉搜索树的遍历

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

```
代码
class TreeNode():
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def find_node(x):
    if not x:
        return
    root=x[0]
```

```
if len(x) = = 1:
     return root
  t=1
  while t<len(x):
     if x[t].value<root.value:
     else:
        break
  root right=x[t:]
  root left=x[1:t]
  root.left=find_node(root_left)
  root.right=find_node(root_right)
  return root
def houxu(x):
  res=[]
  if x.left:
     res+=houxu(x.left)
  if x.right:
     res+=houxu(x.right)
  res.append(x.value)
  return res
n=int(input())
prior=list(map(int,input().split()))
prior_node=[TreeNode(i) for i in prior]
ans=map(str,houxu(find_node(prior_node)))
print(" ".join(ans))
```



05455: 二叉搜索树的层次遍历

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

思路:
代码
from collections import deque
origin=deque(map(int,input().split()))
mid=[]
while origin:
if origin[0] in mid:
origin.popleft()
else:
mid.append(origin.popleft())

class TreeNode():
definit(self,value):
self.value=value
self.left=None
self.right=None
mid2=[TreeNode(i) for i in mid]
final=deque(mid2)
def dfs_buildtree(root=None):
global final
if root is None:
root=final.popleft()
return root
if final[0].value <root.value:< td=""></root.value:<>

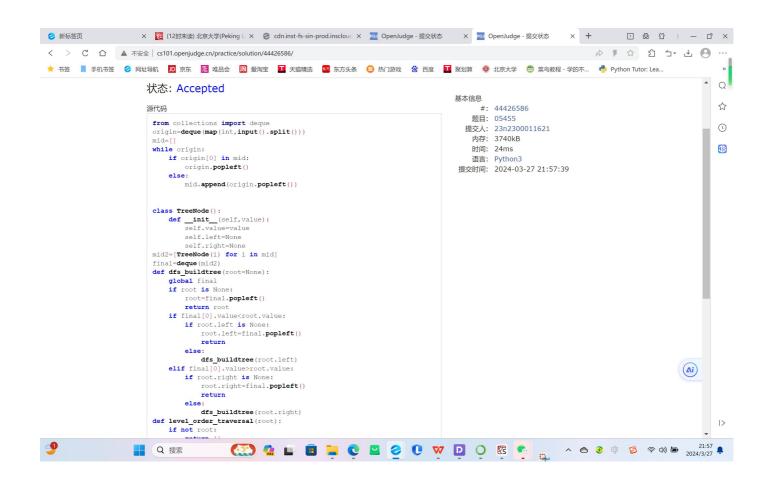
if root.left is None:		
root.left=final.popleft()		
return		
else:		
dfs_buildtree(root.left)		
elif final[0].value>root.value:		
if root.right is None:		
root.right=final.popleft()		
return		
else:		
dfs_buildtree(root.right)		
def level_order_traversal(root):		
if not root:		
return []		

```
result = []
queue = deque([root])
while queue:
  level_size = len(queue)
  current_level = []
  for _ in range(level_size):
     node = queue.popleft()
     current_level.append(node.value)
     if node.left:
```

queue.append(node.left)

```
if node.right:
          queue.append(node.right)
     result.append(current_level)
  return result
root=dfs_buildtree()
while final:
  dfs_buildtree(root)
res1=level_order_traversal(root)
res=[]
for i in range(len(res1)):
  res+=res1[i]
ans=map(str,res)
```

print(" ".join(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:

代码

```
class Binheap():
  def __init__(self):
     self.heaplist=[0]
  def floatup(self,i):
     while i//2>0:
       if self.heaplist[i]<self.heaplist[i//2]:#交换的条件
          self.heaplist[i],self.heaplist[i//2]=self.heaplist[i//2],self.heaplist[i]
       i=i//2
  def insert(self,i):
     self.heaplist.append(i)
     self.floatup(len(self.heaplist)-1)#交换到合适的位置
  def minchild(self,i):
     if i * 2 + 1 > len(self.heaplist)-1:
```

```
return i * 2
  else:
     if self.heaplist[i * 2] < self.heaplist[i * 2 + 1]:
        return i * 2
     else:
        return i * 2 + 1
def sinkdown(self,i):
  while (i*2)<=len(self.heaplist)-1:
     choice=self.minchild(i)
     if self.heaplist[i]>self.heaplist[choice]:#维护最小性
        self.heap list[i], self.heap list[choice] = self.heap list[choice], self.heap list[i] \\
     i=choice
def delmin(self):
  previous=self.heaplist[1]
```

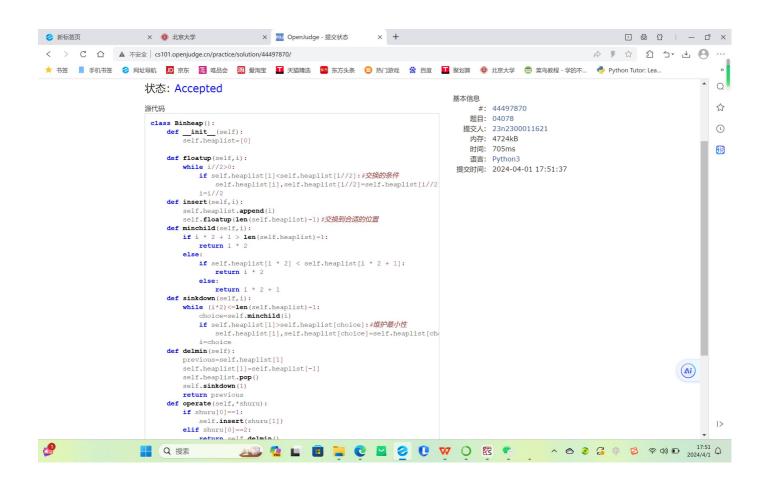
```
self.heaplist[1]=self.heaplist[-1]
     self.heaplist.pop()
     self.sinkdown(1)
     return previous
  def operate(self,*shuru):
     if shuru[0]==1:
       self.insert(shuru[1])
     elif shuru[0]==2:
       return self.delmin()
n=int(input())
S=Binheap()
for _ in range(n):
  m=input()
  if len(m)>=2:
```

a,b=map(int,m.split())

S.operate(a,b)

else:

print(S.operate(int(m)))

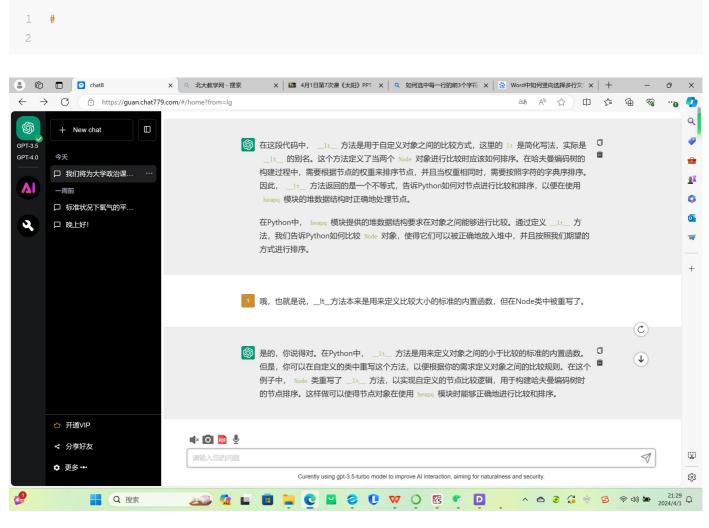


22161: 哈夫曼编码树

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

思路:

代码



(22161: 哈夫曼编码,已阅读题解)

晴问9.5: 平衡二叉树的建立

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

思路:

```
1 #
2
```

代码运行截图 (AC代码截图,至少包含有"Accepted")

(已查看题解)

02524: 宗教信仰

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

思路:

۰

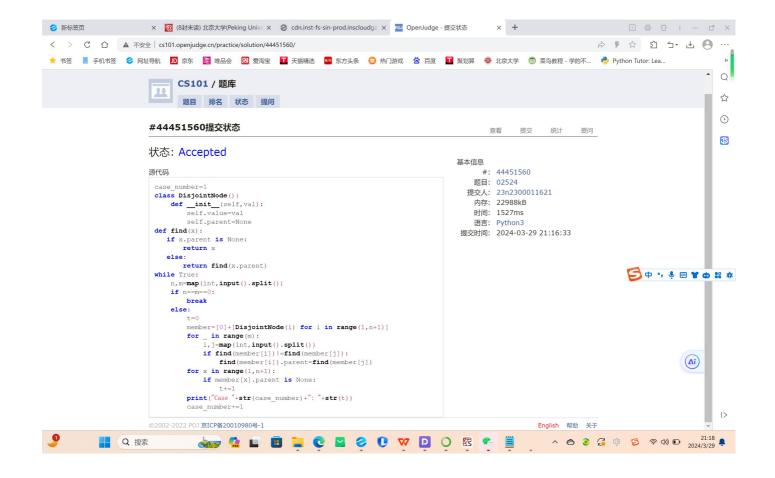
```
代码
case_number=1
class DisjointNode():
  def __init__(self,val):
     self.value=val
     self.parent=None
def find(x):
 if x.parent is None:
    return x
  else:
    return find(x.parent)
while True:
```

break

if n==m==0:

n,m=map(int,input().split())

```
else:
  t=0
  member=[0]+[DisjointNode(i) for i in range(1,n+1)]
  for _ in range(m):
    i,j=map(int,input().split())
    if find(member[i])!=find(member[j]):
       find(member[i]).parent=find(member[j])
  for x in range(1,n+1):
    if member[x].parent is None:
       t+=1
  print("Case "+str(case_number)+": "+str(t))
  case_number+=1
```



2. 学习总结和收获

通常情况下,仅依靠前序,中序,后序中任意的单独一种表达式都是无法确定二叉树的结构的。但二叉搜索树是一个例外:比如,在BST的前序遍历中,第一个比根节点大的数就是BST的右子树。

数算的精华在于复用。我的前序,后序表达式的建立函数已经在连续两周的题目中发光发热,在22275:二叉搜索树的遍历中,同样的方法又被再次使用。基本思路是:在最底层,返回节点值;在高层,表达式在末尾加入每个子节点的递归调用。

为什么要使用并查集?在我的实践中,发现02524:宗教信仰,表示两组元素为同一类的信号出现可能晚于这两组数据,例如: (4,5), (2,3), (5,3), 只有用记录父节点的方法才能去重。

显然,我自己是没有能力独立创造一个建二叉堆的方法的,所以仍然是借鉴了课件。不过我发现了题解代码的两个可改进之处:首先,currentsize属性是冗余的,用内置的len函数就能代替;其次,我感觉buildheap的原理挺复杂,我改为了用operate接收输入数据(这里我回到《Python:从入门到实践》去复习了实参个数不定的处理方法:在形参上加入"*"前缀,则该形参转为一个列表,它会自动append所有处于正确位置的实参。)。

插个题外话: 闫老师提供的代码都有行号,而且是编辑器识别不到的,增加了同学们使用的难度(IDE可不认行号)。 这里分享一个删掉这些数字的方法:按住Alt,向右拖动选中第一行的数字后,向下拖动,然后双击Delete,就可以删 掉这些烦人的数字了。

