1. 题目
LC222.完全二叉树的节点个数
bfs, dfs, binary + greedy, https://leetcode.cn/problems/count-complete-tree-nodes/

如果用 bfs 写是简单级别，其他方法是中级难度。

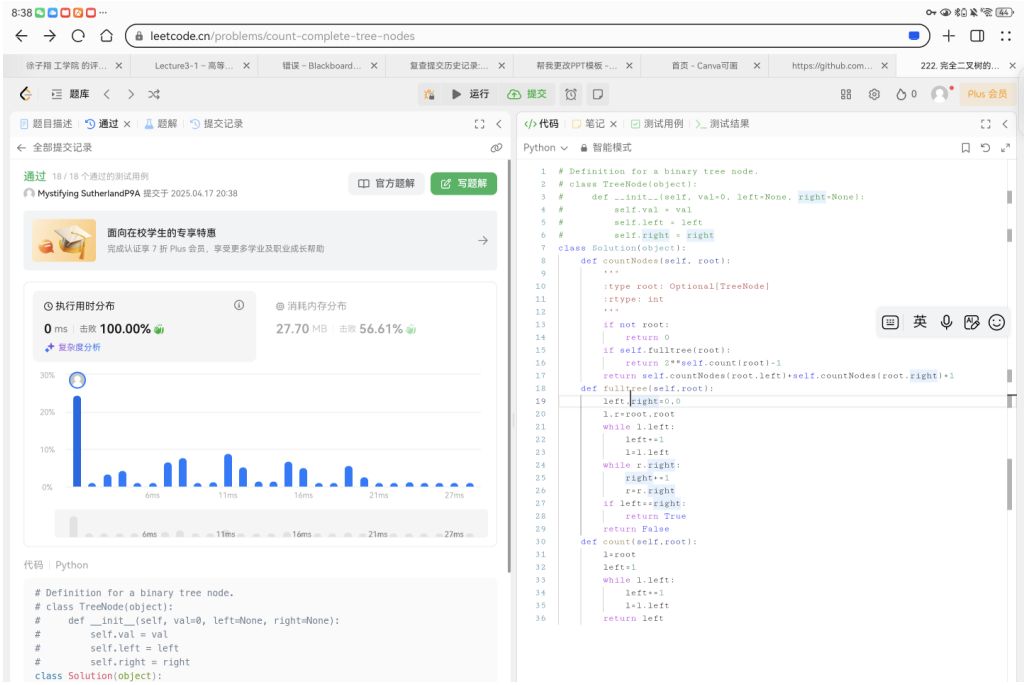
思路:

代码:

```
# Definition for a binary tree node.

# class TreeNode(object):

#         def __init__(self, val=0, left=None, right=None):

#             self.val = val

#             self.left = left

#             self.right = right

class Solution(object):

    def countNodes(self, root):

        """

        :type root: Optional[TreeNode]

        :rtype: int

        """

        if not root:

            return 0

        if self.fulltree(root):

            return 2**self.count(root)-1

        return self.countNodes(root.left)+self.countNodes(root.right)+1

    def fulltree(self,root):
```

```
        left,right=0,0
        l,r=root,root
        while l.left:
            left+=1
            l=l.left
        while r.right:
            right+=1
            r=r.right
        if left==right:
            return True
        return False
    def count(self,root):
        l=root
        left=1
        while l.left:
            left+=1
            l=l.left
        return left
```

代码运行截图 （至少包含有"Accepted"）



LC103.二叉树的锯齿形层序遍历

bfs, https://leetcode.cn/problems/binary-tree-zigzag-level-order-traversal/

思路:

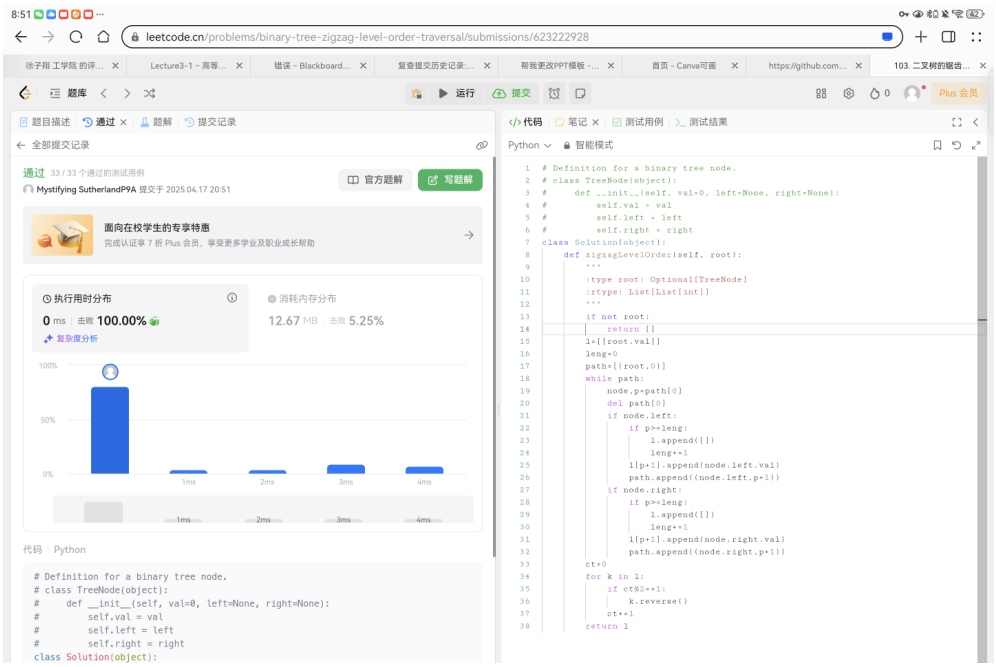
代码:

```
# Definition for a binary tree node.

# class TreeNode(object):

#         def __init__(self, val=0, left=None, right=None):

#                 self.val = val

#                 self.left = left

#                 self.right = right

class Solution(object):

    def zigzagLevelOrder(self, root):

        """

        :type root: Optional[TreeNode]

        :rtype: List[List[int]]
```

```
"""
if not root:
    return []
l=[[root.val]]
leng=0
path=[(root,0)]
while path:
    node,p=path[0]
    del path[0]
    if node.left:
        if p>=leng:
            l.append([])
            leng+=1
        l[p+1].append(node.left.val)
        path.append((node.left,p+1))
    if node.right:
        if p>=leng:
            l.append([])
            leng+=1
        l[p+1].append(node.right.val)
        path.append((node.right,p+1))
ct=0
```

```
        for k in l:
            if ct%2==1:
                k.reverse()
            ct+=1
        return l
```

代码运行截图 （至少包含有"Accepted"）



M04080:Huffman 编码树

greedy, http://cs101.openjudge.cn/practice/04080/

思路：

代码：

```
import heapq
n=int(input())
op=0
l=list(map(int,input().split()))
leng=len(l)
heapq.heapify(l)
while leng>1:
    a=heapq.heappop(l)
    b=heapq.heappop(l)
    heapq.heappush(l,a+b)
```

```
        op+=a+b
        leng-=1
print(op)
```

代码运行截图 （至少包含有"Accepted"）

M05455: 二叉搜索树的层次遍历

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

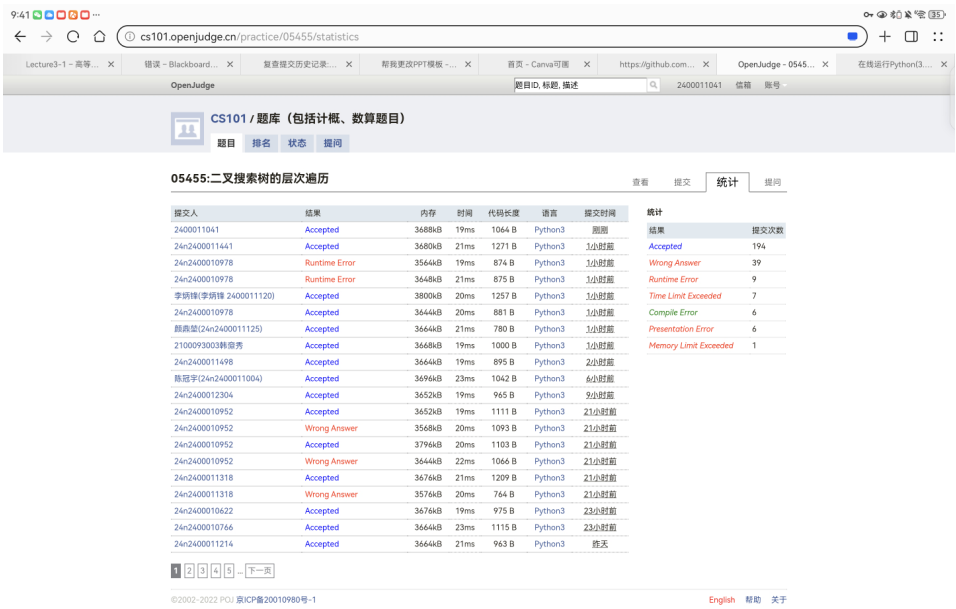
思路:

代码:

```
class node():
        def __init__(self,val):
                self.val=val
                self.right=None
                self.left=None
class bst():
        def __init__(self):
                self.root=None
        def insert(self,node):
                if not self.root:
                        self.root=node
                else:
                        self.inserttree(node,self.root)
        def inserttree(self,node,root):
                if node.val>root.val:
                        if not root.right:
                                root.right=node
                        else:
```

```
                self.inserttree(node,root.right)
        else:
            if not root.left:
                root.left=node
            else:
                self.inserttree(node,root.left)


l=list(map(int,map(int,input().split())))
num=set()
tree=bst()
for k in l:
    if k not in num:
        num.add(k)
        tree.insert(node(k))
root=tree.root
path=[root]
op=str(root.val)+' '
while path:
    n=path[0]
    del path[0]
    if n.left:
        op+=str(n.left.val)+' '
        path.append(n.left)
    if n.right:
        op+=str(n.right.val)+' '
        path.append(n.right)
print(op[:-1])
```

代码运行截图 （至少包含有"Accepted"）



M04078: 实现堆结构
手搓实现，http://cs101.openjudge.cn/practice/04078/

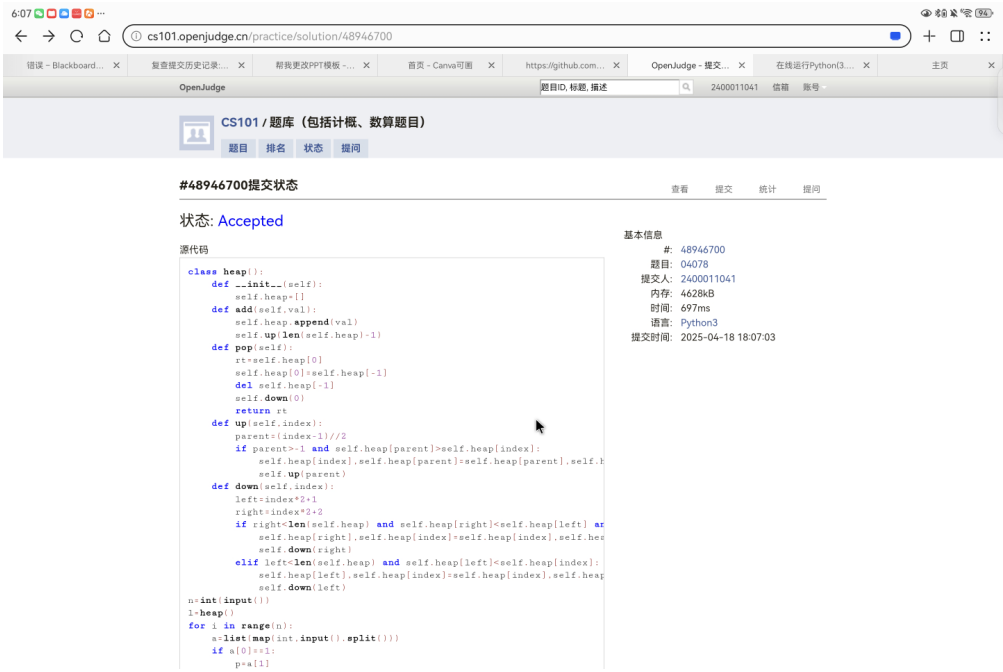
类似的题目是 晴问 9.7: 向下调整构建大顶堆，https://sunnywhy.com/sfbj/9/7

思路：

代码：

```
lass heap():
    def __init__(self):
        self.heap=[]
    def add(self,val):
        self.heap.append(val)
        self.up(len(self.heap)-1)
    def pop(self):
        rt=self.heap[0]
        self.heap[0]=self.heap[-1]
        del self.heap[-1]
        self.down(0)
        return rt
    def up(self,index):
        parent=(index-1)//2
        if parent>-1 and self.heap[parent]>self.heap[index]:
            self.heap[index],self.heap[parent]=self.heap[parent],self.heap[index]
            self.up(parent)
    def down(self,index):
        left=index*2+1
        right=index*2+2
        if right<len(self.heap) and self.heap[right]<self.heap[left] and
self.heap[right]<self.heap[index]:
            self.heap[right],self.heap[index]=self.heap[index],self.heap[right]
            self.down(right)
        elif left<len(self.heap) and self.heap[left]<self.heap[index]:
            self.heap[left],self.heap[index]=self.heap[index],self.heap[left]
            self.down(left)
n=int(input())
l=heap()
for i in range(n):
    a=list(map(int,input().split()))
    if a[0]==1:
        p=a[1]
        l.add(p)
    else:
        print(l.pop())
```

代码运行截图 （至少包含有"Accepted"）



T22161: 哈夫曼编码树
greedy, http://cs101.openjudge.cn/practice/22161/

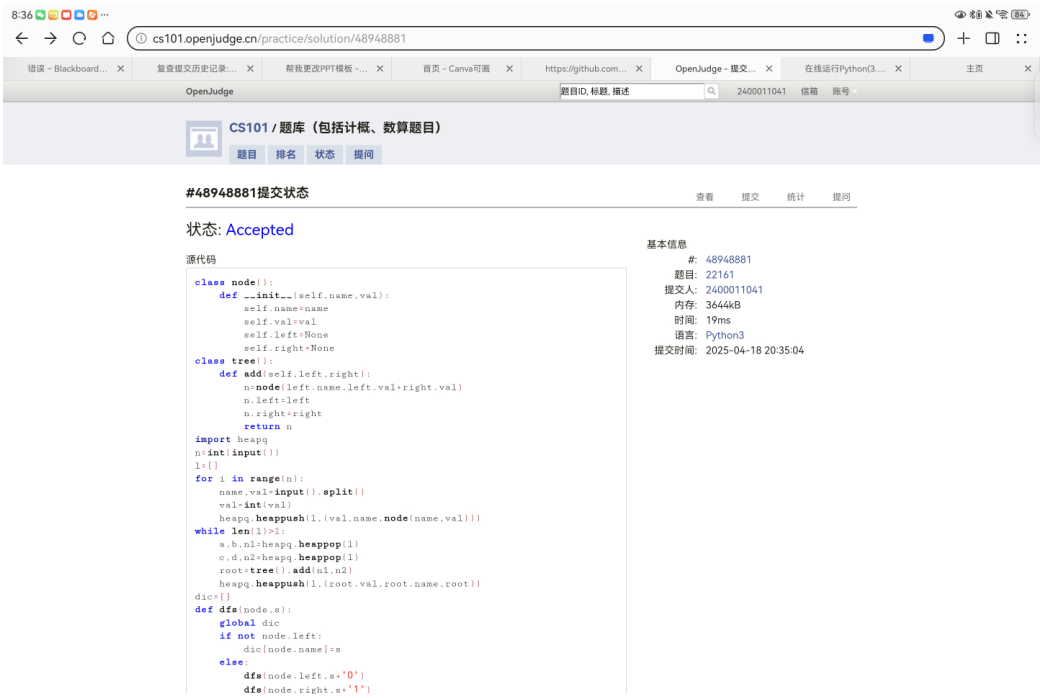
思路:

代码:

```
class node():
    def __init__(self,name,val):
        self.name=name
        self.val=val
        self.left=None
        self.right=None
class tree():
    def add(self,left,right):
        n=node(left.name,left.val+right.val)
        n.left=left
        n.right=right
        return n
import heapq
n=int(input())
l=[]
for i in range(n):
    name,val=input().split()
    val=int(val)
    heapq.heappush(l,(val,name,node(name,val)))
while len(l)>1:
```

```
        a,b,n1=heapq.heappop(l)
        c,d,n2=heapq.heappop(l)
        root=tree().add(n1,n2)
        heapq.heappush(l,(root.val,root.name,root))
dic={}
def dfs(node,s):
        global dic
        if not node.left:
                dic[node.name]=s
        else:
                dfs(node.left,s+'0')
                dfs(node.right,s+'1')
dfs(root,'')
def huf(s):
        global root
        rt=''
        cur=root
        for k in s:
                if k=='1':
                        cur=cur.right
                else:
                        cur=cur.left
                if    not cur.left:
                        rt+=cur.name
                        cur=root
        return rt
def rehuf(s):
        global dic
        p=0
        rt=''
        for i in range(1,len(s)+1):
                if s[p:i] in dic:
                        rt+=dic[s[p:i]]
                p=i
        return rt
while True:
        try:
                s=input()
                if '0'<=s[0]<='1':
                        print(huf(s))
                else:
                        print(rehuf(s))
        except EOFError:
                break
```

代码运行截图 （至少包含有"Accepted"）




```
class node():
    def __init__(self,name,val):
        self.name=name
        self.val=val
        self.left=None
        self.right=None
class tree():
    def add(self,left,right):
        n=node(left.name,left.val+right.val)
        n.left=left
        n.right=right
        return n
import heapq
n=int(input())
l=[]
for i in range(n):
    name,val=input().split()
    val=int(val)
    heapq.heappush(l,(val,name,node(name,val)))
while len(l)>1:
    a,b,n1=heapq.heappop(l)
    c,d,n2=heapq.heappop(l)
    root=tree().add(n1,n2)
    heapq.heappush(l,(root.val,root.name,root))
dic={}
def dfs(node,s):
    global dic
    if not node.left:
        dic[node.name]=s
    else:
        dfs(node.left,s+'0')
        dfs(node.right,s+'1')
```

2. 学习总结和收获

对树有了进一步了解。class 写法确实有其优越性。

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算 2025spring 每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

仍旧期中考，选做暂停。