Assignment5

姓名：尹廖丹华

学号：2000092113

院系：心理与认知科学学院

一、 作业题目

1、27638:求二叉树的高度和叶子数目（1h）

```
class TreeNode:
    def __init__(self):
        self.left = None
        self.right = None

def tree_height(node):
    if node is None:
        return -1
    return max(tree_height(node.left), tree_height(node.right)) + 1

def count_leaves(node):
    if node is None:
        return 0
    if node.left is None and node.right is None:
        return 1
    return count_leaves(node.left) + count_leaves(node.right)

n = int(input())
nodes = [TreeNode() for _ in range(n)]
has_parent = [False] * n

for i in range(n):
    left_index, right_index = map(int, input().split())
    if left_index != -1:
        nodes[i].left = nodes[left_index]
        has_parent[left_index] = True
    if right_index != -1:
        nodes[i].right = nodes[right_index]
        has_parent[right_index] = True

root_index = has_parent.index(False)
root = nodes[root_index]

height = tree_height(root)
leaves = count_leaves(root)

print(f"{height} {leaves}")
```

2、24729:括号嵌套树（1h30min）

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []

def parse_tree(s):
    stack = []
    node = None
    for char in s:
        if char.isalpha():
            node = TreeNode(char)
            if stack:
                stack[-1].children.append(node)
        elif char == '(':
            if node:
                stack.append(node)
                node = None
        elif char == ')':
            if stack:
                node = stack.pop()
    return node


def preorder(node):
    output = [node.value]
    for child in node.children:
        output.extend(preorder(child))
    return ''.join(output)

def postorder(node):
    output = []
```

```
    for child in node.children:
        output.extend(postorder(child))
    output.append(node.value)
    return ''.join(output)

def main():
    s = input().strip()
    s = ''.join(s.split())
    root = parse_tree(s)
    if root:
        print(preorder(root))
        print(postorder(root))
    else:
        print("input tree string error!")

if __name__ == "__main__":
    main()
```

3、02775:文件结构“图” （1h）

```
from sys import exit

class dir:
    def __init__(self, dname):
        self.name = dname
        self.dirs = []
        self.files = []

    def getGraph(self):
        g = [self.name]
        for d in self.dirs:
            subg = d.getGraph()
            g.extend(["|     " + s for s in subg])
```

```
        for f in sorted(self.files):
            g.append(f)
        return g

n = 0
while True:
    n += 1
    stack = [dir("ROOT")]
    while (s := input()) != "*":
        if s == "#": exit(0)
        if s[0] == 'f':
            stack[-1].files.append(s)
        elif s[0] == 'd':
            stack.append(dir(s))
            stack[-2].dirs.append(stack[-1])
        else:
            stack.pop()
    print(f"DATA SET {n}:")
    print(*stack[0].getGraph(), sep='\n')
    print()
```

4、24750:根据二叉树中后序序列建树（2h）

```
def build_tree(inorder, postorder):
    if not inorder or not postorder:
        return []

    root_val = postorder[-1]
    root_index = inorder.index(root_val)

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index + 1:]

    left_postorder = postorder[:len(left_inorder)]
```

```
    right_postorder = postorder[len(left_inorder):-1]

    root = [root_val]
    root.extend(build_tree(left_inorder, left_postorder))
    root.extend(build_tree(right_inorder, right_postorder))

    return root


def main():
    inorder = input().strip()
    postorder = input().strip()
    preorder = build_tree(inorder, postorder)
    print(''.join(preorder))


if __name__ == "__main__":
    main()
```

二、 学习的感想与收获

还是没有学会树的写法，看代码也是看得一头雾水，只能一边百度一边试图看懂，虽然效果也是微乎其微，只能理解个大概…

而且最近 ddl 好多，都没能有时间好好理解代码的流程 5555

这周有时间再继续学习！！！