1. 题目

E108.将有序数组转换为二叉搜索树

https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/

思路： 由于这是一颗平衡二叉树，所以左右子树高度几乎一样，则根节点一定是数组中最接近中间的树，而左右子树又一定是平衡二叉树，所以直接迭代的形成左右子树即可，左右子树的构成也自然是数组左边和右边的两列数。

代码：

```
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        if not nums:
            return None

        mid = len(nums) // 2
        root = TreeNode(nums[mid])
        root.left = self.sortedArrayToBST(nums[:mid])
        root.right = self.sortedArrayToBST(nums[mid+1:])

        return root
```

代码运行截图 （至少包含有"Accepted"）

代码 | Python3

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
```

查看更多

M07161: 森林的带度数层次序列存储

tree, http://cs101.openjudge.cn/practice/07161/

思路： 这个题目非常全面的对树这一数据结构进行考察。第一步是将输入的序列转换为树的数据结构。这里是自己定义出树的类，首先处理从序列建立树的问题，我们注意到序列中的节点是按照bfs的顺序排列的，我们为了提取这些节点想要按照队列的方式进行提取，先进先出以保证从根节点一层一层下降。构建好树之后再按照所需顺序进行输出即可。

代码：

```
from collections import deque
class TreeNode:
    def __init__(self, val=''):
        self.val = val
        self.children = []

def build_tree_from_sequence(sequence_str):
    tokens = sequence_str.split()
    if not tokens:
        return None
```

```
    nodes = []
    for i in range(0, len(tokens), 2):
        val = tokens[i]
        degree = int(tokens[i + 1])
        nodes.append((val, degree))

    root = TreeNode(nodes[0][0])
    queue = deque()
    queue.append((root, nodes[0][1]))

    child_index = 1
    while queue and child_index < len(nodes):
        current_node, remaining_children = queue[0]

        if remaining_children == 0:
            queue.popleft()
            continue

        child_val, child_degree = nodes[child_index]
        child_node = TreeNode(child_val)
        current_node.children.append(child_node)

        queue[0] = (current_node, remaining_children - 1)
        if queue[0][1] == 0:
            queue.popleft()

        if child_degree > 0:
            queue.append((child_node, child_degree))

        child_index += 1

    return root

def postorder_traversal(root, result):
    if not root:
        return
    for child in root.children:
        postorder_traversal(child, result)
    result.append(root.val)
def main():
    n = int(input().strip())
    forest = []
    for _ in range(n):
        sequence = input().strip()
        tree = build_tree_from_sequence(sequence)
        if tree:
            forest.append(tree)
    result = []
    for tree in forest:
        postorder_traversal(tree, result)
    print(' '.join(result))
if __name__ == "__main__":
    main()
```

代码运行截图 （至少包含有"Accepted"）

#50671241提交状态

状态: Accepted

源代码

```
from collections import deque
class TreeNode:
    def __init__(self, val=''):
        self.val = val
        self.children = []

def build_tree_from_sequence(sequence_str):
    tokens = sequence_str.split()
    if not tokens:
        return None

    nodes = []
    for i in range(0, len(tokens), 2):
        val = tokens[i]
        degree = int(tokens[i + 1])
        nodes.append((val, degree))

    root = TreeNode(nodes[0][0])
    queue = deque()
    queue.append((root, nodes[0][1]))

    child_index = 1
    while queue and child_index < len(nodes):
        current_node, remaining_children = queue[0]

        if remaining_children == 0:
            queue.popleft()
            continue
```

M27928: 遍历树

adjacency list, dfs, http://cs101.openjudge.cn/practice/27928/

思路： 两个问题： 1.如何确定根节点开始遍历，我们记录所有以子节点形式出现过的节点在总集中抛去得到根节点 2.如何按照要求输出：对子节点进行排序，并加入根节点，搜索到子节点则进下一层循环，否则直接输出根节点。

代码：

```
n=int(input())
dist={}
all_nodes = set()  # 所有节点
```

```
child_nodes = set()  # 所有子节点
for i in range(n):
    l=list(map(int,input().split()))
    all_nodes.add(l[0])
    if len(l)>1:
        dist[l[0]]=l[1:]
        child_nodes.update(l[1:])
    else:
        dist[l[0]]=[]
root = (all_nodes - child_nodes).pop()
def path(node):
    nums=dist[node]
    nums.append(node)
    nums=sorted(nums)
    for i in nums:
        if i==node:
            print(i)
        else:
            path(i)
path(root)
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
n=int(input())
dist={}
all_nodes = set()  # 所有节点
child_nodes = set()  # 所有子节点
for i in range(n):
    l=list(map(int,input().split()))
    all_nodes.add(l[0])
    if len(l)>1:
        dist[l[0]]=l[1:]
        child_nodes.update(l[1:])
    else:
        dist[l[0]]=[]
root = (all_nodes - child_nodes).pop()
def path(node):
    nums=dist[node]
    nums.append(node)
    nums=sorted(nums)
    for i in nums:
        if i==node:
            print(i)
        else:
            path(i)
path(root)
```

基本信息

#:	50673092
题目:	27928
提交人:	25n殷知远 2300015498
内存:	3876kB
时间:	23ms
语言:	Python3
提交时间:	2025-11-02 18:31:07

M129.求根节点到叶节点数字之和

dfs, https://leetcode.cn/problems/sum-root-to-leaf-numbers/

思路：

本质上是dfs，在dfs过程中用节点值按照位进行计算以获得记录，要点是ans=ans*10+node.val，也是常用手段。

代码

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        def path(node,ans):
            if not node:
                return 0
            ans=ans*10+node.val
            if not node.left and not node.right:
                return ans
            return path(node.left,ans)+path(node.right,ans)
        return path(root,0)
```

代码运行截图（至少包含有"Accepted"）

M24729: 括号嵌套树

dfs, stack, http://cs101.openjudge.cn/practice/24729/

思路：

对于这个括号的结构很经典的根据前后括号对其用栈进行处理，将其解析为树的结构之后就按照普通的各种顺序输出即可。

代码

```
def parse_tree(s, index):
    node = s[index]
    index += 1
    children = []
    if index < len(s) and s[index] == '(':
        index += 1
        while index < len(s) and s[index] != ')':
            if s[index] == ',':
```

```
                index += 1
            child, index = parse_tree(s, index)
            children.append(child)
        index += 1
    return (node, children), index
def preorder(root):
    """前序遍历"""
    node, children = root
    result = [node]
    for child in children:
        result.extend(preorder(child))
    return result
def postorder(root):
    """后序遍历"""
    node, children = root
    result = []
    for child in children:
        result.extend(postorder(child))
    result.append(node)
    return result
def main():
    s = input().strip()
    root, _ = parse_tree(s, 0)
    print(''.join(preorder(root)))
    print(''.join(postorder(root)))
if __name__ == "__main__":
    main()
```

代码运行截图（至少包含有"Accepted"）

状态: Accepted

源代码

```
def parse_tree(s, index):
    node = s[index]
    index += 1
    children = []
    if index < len(s) and s[index] == '(':
        index += 1
        while index < len(s) and s[index] != ')':
            if s[index] == ',':
                index += 1
            child, index = parse_tree(s, index)
            children.append(child)
        index += 1
    return (node, children), index
def preorder(root):
    """前序遍历"""
    node, children = root
    result = [node]
    for child in children:
        result.extend(preorder(child))
    return result
def postorder(root):
    """后序遍历"""
    node, children = root
    result = []
    for child in children:
        result.extend(postorder(child))
    result.append(node)
    return result
def main():
    s = input().strip()
    root, _ = parse_tree(s, 0)
    print(''.join(preorder(root)))
    print(''.join(postorder(root)))
if __name__ == "__main__":
    main()
```

基本信息

#: 50673363
题目: 24729
提交人: 25n殷知远 2300015498
内存: 3656kB
时间: 19ms
语言: Python3
提交时间: 2025-11-02 18:50:36

T02775: 文件结构“图”

tree, http://cs101.openjudge.cn/practice/02775/

思路：

用栈建树再深度搜索输出

代码：

```
def isdir(name):
    return name[0] == 'd'

def dfs(dir, prefix, name, subdirs, subfiles):
    print(prefix + name[dir])

    subprefix = prefix + "|     "
    for subdir in subdirs[dir]:
        dfs(subdir, subprefix, name, subdirs, subfiles)

    for file in sorted(subfiles[dir]):
        print(prefix + file)
```

```
def solve(cas):
    id = 0
    name = ["ROOT"]
    subdirs = [[]]
    subfiles = [[]]
    stk = [0]

    while True:
        line = input()
        if line == "#":
            return False
        if line == "*":
            break

        if isdir(line):
            id += 1
            name.append(line)
            subdirs.append([])
            subfiles.append([])
            subdirs[stk[-1]].append(id)
            stk.append(id)
        elif line == ']':
            stk.pop()
        else:
            subfiles[stk[-1]].append(line)

    print(f"DATA SET {cas}:")
    dfs(0, "", name, subdirs, subfiles)
    print()
    return True

def main():
    cas = 1
    while solve(cas):
        cas += 1

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

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
def is_directory(entry):
    return entry[0] == 'd'

def traverse_directory(current_dir, prefix, dir_names, subdirectories, 
    print(prefix + dir_names[current_dir])

    next_level_prefix = prefix + "|     "
    for subdir_index in subdirectories[current_dir]:
        traverse_directory(subdir_index, next_level_prefix, dir_names, 

    for filename in sorted(files[current_dir]):
        print(prefix + filename)

def process_dataset(dataset_num):
```

基本信息

#: 50682072
题目: 02775
提交人: 25n殷知远 2300015498
内存: 3564kB
时间: 21ms
语言: Python3
提交时间: 2025-11-03 14:53:15

2. 学习总结和个人收获

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2025fall每日选做"、LeetCode、Codeforces、洛谷等网站上的题目。 本次题目中第六题因为其很繁琐的输入输出出现了一定的问题，但问题并不处在数据结构的理解上所以不算严重。 这周在准备期中考试疏于练习，之后补上。