**作业 HW3 实验报告**

姓名：陆明奇 学号：2050283 日期：2021年11月14日

1. **涉及数据结构和相关背景**

树型结构是一类重要的非线性数据结构。其中以树和二叉树最为常用，直观来看，树是以分支关系定义的层次结构。树结构在客观世界中广泛存在，如人类社会的族谱和各种社会组织机构都可用树来形象表示。树在计算机领域也得到广泛应用，如在编译程序中，可用树来表示源程序的语法结构。又如在数据库领域，树形结构也是信息的重要组织形式之一。

**2. 实验内容**

**2.1 二叉树的创建与遍历**

**2.1.1 问题描述**

二叉树是n（n>=0）个结点的有限集合，它或为空树，或是由一个称之为根的结点加上两棵分别称为左子树和右子树的互不相交的二叉树组成。本题练习二叉树的基本操作，包括先序建立二叉树，先序遍历、中序遍历、后序遍历、层次遍历二叉树，输出二叉树的树型图。

**2.1.2 基本要求**

输入一行，输入先序序列，内部结点用一个字符表示，空结点用#表示。输出多行，第1行，输出先序遍历序列。第2行，输出中序遍历序列。第3行，输出后序遍历序列。第4行，输出层次遍历序列。输出二叉树的树型图(逆时针旋转90度)，每一层之间输出5个空格。

**2.1.3 数据结构设计**

本题考察的是二叉树的基本操作，因此采用二叉树这一数据结构。每个结构体结点包含一个数据域和两个指针域分别指向左右孩子。

**2.1.4功能说明（函数、类）**

bool CreateBiTree(BiTree& T)

功能：按先序次序输入二叉树中结点的值（一个字符），空格字符表示空树

Para T：指向二叉树的根节点的指针

返回值：构造二叉树是否成功

void PreOrderTraverse(BiTree T)

功能：先序遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void InOrderTraverse(BiTree T)

功能：中序遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void PostOrderTraverse(BiTree T)

功能：后序遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void LayerOrderTraverse(BiTree T)

功能：层次遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void display(BiTree T, int n)

功能：输出二叉树的树形图（逆时针旋转90度）

Para T：指向二叉树的根节点的指针

Para n：所画子树的根节点所在的层数

返回值：无返回值

**2.1.5 调试分析（遇到的问题和解决方法）**

第一次画二叉树的树形图的想法是逐行打印，但这样需要存储的变量十分多，会把自己绕晕。之后我采用递归的方式来画二叉树的树形图：，先递归画出左子树，再画出根节点，最后递归画出右子树。

**2.1.6 总结和体会**

通过这道题，我熟悉了二叉树的一些基本操作，比如先序建立二叉树，先序遍历、中序遍历、后序遍历、层次遍历二叉树，输出二叉树的树型图等·。难点在于输出二叉树的树形图，这里需要采用递归的方法来画。

**2.2 二叉树的同构**

**2.2.1 问题描述**

给定两棵树T1和T2。如果T1可以通过若干次左右孩子互换变成T2，则我们称两棵树是“同构”的。现给定两棵树，请你判断它们是否是同构的。并计算每棵树的深度。

**2.2.2 基本要求**

输入第一行是一个非负整数N1，表示第1棵树的结点数；随后N行，依次对应二叉树的N个结点（假设结点从0到N−1编号），每行有三项，分别是1个英文大写字母、其左孩子结点的编号、右孩子结点的编号。如果孩子结点为空，则在相应位置上给出“-”。给出的数据间用一个空格分隔。接着一行是一个非负整数N2，表示第2棵树的结点数；随后N行同上描述一样，依次对应二叉树的N个结点。输出共三行。第一行，如果两棵树是同构的，输出“Yes”，否则输出“No”。

后面两行分别是两棵树的深度。

**2.2.3 数据结构设计**

因为本题输入时，输入的是结点左右孩子的编号，而不是左右孩子本身，因此采用静态链表来存储二叉树。每个结构体结点包含一个数据域和两个表示左右孩子的整型。

**2.2.4功能说明（函数、类）**

int BuildTree(TreeNode\* T)

功能：构造一棵二叉树

Para T：指向静态链表首元素的指针

返回值：构造的二叉树的根节点的编号

bool Isomorphic(int R1, int R2)

功能：判断两个二叉树是否同构

Para R1：二叉树1的根节点的编号

Para R2：二叉树2的根节点的编号

返回值：两棵二叉树是否同构

int Depth(TreeNode\* T, int root)

功能：求二叉树的深度

Para T：指向静态链表首元素的指针

Para root：二叉树的根节点的编号

返回值：二叉树的深度

**2.2.5 调试分析（遇到的问题和解决方法）**

之前BuildTree函数一直找不对根节点的编号，经过长时间的调试，发现：由于输入的编号可能为-，我采用了char型变量来读取编号。当判断读入的编号不是-时，我再将其转化为int型的编号。但这样子有一个非常大的问题，就是无法读取两位数及以上的编号。于是我才用char型数组来读取编号，成功找出了正确的根节点。

**2.2.6 总结和体会**

这道题采用了静态链表来存储二叉树，是一次全新的尝试。本题最关键的是需要找出二叉树的根节点，找到根节点之后才能进行比较是否同构和求深度的操作。需要注意的坑是编号不一定只有一位数，需要用char型数组来读取而不能用char型变量。

**2.3 二叉树的非递归遍历**

**2.3.1 问题描述**

二叉树的非递归遍历可通过栈来实现。如果已知中序遍历的栈的操作序列，就可唯一地确定一棵二叉树。请编程输出该二叉树的后序遍历序列。

**2.3.2 基本要求**

第一行输入一个整数n，表示二叉树的结点个数。接下来2n行，每行描述一个栈操作，格式为：push X 表示将结点X压入栈中，pop 表示从栈中弹出一个结点(X用一个字符表示)。输出一行，后序遍历序列。

**2.3.3 数据结构设计**

本题考察的是二叉树的非递归遍历，还是可以采用和第一题一样的二叉树这一数据结构。每个结构体结点包含一个数据域和两个指针域分别指向左右孩子。

**2.3.4功能说明（函数、类）**

bool bool CreateBiTree(BiTree& T, string preorder)

功能：按先序次序输入二叉树中结点的值（一个字符），空格字符表示空树

Para T：指向二叉树的根节点的指针

Para preorder：二叉树的先序遍历序列

返回值：构造二叉树是否成功

void PostOrderTraverse(BiTree T)

功能：后序遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

**2.3.5 调试分析（遇到的问题和解决方法）**

第一次构造先序序列时，将空结点设置为了“#”。随后发现结点的元素也可能为“#”，从而导致错误。随后将空结点改为了“<”,问题得到了解决。

**2.3.6 总结和体会**

中序非递归遍历的出栈入栈操作和先序序列有一定的关系：push x即代表篇先序序列中的x，而pop则代表空结点。由此可构造出先序序列。需要特别注意在出栈和入栈操作结束后还需要在先序序列后再加上一个空结点。本题还需要特别注意空结点的设置，设置的值不能和结点的值相同。

**2.4 中序遍历线索二叉树**

**2.4.1 问题描述**

练习线索二叉树的基本操作，包括二叉树的线索化，中序遍历线索二叉树，查找某元素在中序遍历的后继结点、前驱结点。

**2.4.2 基本要求**：

输入第1行，输入先序序列，内部结点用一个字符表示，空结点用#表示

输入第2行，输入一个字符a，查找该字符的后继结点元素和前驱结点元素

输出第1行，输出中序遍历序列

输出第2行，输出查找字符的结果，若不存在，输出一行Not found，结束

接下来包含N组共2N行（N为树种与a相等的所有节点个数）

每一组按其代表节点的中序遍历顺序排列（即按中序遍历搜索线索树，找到节点即可输出，直到遍历完整棵树）。

输出第2k+3行输出该字符的直接后继元素及该后继元素的RTag，格式为：succ is 字符+RTag

输出第2k+4行输出该字符的直接前驱继元素及前驱元素的LTag，格式为：prev is 字符+LTag

若无后继或前驱，用NULL表示。上述k有（0<=k<N/2）

**2.4.3 数据结构设计**

本题考察的是线索二叉树的基本操作，因此采用线索二叉树这一数据结构。每个结构体结点包含一个数据域，两个指针域，两个标志域LTag和RTag来表示指针是指向后继还是孩子。

**2.4.4功能说明（函数、类）**

bool CreateBiTree(BiTree& T)

功能：按先序次序输入二叉树中结点的值（一个字符），空格字符表示空树

Para T：指向二叉树的根节点的指针

返回值：构造二叉树是否成功

void InOrderTraverse(BiTree T)

功能：中序遍历二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void InThreeading(BiTree P)

功能：中序线索化二叉树

Para T：指向二叉树的根节点的指针

返回值：无返回值

void searchbithrtree(BiTree T, char k)

功能：查找结点字符的前驱和后继结点元素

Para T：指向二叉树的根节点的指针

Para ch：所要查找的结点字符

返回值：无返回值

**2.4.5 调试分析（遇到的问题和解决方法）**

此前寻找某一结点的后继结点时一直出错，原以为只要持续判断RTag遍历其右孩子即可，其实不然。结点的后继应该是遍历其右子树时访问的第一个结点，即右子树中最左下的结点

**2.4.6 总结和体会**

通过这道题，我熟悉了线索二叉树的一些基本操作，比如中序线索化二叉树，查找结点的后继和前驱等·。这道题的难点在于找到某一结点的前驱和后继，需要通过判断RTag和LTag来找出。

**2.5 家族树**

**2.5.1 问题描述**

以下是表示家谱的文本文件的示例。

John

Robert

Frank

Andrew

Nancy

David

每行包含一个人的名字。第一行的名字是这个家谱中最古老的祖先。家谱只包含该祖先的后代。他们的丈夫或妻子没有在家庭树上显示。一个人的孩子比父母缩进一个空格。例如，Robert和Nancy是John的孩子，Frank和Andrew是Robert的孩子。David比Robert缩进了一个空格，但他不是Robert的孩子，而是Nancy的孩子。为了用这一方法表示一个家谱，教授将一些人从家谱中排除，使得家谱中没有人有两个父母。

以下是关于上述家庭陈述语句的一些例子。

John is the parent of Robert.

Robert is a sibling of Nancy.

David is a descendant of Robert.

需要支持的查询有以下几种：

X is a child of Y.

X is the parent of Y.

X is a sibling of Y.

X is a descendant of Y.

X is an ancestor of Y.

注意：一个人的祖先、后代、兄弟可以是自己，但父母、孩子则不行。

**2.5.2 基本要求**：

输入包含若干组测试用例，每个测试用例由一个家谱和一个陈述句集合组成对于测试用例中的每句陈述语句。程序输出一行，给出True或False。

**2.5.3 数据结构设计**

题目中有父亲和孩子这一匹配关系，可以采用map来存储这一关系。用vector来存储读入的姓名。

**2.5.4功能说明（函数、类）**

int spaceNum(string str)

功能：计算读入名字的空格树

Para str：读入的名字

返回值：空格的数量

int judgment(string str1, string str2, string str3)

功能：判断陈述句是否正确

Para str1：X

Para str2：Y

Para str3：两者的关系

返回值：陈述句是否正确

**2.5.5 调试分析（遇到的问题和解决方法）**

之前判断陈述句的结构永远是false，随后发现每一句话的末尾会有一个句号。判断时会把这个句号和名字连在一起进行判断，所以一直输出的都是false。当把句号去除时，问题得到解决。在给vector数组事先确定一个大小时，根据数据范围的需要应设置1000以上，之前设置了100导致有一个点一直超时。

**2.5.6 总结和体会**

通过这道题，我熟悉了map哈希表的一些基本操作，了解到了树在实际问题中的一些应用场景。本题需要特别注意读入的内容，包括换行符和句号等。 在判断X是不是Y的祖先时，直接判断很麻烦，可以将其变换成判断Y是不是X的后代。

**3.实验总结**

这五道题使我对于树中二叉树的一些基本操作，如先序、中序、后序遍历，非递归遍历，线索化等有了更深入的了解。在做题的过程中发现了这些基本操作相互之间的联系。通过这些题目，我原本碎片化的知识得到整合，对于树有了一个整体，系统性的了解。