哈尔滨工业大学(深圳)

《数据结构》实验报告

实验三

树形结构及其应用

学　院: 计算机科学与技术

姓　名: 宗晴

学　号: 200110513

专　业: 计算机科学与技术

日　期: 2021-04-15

一、问题分析

题目 1 要解决的问题是按层次遍历建立二叉树，并输出该二叉树的前序、中序和后序遍历的序列。可以采用二叉链表来存储二叉树，用队列来辅助建立二叉树。

题目 2 要解决的问题是求取给定二叉树的最大路径和。(路径定义为从树的根节点到叶子结点的任意路径，路径和定义为一条路径中各节点的权重之和) 可以采用递归的方式分别求出根节点左右子树的最大路径和，取较大值再加上根节点的权重。

题目 3 要解决的问题是求取给定二叉树的所有左子叶权重之和。(左子叶被定义为二叉树叶子结点中属于左子树的节点) 同样可以利用递归，分别求出根节点左右子树的所有左子叶权重之和，两者相加即为根节点的所有左子叶权重之和。

题目 4 要解决的问题是求取给定二叉树的镜像，并输出镜像二叉树的中序遍历序列。还是可以利用递归，分别求取根节点左右子树的镜像，再将根节点的左右指针交换。

二、详细设计

2.1 设计思想

用二叉链表来存储二叉树，每个节点包含编号 id、数据 val、左指针、右指

针。

题目 1：

（1）按层次遍历建立二叉树：可以利用队列或者数组按层次遍历建立二叉树。本实验中采用了队列。首先判断数组中第一个数是否是-1，若是，则返回空树；否则该数就是一个有效数据。

初始化一个队列 Q，创建根结点存入该数据并将其入队。定义变量 TNode 用来指向有效结点，TNodep 用来指向无效结点。

当数组中还有剩余元素时，进入循环：用 TNode 获取队头元素并将其从队列中删去。（a）若数组中下一个元素值为-1，则表示是空结点，将 TNode 指向的结点的左指针置空，但仍需建立该结点并将其入队；否则建立有效值结点，原结点的左指针指向该结点，并将该结点入队。（b）继续判断数组中是否还有剩余元素，若有，则对原结点的右指针进行上述操作，若没有剩余元素，即数组中没有右孩子的值，也说明是空指针，将原结点的右指针置空。（c）最后判断出队的结点是否是无效结点（即数据域是否为-1），若是无效结点，则将该结点所占内存释放。

继续判断数组中是否还有剩余元素，若有，进入上述循环，否则，循环结束

最后将队列中剩余的所有元素，即最底层元素的左右指针置空，同时释放无效结点所占的内存。返回根结点。

（2）输出前序遍历序列：利用递归。若当前结点不空，则输出当前结点的值，然后分别前序递归遍历其左右子树。

（3）输出中序遍历序列：利用递归。若当前结点不空，则先中序递归遍历

其左子树，再输出当前结点的值，然后中序递归遍历其右子树。

（2）输出后序遍历序列：利用递归。若当前结点不空，则先分别后序递归遍历其左右子树，然后输出当前结点的值。

题目 2：

求取二叉树的最大路径和：利用深度优先遍历，递归实现。如果传入结点为空，则返回 0。否则，分别递归求取其左右子树的最大路径和。返回较大值和当前结点的权重之和。

题目 3：

求取二叉树的所有左子叶权重之和：利用深度优先遍历，递归实现。如果传入结点为空，则返回 0。否则，如果当前结点的左指针非空，且其左孩子的左右指针均为空，表示其左孩子为左子叶，返回其左孩子的权重。否则递归求取其左右子树的所有左子叶权重之和并返回。

题目 4：

求取二叉树的镜像，并输出它的中序遍历序列：利用递归翻转二叉树。如果传入的是空指针，则直接返回。否则递归翻转其左右子树并将两子树位置交换，最后返回翻转后的根结点。

2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）

```
typedef struct TreeNode {
    int id;
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
} TreeNode, *TreeNodePtr;

typedef struct ListNode {
    struct TreeNode *node; // 队列的值的类型是树节点指针
    struct ListNode *next;
} ListNode, *ListNodePtr;

typedef struct Queue {
    ListNodePtr dummyHead;
    ListNodePtr tail;
    int size;
} *QueuePtr;
```

TreeNode 为树的结点，包含编号 id、数据 val、左指针、右指针。

ListNode 为队列的结点，包含树结点指针、指向下一个结点的指针。

QueuePtr 为整个队列，包含头指针、尾指针、队列大小。

(2) 涉及的操作 (一般为自定义函数，可不写过程，但要注明该函数的含义。)

- TreeNodePtr createTreeWithLevelOrder(int *data, int size)

参数：传入存储数据的数组的首地址以及数组大小

功能：通过层次遍历来构建二叉树

返回值：返回二叉树的头结点

- void preOrderTraverse(TreeNodePtr root)

参数：传入二叉树的根节点

功能：打印输出二叉树的前序遍历序列

返回值：无返回值

- void inOrderTraverse(TreeNodePtr root)

参数：传入二叉树的根节点

功能：打印输出二叉树的中序遍历序列

返回值：无返回值

- void postOrderTraverse(TreeNodePtr root)

参数：传入二叉树的根节点

功能：打印输出二叉树的后序遍历序列

返回值：无返回值

- int maxPathSum(TreeNodePtr root)

参数：传入二叉树的根节点

功能：求取二叉树的最大路径和

返回值：返回二叉树的最大路径和

- int sumOfLeftLeaves(TreeNodePtr root)

参数：传入二叉树的根节点

功能：求取二叉树的所有左子叶权重之和

返回值：返回二叉树的所有左子叶权重之和

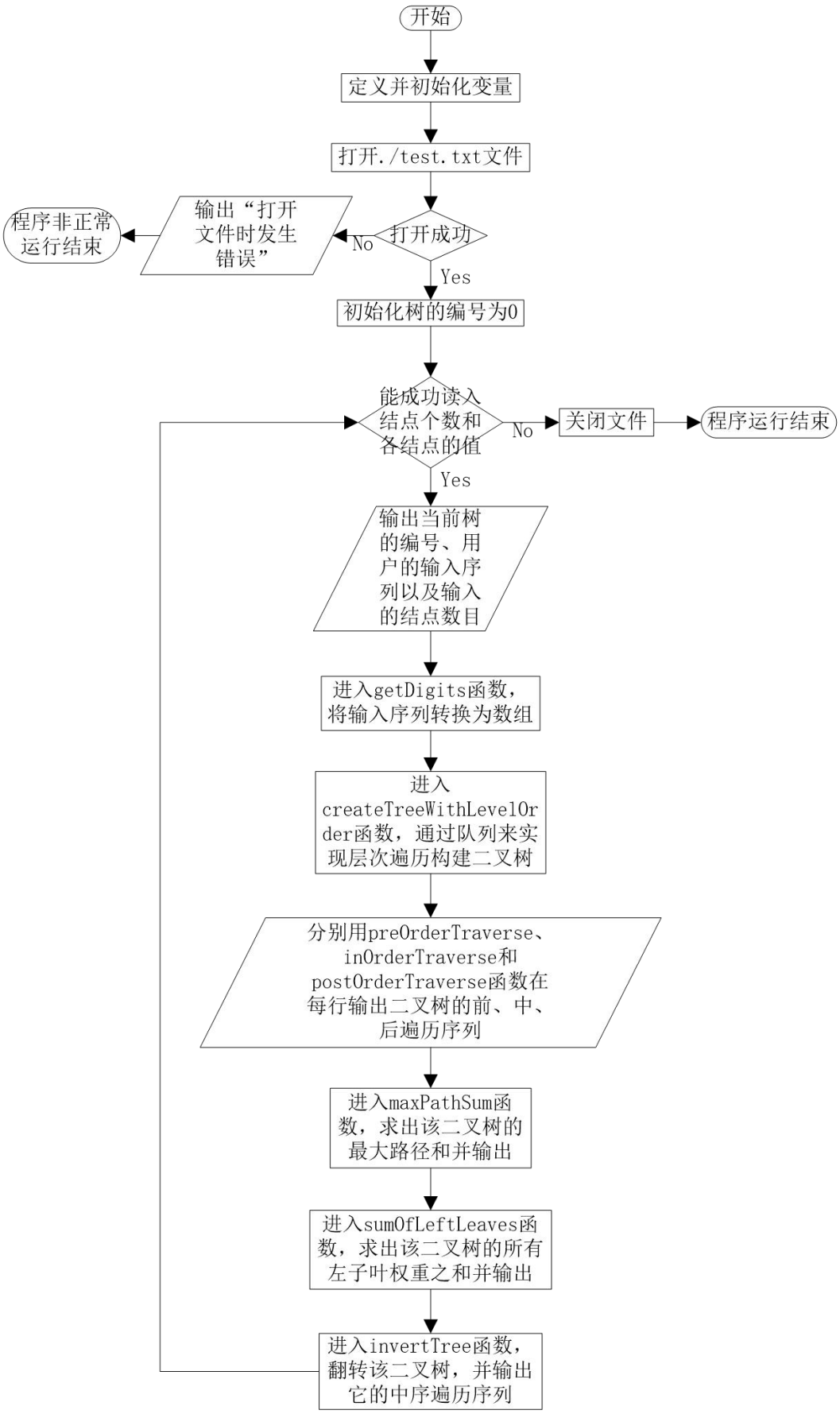
- TreeNodePtr invertTree(TreeNodePtr root)

参数：传入二叉树的根节点

功能：求取二叉树的镜像

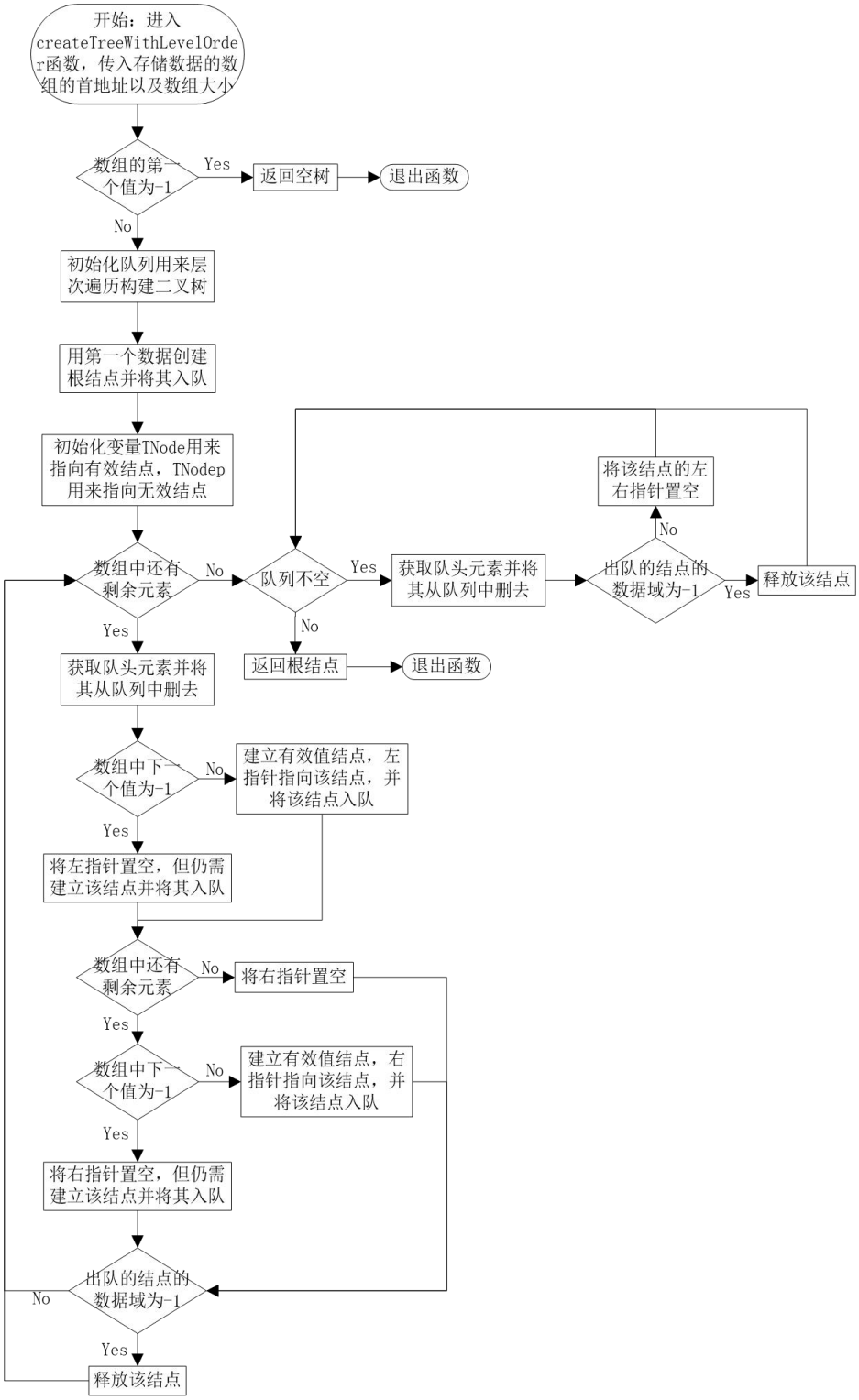
返回值：返回翻转后的二叉树的根节点

2.3 程序整体流程

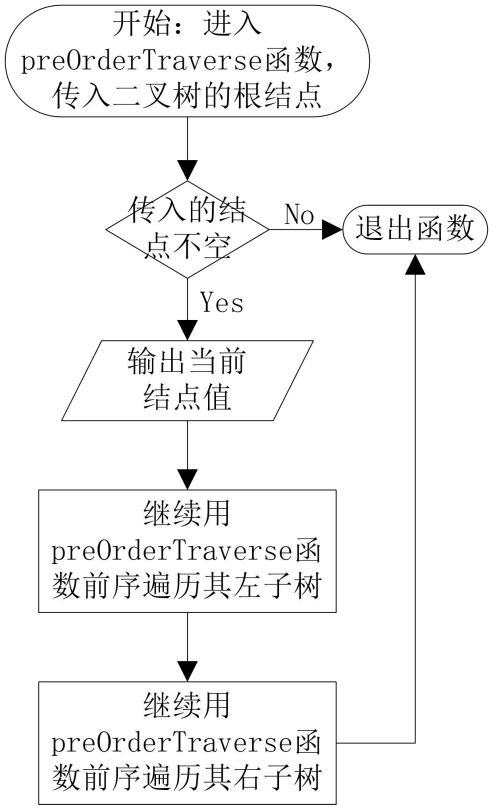

开始
定义并初始化变量
打开./test.txt文件
打开成功
No
输出“打开文件时发生错误”
程序非正常运行结束
Yes
初始化树的编号为0
能成功读入结点个数和各结点的值
No
关闭文件
程序运行结束
Yes
输出当前树的编号、用户的输入序列以及输入的结点数目
进入getDigits函数，将输入序列转换为数组
进入createTreeWithLevelOrder函数，通过队列来实现层次遍历构建二叉树
分别用preOrderTraverse、inOrderTraverse和postOrderTraverse函数在每行输出二叉树的前、中、后遍历序列
进入maxPathSum函数，求出该二叉树的最大路径和并输出
进入sumOfLeftLeaves函数，求出该二叉树的所有左子叶权重之和并输出
进入invertTree函数，翻转该二叉树，并输出它的中序遍历序列

核心算法流程：

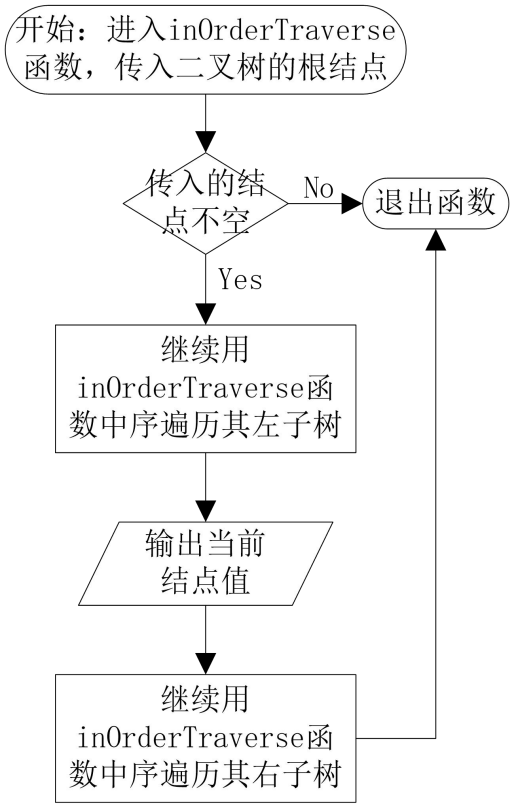
◆ TreeNodePtr createTreeWithLevelOrder(int *data, int size)



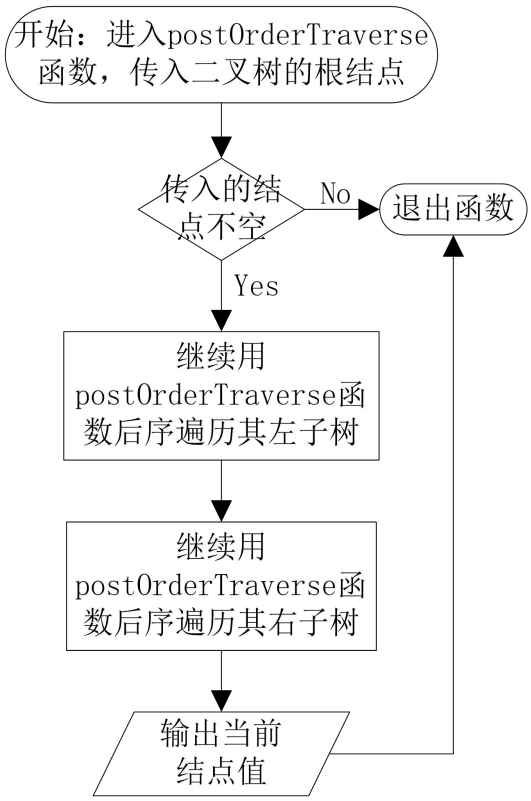
◆ void preOrderTraverse(TreeNodePtr root)



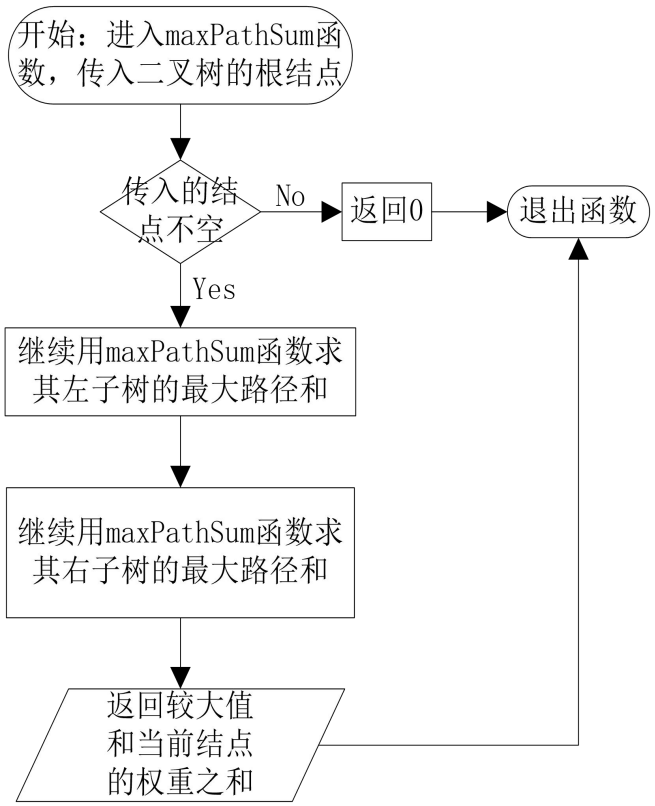
◆ void inOrderTraverse(TreeNodePtr root)



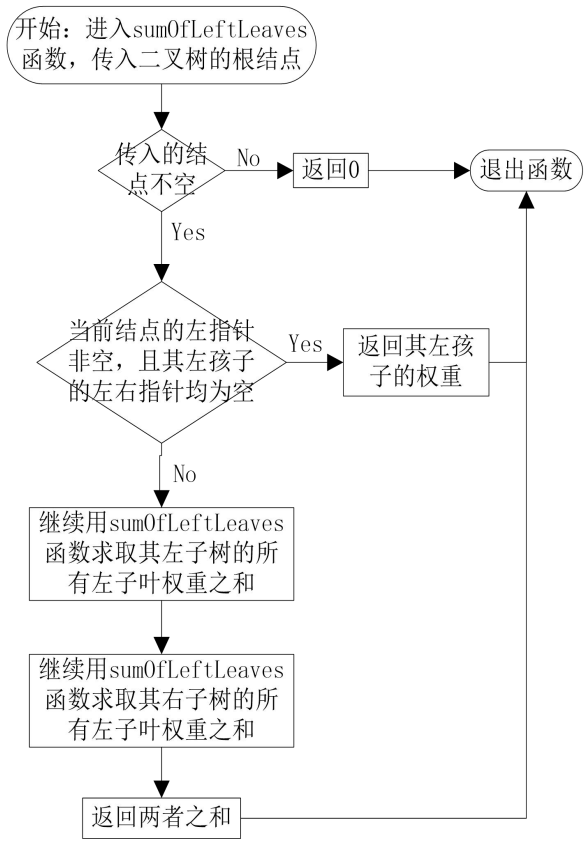
◆ void postOrderTraverse(TreeNodePtr root)



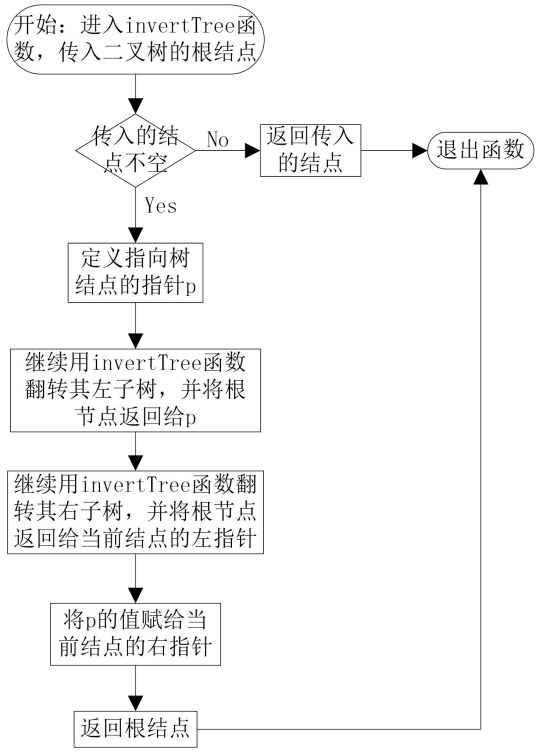
◆ int maxPathSum(TreeNodePtr root)



◆ int sumOfLeftLeaves(TreeNodePtr root)



◆ TreeNodePtr invertTree(TreeNodePtr root)



三、用户手册

输入数据的方式以及实现各种功能的操作方式：

用户在输入时，用每两行数据来表示一棵二叉树，第一行只有一个数据，表示要输入的结点个数；第二行按层次遍历输入每个节点的值，数据个数为第一行输入的值，两个数据之间用空格隔开，注意每一层的空结点也需要输入，用#表示（也算作一个数据）。

对于用户输入的每一棵二叉树，程序会从 0 开始给它们编号，并进行如下的操作：

首先，在第一行中，程序会输出“Case ”以及当前树的编号，然后输出“, data: ”以及用户的输入序列 (即每棵树的第二行输入数据)，接着输出“, nodes number: ”以及输入的结点数目（即每棵树的第一行输入数据)。

然后，程序进入第一部分，在第二行输出“Answer for task 1 is:”。在第三行输出“preOrderTraverse is:”以及该树的前序遍历序列；在第四行输出“inOrderTraverse is:”以及该树的中序遍历序列；在第五行输出“postOrderTraverse is:”以及该树的后序遍历序列。（若为空树，则不输出）

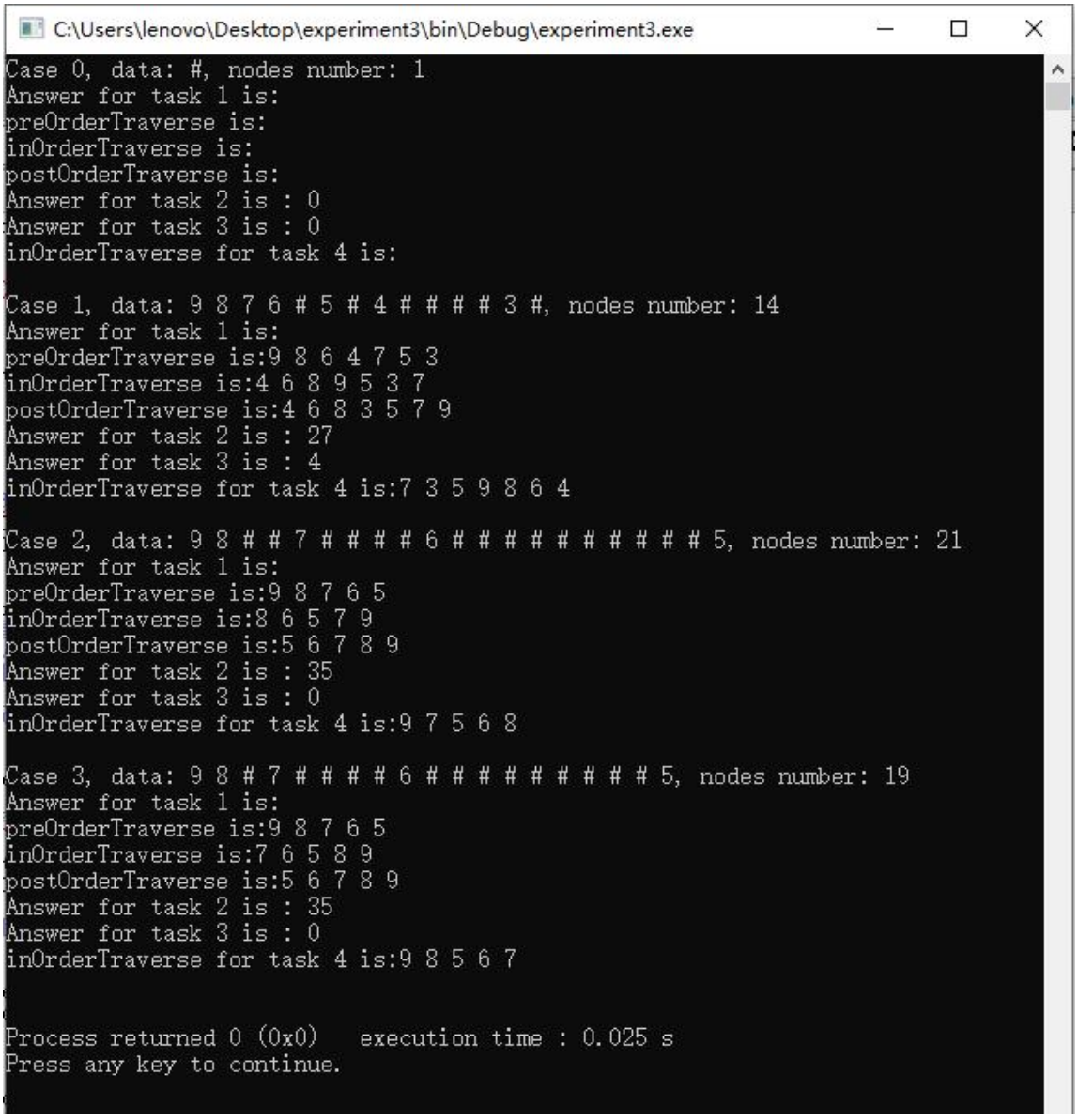
程序进入第二部分，在第六行输出“Answer for task 2 is : ”以及该二叉树的最大路径和。

程序进入第三部分，在第七行输出“Answer for task 3 is : ”以及该二叉树的所有左子叶权重之和。

程序进入第四部分，在第八行输出“inOrderTraverse for task 4 is:”以及

该二叉树的镜像（用中序遍历序列输出）。

四、四、结果



五、总结

该实验涉及的数据结构有链式存储的二叉树以及队列，涉及的算法主要是层

次遍历建立二叉树、前序、中序、后序遍历输出二叉树、深度优先遍历二叉树、递归等。

在这次实验中，我在利用队列层次遍历建立二叉树这一部分耗时较久，主要是不太熟悉题目所给的建立方式。开始遇到空结点时，我并未将其入队，导致后续数据的层数错位，后来我修改了代码，将空结点同样入队，只是不再接到二叉树上，有效地解决了问题。

此外，在递归的构造上（主要是递归的终止条件）我也花了一些时间思考。经过几个递归函数的练习，我对于递归的构造也渐渐熟悉了。

通过这次实验，我熟悉了树的存储结构和基本操作，也将理论课中学到的算法付诸了实践，收获很多。在这次实验中，我较快地理解了助教所给的函数（其中的出队函数和我们平时所常用的写法不同，我在自己写程序前发现了这一点，从而避免了写完整个程序再进行找 bug 的过程），完成速度相较于上次有了很大的进步。