|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构》实验报告** |
|  |
| 实验三  树型结构及其应用  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 邓皓元 | | 学 号: | 200110618 | | 专 业: | 计算机科学与技术 | | 日 期: | 2021-4-13 | |

# 一、问题分析

(1) 题目1

按层次遍历建立二叉树，并输出该二叉树的前序遍历、 中序遍历和后序遍历的序列。

**分析：将输入的数据存入二叉树（层次遍历构建二叉树）（运用两个队列），，再删除val=-1结点。利用递归完成二叉树前序遍历、中序遍历、后序遍历函数的构建**

(2) 题目2

给定一棵二叉树，路径定义为从树的根节点到叶子结点的任意路径，求取该二叉树的最大路径和，路径和定义为一条路径中各节点的权重之和。

**分析：运用深度优先遍历来求取二叉树的最大路径和，利用递归（结点最大路径和=max（结点左孩子最大路径和，结点右孩子最大路径和）+结点val）**

(3) 题目3

给定一棵二叉树，求取该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。

**分析：运用递归（二叉树所有左子叶权重之和=二叉树左孩子左子叶权重之和+二叉树右孩子左子叶权重之和）**

(4) 题目4

给定一棵二叉树，求取该树的镜像，并输出镜像二叉树的中序遍历。求取镜像即翻转该二叉树，如图1中的二叉树翻转后的镜像二叉树如图2所示。

**分析：运用递归翻转结点左右孩子，再对结点左右孩子进行翻转**

# 二、详细设计

## 2.1 设计思想

(1) 题目1

按层次遍历建立二叉树，并输出该二叉树的前序遍历、 中序遍历和后序遍历的序列。

**层次遍历构建二叉树：运用两个队列queue1和queue2，先将数据存入单独二叉树，再将单独的二叉树顺序存入queue2，先将第一个元素（根节点）存入queue1，进行循环：（每次取出queue2的2个元素并删除，每次取出queue1的1个元素并删除，将queue2顺序取出的2个元素作为queue1取出元素的左右孩子，再将queue1取出的元素的左右孩子存入queue1）（直到queue1空）**

**删除二叉树中val=1的结点：利用destoryTree函数删除val=-1结点及其子树**

**前序遍历：运用递归，先printf结点数据，再对左孩子进行函数递归，最后对右孩子进行函数递归**

**中序遍历：运用递归，先对左孩子进行函数递归，再printf结点数据，最后对右孩子进行函数递归**

**后序遍历；运用递归，先对左孩子进行函数递归，再对右孩子进行函数递归，最后printf结点数据**

(2) 题目2

给定一棵二叉树，路径定义为从树的根节点到叶子结点的任意路径，求取该二叉树的最大路径和，路径和定义为一条路径中各节点的权重之和。

**运用递归：（结点最大路径和=max（结点左孩子最大路径和，结点右孩子最大路径和）+结点val）**

(3) 题目3

给定一棵二叉树，求取该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。

**运用递归：（二叉树所有左子叶权重之和=二叉树左孩子左子叶权重之和+二叉树右孩子左子叶权重之和）**

**对结点的判断：若结点左孩子为叶结点，则赋值返回值=左孩子val+右孩子返回值，否则返回值=左孩子返回值+右孩子返回值**

(4) 题目4

给定一棵二叉树，求取该树的镜像，并输出镜像二叉树的中序遍历。求取镜像即翻转该二叉树，如图1中的二叉树翻转后的镜像二叉树如图2所示。

**运用递归：（交换结点左孩子和右孩子，并递归翻转左孩子和右孩子）**

## 2.2 存储结构及操作

(1) 存储结构

**链表：用来构建队列**

**二叉树：本次实验主要使用的存储结构**

**队列：用以构建二叉树**

1. 涉及的操作

**TreeNodePtr destoryTreeNULL(TreeNodePtr root)//删除树中val=-1的结点及其所有子树（运用destoryTree函数）**

**TreeNodePtr createTreeWithLevelOrder(int \*data, int size)//通过队列来实现层次遍历构建二叉树，并返回二叉树的头结点**

**void preOrderTraverse(TreeNodePtr root)//前序遍历二叉树**

**void inOrderTraverse(TreeNodePtr root)//中序遍历二叉树**

**void postOrderTraverse(TreeNodePtr root)//后序遍历二叉树**

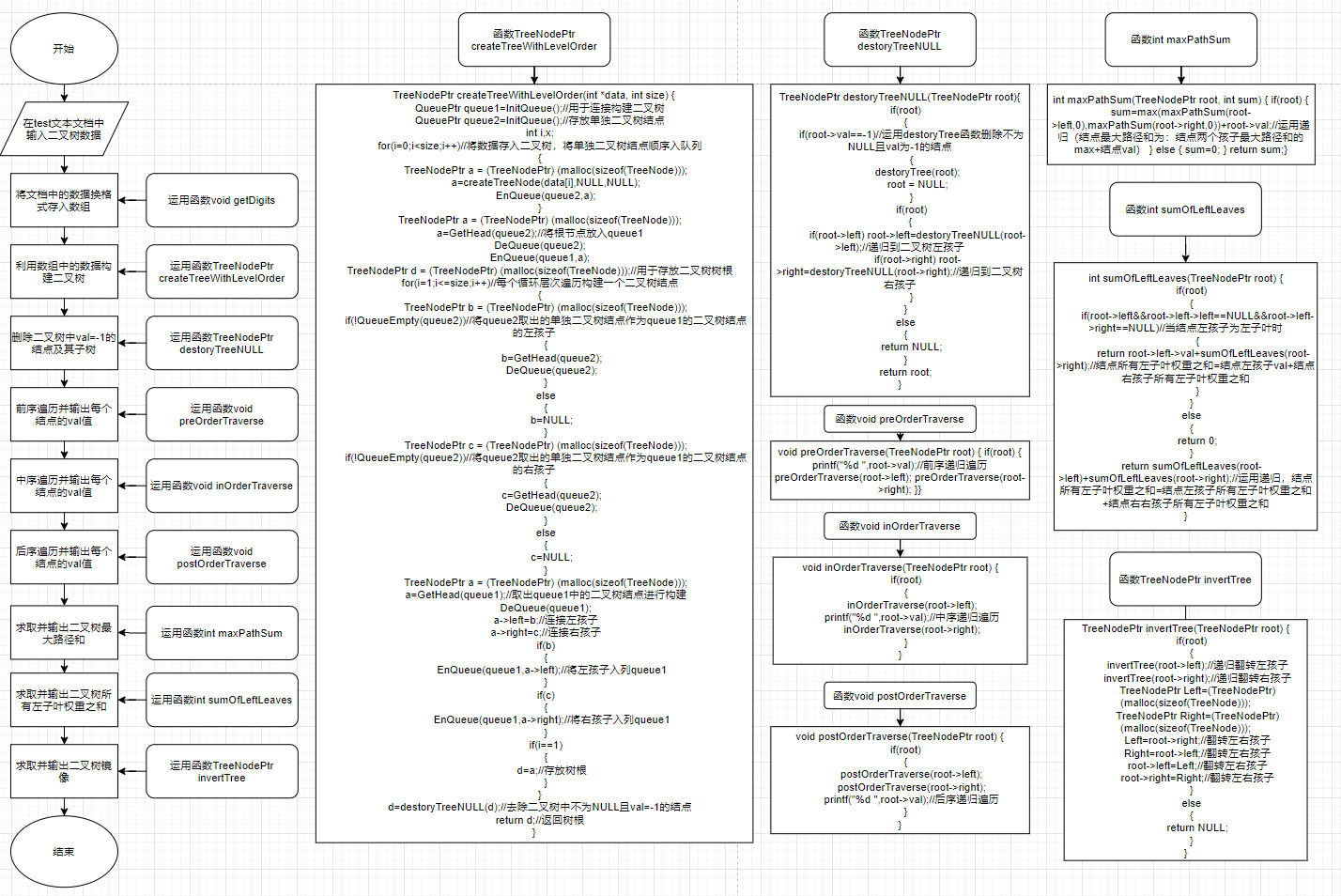
**int maxPathSum(TreeNodePtr root, int sum)//通过深度优先遍历来求取该二叉树的最大路径和**

**int sumOfLeftLeaves(TreeNodePtr root)通过递归求取该二叉树的所有左子叶权重之和**

**TreeNodePtr invertTree(TreeNodePtr root)通过递归求取该树的镜像，即翻转该二叉树**

## 2.3 程序整体流程

画出整体流程，及核心算法流程。



# 三、用户手册

(1)输入数据的方式：

**提前在文本文档test中输入需要输入的二叉树数据。**

**先输入二叉树元素数量，换行，输入数据（每个数据之间间隔一个空格）。**

**输入#表示该树结点为空。**

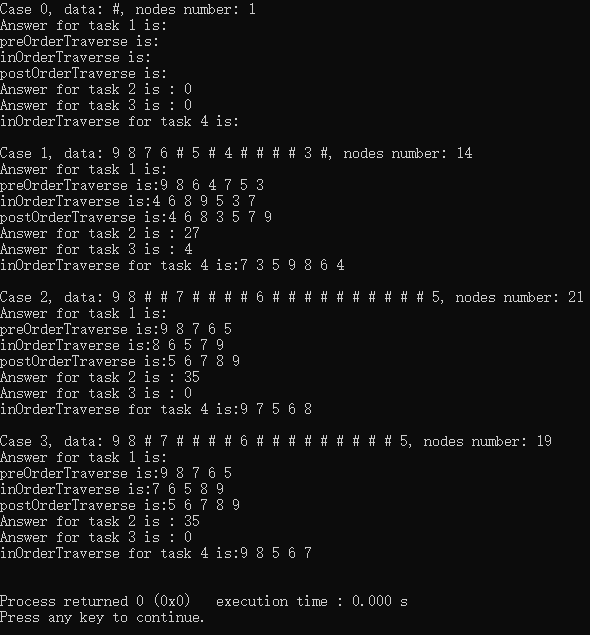
**若要输入下一个二叉树的数据，则换行，再次进行上述操作。**

(2)实现各种功能的操作方式

**无。只需输入二叉树的元素数据，就能自动构建二叉树并输出二叉树的前序遍历、中序遍历、后序遍历、二叉树的最大路径和、二叉树的所有左子叶权重之和、二叉树的镜像**

# 四、结果

程序正确运行的结果截图。



# 五、总结

**该实验涉及到的数据结构和算法，以及遇到的问题和收获。**

**该实验涉及到的数据结构有链表、队列、二叉树。**

**该实验涉及到的算法有构建二叉树（利用队列）、二叉树的前序遍历（递归）、中序遍历（递归）、后序遍历（递归）、求取二叉树的最大路径和（递归）、求取二叉树的所有左子叶权重之和（递归）、求取二叉树镜像（递归）。**

**在本次实验中，我遇到了许多问题，首先是利用队列构建二叉树的问题，得十分巧妙地利用两个队列不仅将数据存入二叉树且将单独二叉树连接，为了方便二叉树的构建，我将val=-1结点都构建了，并新构建了一个函数来删除二叉树中val=-1的结点及其子树。之后我在二叉树的函数运用上思考了许多，用void函数时经常会出错，返回二叉树的指针才能让函数正常进行。**

**收获：从这次实验中，我更加明白了队列和二叉树之间的各种灵活运用，对二叉树上的递归运用更加得心应手，也明白了函数修改二叉树最好需要返回二叉树的指针。**