华南农业大学信息学院

综合性、设计性实验

起止日期:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | | 软件学院 | **专业班级** | 软件工程R1 | | | | | | **学号** | 201734020124 | | | | | **姓名** | | 吴嘉贤 | | |
| **实**  **验**  **题**  **目** | 实现（平衡）二叉排序树的各种算法 | | | | | | | | | | | | | | | | | | |
| **自**  **我**  **评**  **价** | **项 目** | | | | | **算法设计** | | | **独立完成情况** | | | | **算法熟练程度** | | | | | | **测试通过** |
| 成功 | | 失败 | 独立 | | 帮助 | | 掌握 | | 了解 | | 不懂 | |
| 插入新结点 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 前序、中序、后序遍历二叉树（递归） | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 前序、中序、后序遍历的非递归算法 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 层次遍历二叉树 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 在二叉树中查找给定关键字 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 交换各结点的左右子树 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 求二叉树的深度 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 叶子结点数 | | | | |  | |  |  | |  | |  | |  | |  | |  |
| 删除某结点 | | | |  | |  | |  | |  |  | |  | | |  | |  |
| * + A---------完成实验要求的全部功能并运行通过，算法有一定的新意，程序代码符合书写规范，实验报告叙述清晰完整，有详尽的分析和总结。 * B---------完成实验要求的全部功能，程序代码符合书写规范，实验报告叙述 清晰完整。 * C---------完成实验要求的大部分功能，实验报告良好。 * D---------未按时完成实验，或者抄袭。 | | | | | | | | | | | | | | | | | | |
| **成绩** |  | | | | | | | | | | | | | | | | | | |

教师签名：杨秋妹

# 实现（平衡）二叉排序树的各种算法

软件工程R1 吴嘉贤 201734020124 完成日期2018.6.13

1. 分析题目要求
   1. 各个函数命名，包含的参数，返回值类型

//初始化队列，成功返回1，失败返回0

Status InitQueue( LinkQueue &Q );

//入队，成功返回1，失败返回0

Status EnQueue( LinkQueue &Q , BiTree e );

//出队，成功返回1，失败返回0

Status DeQueue( LinkQueue &Q , BiTree &e );

//初始化栈，成功返回1，失败返回0

Status InitStack( Stack &S );

//入栈，成功返回1，失败返回0

Status Push( Stack &S , BiTree e );

//出栈，成功返回1，失败返回0

Status Pop( Stack &S , BiTree &e );

//判断树是否为空，空返回1，非空返回0

Status StackEmpty( Stack S );

//搜索结点，成功返回1，失败返回0

Status SearchBST( BiTree T , int key , BiTree f , BiTree &p );

//插入结点，成功返回1，失败返回0

Status InserBST( BiTree &T , ElemType e );

//初始化二叉树，成功返回1，失败返回0

Status InitBST( BiTree &T , int e );

//输出元素e的值

Status PrintElement( ElemType e );

//非递归前序

void PreOrder( BiTree T , Status(\*Visit)(ElemType) );

//递归前序

void PreOrderTraverse( BiTree T, Status(\*Visit)(ElemType) );

//非递归后序

void PostOrder( BiTree T, Status(\*Visit)(ElemType) );

//递归后序

void PostOrderTraverse( BiTree T , Status(\*Visit)(ElemType) );

//非递归中序

void InOrderTraverse( BiTree T,Status (\*Visit)(int e) );

//递归中序

void InOrder( BiTree T , Status (\*Visit)(int e) );

//层次遍历二叉树

void ccTraverse( BiTree T , Status (\*Visit)(int e) );

//交换左右结点

void ExchangeBiTNode( BiTree &T );

//树深，返回树深

Status BiTreeHigh( BiTree T );

//叶子节点数，返回叶子结点树

Status BiTNodeNumber( BiTree T , int &Node );

//删除结点

void DeleteNode( BiTree &T );

//删除结点

void DeleteBST( BiTree &T , int key );

* 1. 输入的形式和输入值的范围

第一行输入初始化结点个数（1-100），第二行输入初始化的结点的值，第三行输入要查找的结点，第四行输入要插入的结点，第五行输入要删除的结点。

* 1. 输出的形式

按顺序依次为：

先序遍历，中序遍历，后序遍历。

先序遍历，先序遍历非递归。

中序遍历，中序遍历非递归。

后序遍历，后序遍历非递归。

先序遍历，中序遍历，后序遍历，层次遍历。

先序遍历，中序遍历，后序遍历。

先序遍历，中序遍历，后序遍历。

树高，叶子结点树。

* 1. 程序所能达到的功能

（1） 插入新结点

（2） 前序、中序、后序遍历二叉树 （递归）

（3） 前序、中序、后序遍历的非递归算法

（4） 层次遍历二叉树

（5） 在二叉树中查找给定关键字

（6） 交换各结点的左右子树

（7） 求二叉树的深度

（8） 叶子结点数

（9） 删除某结点

1. 解题思路
2. 插入新结点

搜索得到该结点的父节点，然后根据与父节点的大小关系插入左孩子或者右孩子。

1. 前序、中序、后序遍历二叉树 （递归）

前序先输出结点值然后递归的遍历左孩子和右孩子。

中序先递归左孩子再输出结点的值再递归右孩子。

后序先递归左右孩子再输出结点的值。

1. 前序、中序、后序遍历的非递归算法

前序用一个while循环只要树非空或者栈非空就进行循环操作。只要结点非空，就先输出结点值，再把右孩子和左孩子入栈，再出栈一个结点。

中序用一个while循环只要树非空或者栈非空就进行循环操作。只要结点非空就把结点推进栈里，并指向左孩子。否则出栈一个结点并访问结点值再指向右孩子。

后序用一个while循环只要栈非空就进行循环操作。先推出一个结点，判断结点是否可以输出，可以的话输出并标记为最后一个访问的结点。否则只要右结点存在则入栈右结点，左结点存在则入栈左结点。

1. 层次遍历二叉树

用一个while循环只要队列非空就进行循环操作。先输出结点值，如果左孩子存在则入队左孩子，右孩子存在则入队右孩子。

1. 在二叉树中查找给定关键字

如果该结点等于关键字值，则返回1，并令p指向这个结点。否则递归的调用左孩子和右孩子进行搜索。

1. 交换各结点的左右子树

交换左右结点，再递归的对左孩子和右孩子进行交换。

1. 求二叉树的深度

递归的调用函数，如果树为空返回0，否则返回左右树高中的最大值。

1. 叶子结点数

只要左右孩子都为null，叶子结点树就加一。再递归的对左右孩子调用函数。

1. 删除某结点

删除节点有三种情况分析：

    a.叶子节点：直接删除即可。

b.仅有左或右子树的节点：上移子树即可。

c.左右子树都有的节点。 用删除节点的直接前驱或者直接后继来替换当前节点，调整直接前驱或者直接后继的位置。

1. 调试分析
   1. 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析

后序遍历非递归算法实现的时候一直无法实现正常的输出，通过调试之后发现是没有考虑第一个结点要直接再while循环外部推进栈里的原因。

* 1. 使用的测试数据

（1） 7

40 20 60 18 50 56 90183590

（2） 57 4 9 5 8565

* 1. 算法的效率分析和改进设想

大部分操作的时间复杂度都为O（n）。

其中对于遍历的改进思想有判断结点是否是空，空可以不进入递归。

* 1. 经验和体会

二叉树是一种递归的数据结构类型，用递归的方式实现会方便许多，效率也高。

二叉树的删除操作是二叉树操作中比较复杂的操作，要考虑到三种情况。

1. 附录

