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: 李秋阳

学　号: 180110527

专　业: 计算机科学与技术

日　期: 2019-05-08

一、问题分析

实验原题：

（一）

采用交互问答方式按照先序序列建立二叉树，先输入根节点，每次都询问用户某个节点的左孩子是谁，右孩子是谁，其中字符’#’代表空节点。对建好的树进行前序遍历、中序遍历和后序遍历，并输出结果。

（二）

给定一棵二叉树的前序遍历和中序遍历结果（节点值用单个字符表示，不重复）构造这棵二叉树，输出这棵二叉树的前、中、后序遍历结果，并判断构造的二叉树是不是镜像对称的。

实验分析：

一、在给出非空根节点的情况下，通过递归的方式，分别依据输入的根节点的左孩子和右孩子进行建树，然后根据前序、中序、后序遍历的方式输出遍历结果。

二、找到前序遍历中根节点在中序遍历中的位置，其左边为其左子树，右边为其右子树，据此规律递归建树，再对建树结果进行后序遍历。然后递归判断同层次对应节点是否具有相同结构（是否为空等），以此判断是否是镜像对称。

二、详细设计

2.1 设计思想

2.1.1 实验一

在给出非空根节点的情况下，通过递归的方式，以交互式的方式输入根节点的左孩子和右孩子的 data 域数据，在此基础上建树，当所有叶节点的左子树和右子树都为空时递归终止，完成建树。然后根据前序、中序、后序遍历的方式输出遍历结果。

2.1.2 实验二

使用数组读取全部字符串，以前序结果的第一个为根节点作为递归起点，设定前序边界和中序边界，从前序左边界开始装填节点，在中序结果中寻找此节点位置，递归建立左子树：前序边界为(上次前序左边界+1~上次前序左边界+递归根位置-上次中序左边界)、中序边界为(上次中序左边界~递归根位置-1)。递归建立右子树：前序边界为(上次前序左边界+递归根位置-中序左边界~上次前序右边界)，中序边界为(递归根位置+1~上次中序右边界)，递归上述过程，当中序左边界超过右边界时，则没有叶节点，返回 NULL 结束递归。利用递归对同一层次对应节点进行判断，当传入节点都为空时返回 true 结束递归，当出现任意节点不对称时返回 false，退出递归，若递归结束都没有返回 false 而是返回了 true 则说明是镜像对称。

2.2 存储结构及操作

(1) 存储结构

实验一结构体

```
typedef struct BiNode
{
    char data;
    struct BiNode *lchild;
    struct BiNode *rchild;
}BiNode;
```

实验一树的链式储存结构

```
void Creat_BiTree(BiNode* T)
{
    char ch1,ch2;
    printf("Please enter the lchild\n");
    scanf("%c",&ch1);
    getchar();
    if(ch1=='#')
        T->lchild=NULL;
    else
    {
        BiNode*lchild=(struct BiNode*)malloc(sizeof(struct BiNode));
        T->lchild=lchild;
        lchild->data=ch1;
        printf("%c\n",ch1);
        Creat_BiTree(T->lchild);
    }
    printf("Please enter the rchild\n");
    scanf("%c",&ch2);
    getchar();
    if(ch2=='#')
        T->rchild=NULL;
    else
    {
        BiNode*rchild=(struct BiNode*)malloc(sizeof(struct BiNode));
        T->rchild=rchild;
        rchild->data=ch2;
        printf("%c\n",ch2);
        Creat_BiTree(T->rchild);
    }
}
```

实验二结构体

```
typedef char TElemType;
typedef struct TNode *Position;
typedef Position BiTree;
struct TNode{
    TElemType Data;
    BiTree lchild;
    BiTree rchild;
};
```

数组

```
TElemType Msg[MaxData];
TElemType PreOrd[MaxData/2];
TElemType InOrd[MaxData/2];
```

实验二树的链式储存结构

```
BiTree BuildTree(char PreOrd[],int PreStar,int PreEnd,char InOrd[],int InStar,int InEnd)
{
    int root;
    if(InStar>InEnd)
        return NULL;
    BiTree BT=(BiTree)malloc(sizeof(struct TNode));
    BT->Data=PreOrd[PreStar];
    root=FindRoot(InOrd,InStar,InEnd,PreOrd[PreStar]);
    BT->lchild=BuildTree(PreOrd,PreStar+1,PreStar+(root-InStar),InOrd,InStar,root-1);
    BT->rchild=BuildTree(PreOrd,PreStar+(root-InStar)+1,PreEnd,InOrd,root+1,InEnd);
    return BT;
}
```

(2) 涉及的操作

实验一操作

a. void Creat_BiTree(BiNode* T) 递归创建树

b. void Preorder(struct BiNode *T) 前序遍历并输出结果

c. void Inorder(struct BiNode *T) 中序遍历并输出结果

d. void Postorder(struct BiNode *T) 后序遍历并输出结果

实验二操作

e. int FindRoot(TElemType InOrd[],int star,int end,char key) 查找当前元素在中序遍历中的位置

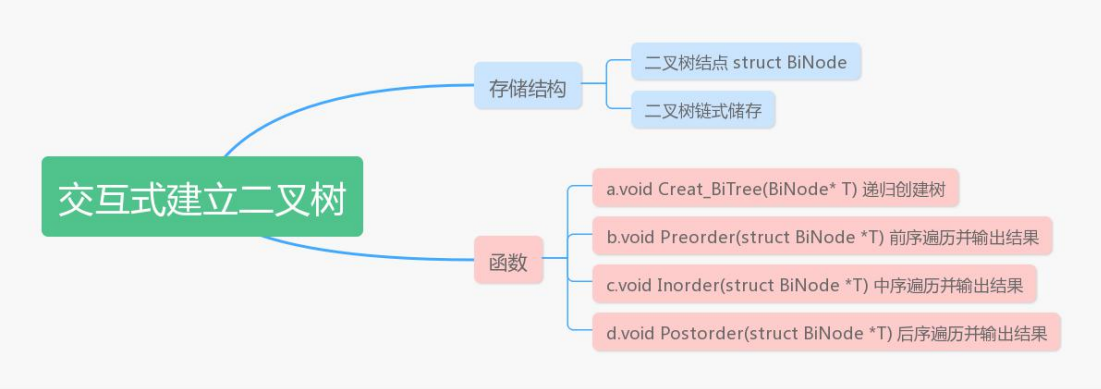
f. BiTree BuildTree(char PreOrd[],int PreStar,int PreEnd,char InOrd[],int InStar,int InEnd) 递归创建树

g. void Postorder_Traversal(BiTree BT) 后序遍历并输出结果

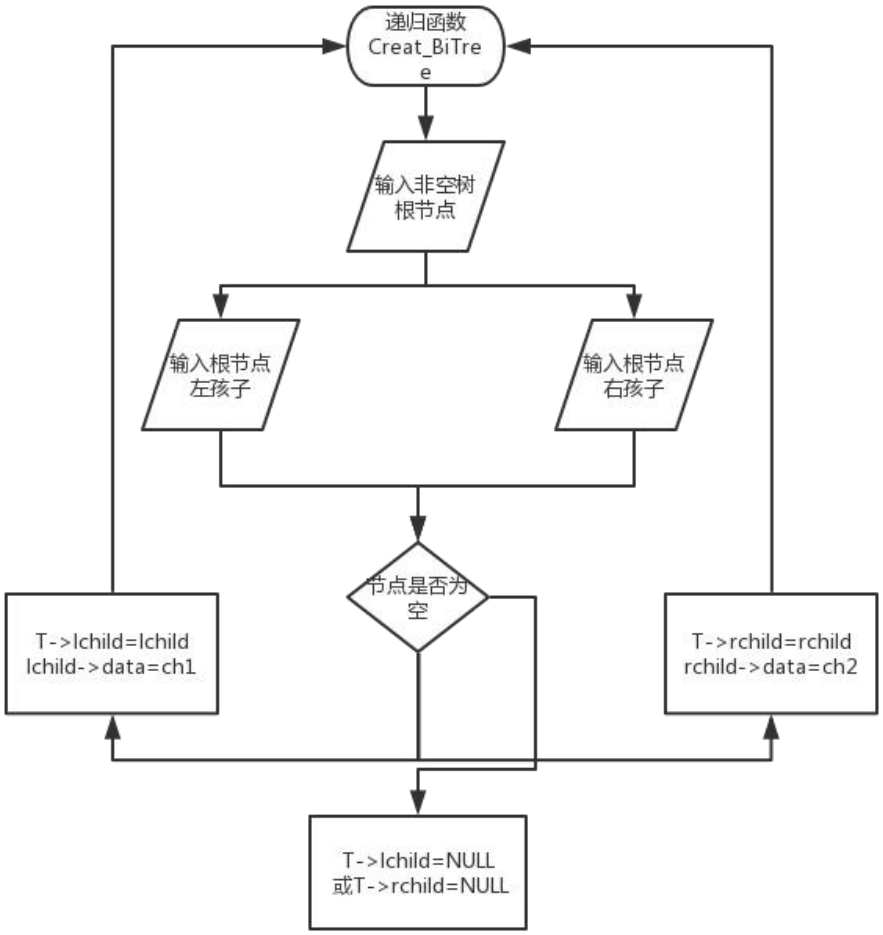
h. bool IsMirror(BiTree x,BiTree y) 检测是否镜像对称

2.3 程序整体流程

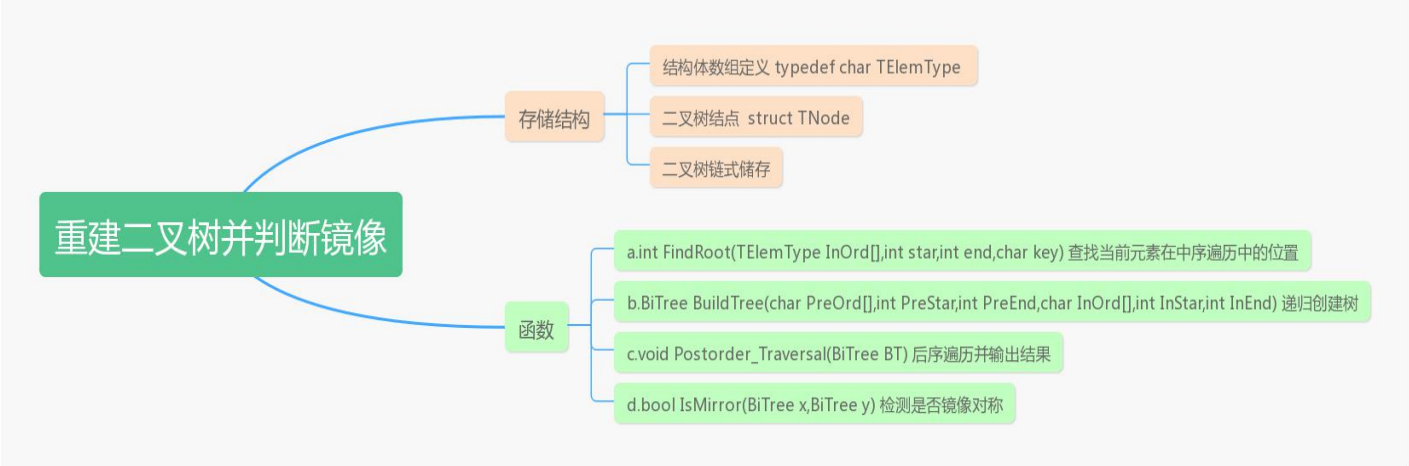
实验一



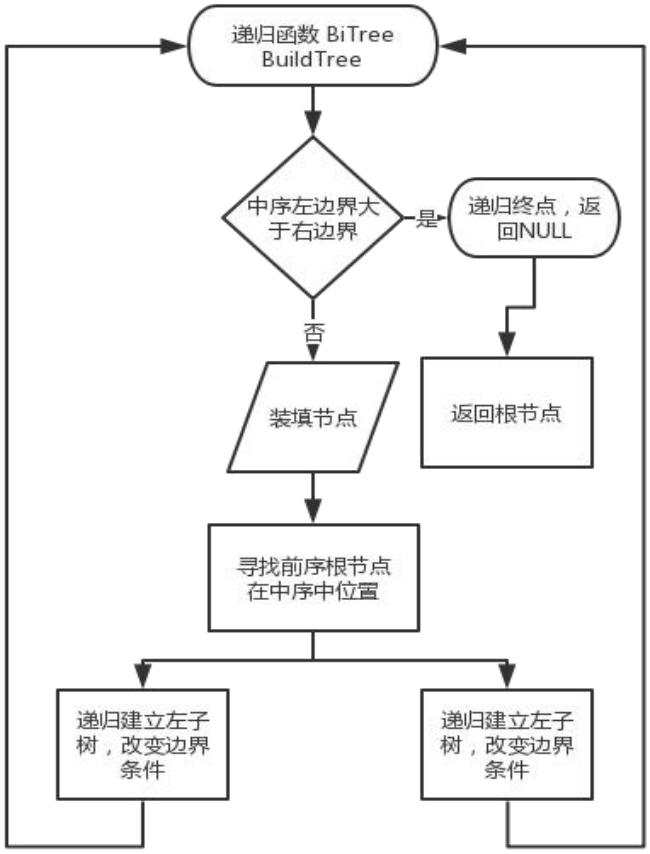
递归建立树

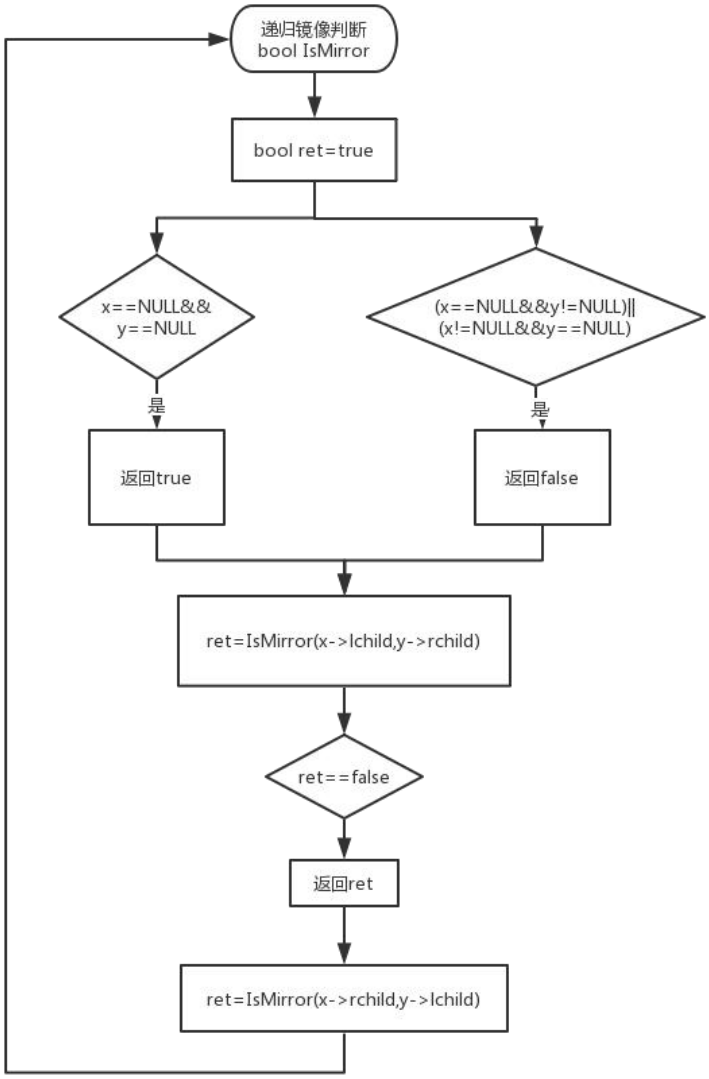


实验二


重建二叉树并判断镜像
存储结构
结构体数组定义 typedef char TElemType
二叉树结点 struct TNode
二叉树链式储存
函数
a.int FindRoot(TElemType InOrd[],int star,int end,char key) 查找当前元素在中序遍历中的位置
b.BiTree BuildTree(char PreOrd[],int PreStar,int PreEnd,char InOrd[],int InStar,int InEnd) 递归创建树
c.void Postorder_Traversal(BiTree BT) 后序遍历并输出结果
d.bool IsMirror(BiTree x,BiTree y) 检测是否镜像对称

递归创建树


递归函数 BiTree BuildTree
中序左边界大于右边界
是
递归终点，返回NULL
否
装填节点
返回根节点
寻找前序根节点在中序中位置
递归建立左子树，改变边界条件
递归建立左子树，改变边界条件



三、用户手册

3.1 实验一

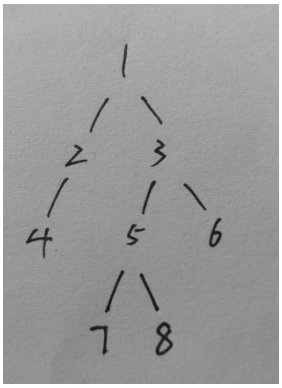
根据提示，先输入整棵树的树根节点，然后根据先序遍历的顺序，输入根节点的左子树和右子树，当输入‘#’说明其左（右）孩子为空。注意，当对所有叶节点的左右子树全部输入‘#’赋空时，输入自动结束，树完成构建，自动输出树的先序、中序、后序遍历结果。

3.2 实验二

根据提示，分别输入一颗二叉树的先序、中序遍历结果，中间用“,”隔开，程序将自动输出其后序遍历并判断是否为镜像对称。注意，二叉树的各层次数据不能重复，请输入不一样的数据。

四、结果

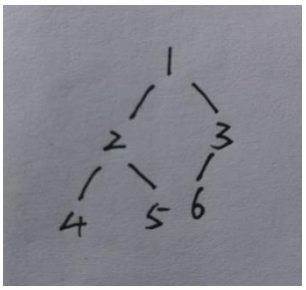
实验一



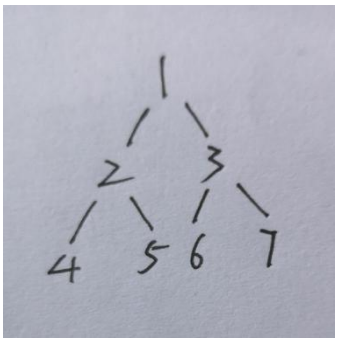

```
start task  Create Tree in PreOrder
1
Please enter the lchild
2
Please enter the lchild
#
Please enter the lchild
#
Please enter the rchild
#
Please enter the rchild
#
Please enter the rchild
3
Please enter the lchild
5
Please enter the lchild
7
Please enter the lchild
#
Please enter the rchild
#
Please enter the rchild
8
Please enter the lchild
#
Please enter the rchild
#
Please enter the rchild
6
Please enter the lchild
#
Please enter the rchild
#
```

```
先序
12435786
中序
42175836
后序
42785631
```

实验二




```
Preorder Traversal,Inorder Traversal
124536,425163
Postorder Traversal:452631
It's not a mirror BiTree.
Process returned 0 (0x0)   execution time : 15.248 s
Press any key to continue.
```




```
Preorder Traversal,Inorder Traversal
1245367,4251637
Postorder Traversal:4526731
It's a mirror BiTree.
Process returned 0 (0x0)   execution time : 44.473 s
Press any key to continue.
```

五、总结

这次实验运用了递归的方式来构建和重建二叉树，同时也用递归的方式来进行先序遍历、中序遍历、后序遍历以及对二叉树是否镜像对称的判断。在构建过程中，通过链式结构来存储树比顺序结构更加节约空间。在使用递归时，对指针的使用以及递归条件的设定需要进行细致的思考，包括如何设置递归入口和递归出口等。

本次实验难度并不是很大，但让我更加深刻地理解了二叉树的性质以及如何根据不同条件构建二叉树，拓展了对递归的用法的认识，受益匪浅。