# 习题3 树结构

3-1设二叉树T中度为1的结点11个，度为2的结点12个，则二叉树T共有(  C    )个叶子结点。

(A) 11

(B) 12

(C) 13

(D) 36

3-2 设树T的度为4，其中度为1，2，3和4的结点个数分别为4，2，1，1，则T中的叶子数为( D )。

(A) 5

(B) 6

(C) 7

(D) 8

3-3 已知一棵度为k的树中，有n1个度为1的结点，n2个度为2的结点，…，nk个度为k的结点。试计算该树的叶子结点数。

设结点总数为n，叶子结点数为n0

则 1\*n1+2\*n2+3\*n3+……k\*nk=n-1

n0+n1+n2+n3+……nk=n

解得n0=n2+2\*n3+3\*n4+……+(k-1)\*nk+1

3-4 证明：如果二叉树T的叶子结点数为n0，度为2的结点数为n2，则n0=n2+1。

设结点总数为n，度为1的结点数为n1.

因为 n0+n1+n2=n, 1\*n1+2\*n2=n-1

得到n0=n2+1

3-5 对于任意非空二叉树，要设计出其后序遍历的非递归算法而不使用栈结构，最适合的方法是对该二叉树采用( B )存储结构。

(A) 二叉链表

(B) 三叉链表

(C) 索引

(D) 顺序

3-6 一棵二叉树的叶子结点在其先序、中序和后序序列中的相对位置( C )。

(A) 肯定发生变化

(B) 可能发生变化

(C) 不会发生变化

(D) 无法确定

3-7 设二叉树T按照二叉链表存储，则下列递归算法的主要功能是( B

    )。

int F(BiTree T)

{

if (!T) return 0;

x=F(T->Lchild);

y=F(T->Rchild);

if (y>x) x=y;

return x+1;

}

(A) 交换二叉树T的左右子树

(B) 计算二叉树T的高度

(C) 计算二叉树T的叶子结点数

(D) 先遍历左子树，再遍历右子树

3-8 已知二叉树T的先序序列为ABCDEF，中序序列为CBAEDF, 则T的后序序列为( A )。

(A) CBEFDA

(B) FEDCBA

(C) CBEDFA

(D) 不确定

3-9 简述由先序序列和中序序列构造二叉树的基本操作方法。

由先序序列第一个数据即为树的根结点，在中序中找到其位置，其左边的数据在其左子树上，其右边的数据在其右子树上。以此方法，分别考虑根结点的左右子树，实现构造二叉树。用递归实现。

3-10 已知二叉树的先序序列为ebadcfhgjik，中序序列为abcdefghijk，试画出该二叉树。

e

b f

a d h

c g j

i k

3-11 已知二叉树T的中序序列和后序序列分别为

(中序) 3, 7, 11, 14, 18, 22, 27, 35

(后序) 3, 11, 7, 14, 27, 35, 22, 18

试画出二叉树T。

18

14 22

7 35

3 11 27

3-12 已知二叉树T按照二叉链表存储，设计算法，计算T中叶子结点的数目。

void CalcuYeZiNode(BiTree \* p,int &number)//number应被初始化为0

{

if(!p)

{

number++;

return;

}

CalcuYeZiNode(p->leftChild,number);

CalcuYeZiNode(p->rightChild,number);

return;

}

3-13 已知二叉树T按照二叉链表存储，设计算法，交换T的左子树和右子树。

void ChangeLeftTreeAndRightTree(BiTree \*p)

{

TNode \*q=p->leftChild;

p->leftChild=p->rightChild;

p->rightChild=q;

return ;

}

3-14 先序后继线索化算法是根据二叉链表建立先序后继线索二叉链表，其基本原则是在前驱空指针域中写入后继线索，即将右子树的( B )指针写入左子树的最后一个叶子结点右指针域。

(A) 线索

(B) 根结点

(C) 前驱结点

(D) 后继结点

3-15 设计算法，在先序线索二叉树中，查找给定结点p在先序序列中的后继。

TNode\* SearchNodeBehind(BiTree \*T,TNode \*p)

{

return p->rightChild;

}

3-16对n (n≥2)个权值均不相同的字符构造哈夫曼树T，不正确的叙述( A )。

(A) T一定是一棵完全二叉树

(B) T中一定没有度为1的结点

(C) T中两个权值最小的结点一定是兄弟结点

(D) T中任一分支结点的权值一定不小于下一层任一结点的权值

3-17 按照“逐点插入方法”建立一个二叉排序树，树的形状取决于(B )。

(A) 数据序列的存储结构

(B) 数据元素的输入次序

(C) 序列中的数据元素的取值范围

(D) 使用的计算机的软、硬件条件

3-18 用利用逐点插入法建立序列(50, 72, 43, 85, 75, 20, 35, 45, 65, 30)对应的二叉排序树以后，查找元素35要在元素间进行( B )次比较。

(A) 3

(B) 4

(C) 5

(D) 8

3-19 给定n个整数，设计算法实现：

(1) 构造一棵二叉排序树；

(2) 从小到大输出这n个数。

来源自实验四第一题

(1)void CreateBiTree(BiTree &T)

{

cout << "请输入想要创建的结点总数:";

int n;

cin >> n;

for (int i = 1; i <= n; i++)

{

int keynumber;

char datanumber;

cout << "请输入第" << i << "个结点的关键字:";

cin >> keynumber;

cout << "请输入第" << i << "个结点的数据:";

cin >> datanumber;

if (i == 1)

{

BiTree p;

InitialBiTree(p);

p->data = datanumber;

p->key = keynumber;

T->rightChild = p;

}

else

InsertBiTNode(T->rightChild, keynumber, datanumber);

}

}

void InsertBiTNode(BiTree &T,int key,char ch) // T为根结点

{

if (!T)

{

BiTree p;

InitialBiTree(p);

p->data = ch;

p->key = key;

T = p;

return;

}

if (key < T->key)

{

if (T->leftChild == NULL)

{

BiTree p;

InitialBiTree(p);

p->data = ch;

p->key = key;

T->leftChild = p;

return;

}

else

InsertBiTNode(T->leftChild, key, ch);

}

else

{

if (T->rightChild == NULL)

{

BiTree p;

InitialBiTree(p);

p->data = ch;

p->key = key;

T->rightChild = p;

return;

}

else

InsertBiTNode(T->rightChild, key, ch);

}

}

(2)

void PreOrder(BiTree &T) //T为根结点

{

if (!T)

return;

cout << "关键字:" << T->key << " 数据:" << T->data << endl;

PreOrder(T->leftChild);

PreOrder(T->rightChild);

return;

}

3-20设计一个求结点x在二叉树中的双亲结点算法。

void GetParent(BiTree \*T,TNode \*child,TNode \* &parent)

{

if(!T)

return ;

if(T->leftChild==child||T->rightChild==child)

{

parent=T;

return ;

}

GetParent(T->leftChild,child,parent);

GetParent(T->rightChild,child,parent);

return ;

}