树

上课题目

中序:BDCAHGKFE

前序:ABCDEFGHK

后序:KHGFEDCBA

讨论题

第六章 树和二叉树 1、证明：一棵二叉树的所有终端结点（叶结点）在前序序列、中序序列以及后 序序列中都按相同的相对位置出现。

2、试写一个判定所给定二叉树是否为二叉排序树的算法，设此二叉树以二叉链 表为存储结构，且树中结点的关键字均不同。 3、证明若二叉排序树中的一个节点存在两个孩子，则它的中序后继节点没有左孩子，且它的中序前趋节点没有右孩子。 4、给定二叉树 T，其中任意两个结点 p、q，设计算法，求 p 和 q 最近的共同祖 先。 5、若一棵二叉树，左右子树均有三个结点，其左子树的前序序列与中序序列相同，右子树的中序序列与后序序列相同，试构造该树。

课后题

习题集：6.8、6.10、6.31、6.36、6.41、6.42、6.43、6.49、6.56、6.61、6.62、6.65、6.69

6.8

证明：由于是满k叉树，因而只有度为0和度为k的结点。设度为0的结点有n0个，度为k的结点有n1个，结点总数为n，则：

结点关系：n = n1 + n0

分支关系：n-1 = k▪n1

因而可得：n0 = (k-1)n1 + 1

6.10

(1)根据题意，此树只有度为0和度为2的结点。设其度为2的结点有n2个，总结点数为n总，则：

结点关系：n总 = n + n2

分支关系：n总-1 = 2 ▪n2

因而可得：n总 = 2n - 1

(2)用数学归纳法证明：

当n=1时，只有一个叶子结点，也就是树中只有根结点，l1=1，因而：

$$\sum_{i=1}^{n} 2^{-(l_i-1)} = 2^{-(l_1-1)} = 2^{-(1-1)} = 1$$

满足题意。

假设当树中有n(n>1)个结点时，等式成立，且p为其任一叶子结点。

此时有：

$$\sum_{i=1}^{n} 2^{-(l_i-1)} = 2^{-(l_1-1)} + 2^{-(l_2-1)} + \cdots + 2^{-(lp-1)} + \cdots + 2^{-(ln-1)} = 1$$

那么当叶子结点个数为n+1时，树的总结点数增2，叶子结点增1。假设增加的叶子结点来源于p的孩子结点。此时，p不再作为叶子结点，其孩子结点充当叶子结点，设其两个孩子结点为x个y，则有等式：lp +1 = lx = ly。

因而有：

$$2^{-(lx-1)} = 2^{-(ly-1)} = 2^{-lp}$$

故：

$$2^{-(lx-1)} + 2^{-(ly-1)} = 2 \cdot 2^{-lp} = 2^{1-l_p} = 2^{-(l_p-1)}$$

因此有：

$$\begin{aligned}\sum_{i=1}^{n+1} 2^{-(l_i-1)} &= 2^{-(l_1-1)} + 2^{-(l_2-1)} + \cdots + 2^{-(lx-1)} + 2^{-(ly-1)} + \cdots + 2^{-(ln-1)} \\ &= 2^{-(l_1-1)} + 2^{-(l_2-1)} + \cdots + 2^{-(lp-1)} + \cdots + 2^{-(ln-1)} \\ &= 1\end{aligned}$$

至此，原等式得证，即

$$\sum_{i=1}^{n} 2^{-(l_i-1)} = 1$$

成立。

6.31

证明：命题等价于证明由一棵二叉树的先序序列和中序序列可唯一确定任一结点的位序，进一步等价于证明由一棵二叉树的先序序列和中序序列可唯一确定任一结点的父结点值及其属于此父结点的左孩子还是右孩子。

(1)根据二叉树性质，每一个结点最多只有一个父结点，所以无论其遍历序列如何，其父结点的值要么不存在（根结点），要么存在且唯一。

(2)二叉树先序遍历序列为根结点-左子树-右子树，中序遍历序列为左子树-根结点-右子树。由此可得，对于任一结点，在先序序列中寻找其前驱，若前驱不存在，说明此结点是树的根结点，若前驱存在，再判断此结点与其前驱在中序序列中的相对次序，若其前驱位于其右边， 则可断定该结点为其父结点的左子树，否则为右子树。

这样一来，根据先序序列和中序序列，就可以唯一确定每一个结点的父结点值及其相对位置（左子树或右子树），进而确定每个结点的位置，由此就可确定唯一的二叉树。

6.36

```
int Resemble(BiTree B1, BiTree B2)

{

    if(BiTreeEmpty(B1) && BiTreeEmpty(B2))          //都为空树

        return 1;

    else

    {

        if(!BiTreeEmpty(B1) && !BiTreeEmpty(B2))    //都不为空树

        {

            if(Resemble(B1->lchild, B2->lchild) && Resemble(B1->rchild, B2->rchild))    //判断左右子树

                return 1;

        }

    }

```

```
    return 0;

}
```

6.41

```
int FindK(BiTree T, Elem *e, int *order, int k) //order用来计数

{

    if(T)

    {

        (*order)++;


        if(*order==k)

        {

            *e = T->data;

            return 1;

        }

        else

        {

            if(FindK(T->lchild, e, order, k))

                return 1;

            if(FindK(T->rchild, e, order, k))

                return 1;

        }

    }


    return 0;
```

```
42
43  }
```

6.42

```
1   int Count(BiTree T)
2
3   {
4
5   •   int count = 0;
6
7   •
8
9   •   if(T)
10
11  •   {
12
13  •       if(T->lchild==NULL && T->rchild==NULL)
14
15  •           count++;
16
17  •       else
18
19  •       {
20
21  •           count += Count(T->lchild);
22
23  •           count += Count(T->rchild);
24
25  •       }
26
27  •   }
28
29  •
30
31  •   return count;
32
33  }
```

6.43

```
1   void Swap(BiTree T)
```

```
{
    BiTree p;

    if(T)
    {
        p = T->lchild;
        T->lchild = T->rchild;
        T->rchild = p;

        Swap(T->lchild);
        Swap(T->rchild);
    }
}
```

6.49

```
int CompleteBiTree(BiTree T)
{
    int i, j;
    BiTree p[100] = {};                    //树指针数组，模拟队列
    int order[100] = {};

    i = j = 0;

    if(T)                                  //遍历的同时为各结点编号
```

```
18
19  {
20
21      p[j] = T;
22
23      order[j] = 1;
24
25      j++;
26
27
28
29      while(i<j)
30
31      {
32
33          if(i>0 && order[i]>order[i-1]+1)
34
35              return 0;            //若结点序号不连续，则非完全二叉树
36
37
38
39          if(p[i]->lchild)
40
41          {
42
43              p[j] = p[i]->lchild;
44
45              order[j] = 2*order[i];
46
47              j++;
48
49          }
50
51
52
53          if(p[i]->rchild)
54
55          {
56
57              p[j] = p[i]->rchild;
58
59              order[j] = 2*order[i]+1;
60
61              j++;
62
63          }
64
65
66
```

```
            i++;

        }

    }

    return 1;

}
```