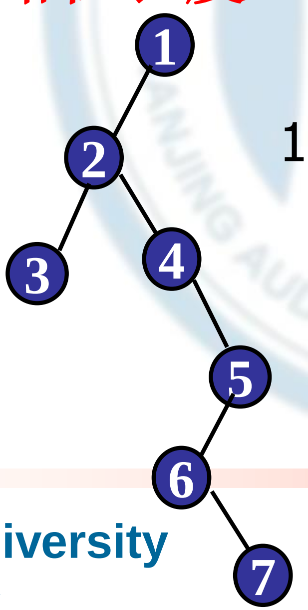
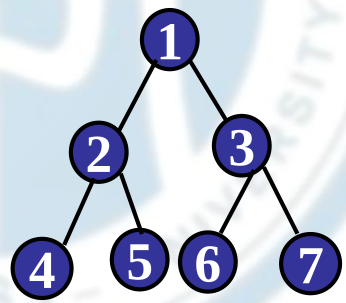
6.6 哈夫曼树及其应用

6.6.1 哈夫曼树（最优二叉树---带权路径长度最短的树）

- 基本概念
  - 路径：从树中一个结点到另一个结点之间的分支。
  - 路径长度：路径上的分支数目称为路径长度。
  - 树的路径长度：从树根到每一结点的路径长度之和。



17

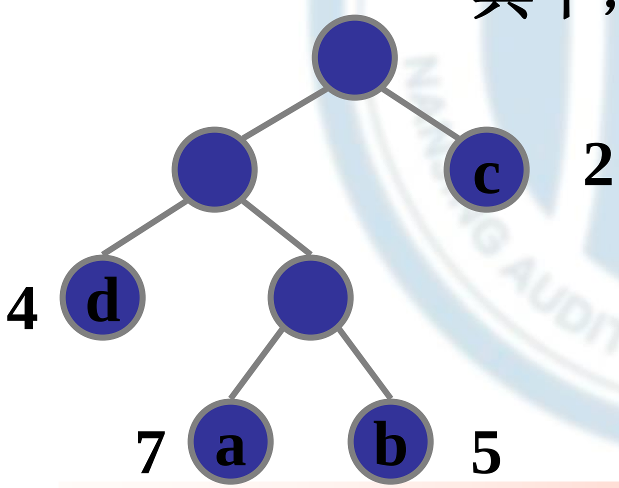


10

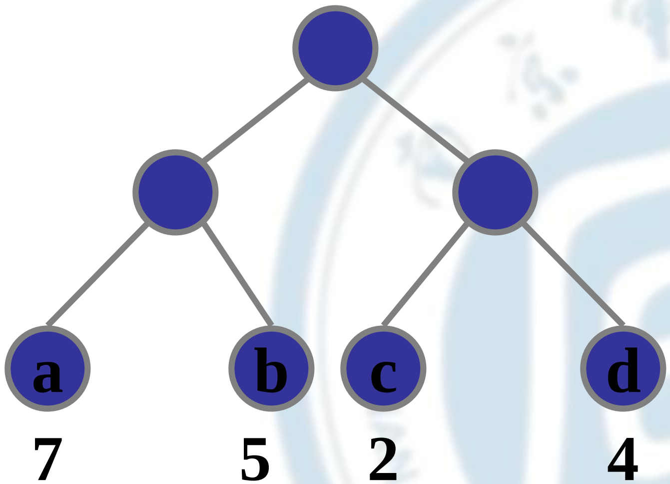
完全二叉树是路径长度最短的二叉树

- 结点的带权路径长度：从该结点到树根之间的路径长度与结点上的权值的乘积。
- 树的带权路径长度：树中所有叶子结点的带权路径长度之和。通常记作 $WPL = \sum_{k=1}^{n} w_k l_k$

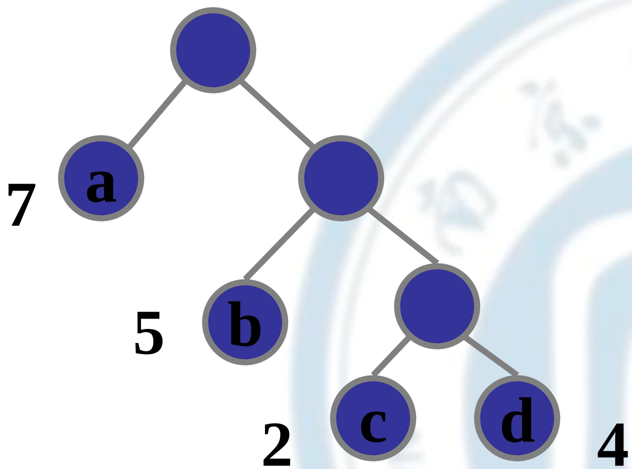
其中，w_k叶子结点的权值，l_k叶子结点的路径长度。



WPL=7*3+5*3+2*1+4*2=46



WPL=7*2+5*2+2*2+4*2=36



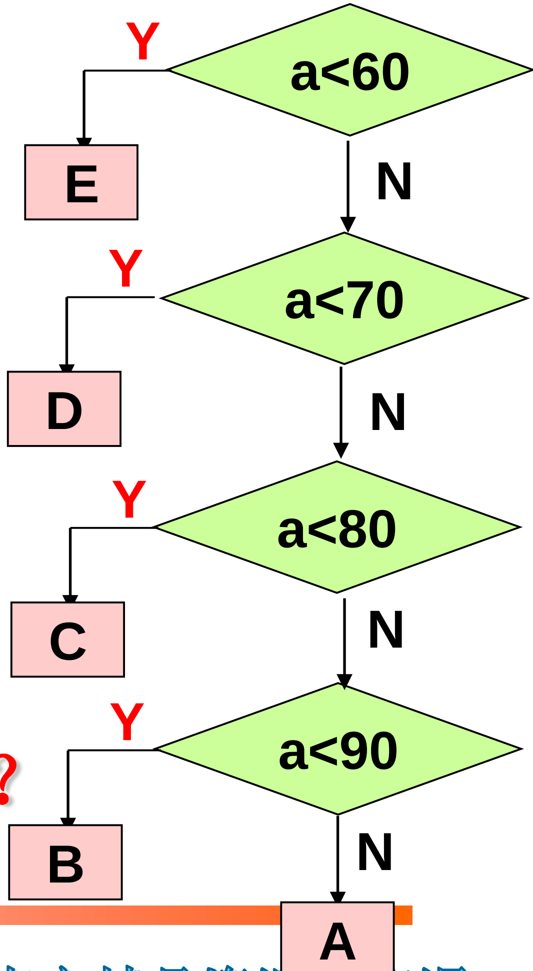
WPL=7*1+5*2+2*3+4*3=35

加权后路径长度最小的并非是完全二叉树，而是**权大的叶子离根最近**的二叉树。

例：百分制成绩转换成五级分制成绩

设实际学生成绩的分布规律如下：

等级	E	D	C	B	A
分数段	0~59	60~69	70~79	80~89	90~100
比例	0.05	0.15	0.40	0.30	0.10



按此程序流程，有80%的数据需进行3次或3次以上比较才能得出结果。

如何设计程序才能使得比较的总次数最少？

哈夫曼树构造算法

- 哈夫曼树：设有n个权值{w1,w2,……wn}，构造一棵有n个叶子结点的二叉树，每个叶子的权值为wi，WPL最小的二叉树。
- 构造哈夫曼树的过程（哈夫曼算法）
  - 根据给定的n个权值{w1,w2,……wn}，构造n棵只有根结点的二叉树，令初始权值为wj；
  - 在森林中选取两棵根结点权值最小的树作左右子树，构造一棵新的二叉树，置新二叉树根结点权值为其左右子树根结点权值之和；
  - 在森林中删除这两棵树，同时将新得到的二叉树加入森林中；
  - 重复上述两步，直到只含一棵树为止，这棵树即哈夫曼树。

哈夫曼树的形态不是唯一的，但对具有一组权值的各哈夫曼树的WPL是唯一的。

例：用w={5,29,7,8,14,23,3,11}构造哈夫曼树

w={5,29,7,8,14,23,3,11}

w={29,7,8,14,23,11,8}

w={29,14,23,11,8,15}

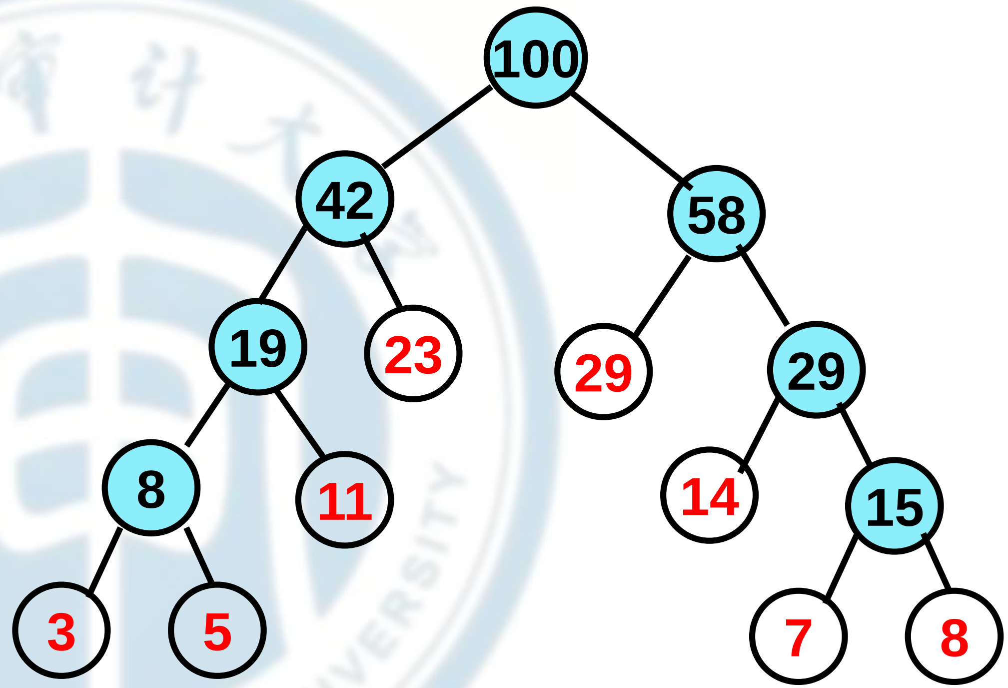
w={29,14,23,15,19}

w={29,23,19,29}

w={29,29,42}

w={42,58}

w={100}

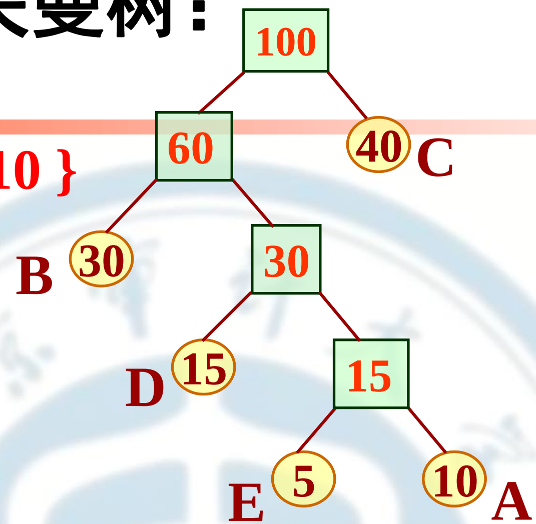


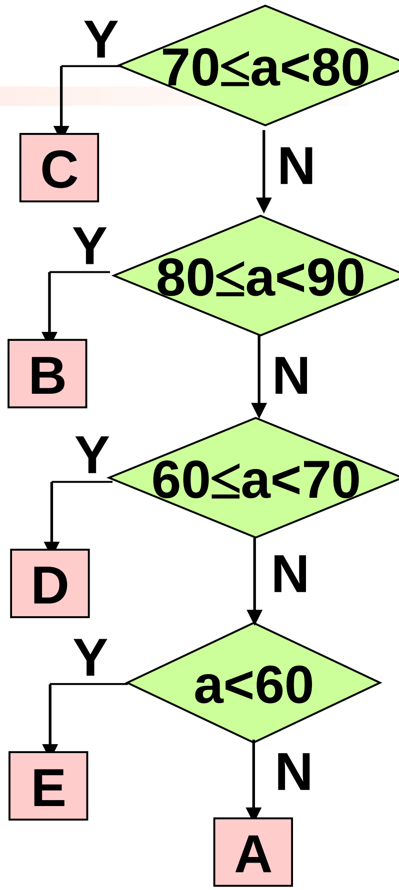
WPL= (23+29) *2 + (11+14) *3+(3+5+7+8)*4=271

以各个分数段上学生所占的百分比作为权值，构造一棵哈夫曼树：

问题的解决

W={ 5, 15, 40, 30, 10 }





（设学生总人数为常数N）

总的比较次数S为：

$$S = Nx(5\%x4+10\%x4+15\%x3+30\%x2+40\%x1)$$

$$= Nx1\%x(5x4+10x4+15x3+30x2+40x1)$$

WPL

S=N x 1% x WPL

6.6.2 哈夫曼编码

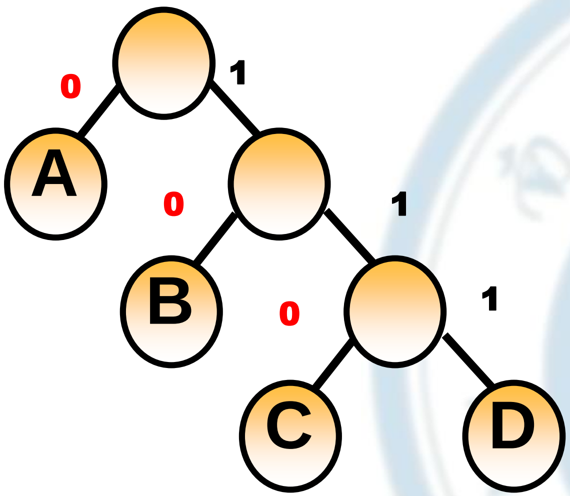
- 背景
  - 历史上，远距离通信的主要手段是电报，即将需传送的文字转换成由二进制的字符组成的字符串。
  - 编码时需要遵循以下原则
    - 解码的结果唯一；
    - 发送的二进制编码尽可能短。
- 两类二进制编码
  - 等长编码：各个字符的编码长度相等。
    - 优点：解码简单。
    - 缺点：编码长度可能不最短。
  - 不等长编码：各个字符的编码长度不等。
    - 优点：编码长度尽可能地短。
    - 缺点：解码困难。

例如：传送电文“ABACCDA”

- 等长编码
  - A：00 B：01 C：10 D：11
  - 编码结果00010010101100，长度为14位。
  - 解码方以两位为一字段解码。
- 不等长编码
  - 原则：出现次数较多的字符编码短，次数较少的字符编码长。
  - A：0 B：00 C：1 D：01
  - 编码结果000011010，长度为9位。
  - 解码方无法解码，因为解码的结果不唯一。

- 前缀编码
  - 任意一个字符的编码都不能是另一个字符的编码的前缀，这种编码称为前缀编码。
- 哈夫曼编码（同时满足代码长度短，且解码唯一）
  - 目标：使电文总长最短。
  - 以字符出现的次数为权，构造一棵赫夫曼树；将树中结点引向其左孩子的分支标“0”，引向其右孩子的分支标“1”；每个字符的编码即为从根到每个叶子的路径上得到的0、1序列，这样得到的编码称为哈夫码编码。
  - 哈夫曼编码为前缀编码。
  - 以这组编码传送电文可使电文总长最短(对所有其它前缀编码而言)。

设有如图所示的哈夫曼树：

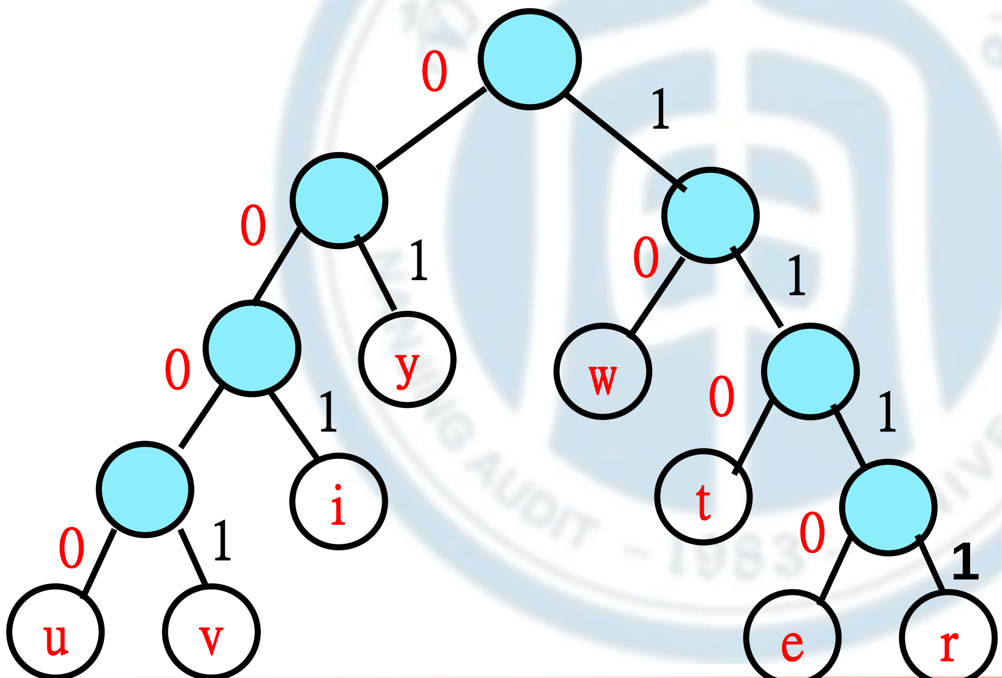


A	B	C	D
0	10	110	111

任一条从根结点到叶子结点的路径都不可能经过其它任一叶子结点。这就保证了这样得到的编码一定是**前缀编码**。

哈夫曼编码

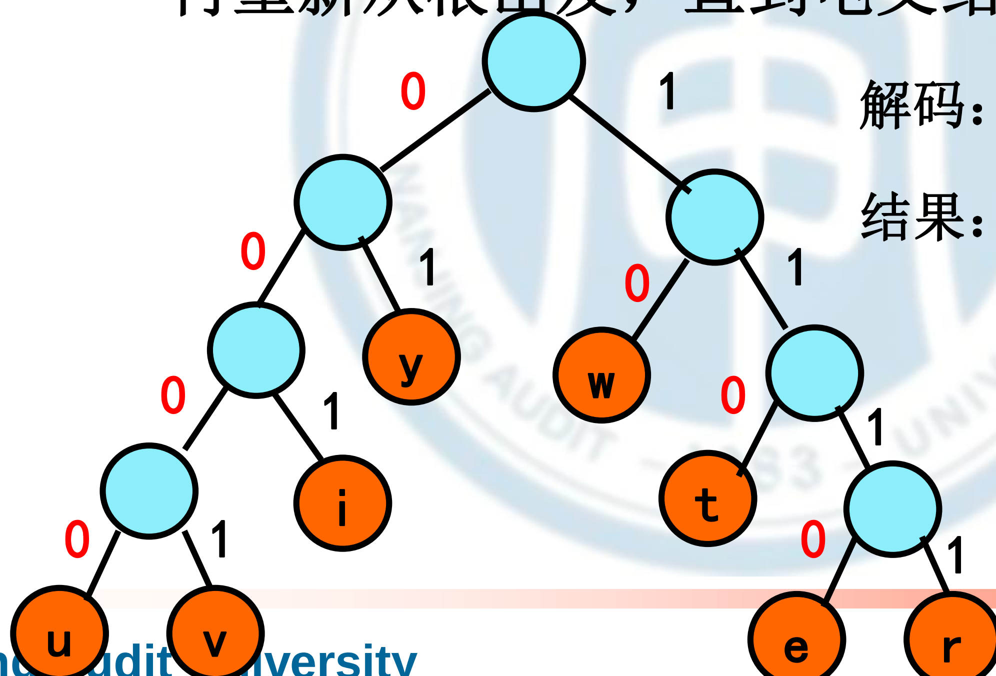
字符集	v	w	e	r	t	y	u	i
频率	5	29	7	8	14	23	3	11



v 0001
w 10
e 1110
r 1111
t 110
y 01
u 0000
i 001

哈夫曼解码

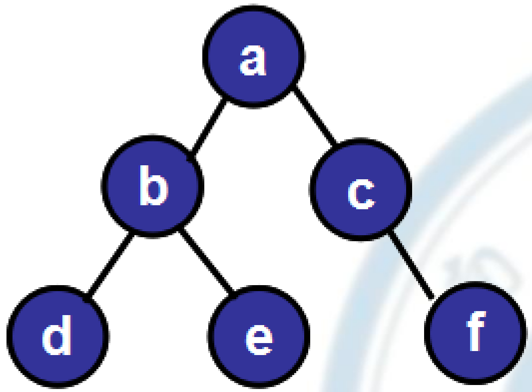
- 从哈夫曼树**根**开始，从待译码电文中逐位取码。
- 若编码是“0”，则向左走；若编码是“1”，则向右走，一旦到达叶子结点，则译出一个字符；
- 再重新从根出发，直到电文结束。

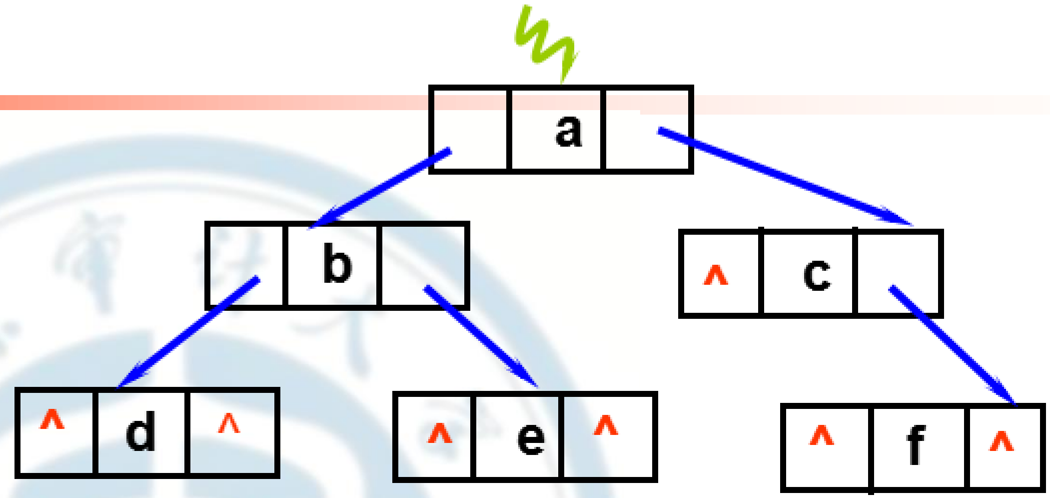


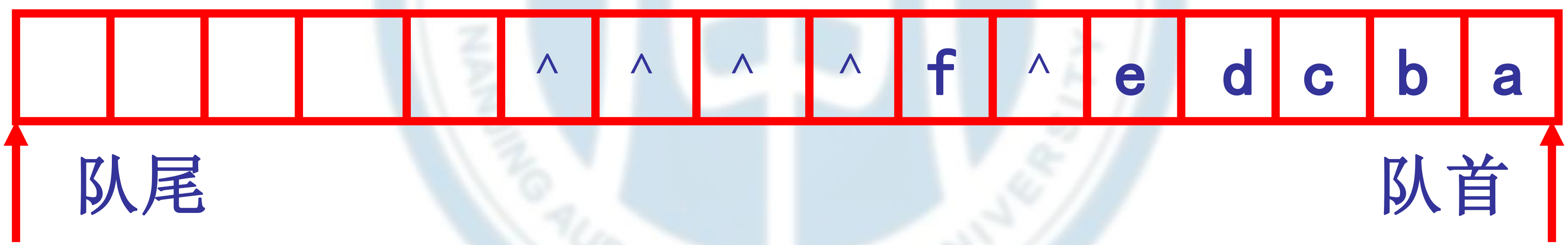
解码：**1111110000100111010**

结果：**review**

判断二叉树是否是完全二叉树







a b c d e

```
Status  IsCompleteTree(BiTree Bt) {
   Sq q; BiTree e;
   InitSq(q);
   InSq(q,Bt);     //根节点入队列
   OutSq(q,e);
   while(e!=NULL) {     //遇到空前，持续出队
       InSq(q,e->lchild);  //不管是不是NULL，左孩子先入队
       InSq(q,e->rchild);   //右孩子入队
       OutSq(q,e);
   }
   while(!Isempty(q)) {   //如果队列非空，留在里面的必定全是 NULL
    OutSq(q,e);
        if (e!=NULL)    //有一个不是 NULL 则判定不是完全二叉树
          return ERROR;
   }
    return OK;
}
```

求树的高度

```
int TreeDepth(CSTree T){
CSTree p;
        int depth,max=0;
        if(!T) // 树空
                return 0;
        for(p=T->firstchild;p;p=p->nextsibling)
                    {           depth=TreeDepth(p);
                   if(depth>max)
                 max=depth;
                }
        return max+1;
}
```

本章小结

- 树结构中的数据元素之间存在着“一对多”的关系，它为计算机应用中出现的具有层次关系的数据，提供了自然的表示方法。
- 二叉树是和树不同的另一种树型结构。
- 二叉树的几个重要特性应该熟练掌握的。
- 树和二叉树的遍历算法是实现各种操作的基础。
- 哈夫曼树和哈夫曼编码的构造方法。