数据结构（C语言版）（第2版）

树和二叉树

哈夫曼树及其应用

主讲教师：汪红松

教 学 内 容 Contents

一、哈夫曼树的基本概念

哈夫曼(Huffman)树又称最优树，是一类带权路径长度最短的树，在实际中有广泛的用途。

哈夫曼树最典型、最广泛的应用是在编码技术上，利用哈夫曼树，可以得到平均长度最短的编码。这在通讯领域是极其有价值的。

计算机程序操作码的优化也可以利用哈夫曼树实现。

一、哈夫曼树的基本概念

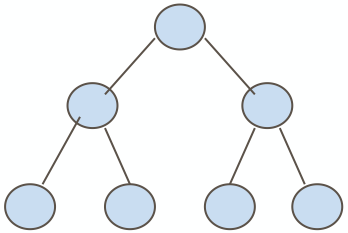
1.基本概念

路径：从树中一个结点到另一个结点之间的分支构成这两个结点之间的路径。

路径长度：路径上的分支数目称作路径长度。

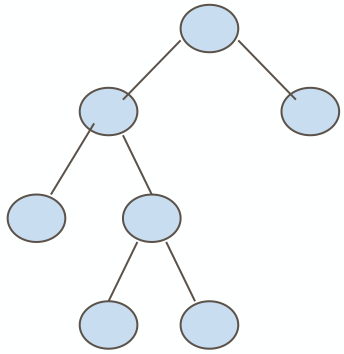
树的路径长度：从根到每个结点的路径长度之和(PL)。

例：



a

PL(a)=1+1+2+2+2+2=10



b

PL(b)=1+1+2+2+3+3=12

一、哈夫曼树的基本概念

1.基本概念

结点的权：给树中每个结点赋予一个具有实际意义的数值，我们称该数值为这个结点的权。

带权路径长度：在树形结构中，我们把从树根到某一结点的路径长度与该结点权的乘积，称做该结点的带权路径长度。

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，称为树的带权路径长度，通常记为WPL：

$$WPL=\sum_{i=1}^{n} w_i \times l_i$$

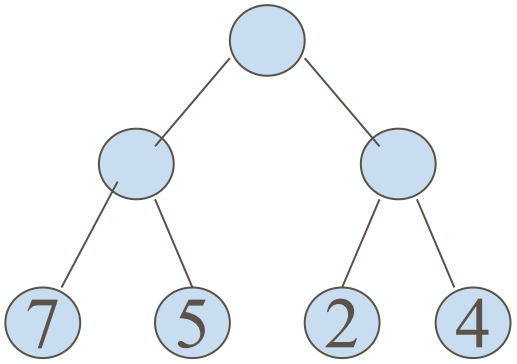
其中：n为叶子结点的个数；

w_i为第i个叶子的权值；

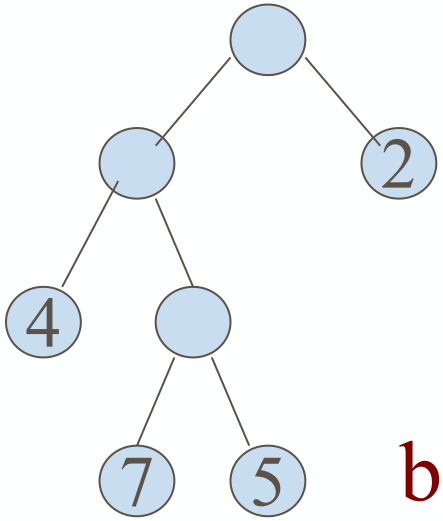
l_i为第i个叶子结点的路径长度。

一、哈夫曼树的基本概念

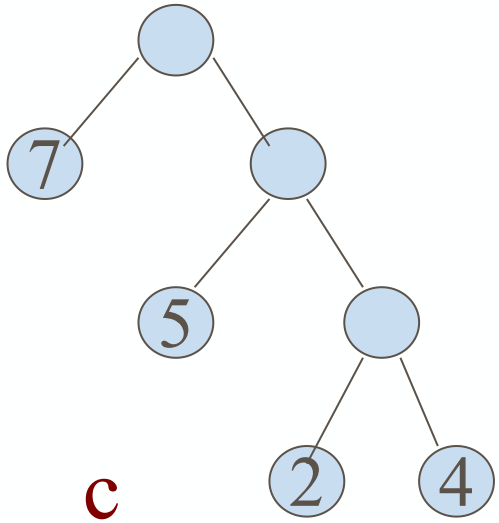
例如



a



b



c

其带权路径长度分别为：

WPL(a)=7×2＋5×2＋2×2＋4×2＝36

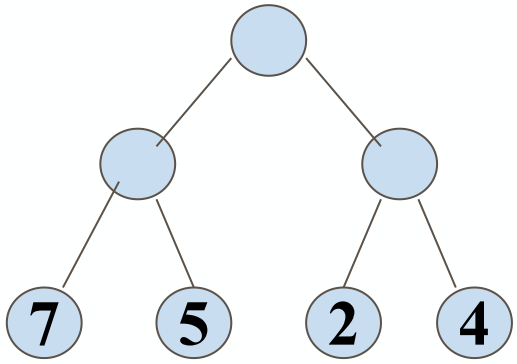
WPL(b)=4×2＋7×3＋5×3＋2×1＝46

WPL(c)=7×1＋5×2＋2×3＋4×3＝35

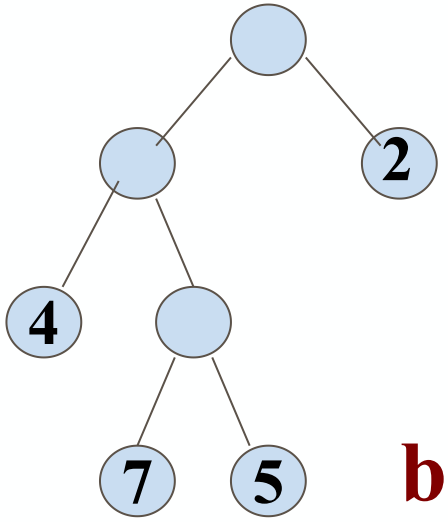
一、哈夫曼树的基本概念

什么样的树的带权路径长度WPL最小？

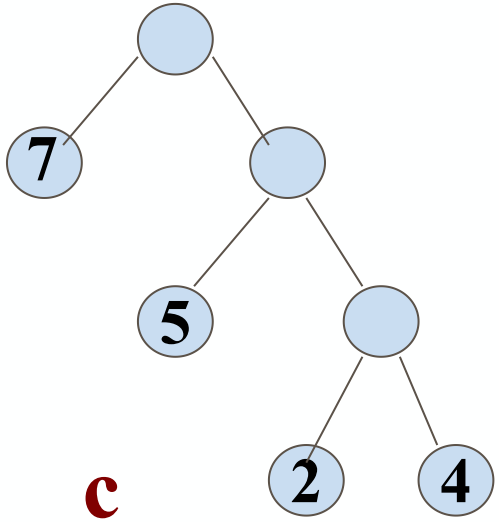
例如：给定一个权值序列{2, 4, 5, 7}，可构造多种二叉树的形态：



a



b



c

其带权路径长度分别为：

WPL(a) = 36　　WPL(b) = 46　　WPL(c)=35

一、哈夫曼树的基本概念

在各种形态的含有 n个叶子结点的 二 叉树中，必存在一棵(或几棵)其带权路径长度值WPL 最小的树，被称为“最优二叉树” 。

特征：

（1）权值越大的叶子结点越靠近根结点，而权值越小的叶子结点越远离根结点。

（2）在最优二叉树中没有度数为 1 的结点;

（3）含 n个叶子结点的最优二叉树的总结点数为 2*n-1 。

“哈夫曼树”——最优二叉树的构造方法最早由哈夫曼研究。

（1）初始化：根据给定的 n 个权值 $\{w_1, w_2, \ldots, w_n\}$，构造$n$ 棵二叉树的集合

$$F = \{T_1, T_2, \ldots, T_n\},$$

其中每棵二叉树中均只含一个带权值为 w_i的根结点，其左、右子树为空树；

二、哈夫曼树的构造　　1.构造哈夫曼树的方法

（2）选取与合并：在 F 中选取其根结点的权值为最小的两棵二叉树，分别作为左、右子树构造一棵新的二叉树，并置这棵新的二叉树根结点的权值为其左、右子树根结点的权值之和；

（3）删除与加入：从F中删去这两棵树，并加入刚生成的新树；

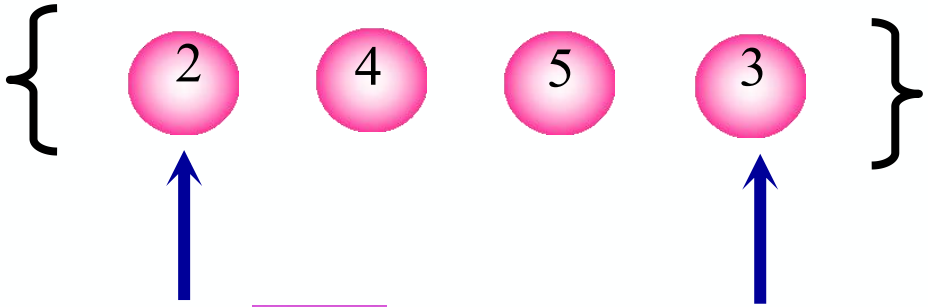
（4）重复 (2) 和 (3) ，直至 F 中只含一棵树为止。

由此得到二叉树就是“最优二叉树”即“哈夫曼树”　。

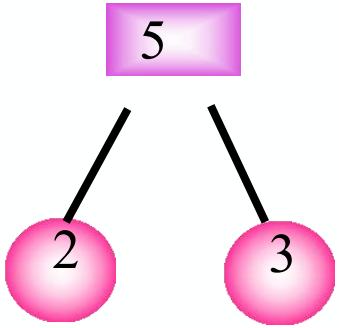
二、哈夫曼树的构造

W＝{2，4，5，3} 哈夫曼树的构造过程

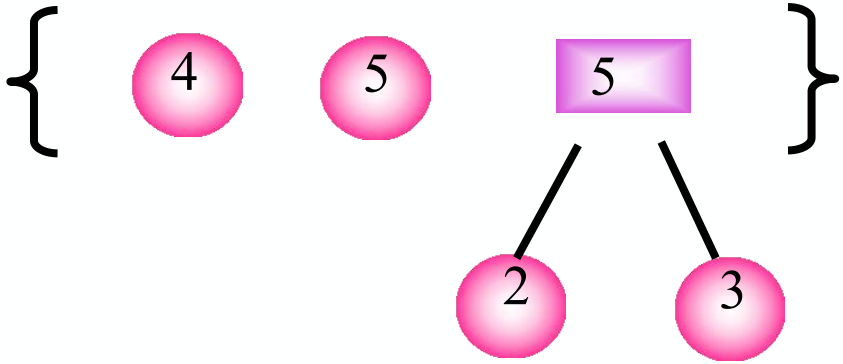
第1步：初始化



第2步：选取与合并

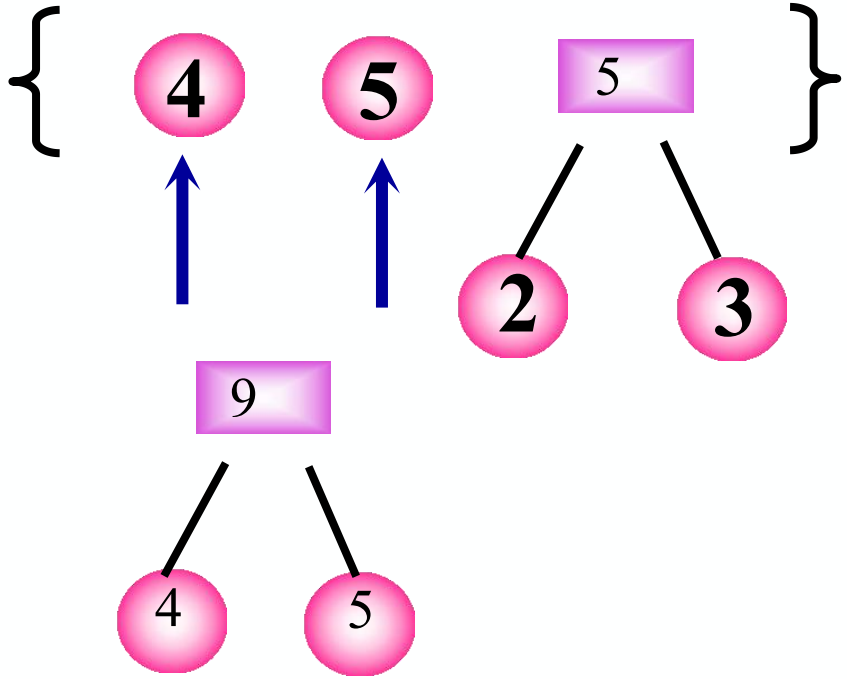


第3步：删除与加入



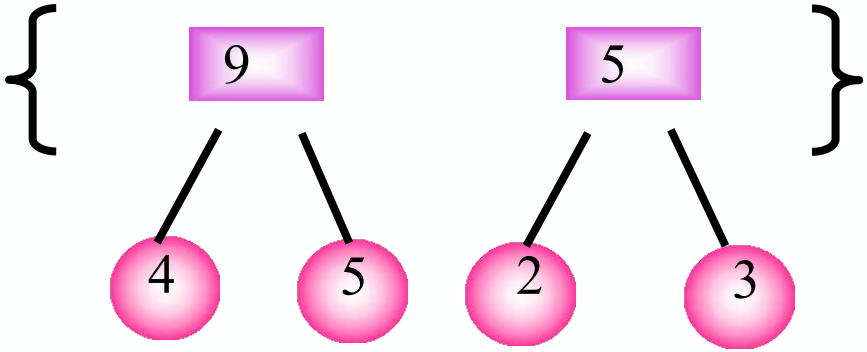
二、哈夫曼树的构造

$W=\{2，4，5，3\}$ 哈夫曼树的构造过程



重复第2步

重复第3步

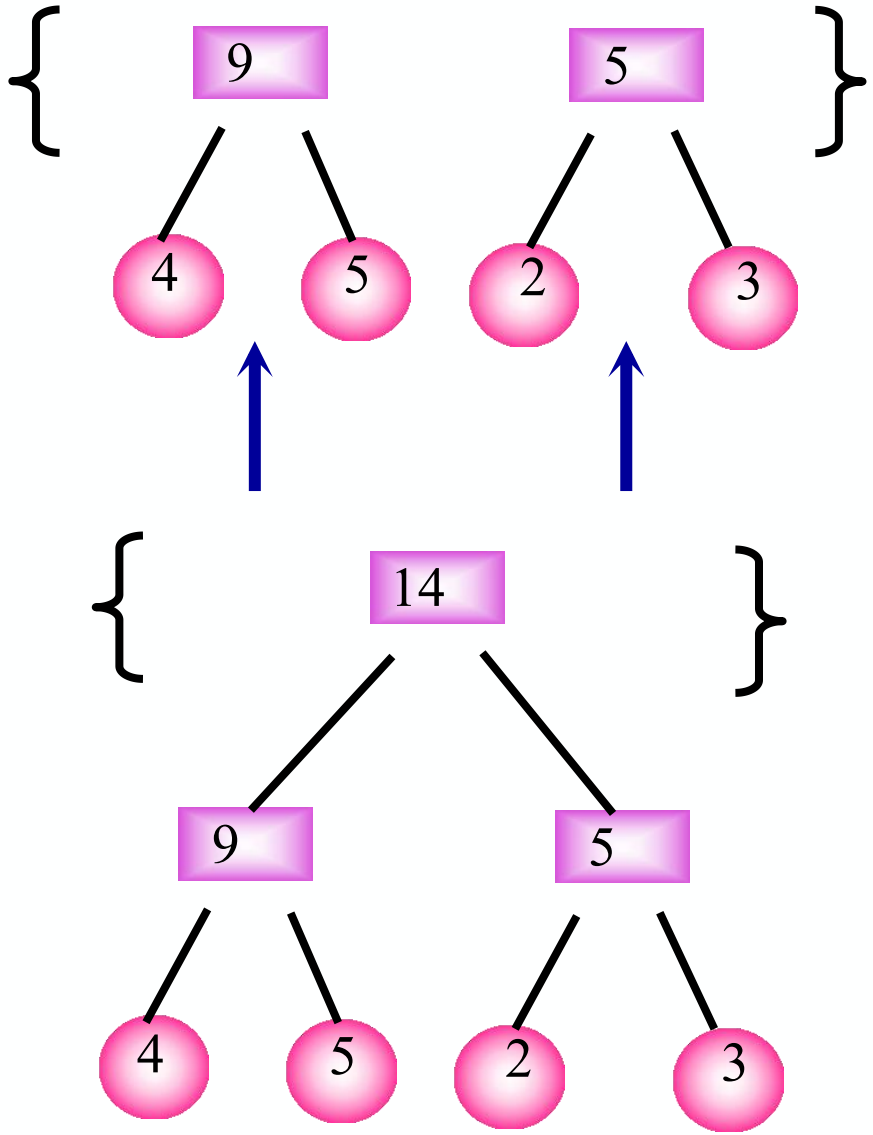


二、哈夫曼树的构造

$W=\{2，3，4，5\}$ 哈夫曼树的构造过程

重复第2步

重复第3步



二、哈夫曼树的构造

2.存储结构

n个叶子结点的哈夫曼树共有2n-1个结点,因此可用有2n-1个元素的数组来存储哈夫曼树，结点间的关系用游标表示，即采用静态链表来存储哈夫曼树。

每个结点需包含其双亲结点信息和孩子结点信息，所以构成一个静态三叉链表。

weight	parent	Lchild	Rchild
权值	双亲序号	左孩子序号	右孩子序号

二、哈夫曼树的构造

3.静态三叉链表结构定义

```
#define N   20
#define M  2*N-1
typedef struct
 { int weight ;
    int parent，Lchild，Rchild ;
}HTNode, HuffmanTree[M+1];      /*0号单元不用*/
```

静态三叉链表数组中前 n 个元素存储叶子结点，后n-1个元素存储分支结点即不断生成的新结点，最后一个元素存储哈夫曼树的根结点。

4.哈夫曼树构造算法

初始化：先将n个元素都视为根结点，即孩子和双亲指针全置0。

建哈夫曼树的过程是：反复在数组中选双亲为0(表示它们当前是树根)且权值最小的两结点，将它们作为左右孩子挂在新的结点之下，新结点权值为左右孩子权值之和。

二、哈夫曼树的构造

```
void CreatHuffmanTree(HuffmanTree &HT,int n)
{
    //构造哈夫曼树HT
   int m,s1,s2,i;
   if(n<=1) return;
   m=2*n-1;
   HT=new HTNode[m+1]; //0号单元未用，所以需要动态分配m+1个单元，HT[m]表示根结点
   for(i=1;i<=m;++i) //将1~m号单元中的双亲、左孩子，右孩子的下标都初始化为0
   {    HT[i].parent=0;  HT[i].lchild=0;  HT[i].rchild=0; }
   for(i=1;i<=n;++i)          //输入前n个单元中叶子结点的权值
      cin>>HT[i].weight;
/*－－－－－初始化工作结束，下面开始创建哈夫曼树－－－－－*/
```

二、哈夫曼树的构造

```
    for(i=n+1;i<=m;++i)
    {    //通过n-1次的选择、删除、合并来创建哈夫曼树
         Select(HT,i-1,s1,s2);
         //在HT[k](1≤k≤i-1)中选择两个其双亲域为0且权值最小的结点,
         // 并返回它们在HT中的序号s1和s2
         HT[s1].parent=i;
         HT[s2].parent=i;
         //得到新结点i，从森林中删除s1，s2，将s1和s2的双亲域由0改为i
         HT[i].lchild=s1;
         HT[i].rchild=s2 ;//s1,s2分别作为i的左右孩子
         HT[i].weight=HT[s1].weight+HT[s2].weight;
                                        //i 的权值为左右孩子权值之和
    }//for
}
```

三、哈夫曼树应用——哈夫曼编码

哈夫曼树最典型的应用是在编码，利用哈夫曼树，可以得到平均长度最短的编码。

平均长度最短的编码一般为不等长编码，为使各编码间无需加分界符即可识别，其编码应是前缀码。

前缀码：如果在一个编码方案中，任一个编码都不是其他任何编码的前缀（最左子串）。

利用哈夫曼树可以构造不等长的二进制编码，并且构造所得的编码是一种最优前缀编码，即可以使所传信息的总长度最短。

三、哈夫曼树应用——哈夫曼编码

对哈夫曼树中每个左分支赋予0，右分支赋予1，则从根到每个叶子的路径上，各分支的值构成该叶子的哈夫曼编码。

例：若要传输如下单词

state, seat, act, tea, cat, set, a, eat

如何使所传送的信息编码长度最短？

为保证信息编码长度最短，先统计各字符出现的次数，然后以此作为权值，构造哈夫曼树。

三、哈夫曼树应用——哈夫曼编码

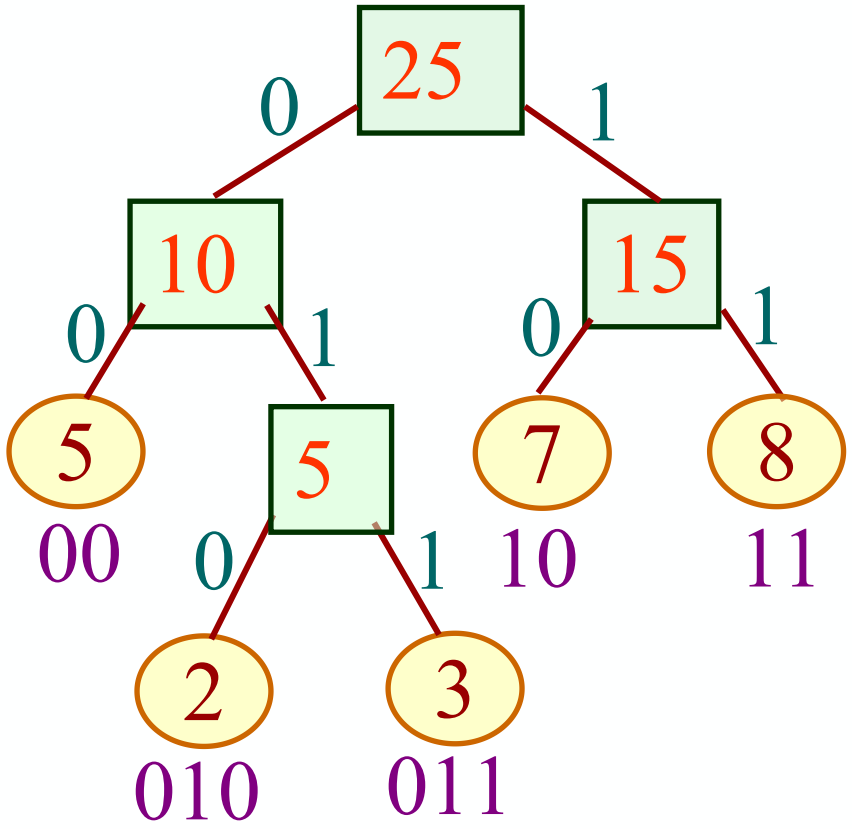
需传输信息： state, seat, act, tea, cat, set, a, eat

各字符权值：

s	t	a	e	c
3	8	7	5	2

编码:左分支:0\右分支:1；
根到叶子路径上的
值构成叶子的编码。

构造哈夫曼树：



各字符编码：

s	t	a	e	c
011	11	10	00	010

三、哈夫曼树应用——哈夫曼编码

哈夫曼树最典型、最广泛的应用是在编码技术上，而操作码的优化也是其应用之一。

例：设有一台模型机，共有7种不同的指令，各指令的使用频率P_i为：

指　　令	I_1	I_2	I_3	I_4	I_5	I_6	I_7
使用频率pi	0.40	0.30	0.15	0.05	0.04	0.03	0.03

以指令的使用频率为权值构造哈夫曼树，并求各指令的哈夫曼编码。

三、哈夫曼树应用——哈夫曼编码

各指令的哈夫曼编码为：

指令	I_1	I_2	I_3	I_4	I_5	I_6	I_7
编码	1	01	001	00011	00010	00001	00000

则指令的平均码长为：

$$\sum_{i=1}^{n} p_i \times l_i = 0.4*7+0.3*2+0.15*3+0.05*5+0.04*5$$
$$+0.03*5+0.03*5 = 2.20$$

若用等长编码，其平均码长为3。

三、哈夫曼树应用——哈夫曼编码

编码：

有了字符集的哈夫曼编码表之后，对数据文件的编码过程是：依次读入文件中的字符，在哈夫曼编码表中找到此字符，并将该字符转换为编码表中存放的编码串。

译码：

对编码后的文件进行译码的过程必须借助于哈夫曼树。具体过程是：依次读入文件的二进制码，从哈夫曼树的根结点出发，若当前读入**0**，则走向左孩子，否则走向右孩子。一旦到达某一叶子时便译出相应的字符编码。然后重新从根出发继续译码，直至文件结束。

小结

1. 哈夫曼树的基本概念
2. 哈夫曼树的构造过程和算法
3. 哈夫曼编码的方法