1.二叉树的典型应用：哈夫曼树和哈夫曼编码译码器

哈夫曼树是二叉树的一种经典应用

哈夫曼树和哈夫曼编码经常搭配使用，用来创建一篇文章对应的加密编码，并且能够对这篇文章进行加密和解密

通过哈夫曼编码加密后的文章完全通过01构成，并且每一篇文章因为内容的不同，即使是相同的字符所对应的的哈夫曼编码也是不同的

所以，既是单纯的得到一篇文章的密文结构，没有得到对应的哈夫曼编码表，也是无法进行解密的

所以哈夫曼树和哈夫曼编码在密码学当中具有非常高的学术研究价值

①前缀编码和非前缀编码

在学习哈夫曼树和哈夫曼编码之前，我们先来学习一下什么是前缀编码和非前缀编码

前缀编码：如果一个字符的加密编码有可能出现在其他字符的加密编码的前面，那么这一整套编码结构就称之为前缀编码

举个例子：加入字符a的编码是10，字符b的编码是1000，那么我们称字符a的编码是字符b的编码的前缀编码

前缀编码在解密过程中很可能出现歧义，比如以上面ab的编码为例，字符串ba的编码结果为：100010

但是在解密过程中，我们既可以把这个密文解密为ba，也可以解密为a00a，那么这样就产生了歧义

非前缀编码：与前缀编码相反的是，非前缀编码指的就是在编码集中，任意两个字符的编码之间都不可能产生前缀编码的情况

也就是说，如果使用非前缀编码的话，在解密过程中是不可能出现歧义的

②构建哈夫曼树

对一篇文章构建哈夫曼编码的过程如如下：

步骤1：首先将这个文章当成一个字符串，遍历这个字符串中所有的字符，并且统计每一个字符出现的次数

步骤2：根据字符出现的次数对字符进行排序

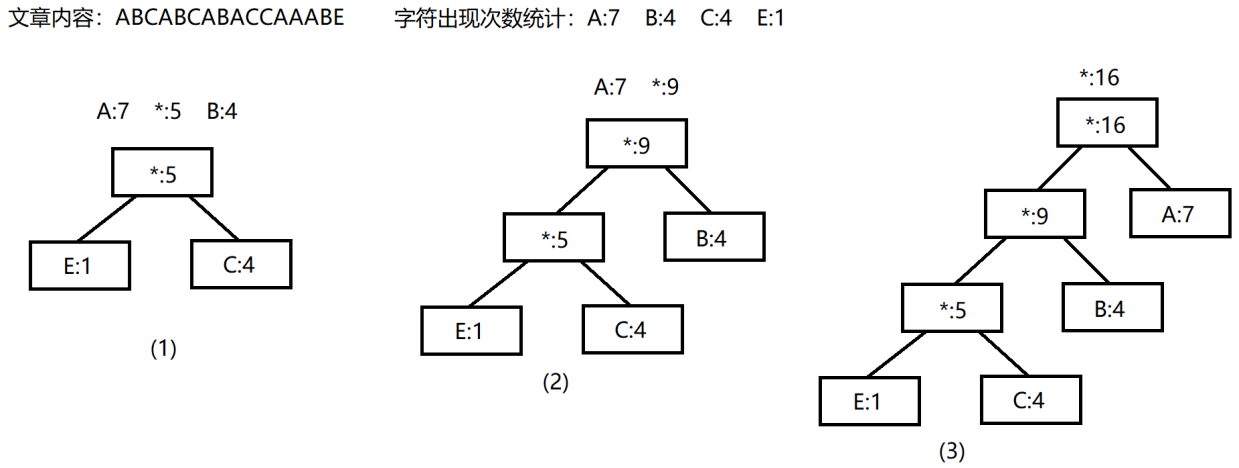
步骤3：选取排序后出现次数最少的两个字符加入哈夫曼树，并将两个字符的出现次数取值相加，合并成为一个中间节点

中间节点仅记录两个节点取值的加和，但是不记录任何字符

步骤4：将中间节点加入字符排序序列中，删除已经使用过的节点，对序列重新排序，重复步骤2-4，构建过程中产生的中间节点也算作在内

步骤5：当所有的字符和中间节点全部合成完毕时，序列中只剩余一个节点，就是哈夫曼树的根节点，哈夫曼树创建完毕

哈夫曼树的构建过程如下图所示：



从哈夫曼树的构建过程我们不难看出如下规律：

1.哈夫曼树的构建过程是自底向上的

2.在一个哈夫曼树中所有具有实际意义的字符，最终一定会存在于叶子结点中

上述这两个步骤也为我们后面生成非前缀的哈夫曼编码打下了基础

③创建字符哈夫曼编码：一种非前缀编码

接下来我们在上面创建的哈夫曼树的基础上，来为每一个存储字符的节点分配哈夫曼编码

每一个字符节点创建哈夫曼编码的过程如下：

步骤1：从根节点开始，向下寻找每一个叶子节点

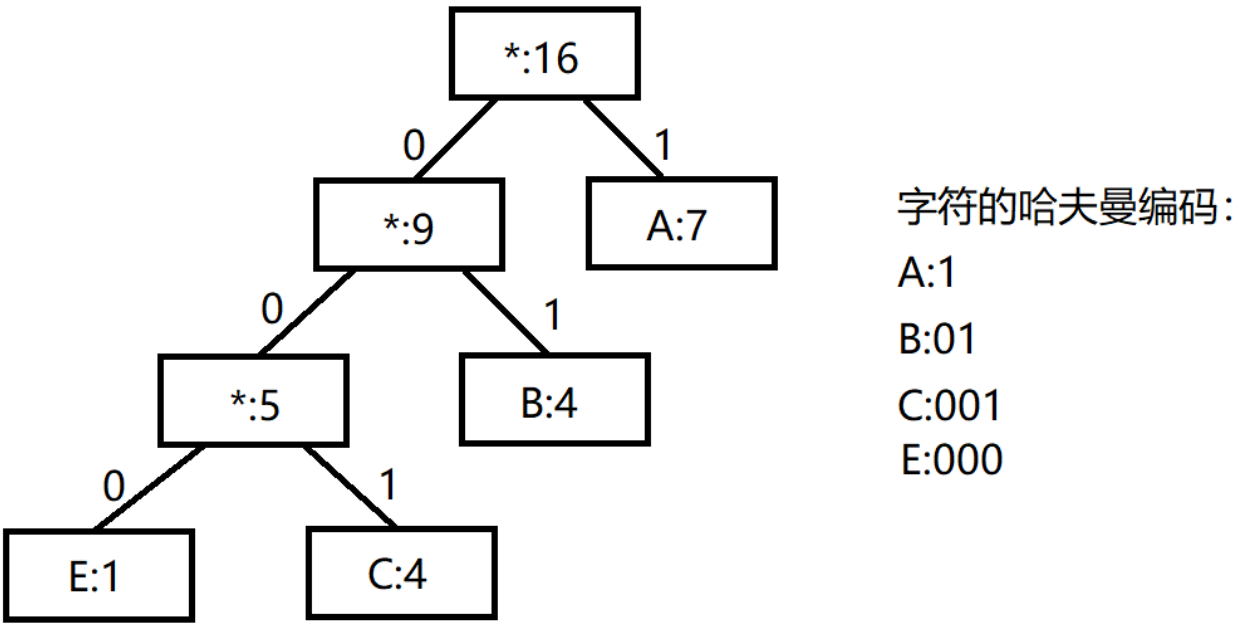
步骤2：如果向左孩子方向走一步，则记0

步骤3：如果向右孩子方向走一步，则记1

步骤4：重复上面步骤2-3，直到遍历完成整个哈夫曼树为止，最终得到根节点通往每一个叶子节点的路径字符串

就是这个叶子节点对应字符的哈夫曼编码

编码结果如图所示：



从上图不难看出：文章中出现的任何字符的编码，都不会构成其他字符编码的前缀编码

④文件内容加密

加密的过程，就是将文章中所有的字符使用其对应的哈夫曼编码进行替换的过程

如上文图中提供的字符串，最终的编码结果为：

10100110100110110010011110100O

⑤加密内容解密

密文解密的时候，必须同时得到密文和所有字符对应的哈夫曼编码表两部分内容

因为每一个文章中，即使是相同的字符，出现的次数也有可能是不相同的

所以只要是通过不同文章得到的哈夫曼编码表之间也是不通用的

解密过程步骤如下：

步骤1：顺序取得密文的一个字符，在编码表中比对，有没有这个密文对应的密码字符串

步骤2：如果没有，则在保留当前密文字符的基础上，获取下一位密文并拼接到这一位密文的后面

步骤3：重复步骤1-2，直到能够在编码表中找到对应的密文字符串，使用编码表中的密文字符串对应的明文替换这一段密文

步骤4：重复步骤1-4，直到密文中所有的内容被翻译成为明文为止

上述步骤如下图所示：

密文字符串：10100110100110110010011110100 0

编码表：A:1　B:01　C:001　E:000

解码过程：

1 -> A　　A0100110100110110010011110100 0

0 -> 无对应

01 -> B　　AB0011010011011001001111010 00

0 -> 无对应

00 -> 无对应

001 -> C　ABC1010011011001001111010 00

1 -> A　　ABCA0100110110010011110100 0

0 -> 无对应

01 -> B　　ABCAB0011011001001111010 00

… …