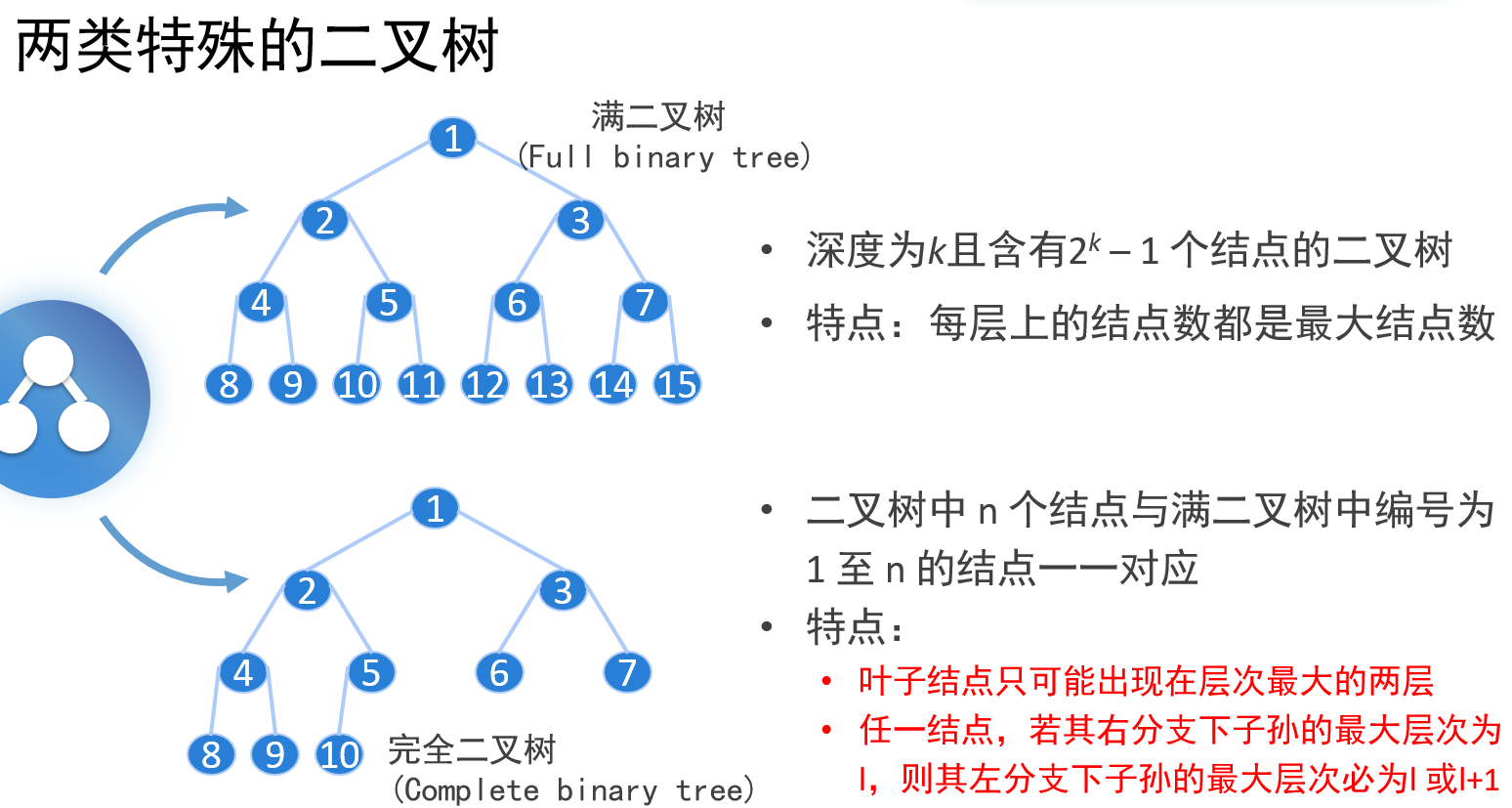
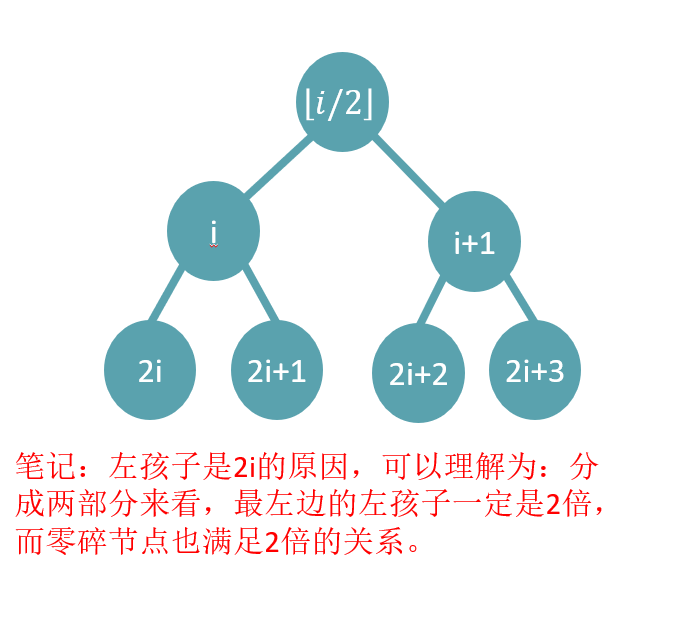
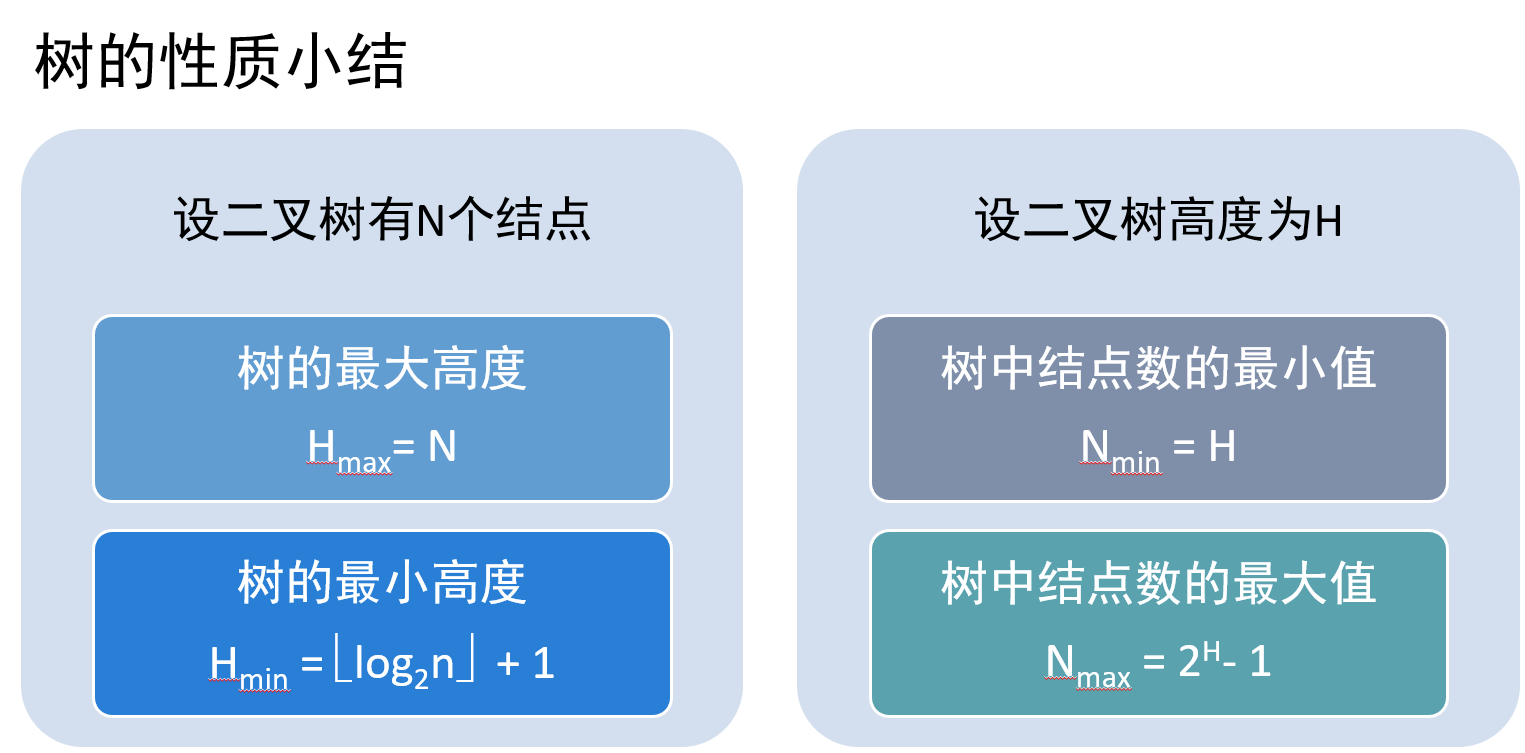
1. 数据结构中树结点的度是指出度，一棵树的度是指最大结点度数。
2. 结点的层次:根为第一层、孩子为第二层……
3. 树的深度是指树中结点的最大层次数
4. 树分为有序树和无序树。有序是指孩子结点不能随意交换顺序。二叉树便是有序树。
5. 二叉树的重要性质
6. 二叉树的第 i 层上至多有个结点。(i≥1)
7. 深度为 k 的二叉树上至多含个结点（k≥1）(对每一层求和)
8. 对任何一棵二叉树，若它含有个叶子结点、个度为 2 的结点，则必存在关系式：(由图的握手定理可推出，注意到根节点虽然出度为2，但图论上的度数为2)
9. 满二叉树与完全二叉树



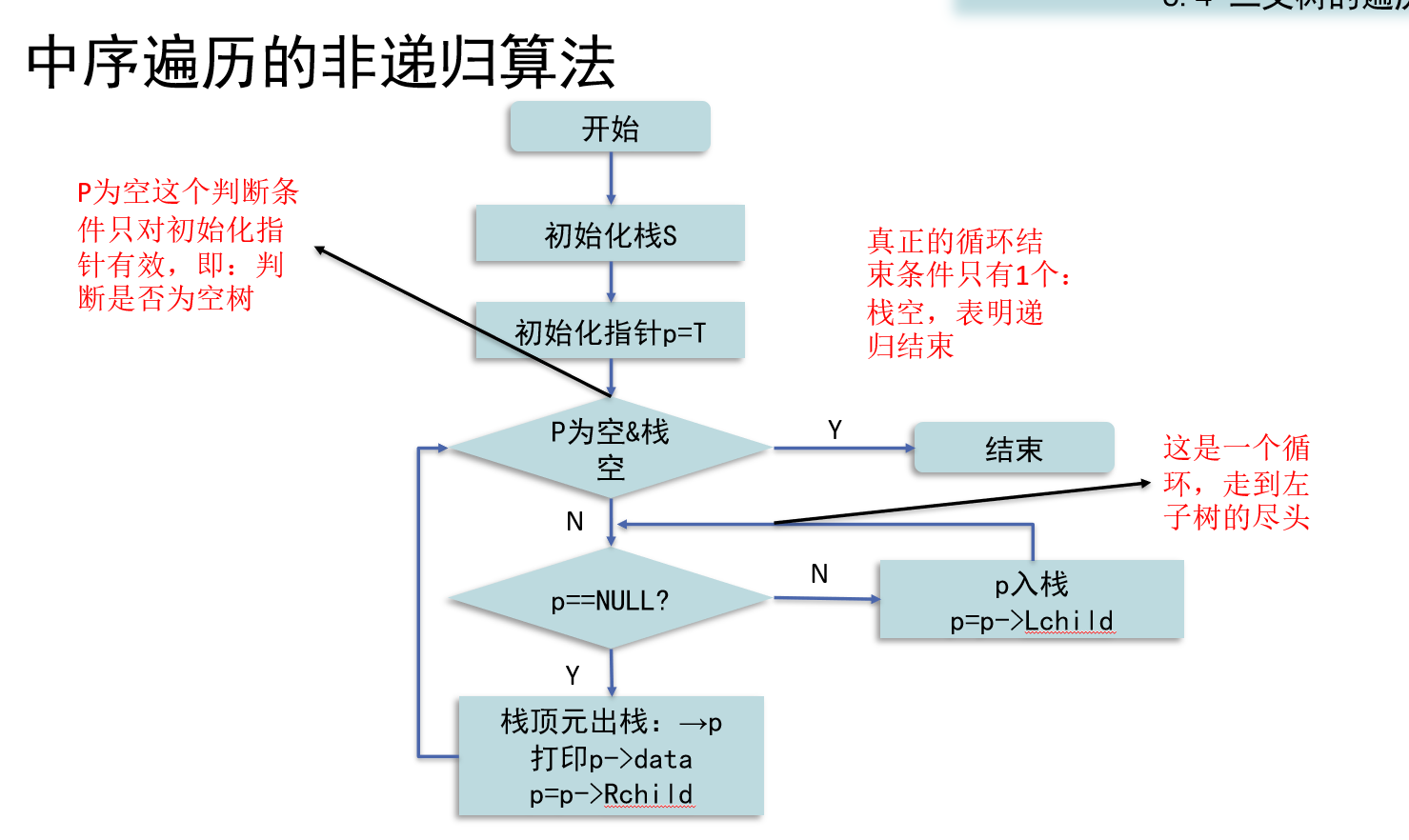
1. 完全二叉树的性质
2. 具有 n 个结点的完全二叉树的深度为(向下取整)
3. 从上到下、从左向右编号后，左孩子是2i，右孩子是2i+1

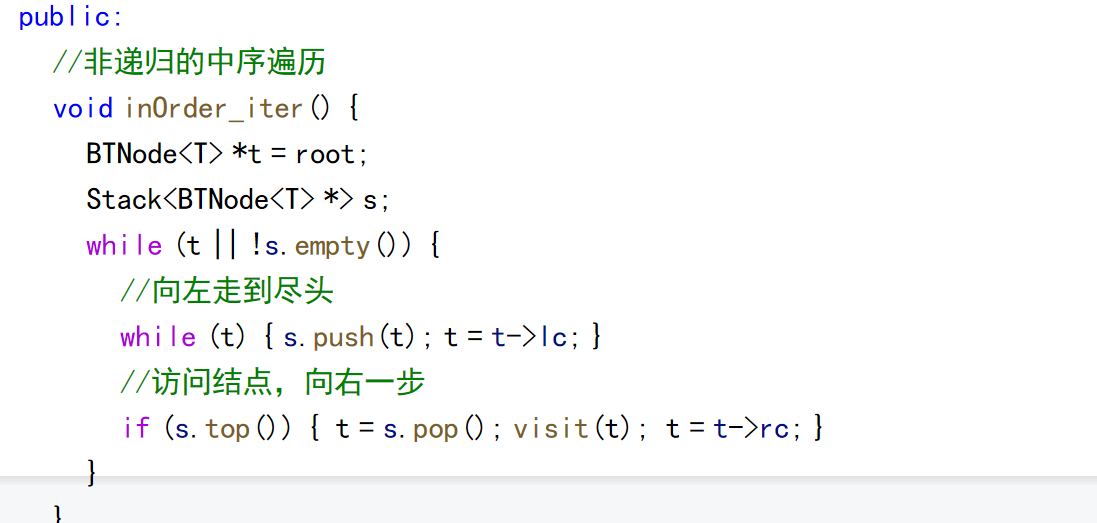


1. 树的性质小结

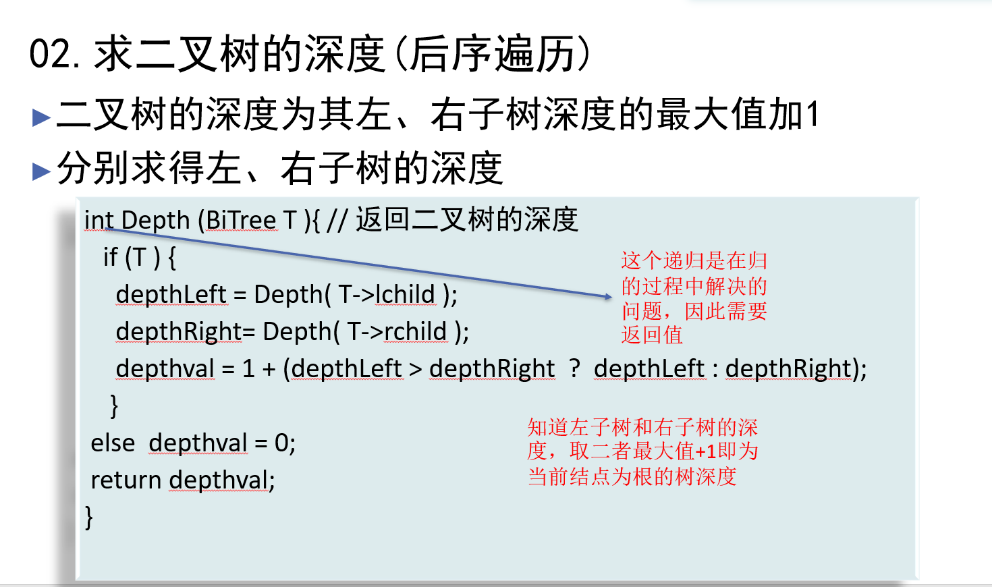


1. 中序遍历的非递归实现

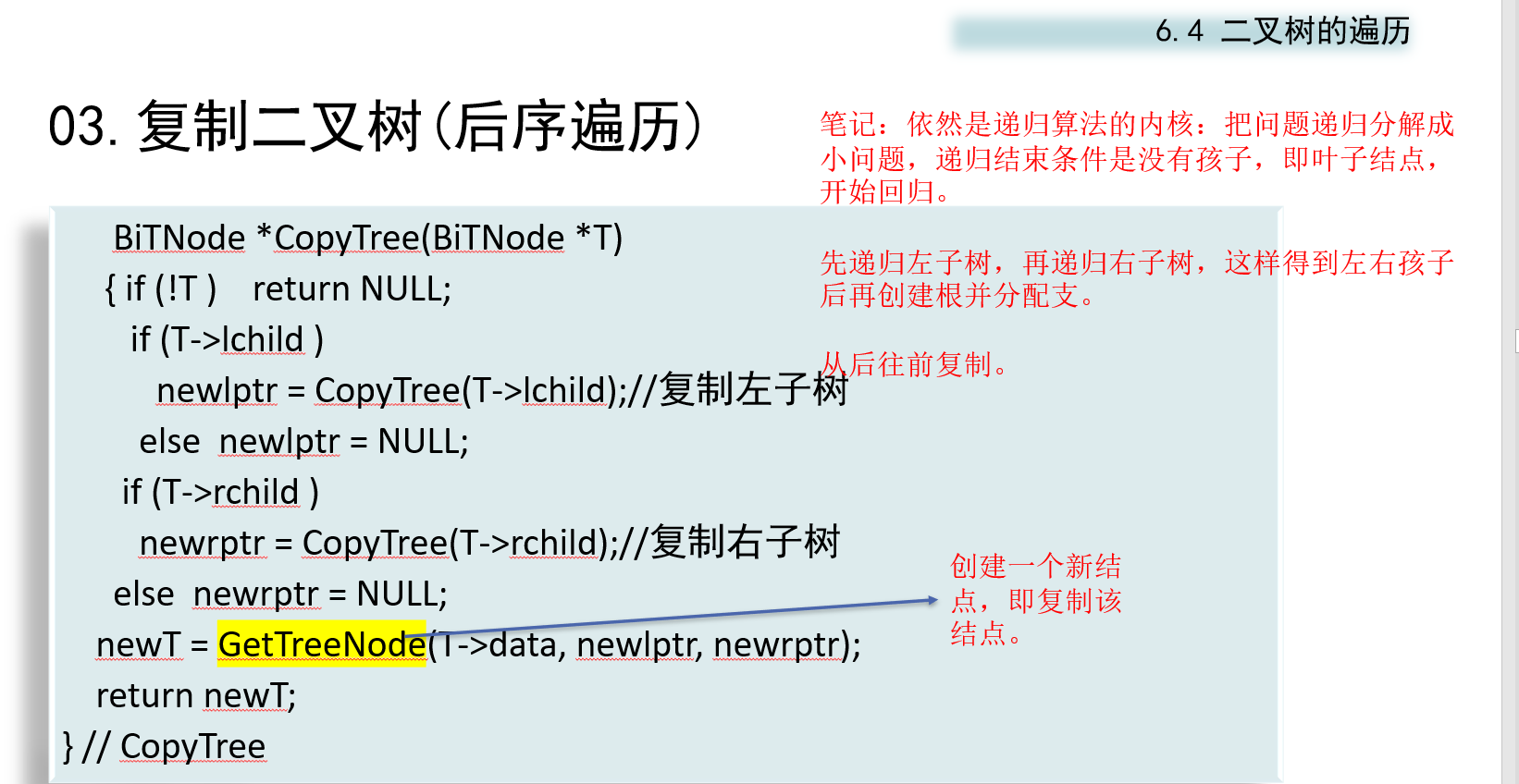


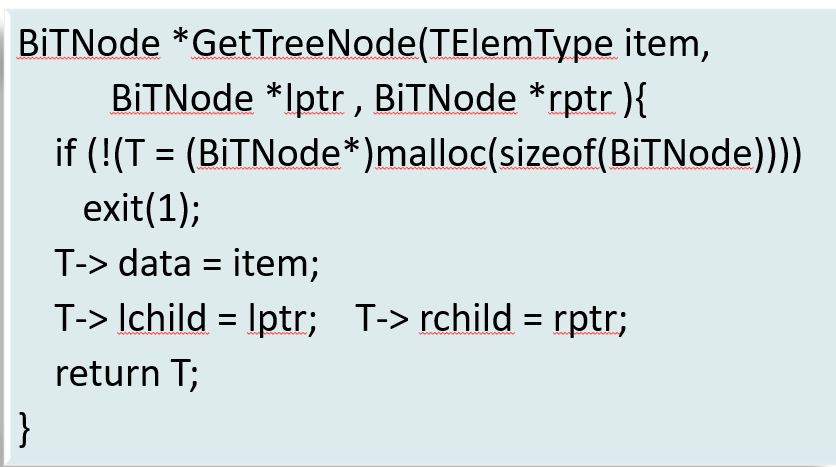


1. 求一颗树的深度

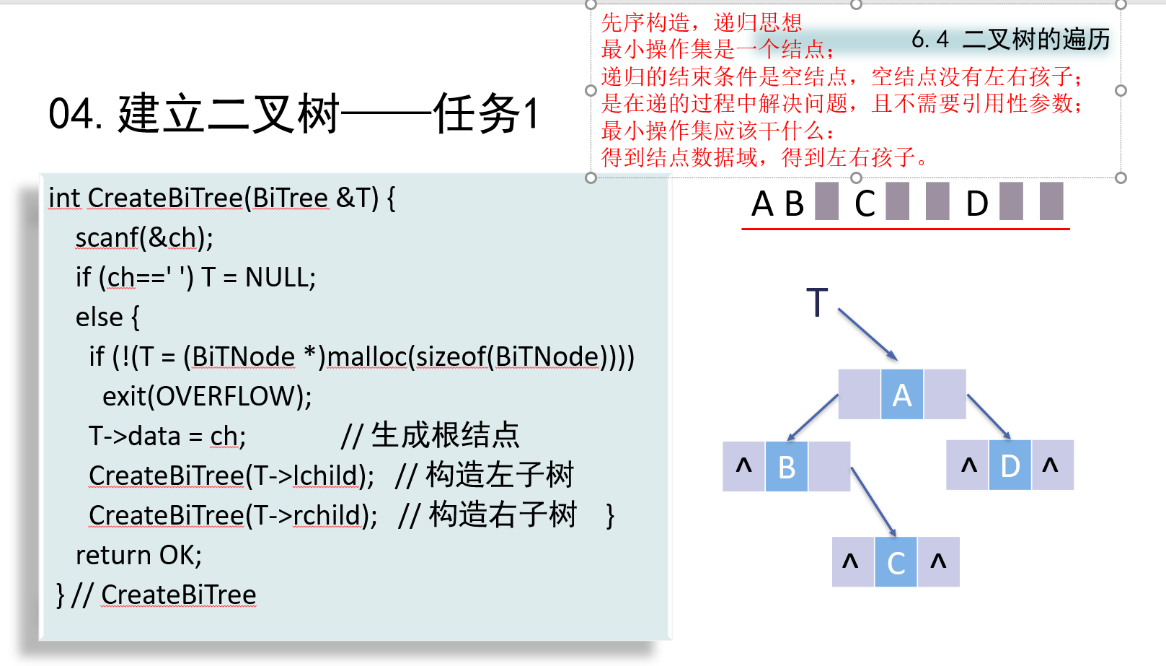


1. 用后序遍历复制一棵树





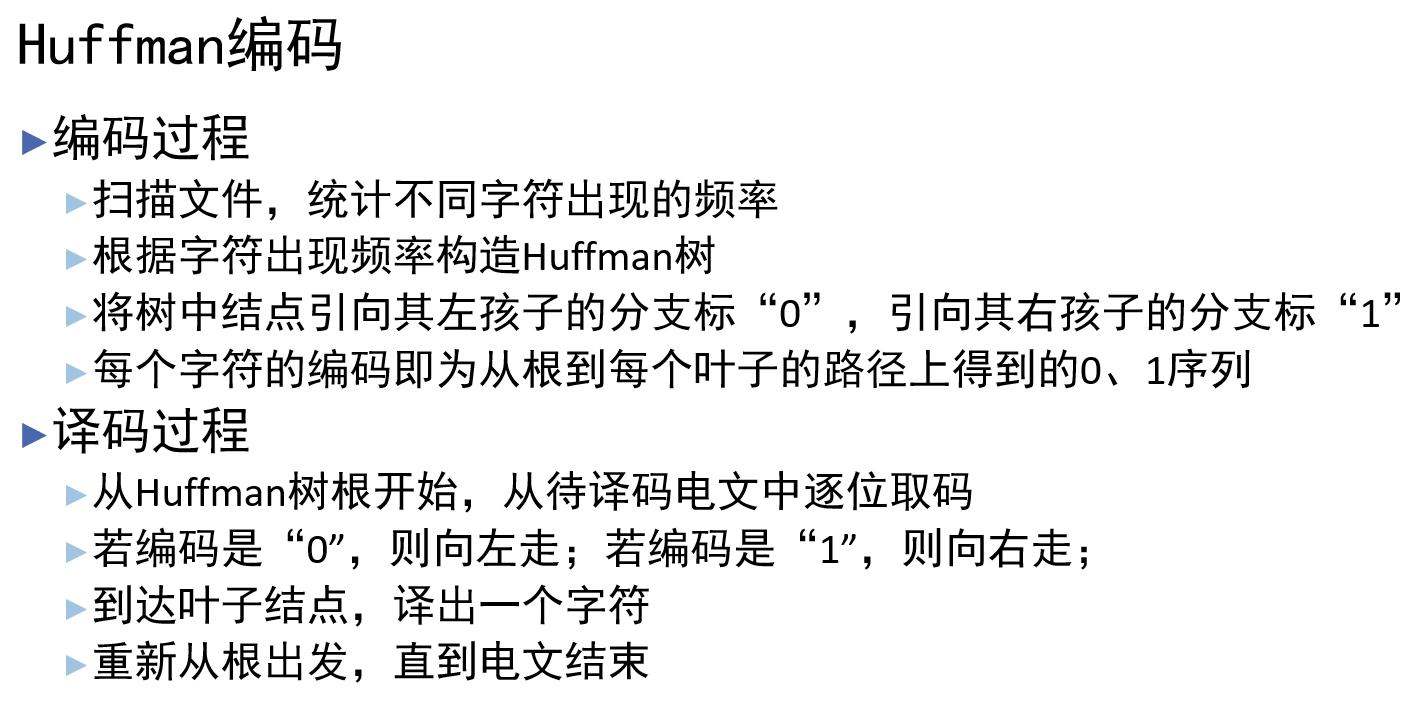
1. 已知先序，建树



1. 根据前序和中序建树
2. 线索二叉树
3. 哈夫曼树
4. 树的路径长度(Weighted Path Length): 从树根到每一个结点的路径长度之和



1. 由已知频率的字符集如何构建一颗哈夫曼树
2. 哈夫曼编码：二进制码、使用频率高的编码更短、每个字符的编码均不同。
3. 一颗哈夫曼树对应一个哈夫曼编码：

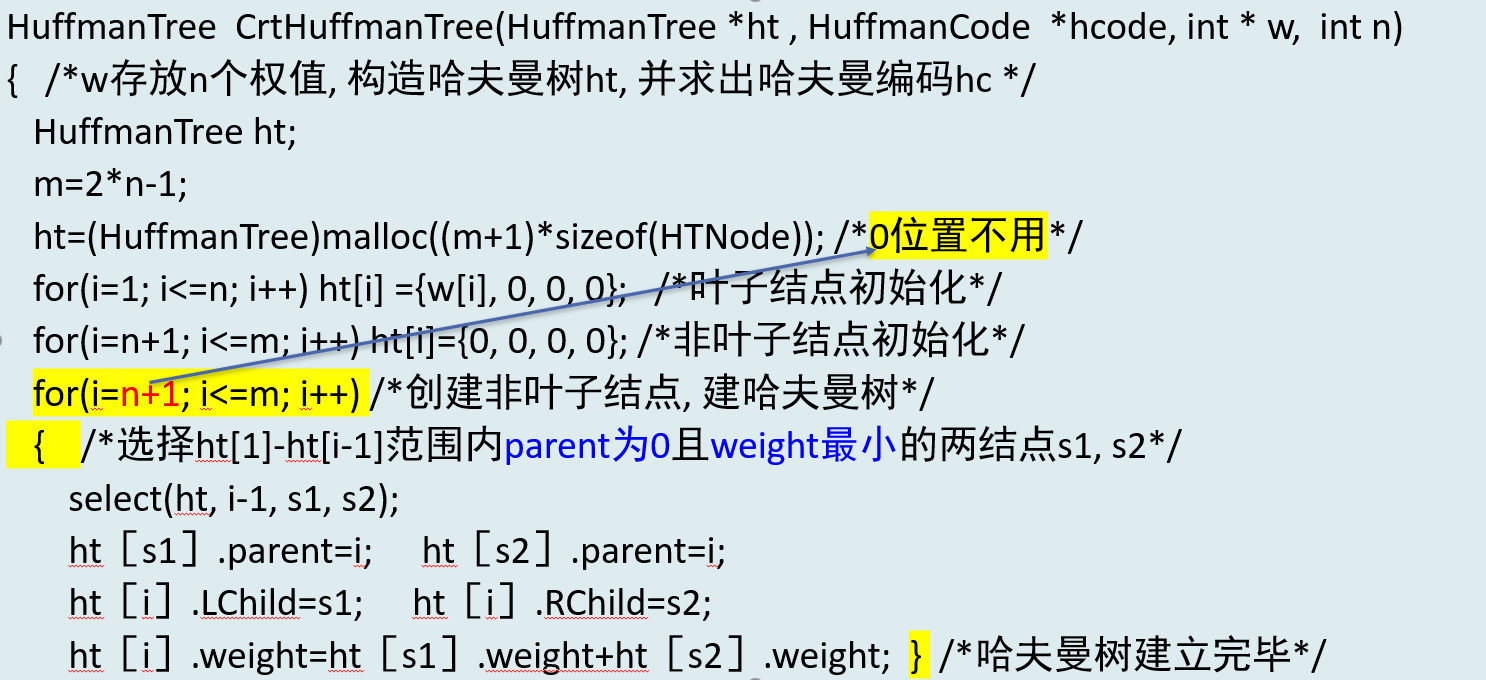


编码与译码是互逆的:

由字符频率→哈夫曼树:从下到上，小小结合。

由哈夫曼树→哈夫曼编码:从上到下，向左为0，向右为1；

1. 关于哈夫曼编码的思考
2. 哈夫曼树没有度为1的结点
3. 字符只能是哈夫曼树中的叶子结点
4. n个叶子结点(n个字符)构成的哈夫曼树共有2n-1个结点。
5. 哈夫曼树的存储结构
6. 静态三叉链表(是一个元素为三叉链表的数组)，数组大小为2n-1，n为字符数量。
7. 代码实现逻辑
8. 创建一个数量为2n-1的静态三叉链表
9. 给n个叶子结点初始化(初始权值、双亲和左右孩子均为0)
10. 给n-1个非叶子结点初始化(权值、双亲和左右孩子均为0)
11. 对非叶子结点进行一次循环赋值并构建哈夫曼树



1. 由哈夫曼树的静态三叉链表求哈夫曼编码

对叶子结点进行一次循环，一个一个叶子结点求，且从后往前求编码：

向上回溯父结点，查看自己是父结点的左孩子还是右孩子，左孩子则是0，右孩子则是1。直至到根节点为止