课时六　树和二叉树（1）

考点	重要程度	占分	题型
1．树的基本概念	★★★★	0~2	选择、填空
2．二叉树	★★★★★	2~4	
3．二叉树的遍历	必考	4~8	解答

视频讲解更清晰
仅4小时

6.1 树的基本概念

树是n(n≥0) 个结点的有限集。当 n=0 时，称为空树。在任意一颗非空树上应满足：

（1）有且仅有一个特定的根的结点。

（2）当 n>1 时，其余结点可分为 m(m>0) 个互不相交的有限集 $T_1,T_2,...T_m$，其中每个集合 本身又是一棵树，并且称为根的子树。

树作为一种逻辑结构，同时也是一种分层结构，具有以下两个特点：

（1）树的根结点没有前驱，除根结点外的所有结点有且只有一个前驱。

（2）树中所有结点可以有零个或多个后继。

6.1 树的基本概念

题 1.题1.树最适合用来表示（ ）。

A.有序数据元素

B. 无序数据元素

C. 元素之间具有分支层次关系的数据

D. 元素之间无联系的数据

答案：C

基本术语：

（1） 根 A 到结点 K 的唯一路径上的任意结点，称为结点 K 的祖先。 路径上最接近结点 K 的结点 E 称为 K 的双亲，而 K 为结点 E 的孩子。

（2） 树中一个结点的孩子个数称为该结点的度，树中结点的最大度数称为树的度。

（3） 度大于 0 的结点称为分支结点（又称非终端结点）；度为 0（没有孩子结点）的结点 称为叶子结点（又称终端结点）。

有相同双亲的结点称为兄弟。

（4） 结点的层次从树根开始定义，根结点为第 1 层，它的子结点为第 2 层。结点的深度是从根结点开始自顶向下逐层累加。 结点的高度是从叶结点开始自底向上逐层累加。 树的高度（或深度）是树中结点的最大层数。

（5） 有序树和无序树。树中结点的各子树从左到右是有次序的，不能互换，称该树为有 序树，否则称为无序树。假设图为有序树，若将子结点位置互换，则变成一颗不同 的树。

（6） 路径和路径长度。树中两个结点之间的路径是由这两个结点之间所经过的结点序列 构成的，而路径长度是路径上所经过的边的个数。

6.1 树的基本概念

树的性质：

（1） 树中的结点数等于所有结点的度数加 。

（2） 度为 的树中第 层上至多有m^{i-1} 个结点$(i\geq 1)$ 。

（3） 高度为 的 叉树至多有 （m^h-1）/(m-1) 个结点。

（4） 具有 个结点的 叉树的最小高度为$[\log_m(n(m-1)+1)]$ 。

题 2.二叉树的深度为k，则二叉树最多有（ ）个结点。

A.2k　　B. 2k-1　　C. 2k-1　　D. 2k-1　　答案：C

题 3.假定在一棵二叉树中，度为2的结点数为15，度为1的结点数为30，则叶子结点数为（ ）个。

A.15　　B. 16　　C. 17　　D. 47　　答案：B

6.2 树的基本概念

二叉树的定义：

二叉树的特点是每个结点至多只能有两棵子树（即二叉树中不存在度大于 2 的结点），并 且二叉树的子树有左右之分，其次序不能任意颠倒。二叉树也已递归的形式定义。 二叉树是 n（n≥0）个结点的有限集合：

（1） 或者为空二叉树；

（2） 或者由一个根结点和两个互不相交的被称为根的左子树和右子树组成，左右子树又 分别是一棵二叉树。

6.2 树的基本概念

几个特殊的二叉树：

（1）满二叉树。一颗高度为h ,且含有 2^h-1个结点的二叉树称为满二叉树，即树中的每 层都含有最多的结点，如下图（a）所示。满二叉树的叶子结点都集中在二叉树的最 下一层，并且除叶子结点之外的每个结点度数均为 2 。

对满二叉树按层序编号：约定编号从根结点（根结点编号为 1）起，自上而下，自左 向右。这样，每个结点对应一个编号，对于编号为 的结点，若有双亲，则其双亲为[i/2] ，若有左孩子，则左孩子为 2i；若有右孩子，则右孩子为 2i+1。

（2）完全二叉树。高度为h、有 个结点的二叉树，当且仅当其每个结点都与高度为h 的满叉树中编号为1~n 的结点一一对应时，称为完全二叉树，如图（b）所示。其特 点如下：

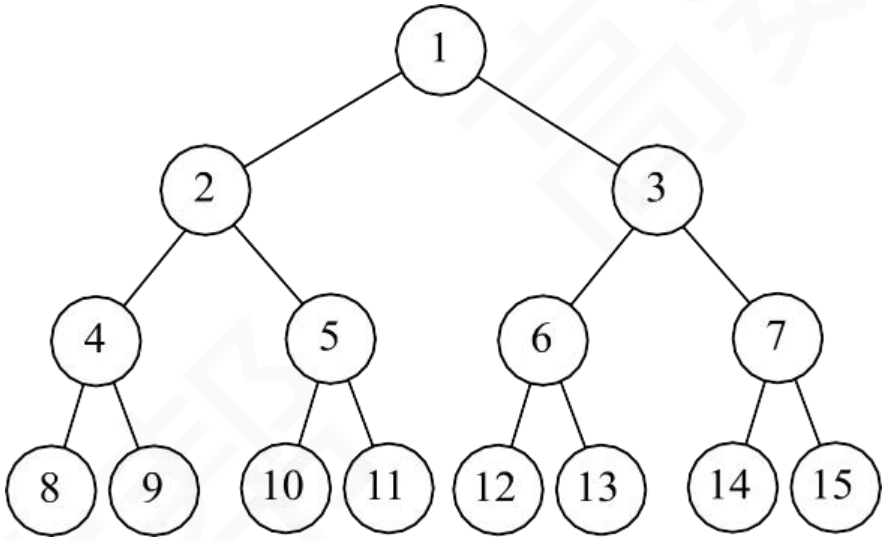
①若 i≤[n/2]，则结点 i 为分支结点，否则为叶子结点。

②叶子结点只可能在层次最大的两层上出现。对于最大层次中的叶子结点，都依次排 列在该层最左边的位置上。

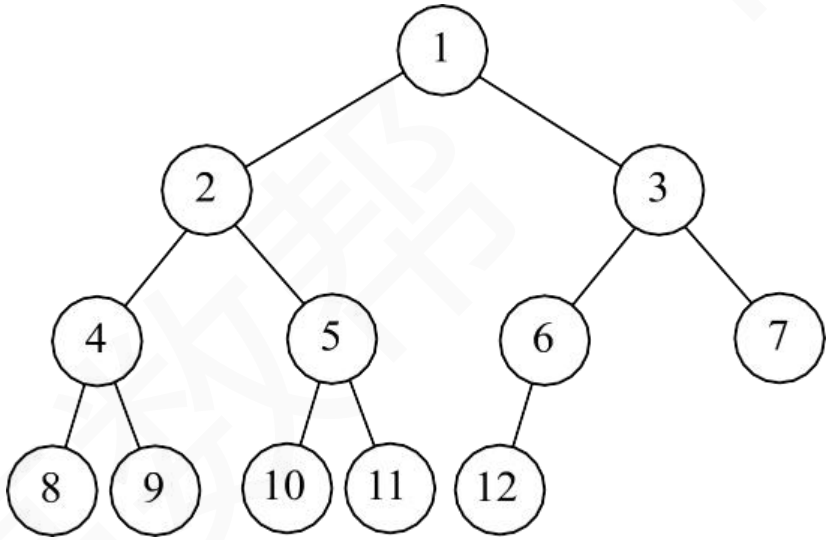
③若有度为 1 的结点，则只可能有一个，且该结点只有左孩子而无右孩子。

④按层序编号后，一旦出现某结点（编号为 i ）为叶子结点或只有左孩子，则编号大 于 的结点均为叶子结点。

⑤若 n 为奇数，则每个分支结点都有左孩子和右孩子；若n为偶数，则编号最大的分 支结点（编号为 n/2）只有左孩子，没有右孩子，其余分支结点左、右孩子都有。



(a)满二叉树



(b)完全二叉树

（3）二叉排序树。左子树上所有结点的关键字均小于根结点的关键字；右子树上的所有 结点的关键字均大于根结点的关键字；左子树和右子树又各是一颗二叉排序树。

（4）平衡二叉树。树上任一结点的左子树和右子树的深度之差绝对值不超过 1。

二叉树的性质：

（1） 非空二叉树上的叶子结点数等于度为 2 的结点数加 1 ， 即$n_0=n_2+1$ 。

（2） 非空二叉树上第 k 层上至多有2^{k-1} 个结点$(k\geq 1)$ 。

（3） 高度为 h 的二叉树至多有2^h-1 个结点$(h\geq 1)$ 。

（4） 对完全二叉树按从上到下、从左到右的顺序依次编号 1,2,…，n，则有以下关系：

①当 i>1 时，结点 的双亲的编号为 [i/2]，即当 i为偶数时，其双亲的编号为 i/2，它是 双亲的左孩子；当 为奇数时，其双亲的编号为 (i-1)/2，它是双亲的右孩子。

②当 2i≤n 时，结点 的左孩子编号为 2i，否则无左孩子。

③当 2i+1≤n 时，结点 的右孩子编号为 2i+1 ，否则无右孩子。

（5） 具有n(n>0) 个结点的完全二叉树的高度为$[\log_2(n+1)]$ 或$[\log_2 n]+1$ 。

6.2 树的基本概念

题 1.一棵二叉树有 67 个结点，这些结点的度要么是 0，要么是 2 。这棵二叉树中度为 2 的结 点有_____个。

答案：33 (67-1)/2=33

题 2.将一棵有 100 个结点的完全二叉树，从根这一层开始，每一层从左到右依次对结点编 号，根结点的编号为 1 ，则编号为 49 的结点的双亲编号为（ ）。

A.23　　B.25　　C.24　　D.无法确定

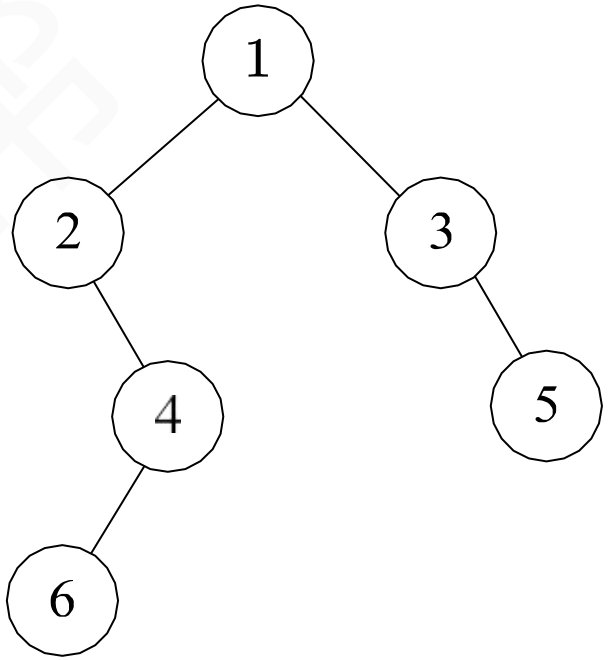
答案：C [49/2]=24

6.2 树的基本概念

二叉树的存储结构：

（1）顺序存储结构：指用一组地址连续的存储单元依次自上而下、自左至右存储完全二 叉树的结点元素，即将完全二叉树上编号为 i 的结点元素存储在一维数组下标 i-1 的分量 中。

1	2	3	0	4	0	5	0	0	6	0

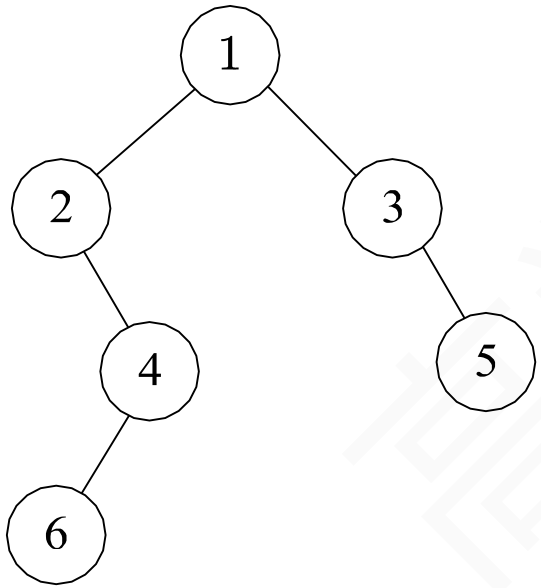


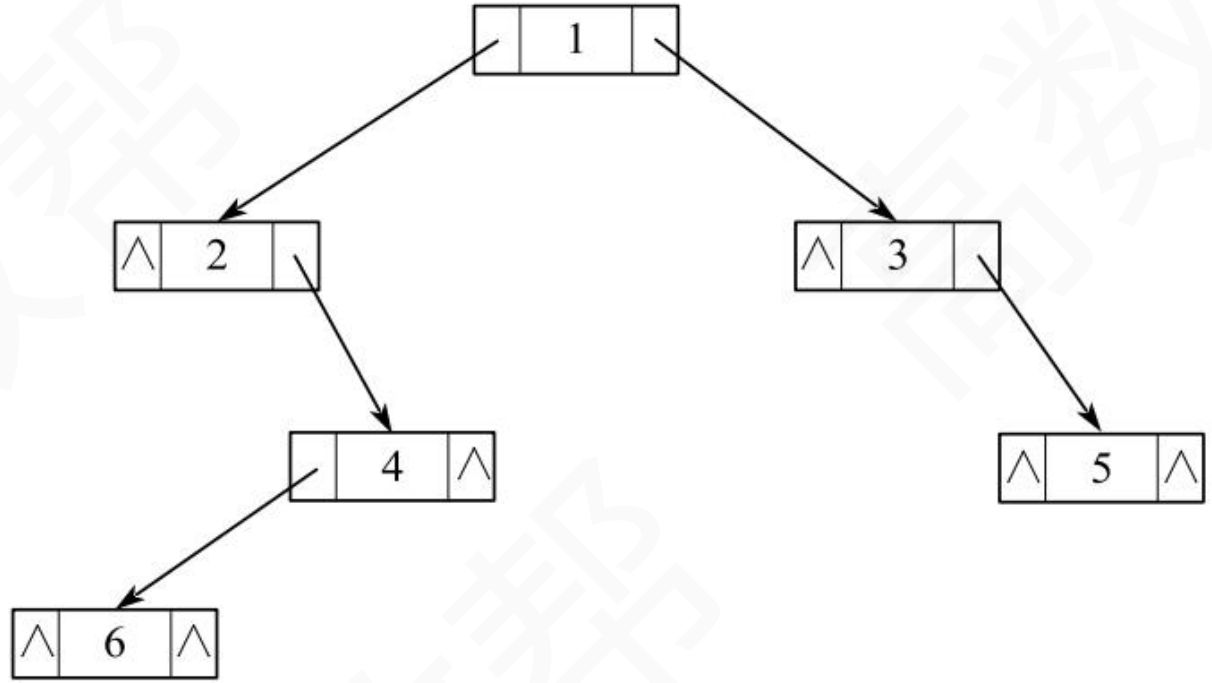
（2）链式存储结构：用链表结点来存储二叉树中的每个结点。在二叉树中，结点结构通 常包括若干数据域和若干指针域，二叉链表至少包含 3 个域：数据域data ,左指针域 lchild和右指针域 rchild。

二叉树的链式存储结构描述如下：

```
typedef struct BiTNode{
    ElemType data;                          //数据域
    struct BiTNode *lchild,*rchild;    //左、右孩子指针
}BiTNode,*BiTree;
```

6.2 树的基本概念





题 4.若二叉树用二叉链表作存储结构，则在 n 个结点的二叉树链表中只有 个非空指针 域。 （ ）

答案： 正确 在含有 n 个结点的二叉链表中，含有 n+1个空链域，含有 n-1 个非空链域。

6.3 二叉树的遍历

二叉树的遍历是指按某条搜索路径访问树中每个结点，使得每个结点均被访问一次， 而且仅被访问一次。

由二叉树的递归定义可知，遍历一颗二叉树便要决定对根结点N ，左子树L ，右子树R 的访问顺序。按照先遍历左子树再遍历右子树的原则，常见的遍历次序有先序(NLR)、 中序(LNR)和后序(LRN)三种遍历算法，其中“序”指的是根结点在何时被访问。

（1）先序遍历 先序遍历(PreOrder)的操作过程如下：

若二叉树为空，则什么也不做；否则，

1）访问根结点；

2）先序遍历左子树；

3）先序遍历右子树；

对应的递归算法如下：

```
void PreOrder(BiTree T){
    if(T!=NULL){
            visit(T);                    //访问根结点
            PreOrder(T->lchild);         //递归遍历左子树
            PreOrder(T->rchild);         //递归遍历右子树
    }
}
```

（2）中序遍历

中序遍历(InOrder)的操作过程如下，

若二叉树为空，则什么也不做，否则：

1）中序遍历左子树；

2）访问根结点；

3）中序遍历右子树；

对应的递归算法如下：

对应的递归算法如下：

```
void InOrder(BiTree T){
    if(T!=NULL){
            InOrder(T->lchild);          //递归遍历左子树
            visit(T);                    //访问根结点
            InOrder(T->rchild);          //递归遍历右子树
}
}
```

（3）后序遍历 后序遍历( )的操作过程如下。

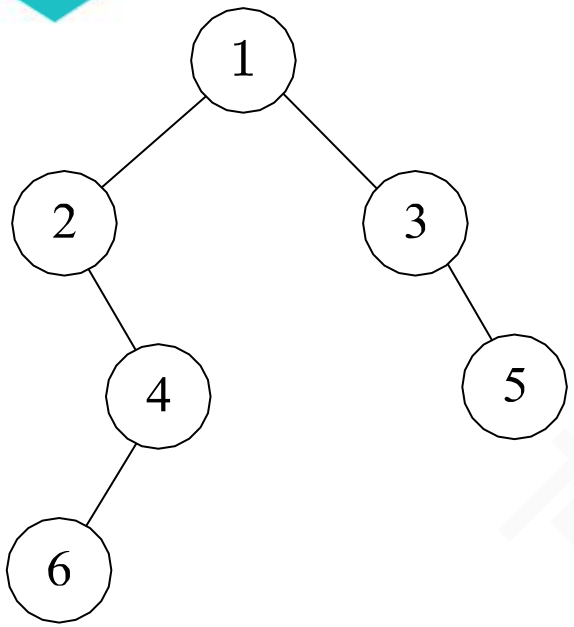
若二叉树为空，则什么也不做；否则，

1）后序遍历左子树;　　2）后序遍历右子树;　　3）访问根结点。

对应的递归算法如下：

```
void PostOrder(BiTree T){
if(T!=NULL){
PostOrder(T->lchild);          //递归遍历左子树
PostOrder(T->rchild);          //递归遍历右子树
visit(T);                      //访问根结点
} }
```

6.3 二叉树的遍历



先序遍历所得到的结点序列为 1 2 4 6 3 5。

中序遍历所得到的结点序列为 2 6 4 1 3 5。

后序遍历所得到的结点序列为 6 4 2 5 3 1。

（4）层次遍历

要进行层次遍历，需要借助一个队列。先将二叉树根结点入队，然后出队，访问出队 结点，若它有左子树，则将左子树根结点入队；若它有右子树，则将右子树根结点入队。 然后出队，访问出队结点……如此反复，直至队列为空。

6.3 二叉树的遍历

二叉树的层次遍历算法如下：

```
void LevelOrder(BiTree T){
InitQueue(Q);                                   //初始化辅助队列
BiTNode *p=T;
EnQueue(Q,p);                                   //将根结点入队
while(!IsEmpty(Q)){                             //队列不空则循环
DeQueue(Q,p);                                   //队头元素出队
visit(p);                                       //访问出队结点
if(p->lchild!=NULL) EnQueue(Q,p->lchild); //左子树不空，左子树根结点入队
if(p->rchild!=NULL) EnQueue(Q,p->rchild); //右子树不空，右子树根结点入队 } }
```

6.3 二叉树的遍历

题 1.对某二叉树进行先序遍历的结果为ABDEFC，中序遍历的结果为DBFEAC，则后序遍历的结果是（ ）。

A.DBFEAC　　B. DFEBCA　　C. BDFECA　　D. BDEFAC

答案： B

题 2.某二叉树结点的中序序列为：ABCDEFG，后序序列为：BDCAFGE，则其左子树中 结点数目为（）。

A.3　　B. 2　　C. 4　　D. 5

答案：C

课时七　树和二叉树（2）

考点	重要程度	占分	题型
1.树和森林	★★★★★	2~4	解答
2.二叉排序树	★★★★★	4~6	
3.哈夫曼树	必考	6~10	

视频讲解更清晰
仅4小时

7.1 树和森林

树的存储结构：双亲表示法、孩子表示法、孩子兄弟表示法。

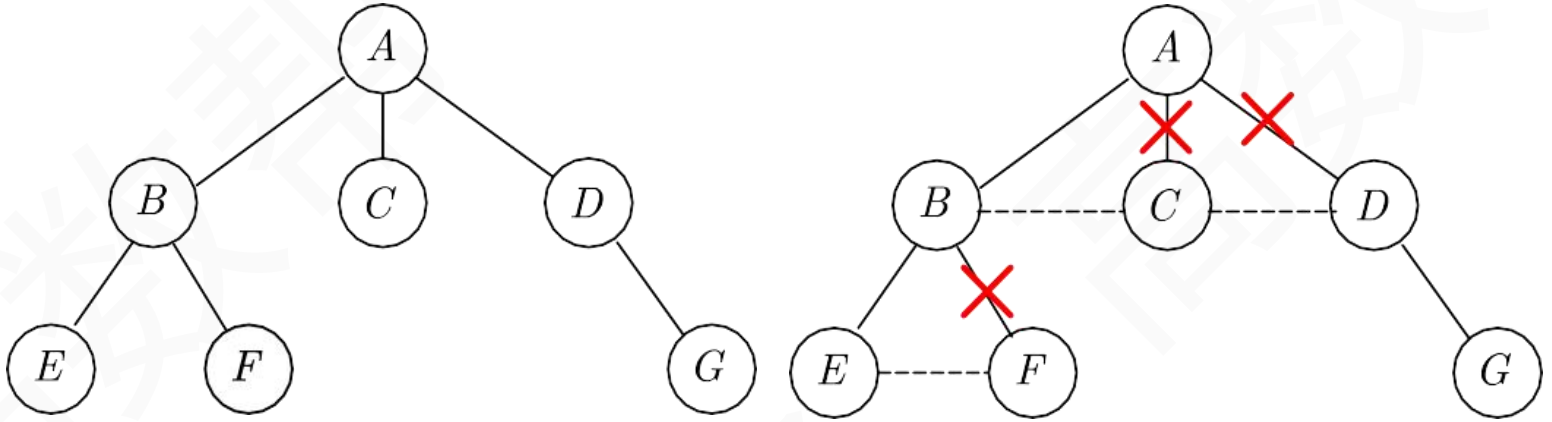
树转换成二叉树的画法：

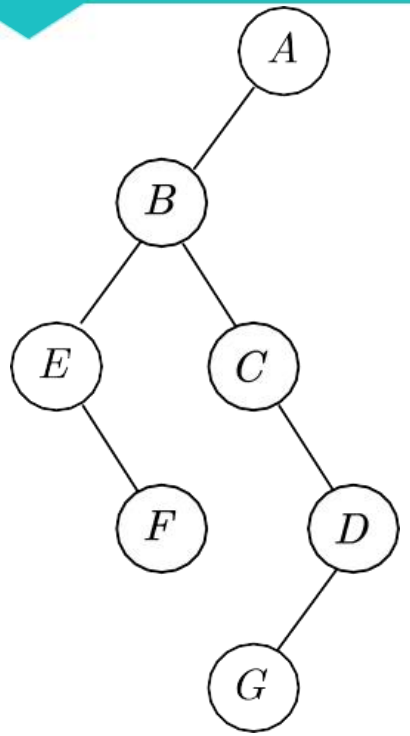
① 在兄弟结点之间加一条线；

② 对每个结点，只保留它与第一个孩子的连线，抹去与 其他孩子的连线；

③ 以树根为轴心，顺时针旋转 45°。

题 1.将树转换成二叉树。





题 2.由树转换成二叉树，其根结点的右子树总是空的。（）

答案：正确

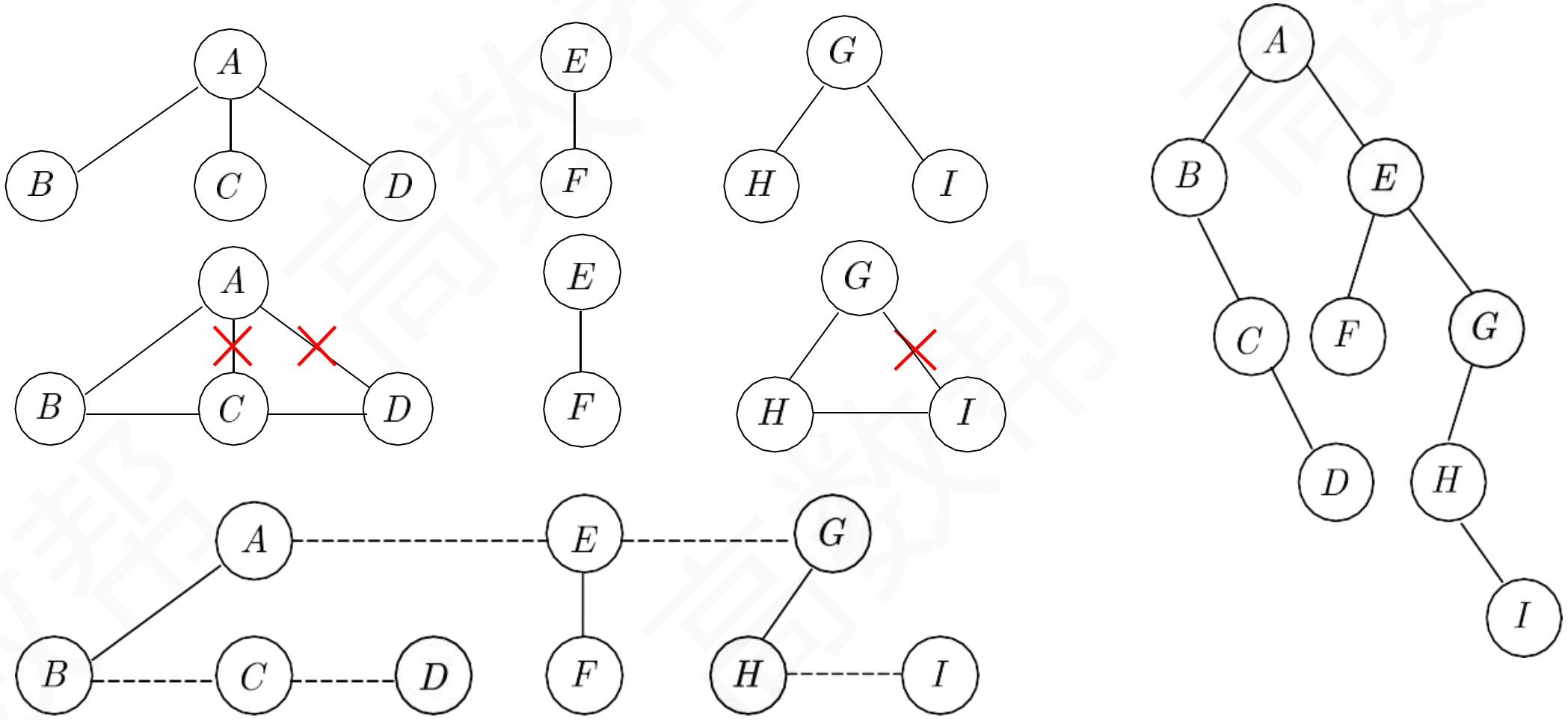
森林转换成二叉树的画法：

① 将森林中的每棵树转换成相应的二叉树；

② 每棵树的根也可视为兄弟结点，在每棵树之间加一根连线；

③ 以第一棵树的根为轴心顺时针旋转 45°。

题 3.将森林转换成二叉树。

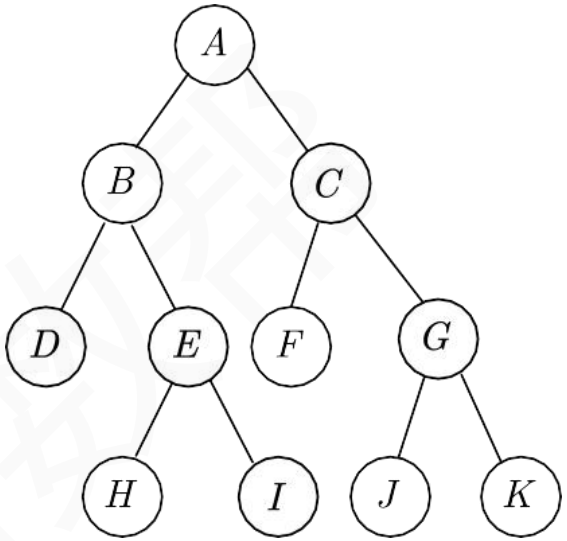


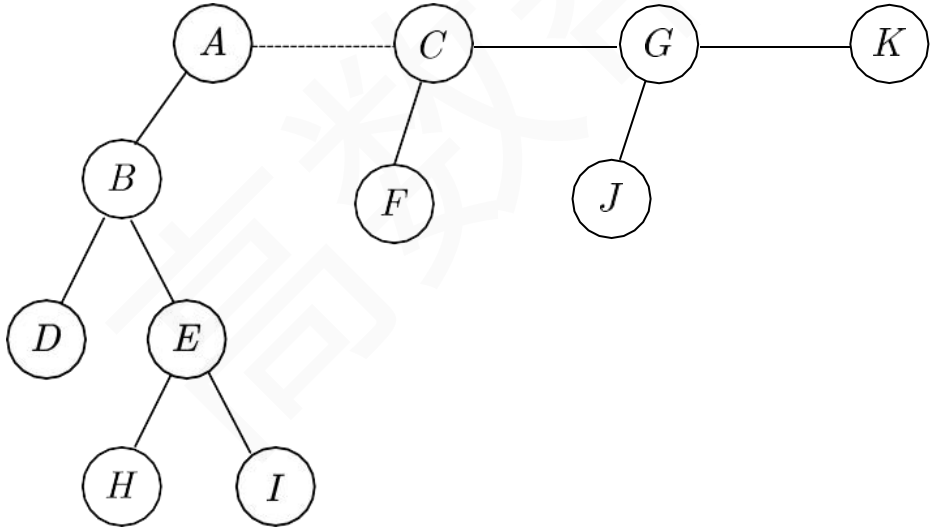
题 4.设森林 F 中有三棵树，第一、第二、第三棵树上的结点个数分别为 M_1,M_2,M_3，则与森林 F 对应的二叉树根结点的右子树上的结点个数为（ ）。

A.M_1　　B.M_1+M_2　　C.M_3　　D.M_2+M_3

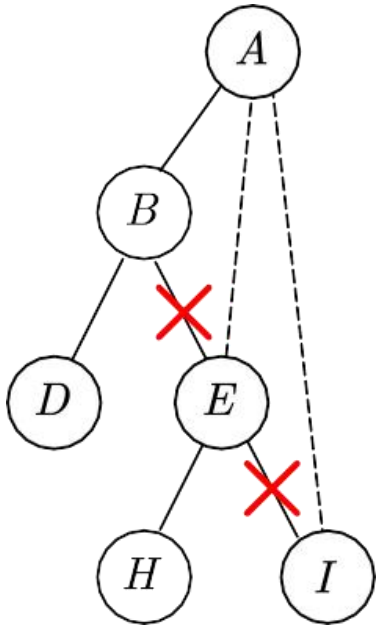
答案：D

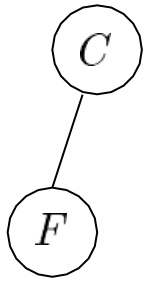
题 5.画出下图中的二叉树对应的森林。

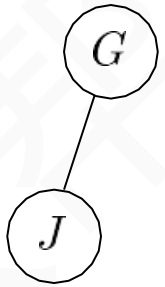


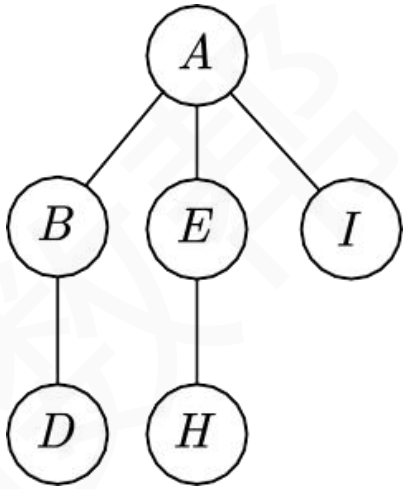


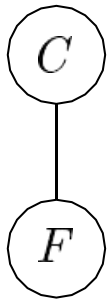
7.1 树和森林

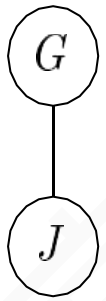








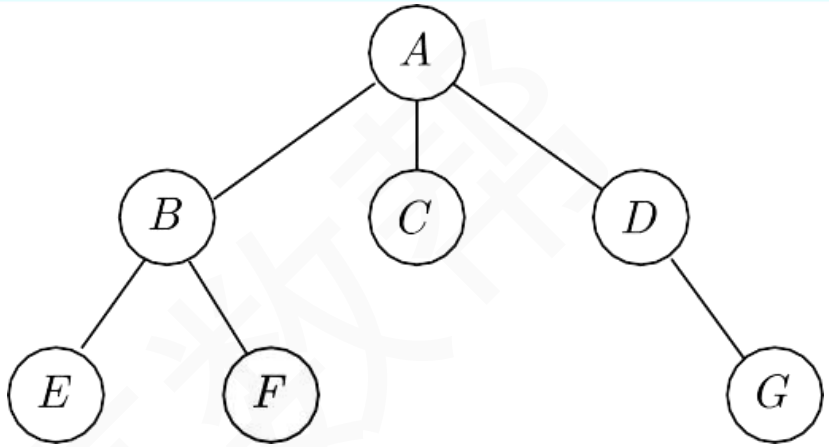




树的遍历是指用某种方式访问树中的每个结点，且仅访问一次。主要有两种方式：

（1） 先根遍历。若树非空，先访问根结点，再依次遍历根结点的每棵子树，遍历子树时 仍遵循先根后子树的规则。其遍历次序与这棵树对应二叉树的先序序列相同。

（2） 后根遍历。若树非空，先依次遍历根结点的每棵子树，再访问根结点，遍历子树时 仍遵循先子树后根的规则。其遍历次序与这棵树对应二叉树的中序序列相同。

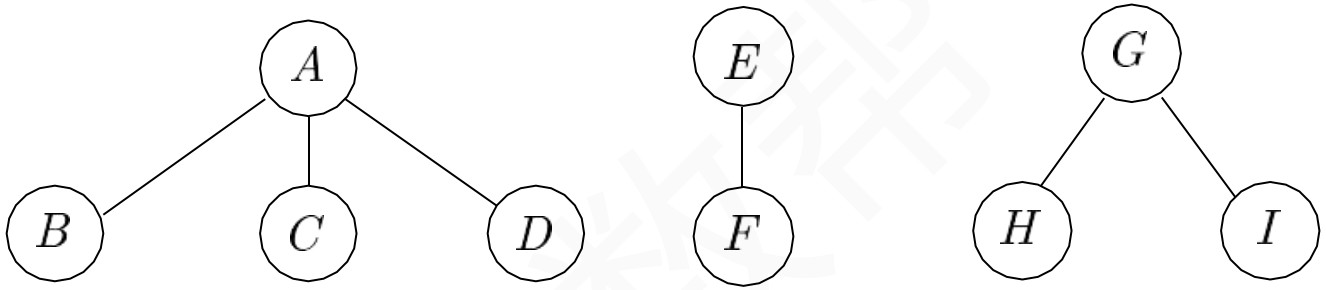


先根遍历序列为 A B E F C D G，后根遍历序列为 E F B C G D A 。

森林的两种遍历方法：

（1） 先序遍历。若森林非空，先访问森林中第一棵子树的根结点，再先序遍历第一棵树 中根结点的子树森林，再先序遍历除去第一棵树之后剩余的树构成的森林。

（2） 中序遍历。若森林非空，先中序遍历森林中第一棵树的根结点的子树森林，再访问 第一棵树的根结点，再中序遍历除去第一棵树之后剩余的树构成的森林。



先序遍历序列为 A B C D E F G H I，中序遍历序列为 B C D A F E H I G。

树	森林	二叉树
先根遍历	先序遍历	先序遍历
后根遍历	中序遍历	中序遍历

7.2 二叉排序树

二叉排序树（二叉查找树）或者是一颗空树，或者是具有下列特性的二叉树。

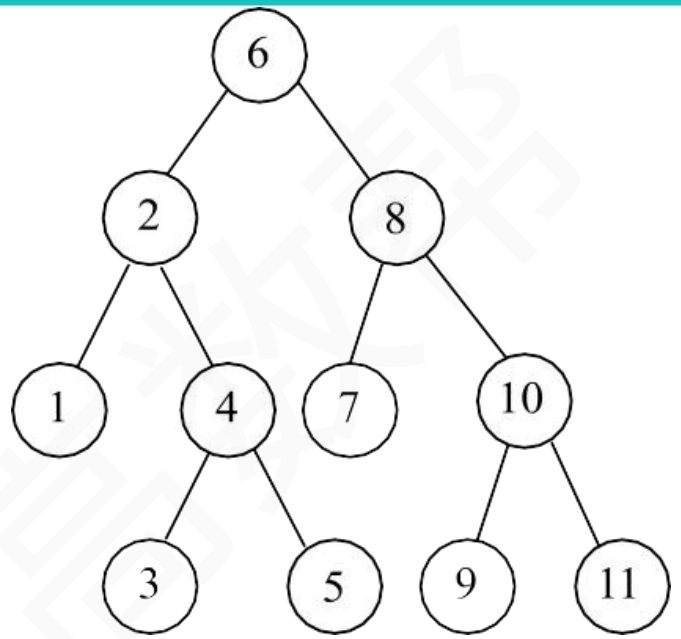
（1）若左子树非空，则左子树上所有结点的值均小于根结点的值；

（2）若右子树非空，则右子树上所有结点的值均大于根结点的值；

（3）左、右子树也分别是一棵二叉排序树。

根据二叉排序树的定义，左子树结点值 < 根结点值 < 右子树结点值，所以对二叉排序树进 行中序遍历，可以得到一个递增的有序序列。

7.2 二叉排序树



二叉排序树的中序遍历为 1 2 3 4 5 6 7 8 9 10 11

题 1.若对一棵二叉排序树构成的序列采用中序遍历，可得到（ ）结果。

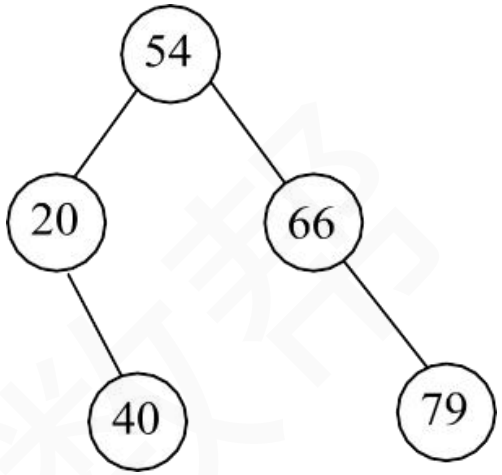
A.降序　　B.升序　　C.无序　　D.不确定

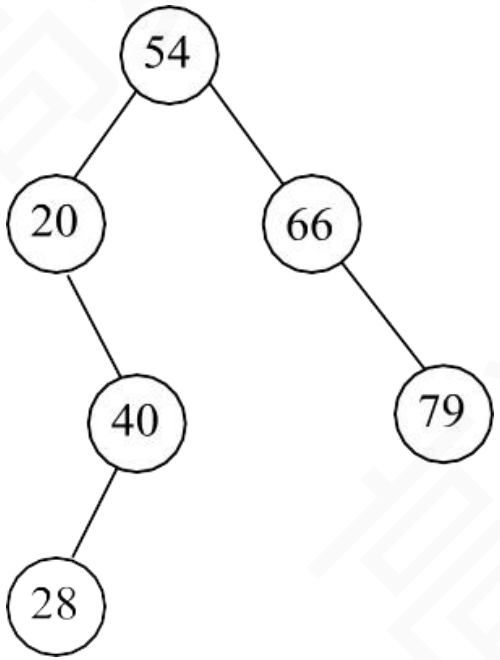
答案：B

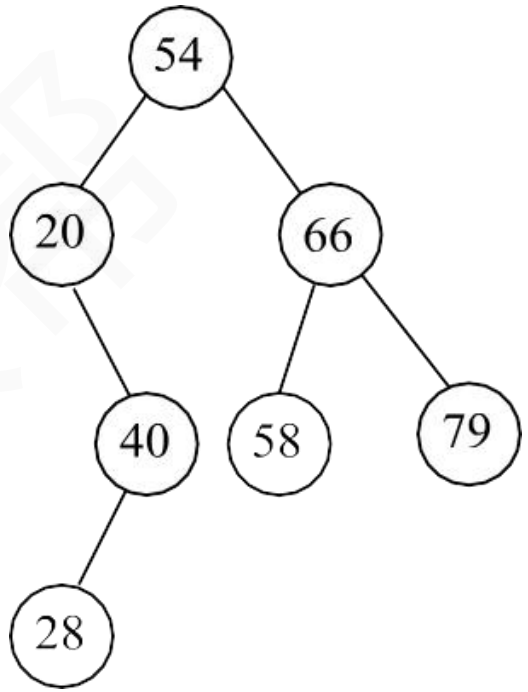
7.2 二叉排序树

二叉树的插入：

若原二叉排序树为空，则直接插入结点；否则，若关键字k小于根结点值，则插入到左子树，若关键字k大于根结点值，则插入到右子树。





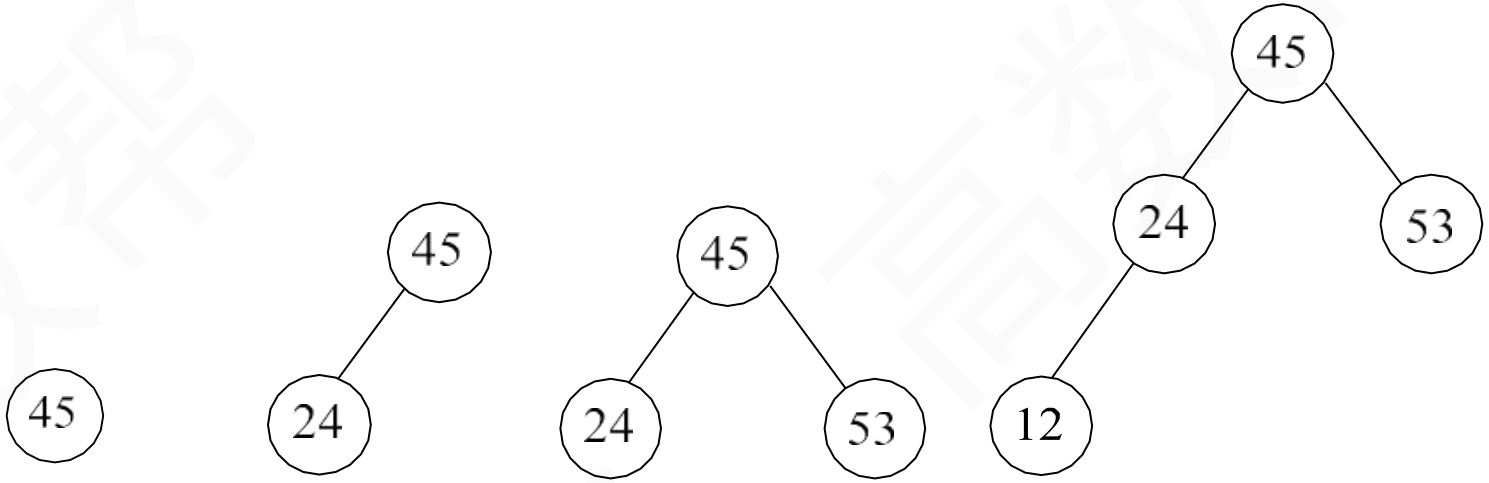


题 2.向二叉排序树插入一个新结点时，新结点一定会成为二叉排序树的一个叶子结点。（）

答案：正确

二叉排序树的构造：从一棵空树出发，依次输入元素，将它们插入二叉排序树的合适位置。

题 3.设查找的关键字序列是{45,24,53,12}，构造一棵二叉排序树。



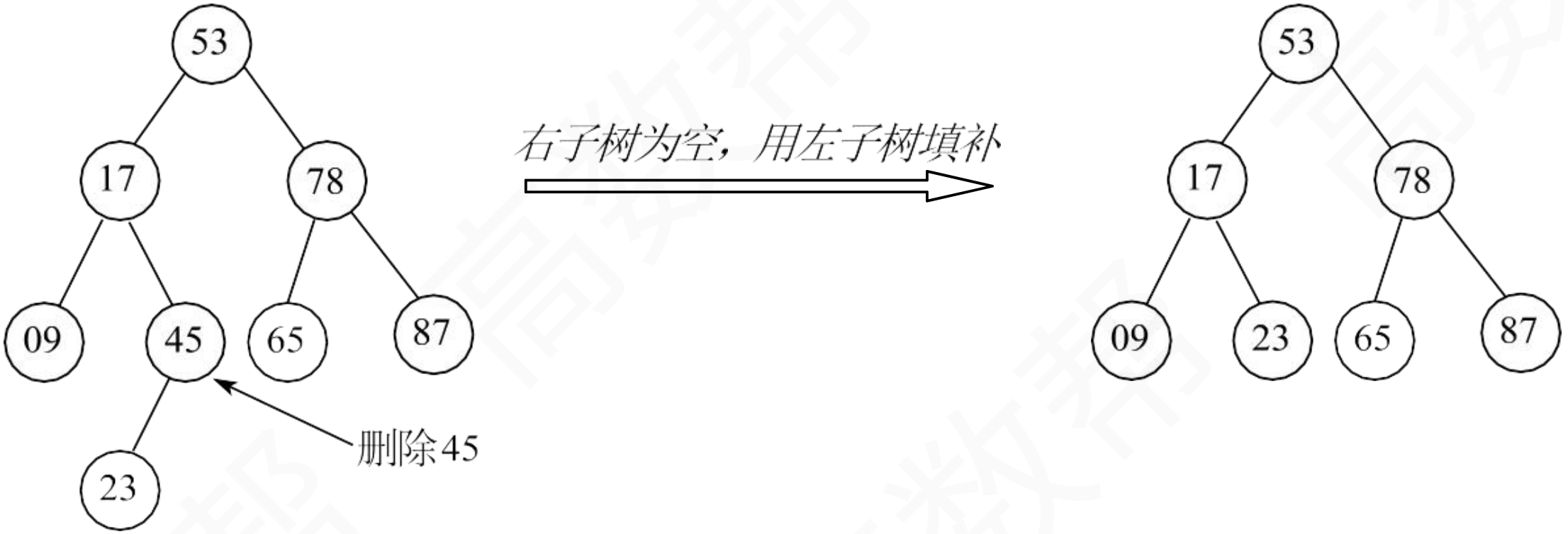
二叉排序树的删除：

（1）若被删除的结点是叶子结点，则直接删除；

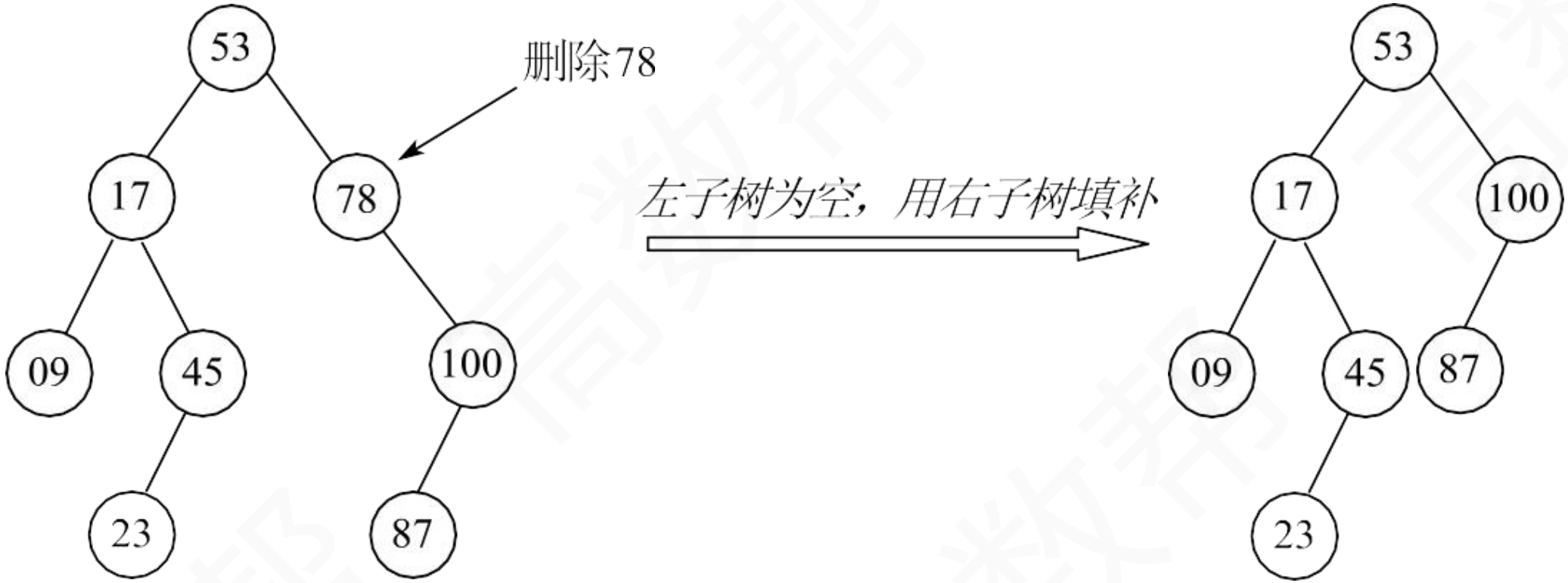
（2）若结点只有一棵左子树或者右子树，则让该结点的子树成为其父结点的子树；

（3）若结点有左、右两棵子树，则令该结点的直接后继（直接前驱）代替该结点，然后 从二叉排序树中删去这个直接后继（直接前驱），这样就转换成了前两种情况。

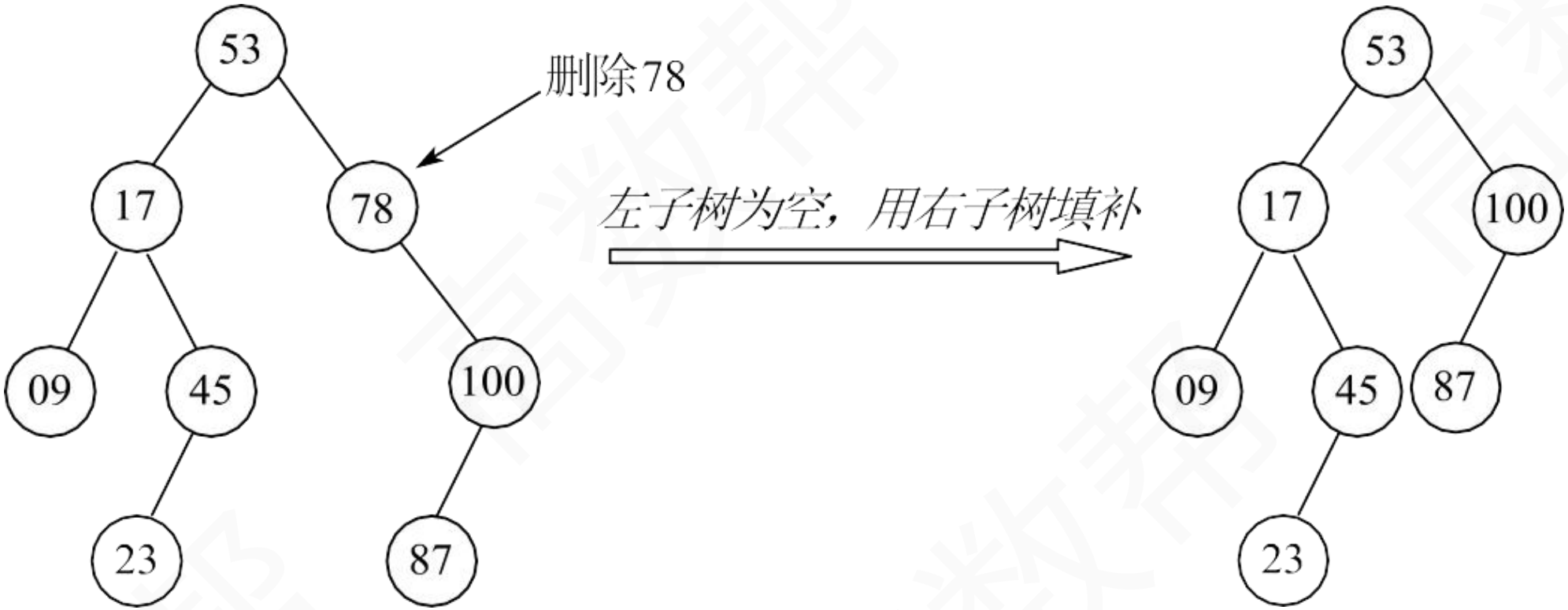
7.2 二叉排序树



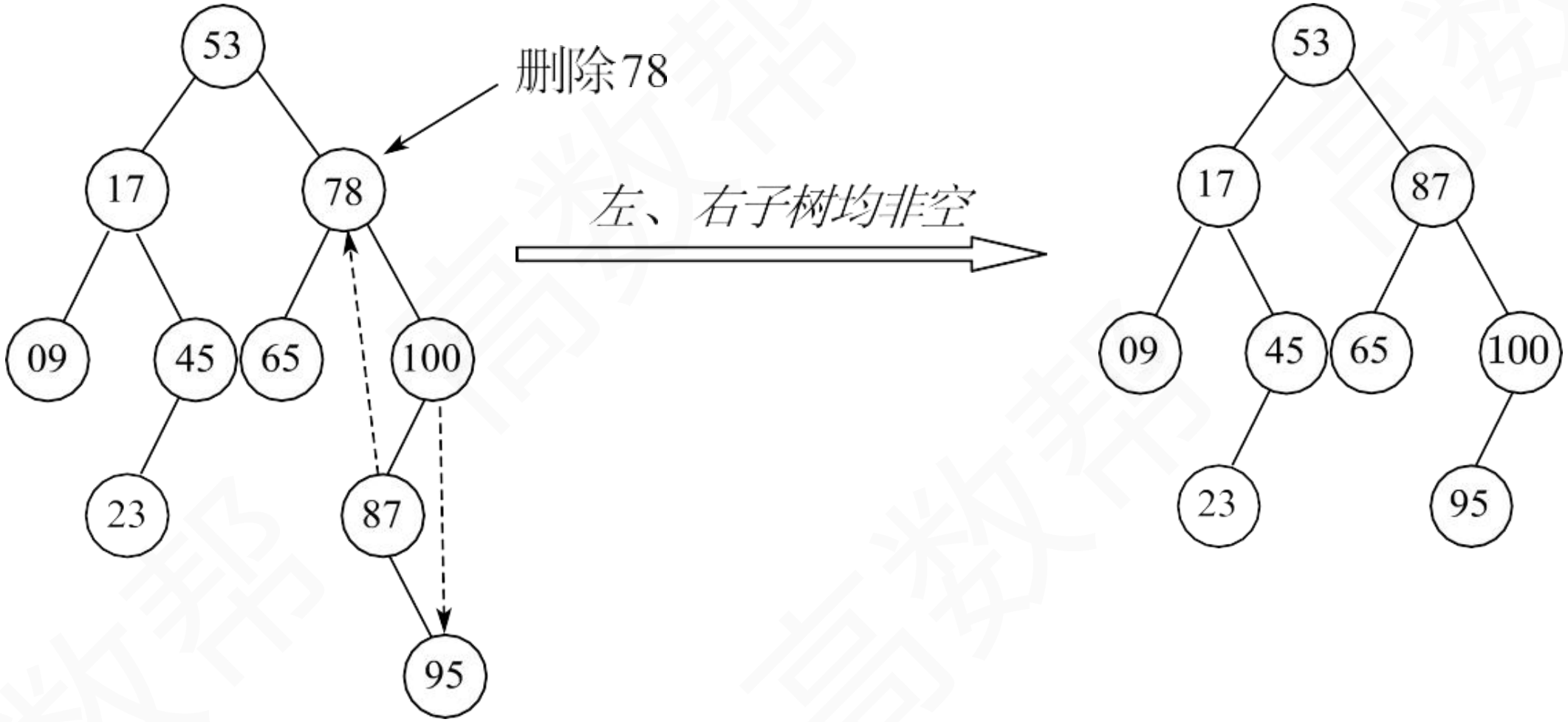
7.2 二叉排序树



7.2 二叉排序树



7.2 二叉排序树


53
17
78
09
45
65
100
23
87
95
删除78
左、右子树均非空
53
17
87
09
45
65
100
23
95

7.3 哈夫曼树

树中结点常被赋予一个代表某种意义的数值，那个数值称为该结点的权。

从树的根到任意结点的路径长度（经过的边数）与该结点上权值的乘积，称为该结点的带 权路径长度。

树中所有叶结点的带权路径长度之和称为该树的带权路径长度，记为 $WPL-\sum_{i-}^{n} w_i l_i^{\circ}$ 。

带权路径长度最小的二叉树称为哈夫曼树，也称最优二叉树。

7.3 哈夫曼树

给定 n个权值分别为$w_1,w_2,\cdots,w_n$ 的结点，构造哈夫曼树的算法描述如下：

（1）将这 n 个结点分别作为 棵仅含一个结点的二叉树，构成森林 F。

（2）构造一个新结点，从 F 中选取两棵根结点权值最小的树作为新结点的左、右子树，并 且将新结点的权值置为左、右子树上根结点的权值之和。

（3）从 F 中删除刚才选出的两棵树，同时将新得到的树加入 F 中。

（4）重复步骤（2）和（3），直至 F 中只剩下一棵树为止。

7.3 哈夫曼树

题 1.有一电文使用五种字符 a,b,c,d,e，其出现频率依次为 4,7,5,2,9 。

（1）试画出对应的哈夫曼树（要求左子树根结点的权小于等于右子树根结点的权）。

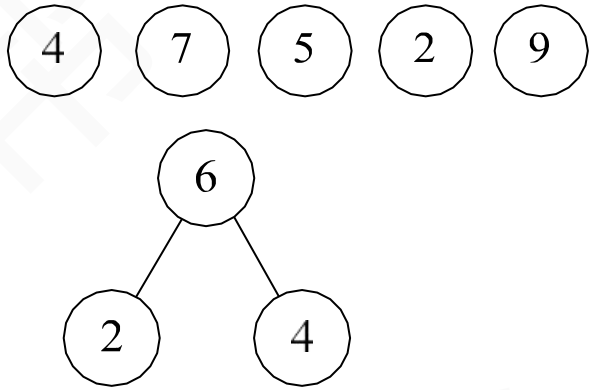
（2）求出每个字符的哈夫曼编码。

（3）译出编码序列 1100011100010101l的相应电文。

（4）求带权路径长度。

视频讲解更清晰
仅4小时

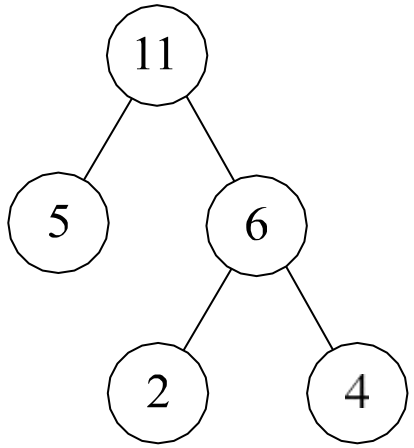
答案：（1）



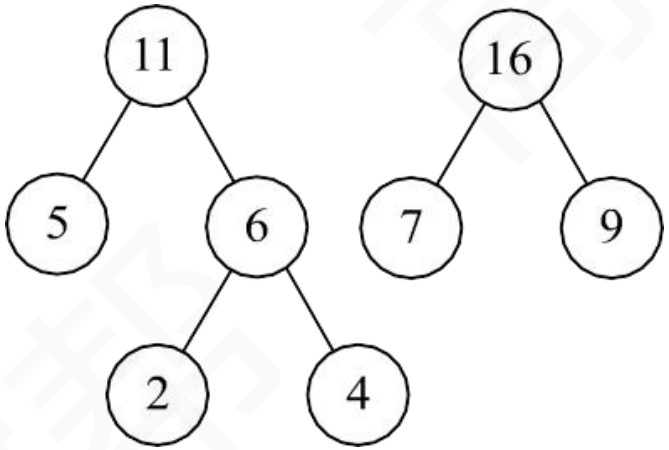
4,7,5,2,9

6,7,5,9

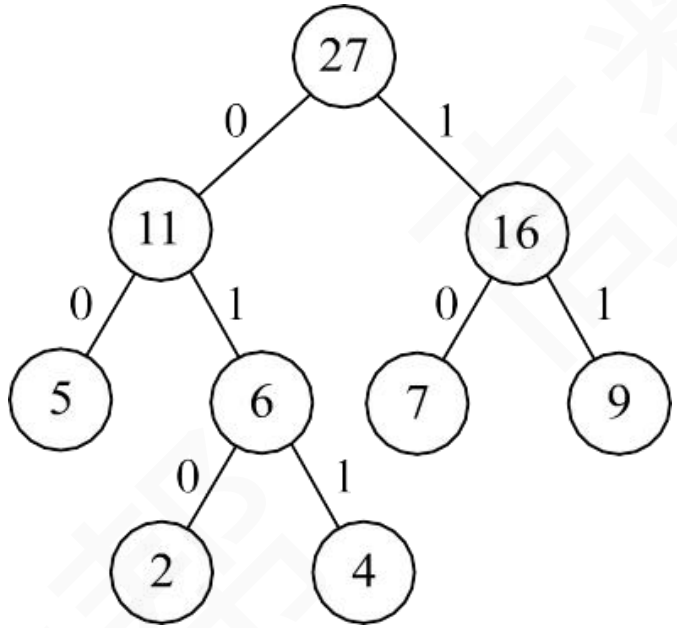
7.3 哈夫曼树



11,7,9



11,16



27

（2） a对应的哈夫曼编码为011；

b对应的哈夫曼编码为10；

c对应的哈夫曼编码为00；

d对应的哈夫曼编码为010；

e对应的哈夫曼编码为11；

（3）编码序列 1100011100010101011的相应电文为ecabcbbe 。

（4）WPL=5×2+2×3+4×3+7×2+9×2=60