第 5 章　树与二叉树

本章的知识结构图

树与二叉树的知识结构图如图 5-1 所示。

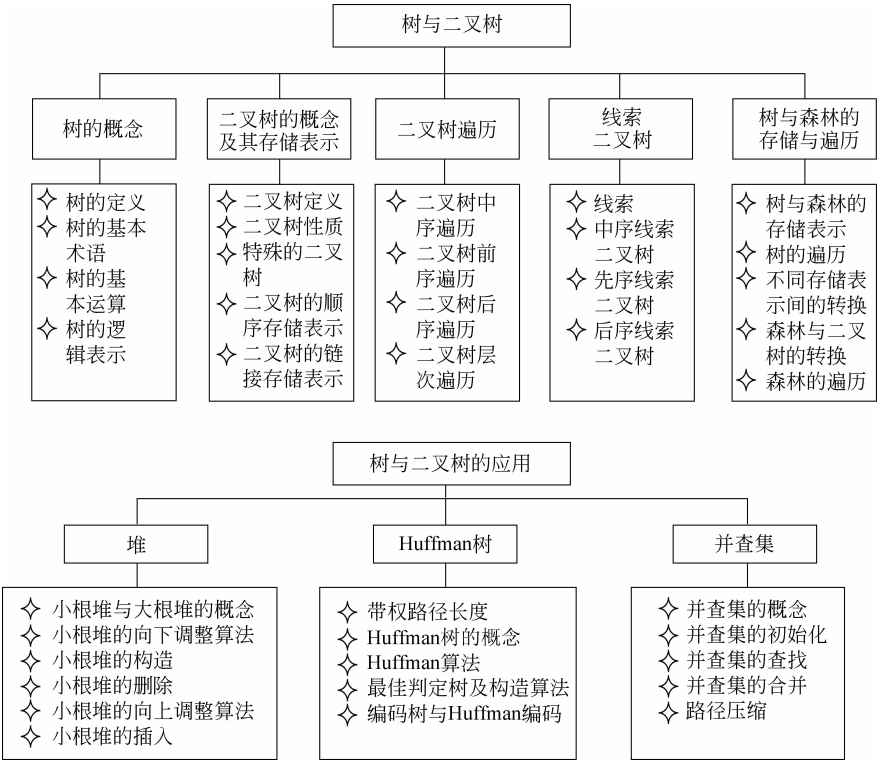


图 5-1　树与二叉树的知识结构图

5.1　树的基本概念

5.1.1　知识点提要

1. 树的定义

(1) 树是树形结构的简称，它是一种重要的非线性结构。树的定义是：

树是 $n(n\geqslant 0)$个结点的有限结合，当 $n=0$ 时称为空树，在任一非空树($n>0$)中，它有且仅有一个称作根的结点，其余的结点可分为 m 棵($m\geqslant 0$)互不相交的子树(称作根的子树)，每棵子树又同样是一棵树。

(2) 树的定义是递归的，树是一种递归的数据结构，递归结束于叶结点。

(3) 当树中某结点有多个子树时,其子树的顺序一般是任意的,这种树是无序树。如果对一个度为 m 的树规定了各个子树的序号,这种树是有序树。

(4) 该定义只给出了树的组成特点。作为一种逻辑数据结构,树是一种分层结构。如果把双亲结点定义为直接前趋,子女结点定义为直接后继,树中的根结点没有直接前趋,除根结点外其他每个结点都有一个且只有一个直接前趋;除叶结点没有直接后继外,其他每个结点可以有一个或多个直接后继。因此,在有 n 个结点的树中有 $n-1$ 条边。

2. 树的基本术语

(1) 双亲结点、子女结点、兄弟结点、祖先与子孙结点的定义。

(2) 结点的度。

(3) 叶结点(终端结点)、根结点、分支结点(非终端结点)。

(4) 结点的层次。

(5) 结点的深度和高度、树的深度和高度。

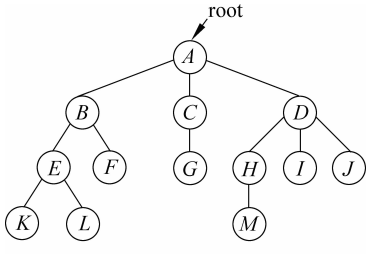
(6) 路径和路径长度。树中两个结点间的路径由这两个结点间所经过的结点序列来表征,其路径长度则由这些结点经过的分支条数来确定。树的外部路径长度是各外结点到根结点的路径长度之和,树的内部路径长度是各内结点到根结点的路径长度之和。

(7) 有序树和无序树。

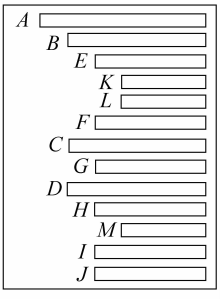
(8) 森林:是 $m(m\geqslant 0)$棵互不相交的树的集合。

3. 树的逻辑表示

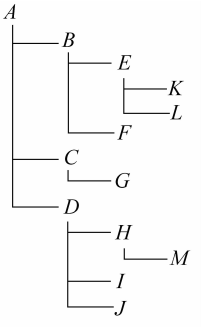
树的逻辑表示有如图 5-2 所示的 5 种。



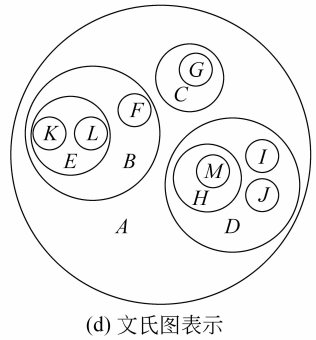
(a) 树形表示



(b) 凹入表表示



(c) 树的目录表表示



(d) 文氏图表示

$$A(\underbrace{B(E(K,L),F)}_{T_1},\underbrace{C(G)}_{T_2},\underbrace{D(H(M),I,J)}_{T_3})$$

(e) 广义表表示

图 5-2　树的逻辑表示

4. 树的基本操作

- bool CreateTree(position& t);　　//建立一棵树,t 返回树根指针
- bool DelTree(position& t);　　//释放以 t 为根的树的所有结点
- void DelSubTree(position& t,position p);　　//删除以 t 为根的树中以 p 为根的子树
- TElelmType GetData(position p);　　//取得结点 * p 的值
- position FirstChild(position p);　　//返回 p 第一个子女的地址
- position NextSibling(position p);　　//返回 p 下一兄弟的地址
- position Parent(position p);　　//返回 p 双亲结点地址
- bool InsertChild(position t,TElemType x);　　//在结点 t 下插入值为 x 的新子女
- bool DeleteChild(position t,TElemType x);　　//删除结点 t 的元素值为 x 的子女
- bool IsEmpty(position t);　　//判树 t 空否
- bool IsLeaf(position p);　　//判结点 * p 是否叶结点

5.1.2　选择题

1. 树最合适用来表示(　　)。

A. 有序数据元素　　B. 元素之间具有分支层次关系的数据

C. 无序数据元素　　D. 元素之间无联系的数据

2. 树 T 的结点用"层号+数据"表示为 $1a,2b,3d,3e,2c$,对应于如下的(　　)。

A. $a(b(d,e),c)$　　B. $a(b),(d,e),c$

C. $a(b,d(e),c)$　　D. $1a(2b(3d,3e),2c)$

3. 一棵有 n 个结点的树的所有结点的度数之和为(　　)。

A. $n-1$　　B. n　　C. $n+1$　　D. $2n$

4. 设树中某结点不是根结点,则离它最近的祖先结点是(　　)。

A. 双亲结点　　B. 双亲结点的双亲结点

C. 该结点本身　　D. 不是根结点

5. 两个结点之间的路径长度是指结点之间(　　)。

A. 所经历的结点的个数　　B. 所经历的边的条数

C. 所经历的边上权值之和　　D. 绕过根结点的路径上边的条数

6. 树的路径长度是从根结点到树中每一结点的路径长度的(　　)。

A. 最小值　　B. 最大值　　C. 总和　　D. 平均值

7. 设一棵度为 m 的树有 n 个结点,则(　　)。

A. 树的高度最大是 $n-m+1$　　*B*. 树的高度最大是 $n-m$

C. 第 i 层最多有 $m(i-1)$个结点　　*D*. 至少在某一层上正好有 m 个结点

8. 设一棵度为 m 的树的高度为 h,则(　　)。

A. 至多有 m^h-1 个结点　　B. 至多有 m^h 个结点

C. 至少有 $h+m-1$ 个结点　　D. 至少有 $h+m$ 个结点

9. 在一棵度为 4 的树 T 中,若有 20 个度为 4 的结点,10 个度为 3 的结点,1 个度为 2 的结点,10 个度为 1 的结点,则树 T 的叶结点个数是(　　)。

A. 41　　B. 82　　C. 113　　D. 122

10. 设一棵树中只有度为0和度为3的结点,则该树的第i层($i \geqslant 1$)的结点个数最多为(　　)。

A. $3^{i-1}-1$　　B. $3^{i}-1$　　C. 3^{i-1}　　D. 3^{i}

11. 假定一棵三叉树的结点数为50,则它的最小高度为(　　)。

A. 3　　B. 4　　C. 5　　D. 6

12. 设在一棵三叉树中,度为3的结点数等于度为2的结点数,且树中叶结点的数目为13,则度为2的结点数目为(　　)。

A. 2　　B. 3　　C. 4　　D. 5

13. 在一棵有k层的满三叉树中,结点总数为(　　)。

A. $(3^{k}-1)/2$　　B. $3^{k}-1$　　C. $(3^{k}-1)/3$　　D. 3^{k}

14. 按照树的定义,具有3个结点的树有(　　)种形态(不考虑数据信息的组合情况)。

A. 2　　B. 3　　C. 4　　D. 5

15. 一棵含有n个结点的m叉树,可能达到的最大深度为(①),最小深度为(②)。

A. $\lceil \log_m(n(m-1)+1) \rceil$　　B. $\log_m(nm-1)+1$

C. m　　D. n

【参考答案】

1. B　2. A　3. A　4. A　5. B　6. C　7. A　8. C　9. B　10. C　11. C　12. C　13. A　14. A　15. ① D ② A

5.1.3　判断题

1. 树中任一结点的子树不必是有序的。
2. 树可以看做是特殊的无向图。
3. 树中的每一个结点都有一个双亲结点,有零个或多个子女结点。
4. 有n个结点的树中有$n-1$条分支。
5. 度为m的树至少有一个度为m的结点,不存在度大于m的结点。
6. 在树中,同一层的结点之间都存在兄弟关系。
7. 一棵m叉树就是度等于m的树。
8. 度为m的树是一棵m叉树
9. 结点的高度等于结点的深度。
10. 树的高度等于树的深度。
11. 树中任意两个结点之间的路径是唯一的。
12. 树中任意两个结点之间的路径是用路径上的结点来标识的。
13. 结点间的路径的长度等于该路径上的分支条数。
14. 树的叶结点之间没有路径存在。
15. 树中结点之间一定存在共同祖先。
16. 树中任意两个结点一定存在最近共同祖先。
17. 一棵具有n个结点的度为4的树的高度最小为$\lceil \log_4(3n+1) \rceil$。

【参考答案】

1. 错　2. 对　3. 错　4. 对　5. 对　6. 错　7. 错　8. 对　9. 错　10. 对　11. 对

12. 对　13. 对　14. 错　15. 对　16. 对　17. 对

5.1.4　简答题

1. 树的叶结点无子女，是否可称它无子树？

2. 一个结点的祖先结点和子女结点是否包括该结点本身？

3. 在结点个数为 $n(n>1)$ 的各棵树中，深度最小的树的深度是多少？它有多少叶结点？多少分支结点？深度最大的树的深度是多少？它有多少叶结点？多少分支结点？

4. 如果一棵树有 n_1 个度为 1 的结点，有 n_2 个度为 2 的结点……有 n_m 个度为 m 的结点，试问有多少个度为 0 的结点？试推导之。

5. 一棵有 n 个结点的树中所有非叶结点的度均为 m，则该树的叶结点有多少？

6. 对于一棵 m 叉树(设根在第 1 层)，第 i 层最多有多少结点？

7. 一棵高度为 h 的 m 叉树最多有多少结点？

8. 一棵高度为 h 的 m 叉树最少有多少结点？

9. 一棵有 n 个结点的完全 m 叉树的高度是多少？

10. 一棵有 n_0 个叶结点的完全 m 叉树的高度是多少？

11. 对于一棵 m 叉树(设根在第 1 层)，从 0 开始自上向下分层给各结点编号。问编号为 k 的结点的双亲结点编号是多少？

12. 对于一棵 m 叉树(设根在第 1 层)，从 0 开始自上向下分层给各结点编号。问编号为 k 的结点的第 1 个子女的编号是多少：

13. 对于一棵 m 叉树(设根在第 1 层)，从 0 开始自上向下分层给各结点编号。问编号为 k 的结点在第几层？

14. 对于一棵高度为 h 的满 m 叉树，如果按层次自顶向下，同一层自左向右，顺序从 1 开始对全部结点进行编号，问编号为 i 的结点的第一个子女(若存在)的编号是多少？

15. 对于一棵高度为 h 的满 m 叉树，如果按层次自顶向下，同一层自左向右，顺序从 1 开始对全部结点进行编号，问编号为 i 的结点的双亲结点(若存在)的编号是多少？

16. 对于一棵高度为 h 的满 m 叉树，如果按层次自顶向下，同一层自左向右，顺序从 1 开始对全部结点进行编号，问编号为 i 的结点有右兄弟的条件是什么？其右兄弟结点的编号是多少？

17. 对于一棵高度为 h 的满 m 叉树，如果按层次自顶向下，同一层自左向右，顺序从 1 开始对全部结点进行编号，问编号为 i 的结点在第几层？

18. 若森林共有 n 个结点和 e 条边($e<n$)，则该森林中有多少棵树？

【参考答案】

1. 如果树可以为空，则树的叶结点虽然没有子女，但不能说它没有子树，只能说它的子树为空。

2. 树的结点是自身的祖先结点，除自己外其他祖先结点为“真祖先”；子女结点的定义也有这种情形。但通常在数据结构中所指祖先和子女是指真祖先和真子女。

3. 结点个数为 n 时，深度最小的树的宽度(每层结点个数)最大，其深度为 2，有 2 层；它有 $n-1$ 个叶结点，1 个分支结点。让树中每一层只有 1 个结点，树的深度最大，其深度为 n，有 n 层；它有 1 个叶结点，$n-1$ 个分支结点。分支结点包括根结点。

4. 设树中总结点数为 $n=n_0+n_1+n_2+\cdots+n_m$,总分支数为 $e=n-1$,则 $e=n_0+n_1+n_2+\cdots+n_m-1=mn_m+(m-1)*n_{m-1}+\cdots+2n_2+n_1$,有 $n_0=\left(\sum_{i=2}^{m}(i-1)n_i\right)+1$。

5. 对于只有度为 0 和度为 m 的树,$n_0=(m-1)n_m+1$,又 $n=n_0+n_m$,即 $n_m=n-n_0$,有 $n_0=(m-1)(n-n_0)+1$,$mn_0=(m-1)n+1$,最后得 $n_0=((m-1)n+1)/m$,此即叶结点数。

6. m 叉树(设根在第 1 层)的第 i 层最多有 m^{i-1} 个结点。

7. 计算公比为 m 的等比级数前 h 项的和:$m^0+m^1+\cdots+m^{h-1}=(m^h-1)/(m-1)$。因此高度为 $h(h\geqslant 2)$ 的 m 叉树最多有 $(m^h-1)/(m-1)$ 个结点。

8. 若 m 叉树从根开始,每层仅一个结点,形成单支树,最少有 h 个结点。

9. 设完全 m 叉树的高度为 h,则从 $(m^{h-1}-1)/(m-1)+1\leqslant n\leqslant(m^h-1)/(m-1)$ 可推得 $h=\lceil\log_m(n(m-1)+1)\rceil$ 或 $h=\lfloor\log_m((n-1)(m-1)+1)\rfloor+1$。

10. 设完全 m 叉树的高度为 h,则树中第 h 层最少一个叶结点,最多 m^{h-1} 个叶结点,有不等式 $m^{h-2}-1+1\leqslant n_0\leqslant m^{h-1}$,推得 $h-2\leqslant\log_m n_0\leqslant h-1$,即 $1+\log_m n_0\leqslant h\leqslant 2+\log_m n_0$。

11. 因为 m 叉树的根所在层次设为 1,从 0 开始自上向下分层给各结点编号,当 $k=0$ 时结点 k 是根,没有双亲结点;$k>0$ 时结点 k 的双亲结点是 $\lfloor(k-1)/m\rfloor$。

12. 若设 m 叉树有 n 个结点,且根结点编号为 0,对于编号为 k 的结点,如果 $k*m+1<n$,则编号为 k 的结点的第 1 个子女的编号是 $k*m+1$。

13. 设结点 k 在第 i 层,到第 $i-1$ 层为止的 m 叉树最多有 $(m^{i-1}-1)/(m-1)-1$ 个结点,到第 i 层为止树最多有 $(m^i-1)/(m-1)-1$ 个结点,由此可得 $(m^{i-1}-1)/(m-1)\leqslant k<(m^i-1)/(m-1)$,$i-1\leqslant\log_m(k(m-1)+1)<i$,因此 k 结点在 $i=1+\lfloor\log_m(k(m-1)+1)\rfloor$ 层。

14. 结点 i 的第 1 个子女编号为 $j=(i-1)\times m+2$。

15. 结点 i 的第 1 个子女编号为 $j=(i-1)\times m+2$,反过来,结点 i 的双亲结点的编号是 $\lfloor(i-2)/m\rfloor+1$,根结点没有双亲指针,所以,以上计算式要求 $i>1$。如果考虑对于 $i=1$ 也适用(根的双亲指针设为 0),可将上式改为 $\lfloor(i+m-2)/m\rfloor$。

16. 若结点 i 从 1 开始编号,则当 $(i-1)\%m\neq 0$ 时结点 i 有右兄弟,右兄弟编号为 $i+1$。

17. 设结点 i 在第 d 层,则从 $(m^{d-1}-1)/(m-1)<i\leqslant(m^d-1)/(m-1)$ 可推得 $d-1<\log_m(i(m-1)+1)\leqslant d$,$d=\lceil\log_m(i(m-1)+1)\rceil$,其中,$\lceil\rceil$ 表示向上取整。

18. 设森林中有 m 棵树,各棵树的结点个数分别为 $n_1,n_2,\cdots,n_m$,那么,各棵树的边数分别为 $n_1-1,n_2-1,\cdots,n_m-1$。因为 $n=n_1+n_2+\cdots+n_m$,$e=(n_1-1)+(n_2-1)+\cdots+(n_m-1)=n_1+n_2+\cdots+n_m-m=n-m$,所以 $m=n-e$。

5.1.5 算法题

1. 设计一个递归算法,计算树 T 的高度。

2. 设计一个递归算法,统计树 T 的叶结点个数。

3. 设计一个递归算法,统计树 T 的结点总数。

【参考答案】

1. 若设树的结构类型是 Tree,算法的递归结束条件是递归到树空,返回高度 0;否则若递归到叶结点,返回高度 1;否则递归计算所有子树的高度,取其大值加一(根结点还有一层),即为树的高度。算法描述如下:

```
int Height (Tree T) {
    if (IsEmpty (T)) return 0;
    else if (IsLeaf (T)) return 1;
    else {
        int h=0,k;
        for (Tree p=FirstChild (T); p !=NULL; p=NextSibling (T,p))
            if ((k=Height (p))>h) h=k;  //计算子树的高度 k,取最大者
        return h+1;                     //返回树 T 的高度
    }
}
```

2. 若设树的结构类型是 Tree,算法的递归结束条件是递归到树空,返回叶结点个数 0;否则若递归到叶结点,返回叶结点个数 1;否则递归计算所有子树的叶结点个数,把它们累加起来,返回结果即可。算法描述如下:

```
int Leave (Tree T) {
    if (IsEmpty (T)) return 0;
    else if (IsLeaf (T)) return 1;
    else {
        int sum=0;
        for (Tree p=FirstChild (T); p !=NULL; p=NextSibling (T,p))
            sum=sum+Leave (p);          //计算子树叶结点数,累加
        return sum;                     //返回树 T 的叶结点个数
    }
}
```

3. 若设树的结构类型是 Tree,算法的递归结束条件是递归到树空,返回结点个数 0;否则递归计算所有子树的结点个数,把它们累加起来再加 1(根结点个数),返回结果即可。算法描述如下。

```
int Nodes (Tree T) {
    if (IsEmpty (T)) return 0;
    else {
        int sum=0;
        for (Tree p=FirstChild (T); p !=NULL; p=NextSibling (T,p))
            sum=sum+Nodes (p);          //计算子树结点个数,累加
        return sum+1;                   //返回树 T 的结点个数
    }
}
```

5.2　二叉树及其存储表示

5.2.1　知识点提要

1. 二叉树的定义

(1) 一棵二叉树或者是一棵空树,或者是一棵由一个根结点和两棵互不相交的分别称

作根的左子树和右子树所组成的非空树,其左子树和右子树又同样是一棵二叉树。

(2) 这是一个递归的定义,递归结束于空的子树(注意,不能说没有子树)。二叉树是有序树,根结点的两棵子树要区分左子树和右子树,它们的地位不能互换。

2. 特殊的二叉树

设二叉树 T 有 n 个结点,令 $k=\lceil \log_2(n+1) \rceil$,$r=n-(2^{k-1}-1)$,则 k 代表离树根最远的结点所处层次,即二叉树的深度,r 代表第 k 层结点个数。如果其中 $2^{k-1}-1$ 个结点放满第 1 到第 $k-1$ 层,则

(1) 若 $0<r\leqslant 2^{k-1}$,且这 r 个结点集中存放在第 k 层的左侧,则称 T 是一棵完全二叉树。

(2) 特别地,若 $r=2^{k-1}$,则称 T 是一棵满二叉树,满二叉树是完全二叉树。

(3) 正则二叉树又称严格二叉树,它只有度为 2 和度为 0 的结点。

(4) 若二叉树有 h 层,上面 $h-1$ 层都是满的,第 h 的结点不是集中存放在第 k 层的左侧,而是散见于第 h 层的各处,则称这种二叉树为理想平衡树或丰满树。

3. 二叉树的性质

性质 1　二叉树上第 i 层上至多有 2^{i-1} 个结点($i\geqslant 1$,根结点所在层次为 1)。

性质 2　高度为 h 的二叉树至多有 2^h-1 个结点($h\geqslant 0$,空树的高度为 0)。

性质 3　设二叉树中度为 2 的结点有 n_2 个,度为 1 的结点有 n_1 个,度为 0(叶结点)的结点有 n_0 个,则 $n_0=n_2+1$。

性质 4　具有 n 个($n\geqslant 0$)结点的完全二叉树的深度为$\lceil \log_2(n+1) \rceil$或$\lfloor \log_2 n \rfloor+1$。

性质 5　如果将一棵有 n 个结点的完全二叉树按层次自顶向下,同一层自左向右连续给结点编号为 0,1,…,$n-1$,并简称编号为 i 的结点为结点 i($0\leqslant i\leqslant n-1$)。则对于编号为 i($0\leqslant i\leqslant n-1$)的结点:

- 若 $i=0$,则结点 i 为根结点,无双亲;若 $i>0$,则结点 i 的双亲为$\lfloor (i-1)/2 \rfloor$。
- 若 $2i+1<n$,则结点 i 的左子女结点为 $2i+1$。
- 若 $2i+2<n$,则结点 i 的右子女结点为 $2i+2$。
- 若结点编号 i 为奇数,且 $i\geqslant 1$,则它是双亲的左子女,其右兄弟为结点 $i+1$。
- 若结点编号 i 为偶数,且 $i<n$,则它是双亲的右子女,其左兄弟为结点 $i-1$。
- 离根最远的非叶结点的编号 $i=\lfloor n/2 \rfloor-1$ 或 $i=\lfloor (n-2)/2 \rfloor$。
- 结点 i 所在层次为$\lfloor \log_2(i+1) \rfloor+1$。

理解二叉树性质的要点:

(1) 二叉树的层次是其所有性质的核心。性质 1 是估计第 i 层最多可有 2^{i-1} 个结点,性质 2 就进一步估计所有 h 层最多有 2^h-1 个结点。

(2) 性质 3 给出二叉树中度为 0 的结点和度为 2 的结点之间的关系,即叶结点的结点数等于度为 2 的结点的结点数加 1。这个关系很有用。例如,对于一棵有 n 个结点的完全二叉树,应用这个性质,可得如下两个结论:

- 若 n 为奇数,则树中只有度为 0 和度为 2 的结点,且度为 0 的结点有$\lceil n/2 \rceil$个,度为 2 的结点有$\lfloor n/2 \rfloor$个。
- 若 n 为偶数,则树中除了度为 0 和度为 2 的结点,还有 1 个度为 1 的结点。

(3) 如果一棵二叉树是完全二叉树,可使用性质 4 根据结点个数 n 直接计算树的高度

h，即结点总的层次数，反之亦可根据 h 计算 n。性质 4 也适用于理想平衡树。

(4) 根据性质 5 可以得出以下结论。对完全二叉树中编号为 $i(0 \leqslant i \leqslant n-1)$的结点有：

- 若 $i \leqslant \lfloor (n-2)/2 \rfloor$，则编号为 i 的结点为分支结点，否则为叶结点。
- 若 n 为奇数，则每个分支结点都有左子女和右子女；若 n 为偶数，则编号最大的分支结点(编号为$(n-2)/2$)只有左子女，没有右子女，其余分支结点左、右子女都有。
- 除根结点外，若一个结点的编号为 i，则它的双亲结点的编号为$\lfloor (i-1)/2 \rfloor$。就是说，当 i 为偶数时，其双亲结点的编号为$(i-2)/2$，它是双亲结点的右子女；当 i 为奇数时，其双亲结点的编号为$(i-1)/2$，它是双亲结点的左子女。

4. 二叉树的顺序存储结构

这种存储结构一般用于完全二叉树。它使用一组地址连续的存储单元存储二叉树中的数据元素。基于二叉树的性质 5，把结点排在一个适当的线性序列中，并通过结点在这个序列中的相互位置反映出结点之间的逻辑关系。

深度为 d 的完全二叉树，除第 d 层外，其余各层中含有最大的结点个数，即每一层的结点个数恰为其上一层结点个数的两倍。

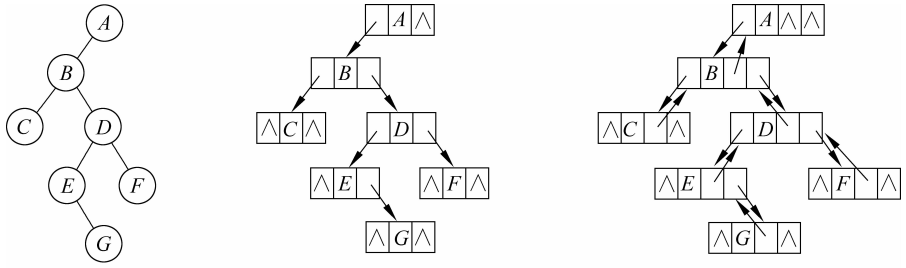
二叉树的顺序存储表示的结构定义(存放于 SqBTree.h 头文件内)如下：

```
#include<stdio.h>
#define maxSize 127                    //默认存储大小,按满二叉树定义
typedef char DataType;                 //假设元素数据类型为 char
typedef struct {
    DataType data[maxSize];            //存储数组
    int n;                             //当前结点个数
} SqBTree;
```

顺序存储结构对于一般二叉树不适用。对于一般的二叉树，采用链式存储结构。

5. 二叉树的链式存储结构

由于二叉树中结点包含有数据元素、左子女地址、右子女地址及双亲结点地址等信息，因此可以用二叉链表或三叉链表来存储二叉树，链表的头指针指向二叉树的根结点，如图 5-3 所示。



(a) 二叉树　(b) 对应二叉链表　(c) 对应三叉链表

图 5-3　二叉树的链表表示

二叉树的二叉链表存储表示的结构类型定义(存放于 BinTree.h 头文件内)如下：

```
#include<stdio.h>
#include<stdlib.h>
```

```
typedef int DataType;                    //结点数据类型
typedef struct Node {                    //二叉链表结点类型定义
    DataType data;                       //结点的数据
    struct Node * lchild, * rchild;      //结点的左、右子树指针
} BiTNode, * BinTree;                    //二叉链表定义
```

二叉树的三叉链表存储表示的结构类型定义(存放于 BinPTree.h 头文件内)如下:

```
typedef struct Node {                    //二叉树结点类型定义
    DataType data;                       //结点的数据
    struct Node * lchild, * rchild;      //结点的左、右子树指针
    struct Node * parent;                //双亲指针
} BiTPNode, * BinPTree;                  //三叉链表定义
```

在不同的存储结构中,实现二叉树的运算方法亦不同,具体应采用什么存储结构,除考虑二叉树的形态外还应考虑需要进行的运算。

6. 理解二叉树存储表示的要点

(1) 使用二叉链表查找某结点的左、右子女的时间代价为 $O(1)$,寻找其双亲结点的时间代价为 $O(n)$,因为要遍历二叉树,看谁的子女是它谁就是它的双亲结点。

(2) 为使查找双亲结点的时间代价也达到 $O(1)$,需要使用三叉链表存储二叉树,每个结点有 3 个指针:parent(双亲指针)、lchild(左子女指针)、rchild(右子女指针)。

7. 二叉树的基本运算

- void InitBinTree(BiTNode * & T);　　//初始化一棵根为 T 的二叉树
- void CreateBinTree(BiTNode * & T);　　//创建一棵根为 T 的二叉树
- BiTNode * LeftChild(BiTNode * p);　　//求结点 p 左子女结点地址
- BiTNode * RightChild(BiTNode * p);　　//求结点 p 右子女结点地址
- BiTNode * Parent(BiTNode * p);　　//求结点 p 的父结点地址
- bool GetData(BiTNode * p, TElemType& x);　　//返回结点 p 中存放的值
- void PrintBinTree(BiTNode * & T);　　//输出以结点 T 为根的子树
- int Height(BiTNode * & T);　　//求以结点 T 为根的子树高度
- void PreOrder(BiTNode * T);　　//先序遍历
- void InOrder(BiTNode * T);　　//中序遍历
- void PostOrder(BiTNode * T);　　//后序遍历
- void LevelOrder(BiTNode * T);　　//层次序遍历

5.2.2 选择题

1. 下面关于二叉树的叙述中正确的是(　　)。

   A. 二叉树中,度为 0 的结点个数等于度为 2 的结点个数加 1

   B. 二叉树中结点个数必大于 0

   C. 完全二叉树中,任何一个结点的度或者为 0,或者为 2

   D. 二叉树的度是 2

2. 在以下有关二叉树的叙述中正确的是(　　)。