

离散数学第三部分之二

树、模型 与 应用

■ 树的定义

定义1 连通无回路的无向图称为无向树，简称树，常用 T 表示。

在树中，悬挂结点称为树叶，度数大于1的结点称为分支点或内结点，悬挂边称为叶边。连通分支数大于1，且每个连通分支都是树的无向图称为森林。

例1 在图10-31中，(a)和(b)是树，(c)是森林。

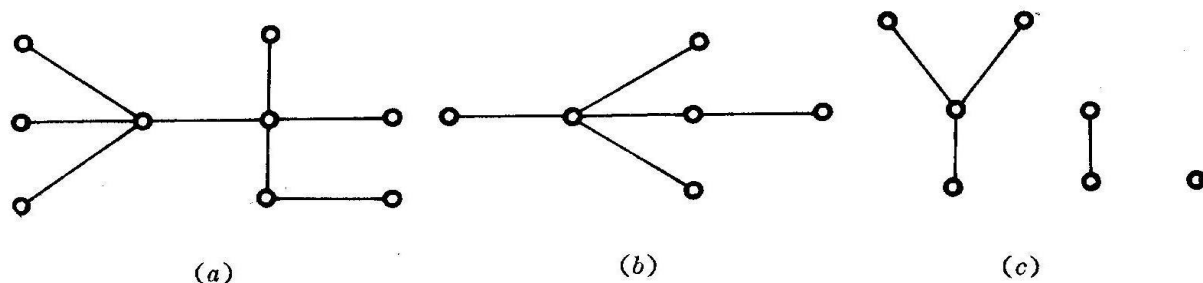
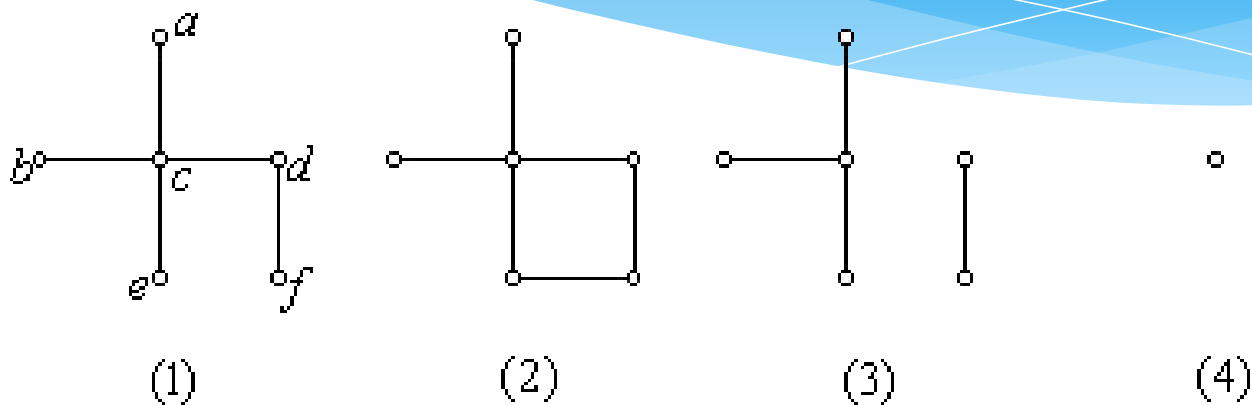


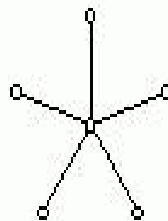
图10-31



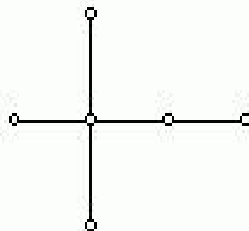
例、

上图中，(1)与(4)是树，(4)为平凡树。(2)有回路，不是树。(3)不连通，也不是树，但它有2个连通分支，每个连通分支都是树，故(3)是森林。

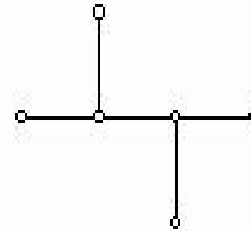
例、画出所有的 6个顶点的非同构的树。



T_1



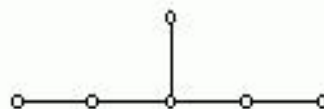
T_2



T_3



T_4



T_5



T_6



树的基本术语

1. 结点的度与树的度：树中某个结点的子树的个数称为该结点的**度**。树中各结点的度的最大值称为树的度,通常将度为 m 的树称为 **m 次树**。

2. **分支结点与叶结点**：度不为零的结点称为非终端结点,又叫**分支结点**。度为零的结点称为终端结点或**叶结点**。在分支结点中,每个结点的分支数就是该**结点的度**。如对于度为1的结点,其分支数为1,被称为**单分支结点**；对于度为2的结点,其分支数为2,被称为**双分支结点**,其余类推。



路径与路径长度：对于任意两个结点 k_i 和 k_j ,若树中存在一个结点序列 $k_i, k_{i1}, k_{i2}, \dots, k_{in}, k_j$,使得序列中除 k_i 外的任一结点都是其在序列中的前一个结点的后继,则称该结点序列为由 k_i 到 k_j 的一条路径,用路径所通过的结点序列 $(k_i, k_{i1}, k_{i2}, \dots, k_j)$ 表示这条路径。**路径的长度**等于路径所通过的结点数目减1(即路径上分支数目)。可见,路径就是从 k_i 出发“自上而下”到达 k_j 所通过的树中结点序列。显然,从树的根结点到树中其余结点均存在一条**路径**。



孩子结点、双亲结点和兄弟结点：在一棵树中,每个结点的后继,被称作该结点的**孩子结点**(或子女结点)。相应地,该结点被称作孩子结点的**双亲结点**(或父母结点)。具有同一双亲的孩子结点互为**兄弟结点**。进一步推广这些关系,可以把每个结点的所有子树中的结点称为该结点的**子孙结点**,从树根结点到达该结点的路径上经过的所有结点被称作该结点的**祖先结点**。



结点的层次和树的高度：树中的每个结点都处在一定的层次上。结点的层次从树根开始定义,根结点为第1层,它的孩子结点为第2层,以此类推,一个结点所在的层次为其双亲结点所在的层次加1。树中结点的最大层次称为**树的高度**(或树的深度)。

有序树和无序树：若树中各结点的子树是按照一定的次序从左向右安排的,且相对次序是不能随意变换的,则称为有序树,否则称为**无序树**。



■ 树的基本性质

- (1) 树必连通，但无回路（圈）。
- (2) n 个顶点的树必有 $n-1$ 条边。
- (3) 树 中任意两个顶点之间，恰有且仅有一条链（初等链）。
- (4) 树 连通，但去掉任一条边， 必变为不连通。
- (5) 树 无回路（圈），但不相邻的两个点之间加一条边，恰得到一个回路（圈）。



性质1 度为 m 的树中第 i 层上至多有 m^{i-1} 个结点,这里应有 $i \geq 1$ 。

证明(采用数学归纳法)

对于第一层,因为树中的第一层上只有一个结点,即整个树的根结点,而由 $i=1$ 代入 m^{i-1} ,得 $m^{i-1}=m^{1-1}=1$,也同样得到只有一个结点,显然结论成立。

假设对于第 $(i-1)$ 层($i > 1$)命题成立,即度为 m 的树中第 $(i-1)$ 层上至多有 m^{i-2} 个结点,则根据树的度的定义,度为 m 的树中每个结点至多有 m 个孩子结点,所以第 i 层上的结点数至多为第 $(i-1)$ 层上结点数的 m 倍,即至多为 $m^{i-2} \times m = m^{i-1}$ 个,这与命题相同,故命题成立。



性质2 高度为 h 的 m 次树至多有 $\frac{m^h-1}{m-1}$ 个结点。

证明：由树的性质2可知,第 i 层上最多结点数为 m^{i-1} ($i=1,2,\dots,h$),显然当高度为 h 的 m 次树(即度为 m 的树)上每一层都达到最多结点数时,整个 m 次树具有最多结点数,因此有:

整个树的最多结点数=每一层最多结点数之和
 $=m^0+m^1+m^2+\dots+m^{h-1} = \frac{m^h-1}{m-1}$ 。



性质 3 具有 n 个结点的 m 次树的最小高度为 $\lceil \log_m(n(m-1)+1) \rceil$ 。

证明：设具有 n 个结点的 m 次树的高度为 h ，若在该树中前 $h-1$ 层都是满的，即每一层的结点数都等于 m^{i-1} 个 ($1 \leq i \leq h-1$)，第 h 层 (即最后一层) 的结点数可能满，也可能不满，则该树具有最小的高度。其高度 h 可计算如下：



根据树的性质3可得: $\frac{m^{h-1}-1}{m-1} < n \leq \frac{m^h-1}{m-1}$

乘(m-1)后得: $m^{h-1} < n(m-1)+1 \leq m^h$

以m为底取对数后得: $h-1 < \log_m(n(m-1)+1) \leq h$

即 $\log_m(n(m-1)+1) \leq h < \log_m(n(m-1)+1)+1$

因h只能取整数,所以

$$h = \lceil \log_m(n(m-1)+1) \rceil$$

结论得证。



例 含n个结点的三次树的最小高度是多少？

解： 设含n个结点的(为完全三次树时高度最小)的三次树的最小高度为h,则有：

$$1+3+9+\dots+3^{h-2} < n \leq 1+3+9+\dots+3^{h-1}$$

$$(3^{h-1}-1)/2 < n \leq (3^h-1)/2$$

$$3^{h-1} < 2n+1 \leq 3^h$$

$$\text{即： } h = \lceil \log_3(2n+1) \rceil$$



由定义不难得出，树一定是连通的简单平面图。

定理1 设 $T=\langle V, E \rangle$ 是 n 阶 m 条边的无向图，则下面命题等价：

- (1) T 是树。
- (2) T 无回路且 $m=n-1$ 。
- (3) T 连通且 $m=n-1$ 。
- (4) T 是连通的，且 T 的每条边都是割边。
- (5) T 无回路，但任意增加一条边产生惟一一条回路。
- (6) T 中每对结点间恰有一条路。



证明 (1) \Rightarrow (2): 只需证 $m=n-1$ 。因为 T 是树, 所以 T 是连通平面图, 且 T 只有一个无限面, 由欧拉公式得 $n-m+1=2$, 即 $m=n-1$ 。

(2) \Rightarrow (3): 只需证 T 连通。若 T 不连通, 不妨设有 k 个连通分支($k \geq 2$), 分别记为 T_1 、 T_2 、...、 T_k , T_i 的结点数和边数分别记为 n_i 和 m_i ($i=1, 2, \dots, k$), 则有 $n = \sum_{i=1}^k n_i$, $m = \sum_{i=1}^k m_i$, $m_i = n_i - 1$ ($i=1, 2, \dots, k$)。于是, $m = \sum_{i=1}^k m_i = \sum_{i=1}^k (n_i - 1) = n - k < n - 1$, 矛盾。所以 T 连通。



(3) \Rightarrow (4): 只需证 T 的每条边都是割边。任取 T 的边 e , 则 $T - \{e\}$ 的边数为 $m-1$, 结点数为 n , 于是 $m-1 = n-2 < n-1$, 所以 $T - \{e\}$ 是不连通的, 因而 e 是割边。故 T 的每条边都是割边。

(4) \Rightarrow (5): 由于 T 的每条边都是割边, 因而 T 中无回路。在 T 中添加新边 $[u, v]$, 由 T 是连通的, 则存在 u 到 v 的一条路 Γ , 于是 $\Gamma + [u, v]$ 构成一回路。若得到的回路不惟一, 不妨记为 Γ_1 和 Γ_2 , 则 $\Gamma_1 - \{[u, v]\}$ 与 $\Gamma_2 - \{[u, v]\}$ 构成一回路, 其上的边都不是割边, 矛盾。故在 T 中任意增加一条边产生惟一一条回路。



(5) \Rightarrow (6): 若 T 中存在两结点 u 和 v 之间不存在路, 则添加边 $[u, v]$ 不产生回路, 矛盾。所以 T 中任意两个结点之间存在路。若 u 和 v 之间存在两条不同的路 Γ_1 和 Γ_2 , 则 Γ_1 和 Γ_2 形成回路, 矛盾。所以 T 中每对结点间恰有一条路。

(6) \Rightarrow (1): 因为 T 中每对结点间恰有一条路, 所以 T 是连通的。若 T 有回路, 则回路上的任意两个结点之间存在两条路, 矛盾, 所以 T 没有回路。因此, T 是树。




由于树少一边就不连通，多一边就有回路，所以树是以“最经济”的方式把各结点连接起来的图。因此，它可以用作典型的数据结构，各类网络的主干网也通常都是树结构。

定理2 若 $T = \langle V, E \rangle$ 是树，且 $|V| \geq 2$ ，则 T 中至少存在两片树叶。

证明 由 T 是树得， $|E| = |V| - 1$ 。设 T 有 k 片树叶，应用握手定理可得 $2|E| = 2(|V| - 1) = \sum_{v \in V} d(v) = k + 2(|V| - k)$ ，于是 $k \geq 2$ 。



- 
- 例、(1) 一棵树有7片叶，3个3度顶点，其余都是4度顶点，求4度顶点多少个？
- (2) 一棵树有2个 4度顶点， 3个3度顶点，其余都是树叶，求这棵树共有多少个顶点？



■ 根树

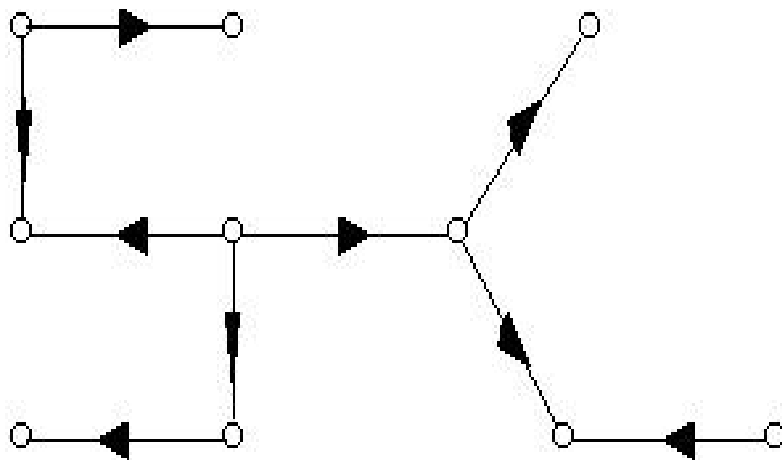
前面我们讨论的树，都是一个无向图，下面我们讨论有向图的树。

定义 如果一个有向图在不考虑边的方向时是一棵数，那么，这个有向图称为有向树。

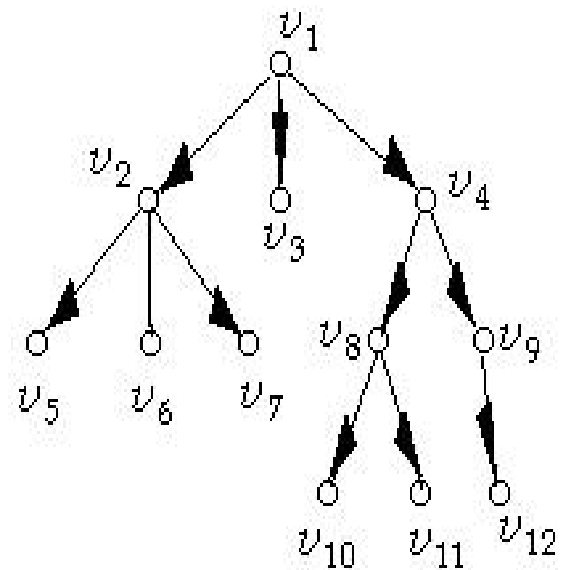


定义 一棵有向树，如果恰有一个结点的入度为0，其余所有结点的入度都为1，则称为根树。入度为0的结点称为根，出度为0的结点称为叶，出度不为0的结点称为分支点或内点。





(a)



(b)

有向树和根树



例如上图 (b) 表示的是一棵根树，其中 v_1 为根， v_2 ， v_4 ， v_8 ， v_9 为分枝点，其余结点为叶。

在根树中，任意一个结点 v 的层次，就是从根到该结点的单向通路的长度。

在图 (b) 中，有三个结点的层次为1，有五个结点的层次为2，有三个结点的层次为3。

从根树的结构可以看出，树中每一个结点可以看作是原来树中的某一棵子树的根，由此可知，根树亦可递归定义为：

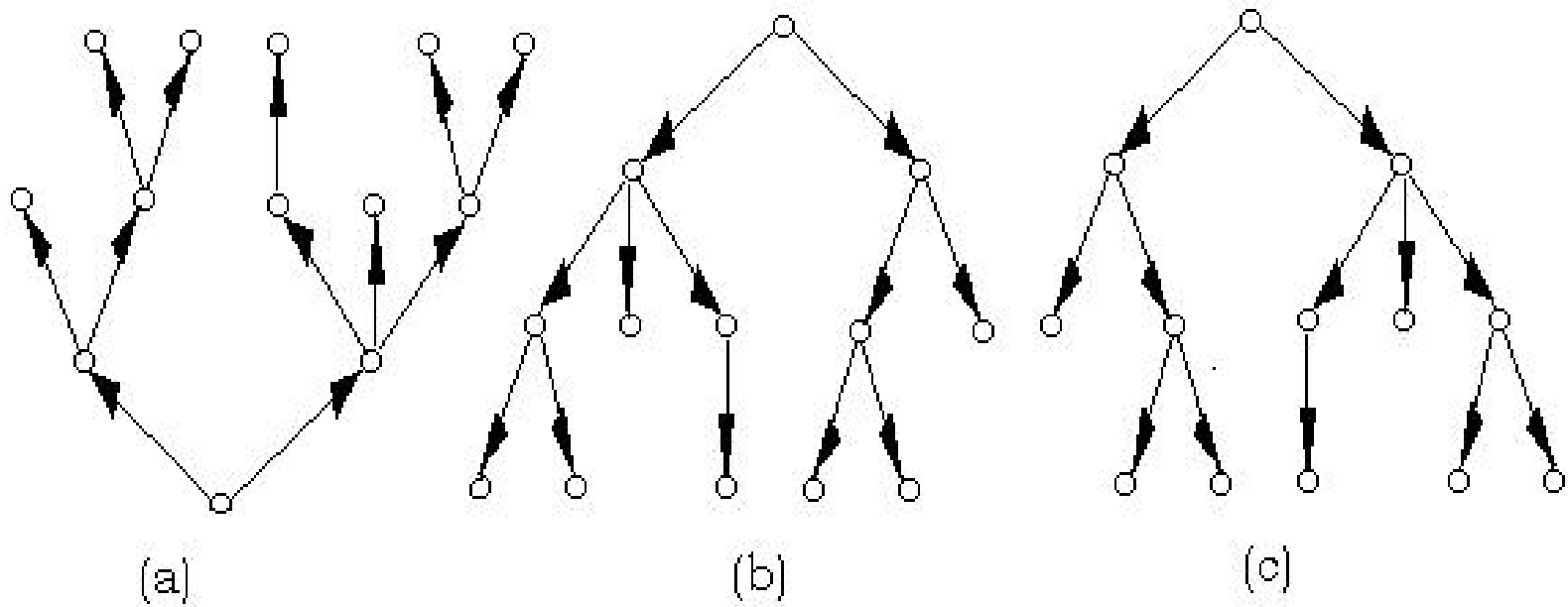


定义 根树包括一个或多个结点，这些结点中某一个称为根，其它所有结点被分成有限个子根树。

这个定义把 n 个结点的根树用结点数少于 n 的根树来定义，最后得到每一棵都是一个结点的根树，它就是原来那棵树的树叶。

对于一棵根树，如果用图形来表示，可以有树根在下或树根在上的两种画法。





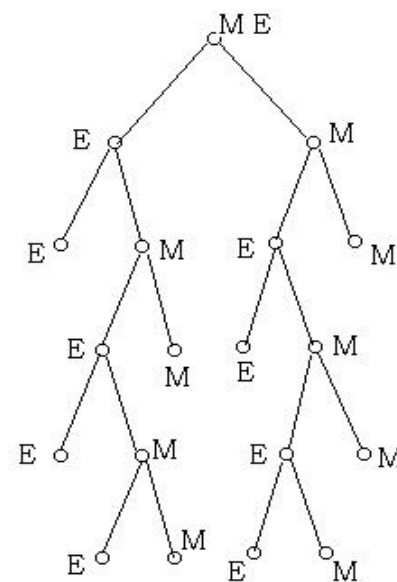
设 a 是根树中的一个分枝点，若从 a 到 b 有一条边，即 $a < b$ ，则结点 a 称为结点 b 的“父亲”，而结点 b 称为结点 a 的“儿子”。假若 $a < c$ ，则结点 a 称为结点 c 的“祖先”，而结点 c 称为结点 a 的“后裔”。同一个分枝点的“儿子”称为“兄弟”。



定义 在根树中，若每一个结点的出度小于或等于 m ，则这棵树称为 m 叉树。如果每一个结点的出度恰好等于 m 或零，则这棵树称为完全 m 叉树。若所有的树叶层次相同，则这棵树称为正则 m 叉树，若 $m=2$ 时，称为二叉树。

有许多实际问题可用二叉树或 m 叉树来表示。





冬



在树的实际应用中，我们经常研究完全 m 叉树。

定理 设有完全 m 叉树，其树叶数为 t ，分枝点数为 i ，则 $(m-1)i=t-1$ 。

证明 若把 m 叉树当作是每局有 m 位选手参加比赛的单淘汰赛计划表，树叶数为 t 表示参加比赛的选手数，分枝点数为 i 表示比赛的局数，

因为每局比赛将淘汰 $(m-1)$ 位选手，比赛的结果共淘汰 $(m-1)i$ 位选手，最后剩下一个冠军，

因此 $(m-1)i+1=t$

即 $(m-1)i=t-1$ 。



例. 设有28盏灯，拟共用一个电源插座，问需用多少块具有四种插座的接线板。

解 将四叉树每个分枝点看作是具有四插座的接线板，树叶看作电灯，则 $(4-1) i = 28-1$, $i=9$ ，所以需要九块具有四插座的接线板。



例 设有一台计算机，它有一条加法指令，可计算三个数的和，如果要计算九个数的和，至少要执行几次加法命令。

解 若把九个数看作是完全三叉树的九片树叶，则有 $(3-1) \cdot i = 9-1$, $i=4$ ，所以，需要执行四次加法指令。



■ 二叉树概念和性质

二叉树

二叉树也称为二次树或二分树,它是有限的结点集合,这个集合或者是空,或者由一个根结点和两棵互不相交的称为左子树和右子树的二叉树组成。

二叉树的定义是一种递归定义。

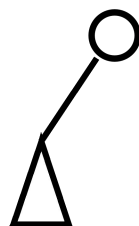
二叉树有五种基本形态, 如下图所示, 任何复杂的二叉树都是这五种基本形态的复合。



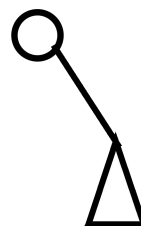
(a)



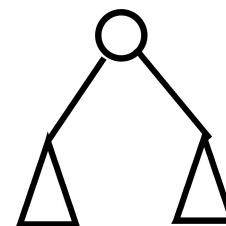
(b)



(c)

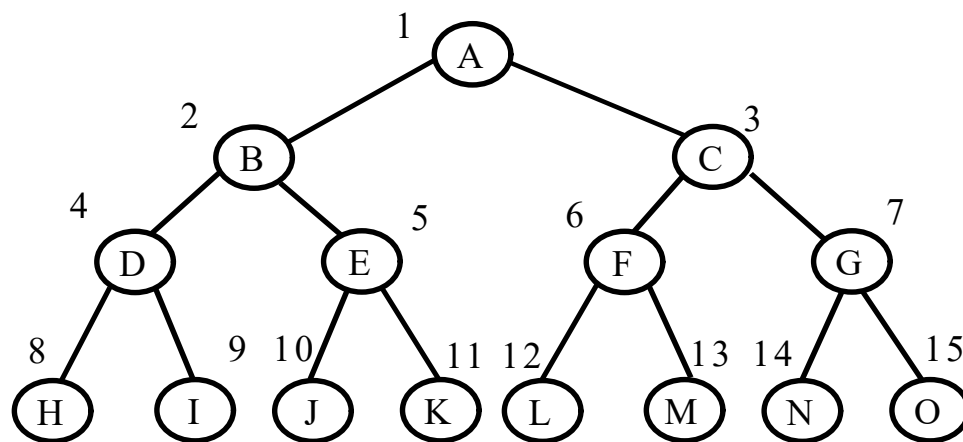


(d)



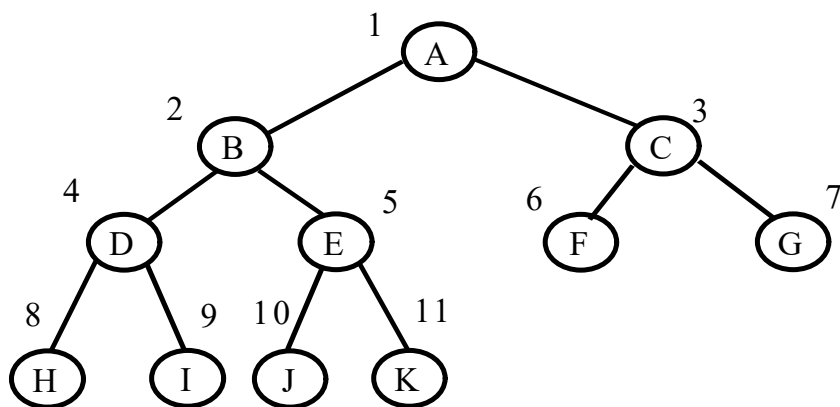
(e)

在一棵二叉树中, 如果所有分支结点都有左孩子结点和右孩子结点, 并且叶结点都集中在二叉树的最下一层, 这样的二叉树称为**满二叉树**。下图所示就是一棵满二叉树。可以对满二叉树的结点进行连续编号, 约定编号从树根为1开始, 按照层数从小到大、同一层从左到右的次序进行。图中每个结点外边的数字为对该结点的编号。



满二叉树

若二叉树中最多只有最下面两层的结点的度数可以小于2, 并且最下面一层的叶结点 都依次排列在该层最左边的位置上, 则这样的二叉树称为完全二叉树。如下图所示为一棵完全二叉树。同样可以对完全二叉树中每个结点进行连续编号, 编号的方法同满二叉树相同。图中每个结点外边的数字为对该结点的编号。



完全二叉树


性质1 非空二叉树上叶结点数等于双分支结点数加1。

证明：设二叉树上叶结点数为 n_0 ,单分支结点数为 n_1 ,双分支结点数为 n_2 ,则总结点数 $=n_0+n_1+n_2$ 。在一棵二叉树中,所有结点的分支数(即度数)应等于单分支结点数加上双分支结点数的2倍,即总的分支数 $=n_1+2n_2$ 。

由于二叉树中除根结点以外,每个结点都有惟一的一个分支指向它,因此二叉树中有：**总的分支数=总结点数-1**。

由上述三个等式可得： $n_1+2n_2=n_0+n_1+n_2-1$

即： $n_0=n_2+1$



性质2 非空二叉树上第 i 层上至多有 2^{i-1} 个结点,这里应有 $i \geq 1$ 。

由树的性质2可推出。


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

由树的性质3可推出。

性质4 对完全二叉树中编号为 i 的结点($1 \leq i \leq n, n \geq 1, n$ 为结点数)有:

(1)若 $i \leq \lfloor n/2 \rfloor$,即 $2i \leq n$,则编号为 i 的结点为分支结点,否则为叶子结点。

(2)若 n 为奇数,则每个分支结点都既有左孩子结点,也有右孩子结点;若 n 为偶数,则编号最大的分支结点(编号为 $\lfloor n/2 \rfloor$)只有左孩子结点,没有右孩子结点,其余分支结点都有左、右孩子结点。



(3)若编号为 i 的结点有左孩子结点,则左孩子结点的编号为 $2i$; 若编号为 i 的结点有右孩子结点,则右孩子结点的编号为 $(2i+1)$ 。

(4)除树根结点外,若一个结点的编号为 i ,则它的双亲结点的编号为 $\lfloor i/2 \rfloor$,也就是说,当 i 为偶数时,其双亲结点的编号为 $i/2$,它是双亲结点的左孩子结点,当 i 为奇数时,其双亲结点的编号为 $(i-1)/2$,它是双亲结点的右孩子结点。

■ 树的遍历问题

对于二叉树，一个十分重要的问题是访问树的结点，这就是二叉树的遍历问题。下面介绍二叉树遍历的3种方法：

(1) 先序遍历法：访问树根、先序遍历左子树、先序遍历右子树；

(2) 中序遍历法：中序遍历左子树、访问树根、中序遍历右子树；

(3) 后序遍历法：后序遍历左子树、后序遍历右子树、访问根树。



■ 中序遍历

中序遍历二叉树算法的定义：
若二叉树为空，则空操作；
否则

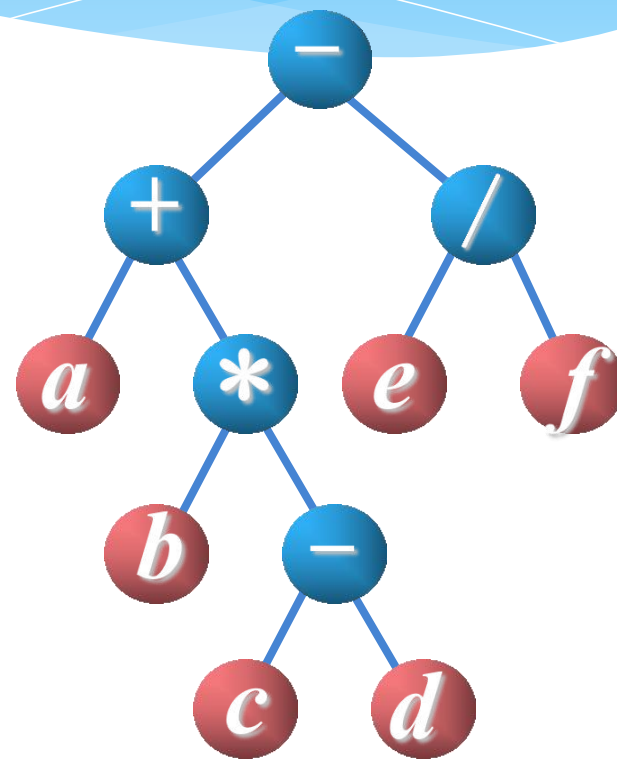
中序遍历左子树 (L)；

访问根结点 (V)；

中序遍历右子树 (R)。

遍历结果

$a + b * c - d - e / f$



前序遍历二叉树算法的定义：

若二叉树为空，则空操作；

否则

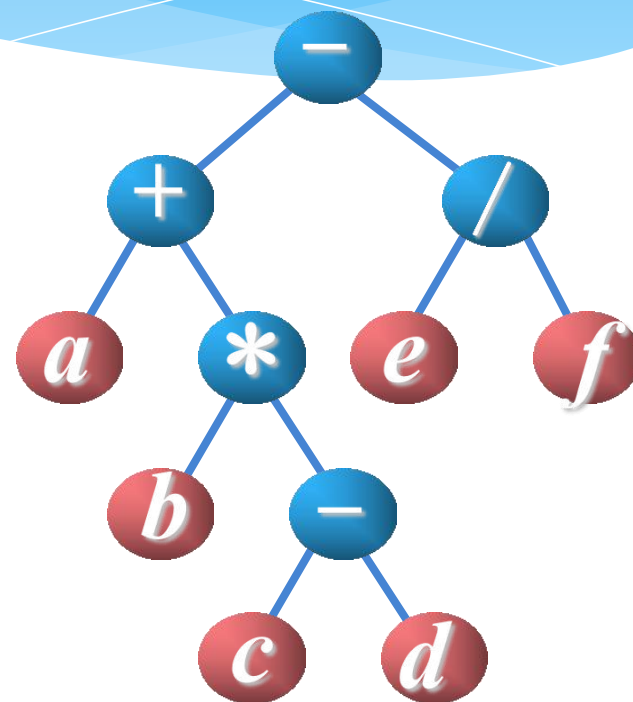
访问根结点 (V)；

前序遍历左子树 (L)；

前序遍历右子树 (R)。

遍历结果

$- + a * b - c d / e f$



后序遍历二叉树算法的定义：

若二叉树为空，则空操作；

否则

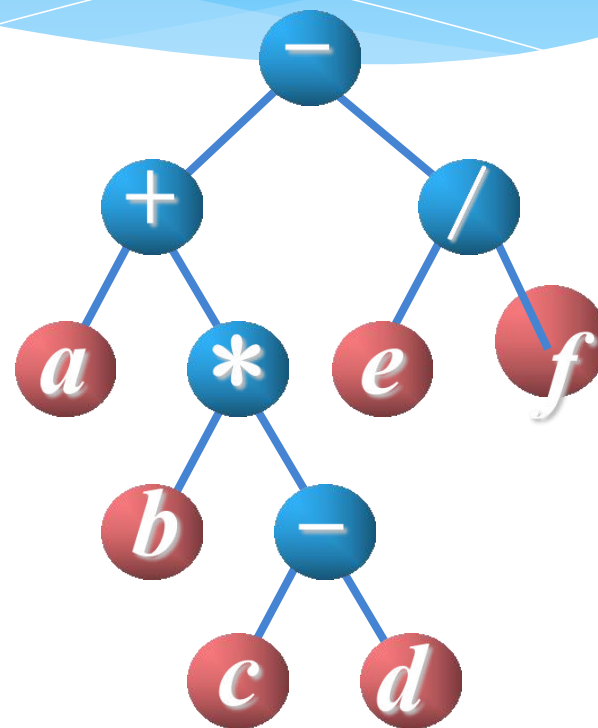
后序遍历左子树 (L)；

后序遍历右子树 (R)；

访问根结点 (V)。

遍历结果

$a\ b\ c\ d\ -\ *\ +\ e\ f\ /\ -$

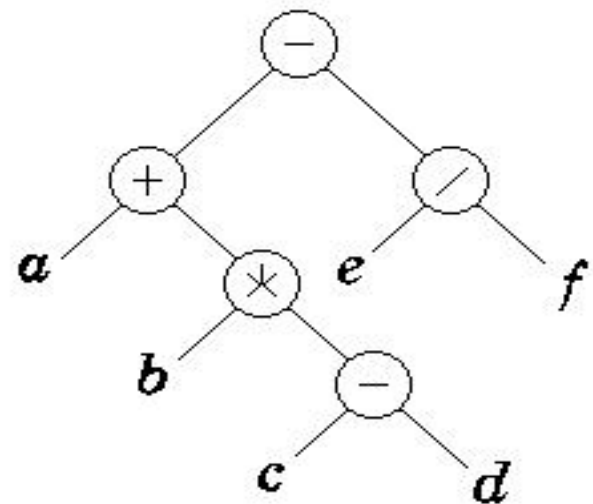


■ 层序遍历

按自上而下，每层从左到右顺序访问结点。

例如：右图的遍历结果：

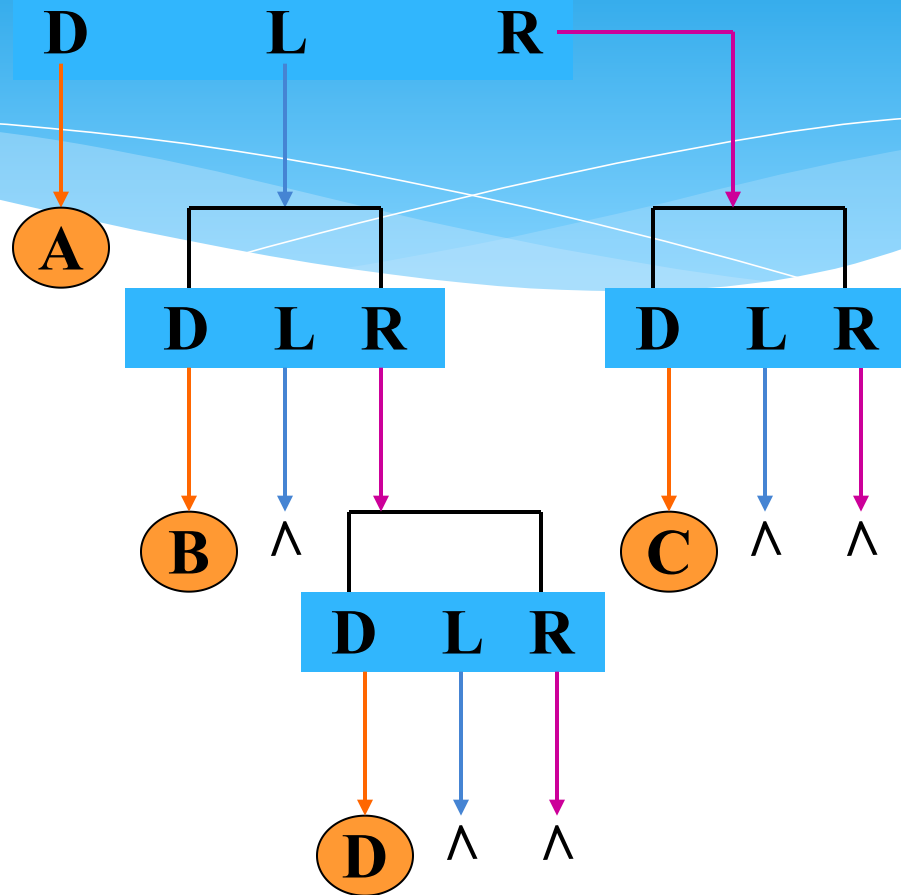
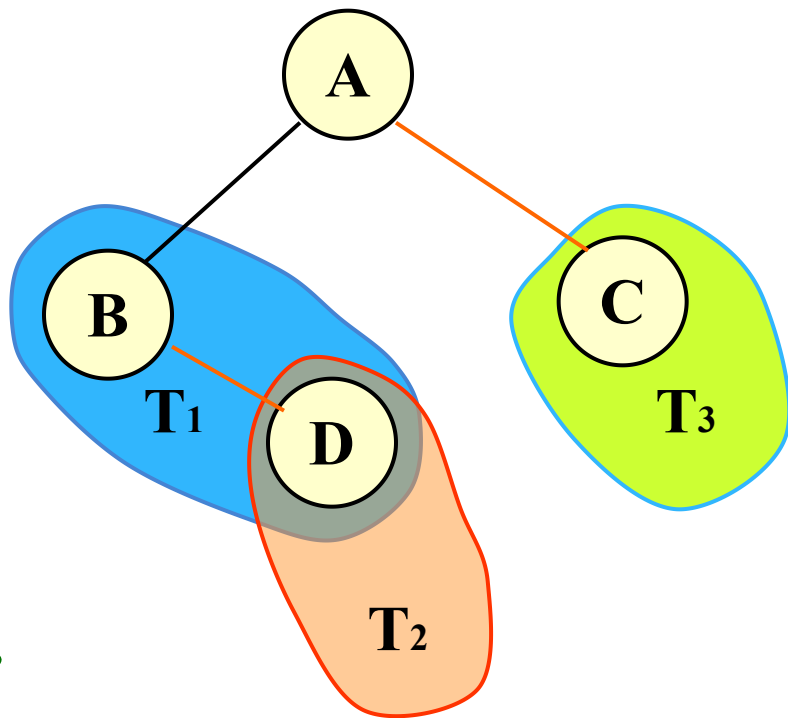
*- + / a * e f b - c d*



前序遍历 (D L R) **ABDC**

中序遍历 (L D R) **BDAC**

后序遍历 (L R D) **DBCA**

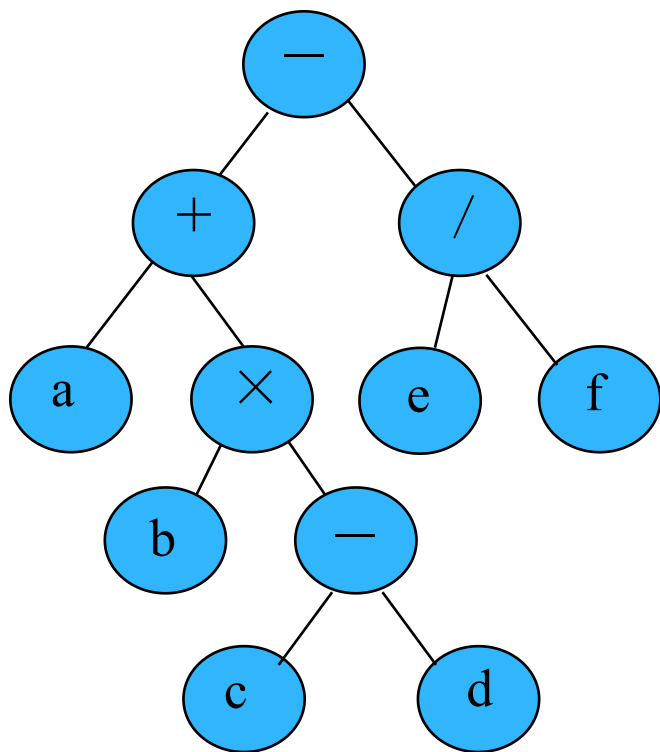


以前序遍历D L R为例演示遍历过程



遍历二叉树

例：表达式 $a+b \times (c-d)-e/f$



遍历结果:

中序: $a+b \times c-d-e/f$

后序 (逆波兰式):

$abcd - \times + ef / -$

先序: $- +a \times b - cd / ef$

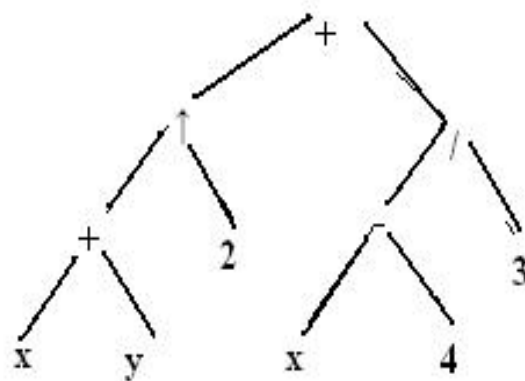


* 前缀、后缀和中缀记法

一些运算表达式可以用根树表示，内点表示运算关系，叶子表示变量或者数字。运算均作用在左子树和右子

例 表达式 $((x+y) \uparrow 2)$

构造方法：自下往



十么



对一个表达式的二叉树的中序遍历产生原来的表达式，因此，上述带有括号的表达式为中缀形式。此时，当以前序遍历表达式的根树时，即获得其前缀形式，其前缀形式的表达式称为 波兰记法。于此对应的是后缀形式（逆波兰记法）。

例： $((x+y) \uparrow 2) + ((x-4)/3)$ 的前缀形式是：
 $+ \uparrow + x y 2 / - x 4 3$



* 特别地

在前缀表达式里，二元运算符在俩个运算对象之前，因此，可以从右向左地求前缀形式的表达式的值。

计算方法为：当遇到一个运算符时，对其右边的两个相邻对象进行相应的计算。

例：前缀表达式”+-* 2 3 5 / ↑ 2 3 4”的值



计算步骤如下

- * $2 \uparrow 3 = 8$
- * $8 / 4 = 2$
- * $2 \times 3 = 6$
- * $6 - 5 = 1$
- * $1 + 2 = 3$




- * 在表达式的后缀形式里，二元运算是在它的两个运算对象之后，所以，为了计算一个后缀形式的表达式的值，就从左向右执行：每当一个运算符跟在两个运算对象后面时，就执行这个运算。

例：计算后缀表达式的值：

7 2 3 * - 4 ↑ 9 3 / +



- 
- ① $2 \times 3 = 6$
 - ② $7 - 6 = 1$
 - ③ $1 \times 1 \times 1 \times 1 = 1$
 - ④ $9 / 3 = 3$
 - ⑤ $1 + 3 = 4$



本章总结

- * 树的概念
- * 二叉树的定义
- * 树的不同遍历和计算
- * 树的高度

