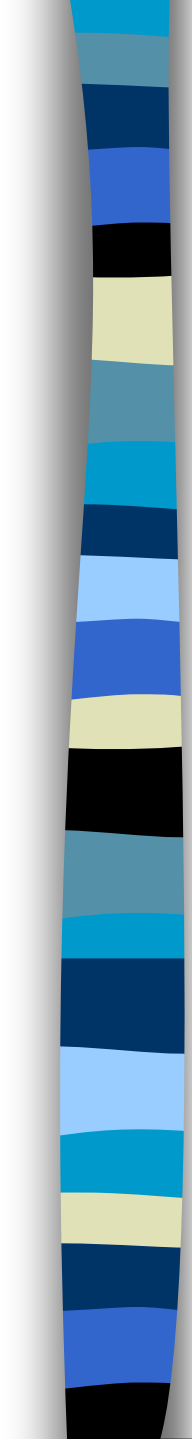




第十六章 树



树是图论中的一个重要概念。

树在许多学科中得到了广泛的应用。

树在计算机的数据处理、数据压缩、数据编码等领域有着重要的应用。

本章学习树的基本概念和性质

1、无向树及生成树

2、根树及其应用



16.1 树及其性质

定义16.1（树和森林）

连通且无回路的无向图称为**无向树**，简称为**树**，常用**T**表示树。

平凡图为树，称为**平凡树**。

非连通且每个连通分支是树的无向图称为**森林**。

T中度数为1的顶点（悬挂顶点）称为**树叶**，度数大于1的顶点称为**分支点**。

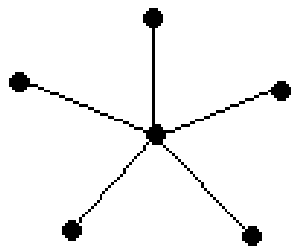
称只有一个分支点，且分支点的度数为 $n-1$ 的 n （ $n \geq 3$ ）阶的树为**星形图**，称唯一的分支点为**星心**。



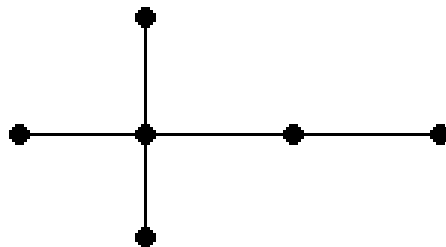
树的性质（定理16.1，定理16.2）

- （1）树中任意两个顶点之间存在唯一的路径；
- （2） $m=n-1$ ，其中 m 和 n 分别为树的边数和顶点数；
- （3）树是连通的且任何边均为桥，即在树中删去任何一边后就不连通；
- （4）在树中任何两个不同的顶点之间加一条新边后，得到且仅得到一个含新边的圈。
- （5）非平凡树中至少有两片树叶。

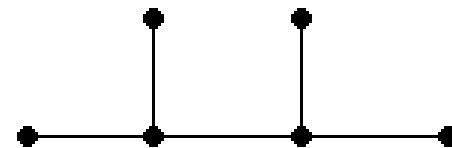
例16.1 画出6阶所有的非同构的无向树。



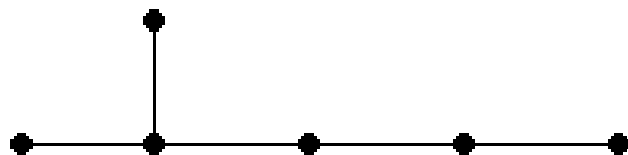
$(5, 1, 1, 1, 1, 1)$



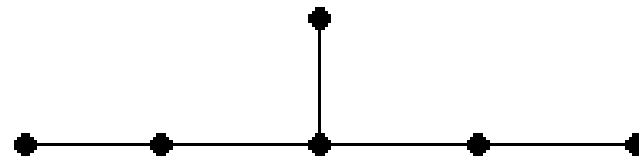
$(4, 2, 1, 1, 1, 1)$



$(3, 3, 1, 1, 1, 1)$



$(3, 2, 2, 1, 1, 1)$



$(3, 2, 2, 1, 1, 1)$



$(2, 2, 2, 2, 1, 1)$



16. 2 生成树

什么是生成子图？

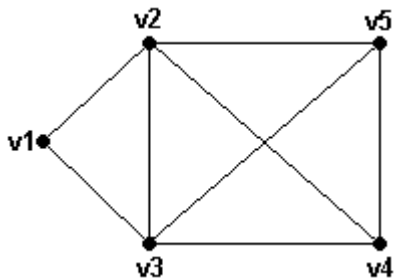
设 $G=\langle V, E \rangle$ 和 $G'=\langle V', E' \rangle$ 是两个图，若 $G' \subseteq G$ 且 $V'=V$ ，则称 G' 是 G 的生成子图或支撑子图。



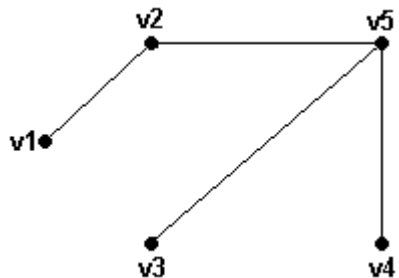
定义16.2 (生成树) 若 T 是无向图 G 的生成子图并且是树，则称 T 为 G 的生成树或支撑树。

设 G 的一棵生成树为 T ，则 T 中的边称为 T 的树枝，在 G 中而不在 T 中的边称为 T 的弦。

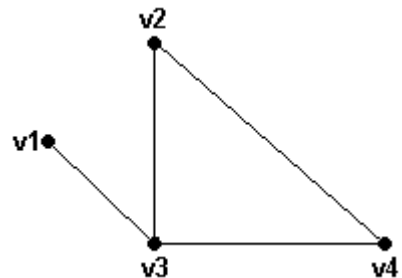
T 的所有弦的集合的导出子图称为 T 的余树，记为 \bar{T} 。



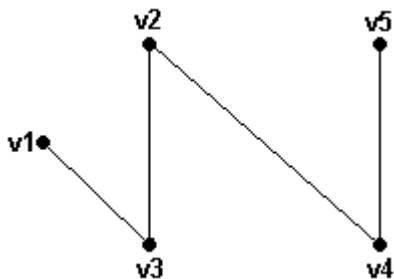
(1)



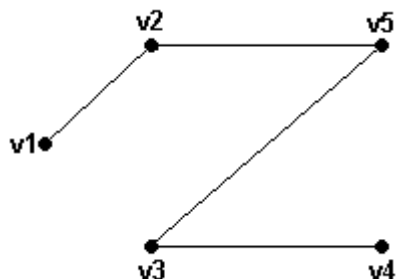
(2)



(3)



(4)



(5)

(2)、(4)、(5) 为 (1) 的生成树

(3) 为 (2) 的余树

(4) 和 (5) 互为余树

说明:

(1) 图的生成树可能不唯一。

(2) 生成树的余树不一定是树，也不一定连通。



定理16.3 无向图 G 具有生成树，当且仅当 G 是连通图。

推论1 设 G 为 n 阶 m 条边的无向连通图，则 $m \geq n-1$ 。

推论2 设 G 为 n 阶 m 条边的无向连通图， T 为 G 的生成树，则 T 的余树中含 $m-n+1$ 条边（即 T 有 $m-n+1$ 条弦）。

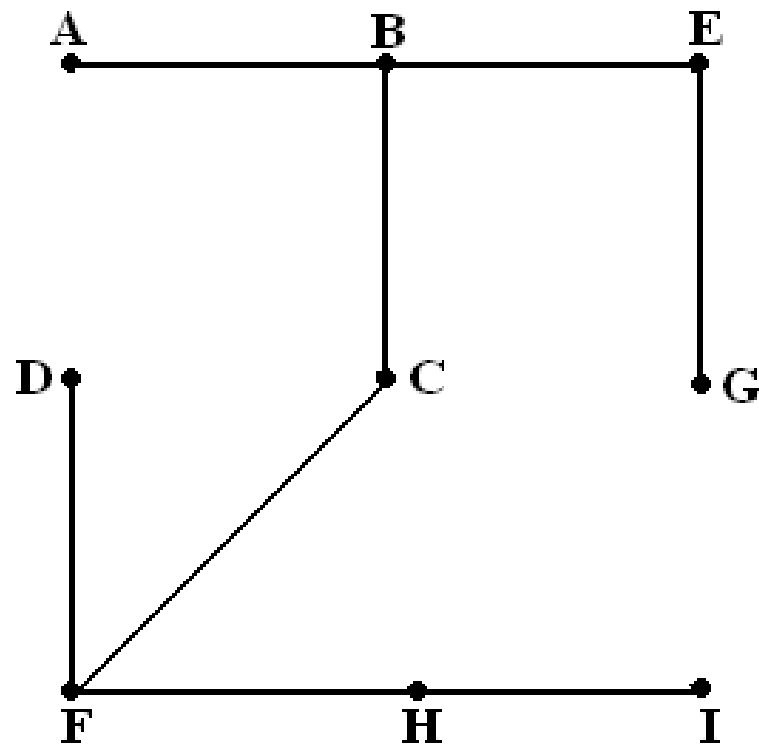
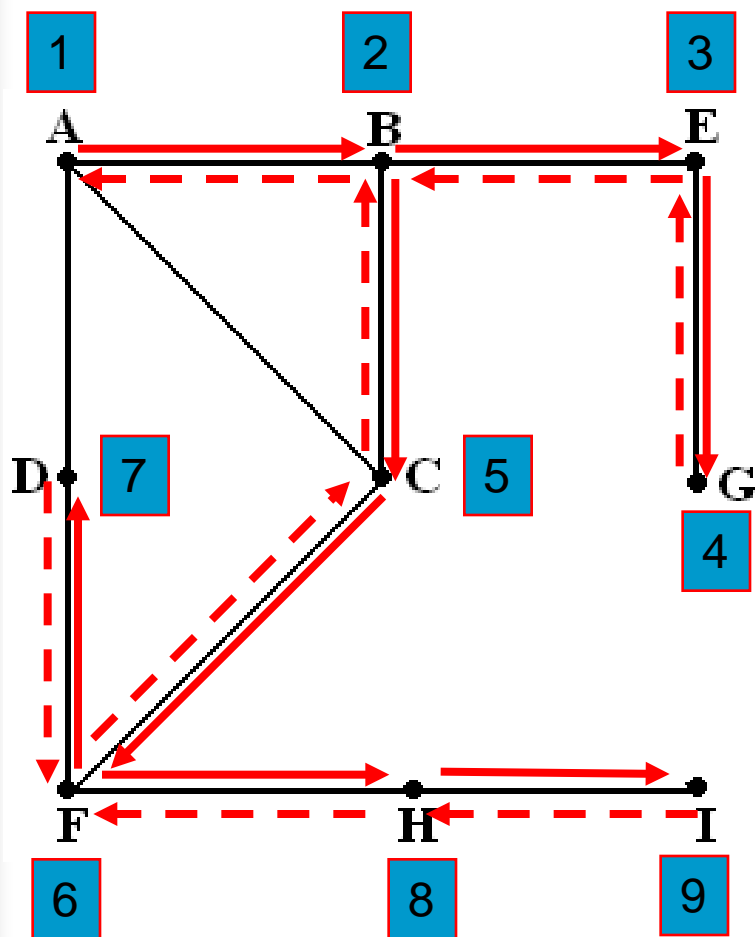
生成树的生成算法

(1) **破圈法**：若 G 中存在回路，删除回路上任意一条边，不影响图的连通性。若还有回路，就在删除回路上的一条边，直到图中无回路为止。最后得到的图是连通无回路的 G 的生成树。

(2) **深度优先搜索法**

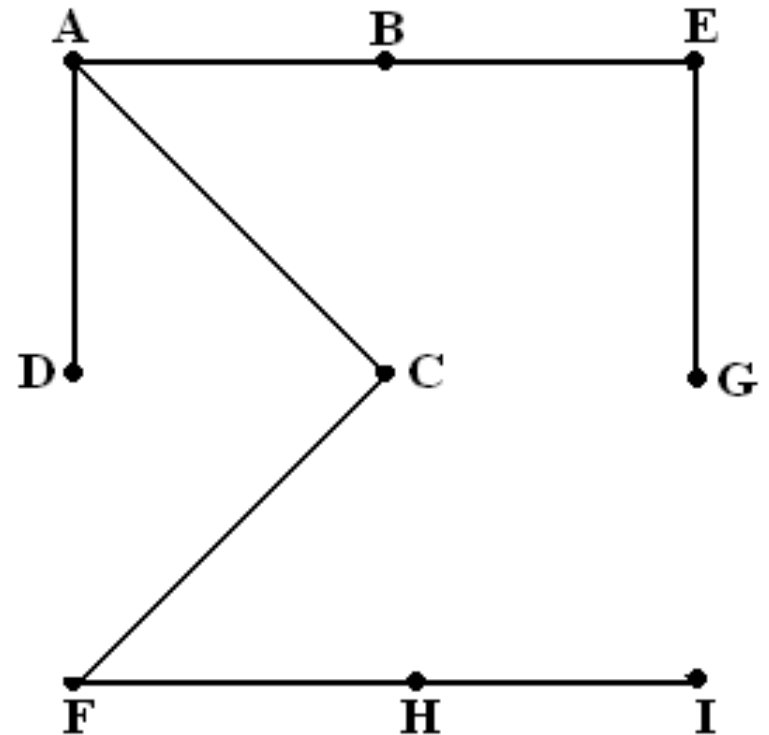
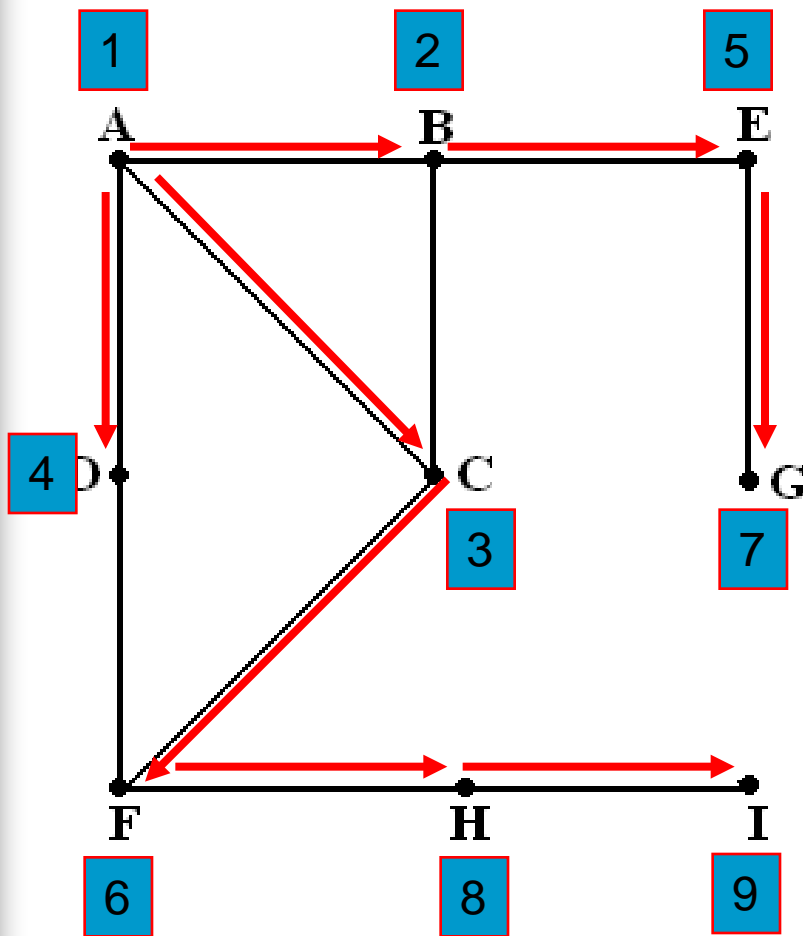
(3) **广度优先搜索法**

例 利用深度优先搜索法生成下图的生成树



深度优先生成树

例 利用广度优先搜索法生成下图的生成树



广度优先生成树



定义16.5（最小生成树） 设 $G=\langle V, E, W \rangle$ 是一无向连通带权图， T 是 G 的一棵生成树， T 的各边权之和称为 T 的**权**，记作 **$W(T)$** 。

在 G 的所有生成树中权最小的树称为 G 的**最小（代价）生成树**（*Minimum-cost Spanning Tree*）。

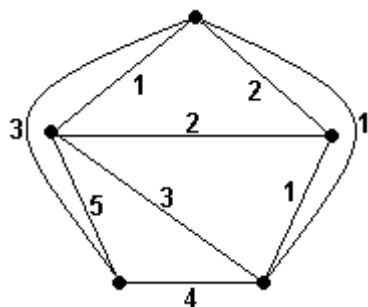
求最小生成树的克鲁斯卡尔 (Kruskal) 算法

----避圈法

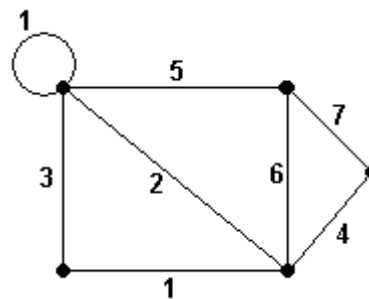
设 n 阶无向连通带权图 $G=\langle V, E, W \rangle$

- (1) 在 G 中选取最小权的边（非环），记作 e_1
- (2) 在 G 中选取第 i 条边（ $i=2, 3, \dots, n-1$ ），记作 e_i ，该边使得 e_1, e_2, \dots, e_i 不能构成回路，并且 e_i 是满足该条件的权最小的一条边。

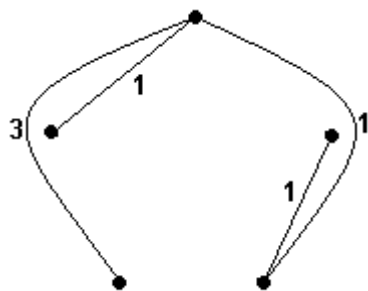
例16.3 求下图中最小生成树。



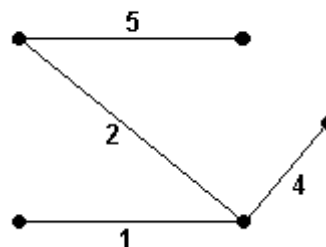
(1)



(2)



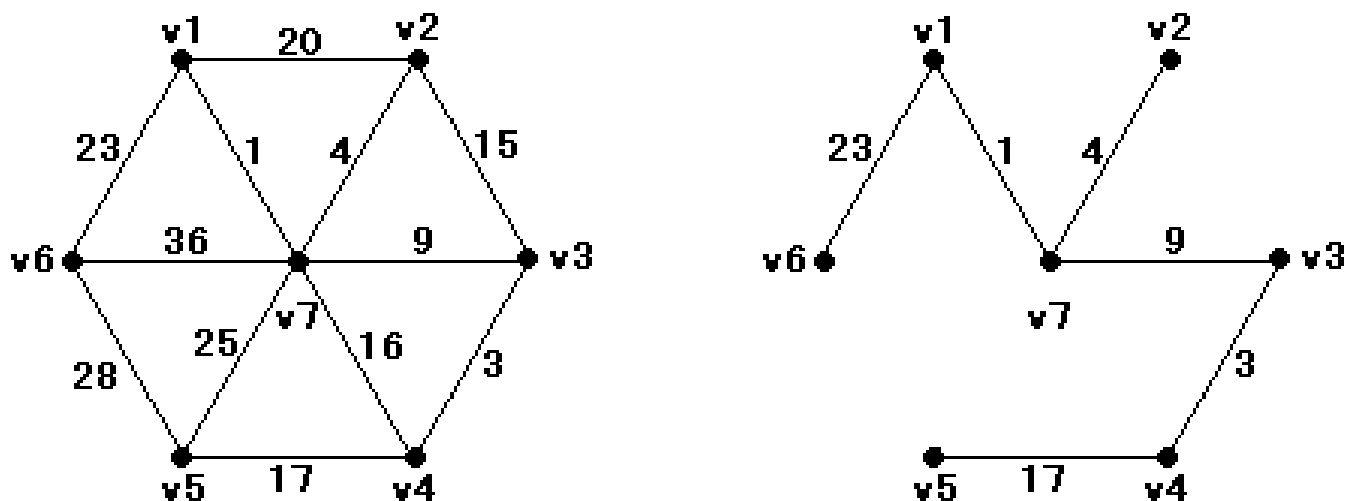
(3)



(4)

(1) (2) 的最小生成树分别为 (3) (4) , 权分别为6, 12。

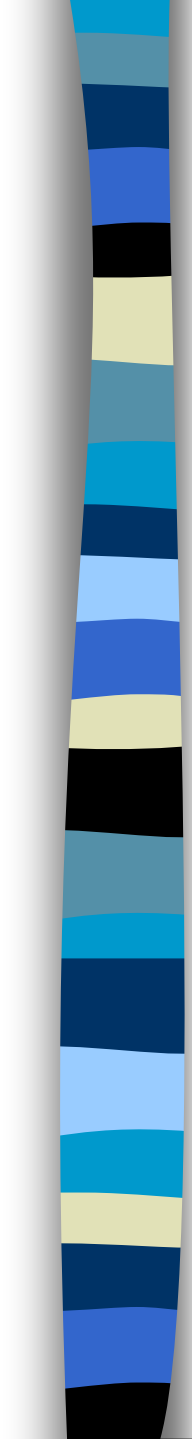
例：下图表示某七个城市 $v_1, v_2, v_3, v_4, v_5, v_6, v_7$ 及预先预算出的它们之间的一些直接通信线路的造价，试给出一个设计方案，使得各城市之间能够通信，而又使总造价最小。



解：易见所求为该图的一棵最小生成树，如图所示
总造价为57



16.3 根树及其应用



定义（有向树） 设 D 是有向图，如果 D 的基图是无向树，则称 D 为**有向树**。

在有向树中最重要的是**根树**。

定义16.6（根树） 一棵非平凡的有向树，如果恰有一个顶点的入度为0，其余所有顶点的入度均为1，则称该树为**根树**。

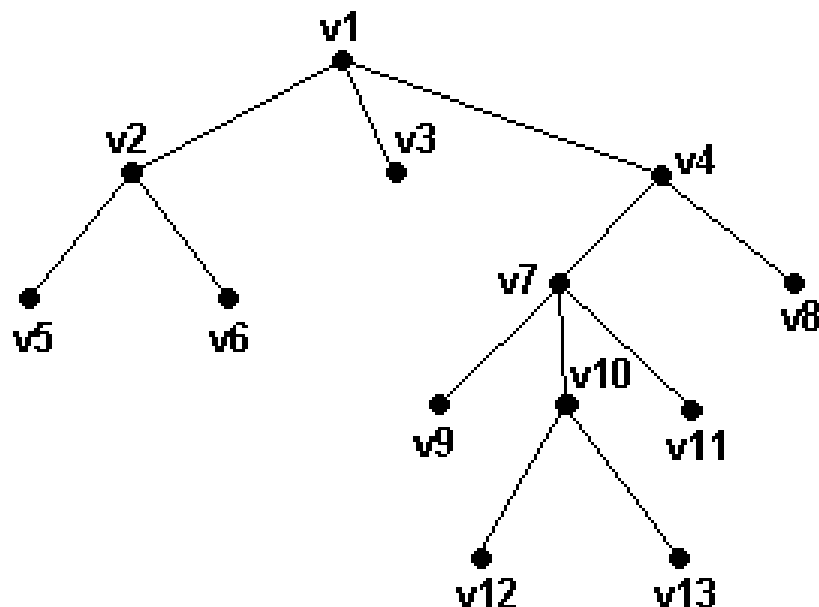
入度为0的顶点称为**树根**，入度为1出度为0的顶点称为**树叶**，入度为1出度不为0的点称为**内点**，内点和树根统称为**分支点**。

树根到一个顶点的有向通路的长度称为该顶点的**层数**。

层数最大顶点的层数称为**树高**。

平凡树也称为根树。

说明：把有向树的根画在最上方，边的箭头全指向下，则可以省略全部箭头。



图中v1是树根

v2, v4, v7, v10是内点

v3, v5, v6, v8,

v9, v11, v12, v13是树叶

树高4层，在v12或v13处达到



根树可以看成**家族树**。

定义16.7 设 T 一棵非平凡的根树, $\forall v_i, v_j \in V(T)$,
若 v_i 可达 v_j , 则称 v_i 为 v_j 的**祖先**, v_j 为 v_i 的**后代**;
若 v_i 邻接到 v_j (即 $\langle v_i, v_j \rangle \in E(T)$), 则称 v_i 为 v_j 的**父亲**, v_j 为 v_i 的**儿子**。

若 v_i 和 v_j 的父亲相同, 则称 v_i 和 v_j 是**兄弟**。

定义16.8 (子树) 设 T 为一棵根树, 则其任一顶点 v 及其后代导出的子图称为根树的**子树**。

定义（有序树） 在根树 T 中，若将层数相同的顶点都标定次序，则称 T 为**有序树**。

根据**每个分支点的儿子数**以及**是否有序**，可将根树分成如下若干类：

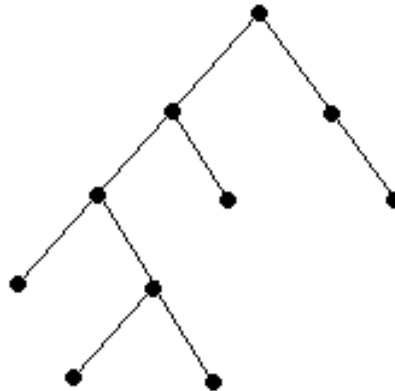
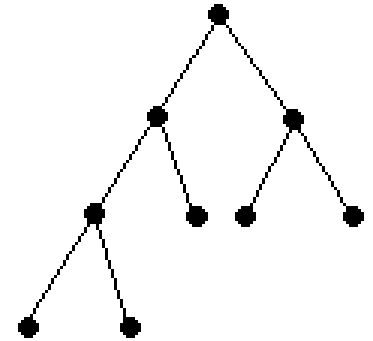
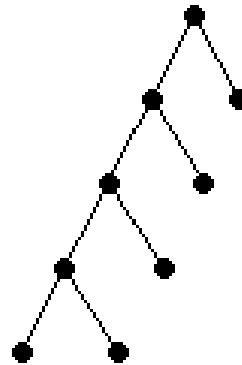
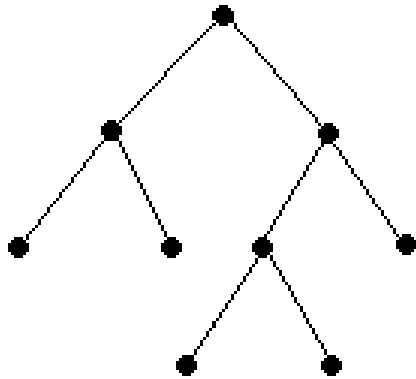
定义（跟树分类） 设 T 为一棵根树

(1) 若 T 的每个分支点至多有 r 个儿子，则称 T 为 **r 叉树**。又若 r 叉树是有序的，则称它为 **r 叉有序树**。

(2) 若 T 的每个分支点恰好有 r 个儿子，则称 T 为 **r 叉正则树**。又若 r 叉正则树是有序的，则称它为 **r 叉正则有序树**。

(3) 若 T 为 r 叉正则树，且每个树叶的层数均为树高，则称 T 为 **r 叉完全正则树**。又若 r 叉完全正则树是有序的，则称它为 **r 叉完全正则有序树**。

在所有的 r 叉树中，**2叉树**最重要。





最优2叉树及其应用

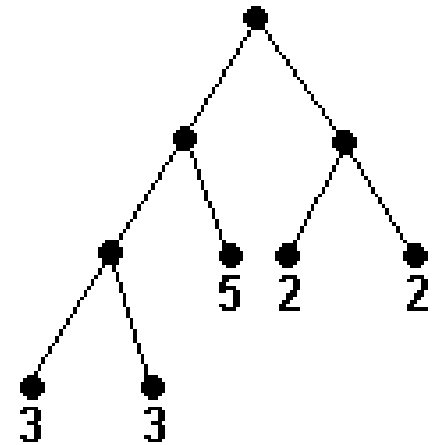
定义16.9 (带权2叉树)

设2叉树T有t片树叶 v_1, v_2, \dots, v_t , 权分别是 w_1, w_2, \dots, w_t , 则称T为**带权2叉树**。

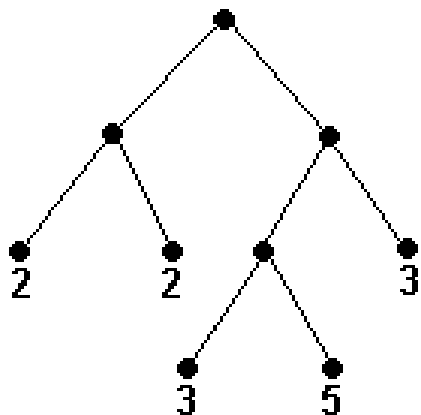
并称

$$W(t) = \sum_{i=1}^t w_i l(v_i)$$

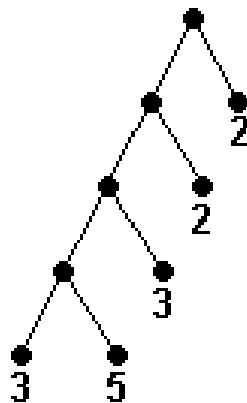
为T的**权**，其中 $l(v_i)$ 是 v_i 的层数。



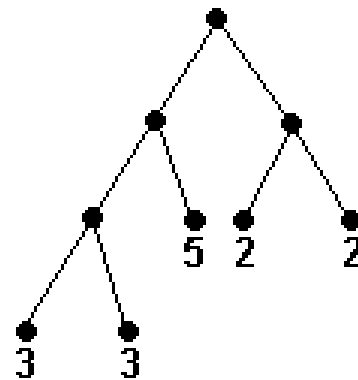
例 下面三棵2叉树T1, T2, T3都是带权2, 2, 3, 3, 5的2叉树。



T1



T2



T3

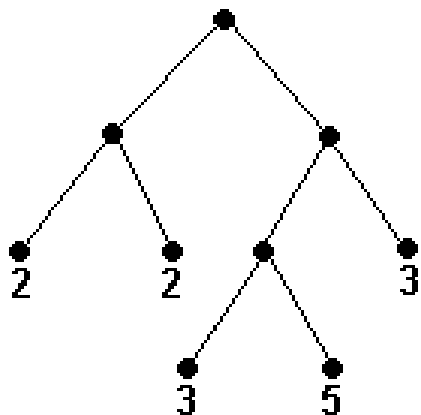
$$W(T1) = 2*2 + 2*2 + 3*3 + 5*3 + 3*2 = 38$$

$$W(T2) = 3*4 + 5*4 + 3*3 + 2*2 + 2*1 = 47$$

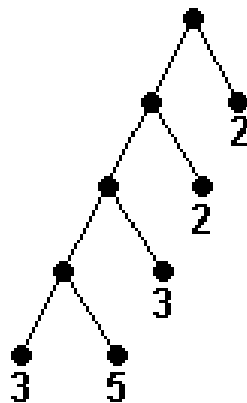
$$W(T3) = 3*3 + 3*3 + 5*2 + 2*2 + 2*2 = 36$$

定义 (最优2叉树) 在所有t片树叶, 带权 w_1, w_2, \dots, w_t 的2叉树中, 权最小的2叉树称为**最优2叉树**。

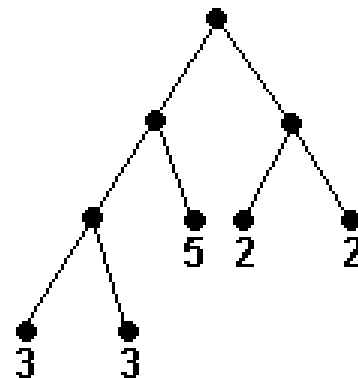
例 下面三棵2叉树T1, T2, T3都是带权2, 2, 3, 3, 5的2叉树。



T1



T2



T3

$$W(T1) = 2*2 + 2*2 + 3*3 + 5*3 + 3*2 = 38$$

$$W(T2) = 3*4 + 5*4 + 3*3 + 2*2 + 2*1 = 47$$

$$W(T3) = 3*3 + 3*3 + 5*2 + 2*2 + 2*2 = 36$$

以上三棵2叉树都不是带权2, 2, 3, 3, 5的最优2叉树
事实上带权2, 2, 3, 3, 5的最优2叉树的权为34。

如何求最优2叉树?

最优2叉树求法-Huffman算法

给定权 w_1, w_2, \dots, w_t , 且 $w_1 \leq w_2 \leq \dots \leq w_t$ 。

(1) 连接权为 w_1, w_2 的两片树叶, 得到一个分支点, 其权为 $w_1 + w_2$

(2) 在 $w_1 + w_2, w_3, \dots, w_t$ 中选出两个最小的权, 连接它们对应的顶点, 得到新分支点及所带的权。

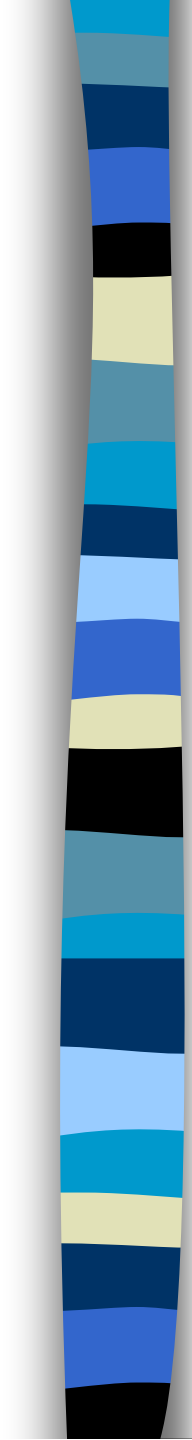
(3) 重复(2), 直到形成 t 片树叶的2叉树为止, 则所得2叉树就是带权 w_1, w_2, \dots, w_t 的最优2叉树。

例16.4 求带权2, 2, 3, 3, 5的最优2叉树。



练习

- (1) 求带权1, 3, 4, 5, 6的最优2叉树。
- (2) 求带权2, 3, 5, 7, 8, 9的最优2叉树。



在通信中，常用二进制编码表示符号。

例如，4位二进制码可以表示 $2^4=16$ 个不同字母

表示26个英文字母至少要用五位二进制码，这种方法称为**等长码表示法**。

但有的英文字母使用频率很高，例如e（13.1%），t（10.5%），还有a, i, r, s等，而有的英文字母使用频率很低，例如j, z, q只使用0.1%。

是否可以用**不等长字符串进行编码（不等长码）**，**频率高的**用的字符串**短些**，**频率低的**字符串可略**长些**，从而使通信数据编码的总长有所降低。

例: a b c d e
 00 110 010 10 01

把集合 {00, 110, 010, 10, 01} 称为**码**

但不等长码有时会造成**二义性**。

如收到电文是010010，就有二义性。

电文如理解为01, 00, 10，电文就是ead。

电文如理解为010, 010，电文就是cc

造成二义性的原因是e所对应的编码“01”是c所对应编码“010”的**前缀**，

如把c所对应的编码改为111，则不等长码 {00, 110, 111, 10, 01} 中就没有哪个编码是另一个编码的前缀，就不会造成二义性。

这种方法就是**前缀码**。

定义16.10 设 $a_1 a_2 \dots a_{n-1} a_n$ 为长 n 的符号串，称其子串 $a_1, a_1 a_2, \dots, a_1 a_2 \dots a_{n-1}$ 分别为该符号串的长度为 $1, 2, \dots, n-1$ 的**前缀**。

设 $A = \{\beta_1, \beta_2, \dots, \beta_m\}$ 为一个符号串集合，若对于任意的 $\beta_i, \beta_j \in A (i \neq j)$ ， β_i, β_j 互不为前缀，则称 A 为**前缀码**。

若符号串 $\beta_i (i=1, 2, \dots, m)$ 中只出现 $0, 1$ 两个符号，则称 A 为**二元前缀码**。

例如： (1) $\{1, 00, 011, 0101, 01001, 01000\}$

(2) $\{1, 00, 011, 0101, 0100, 01001, 01000\}$

(1) 为前缀码

(2) 不是前缀码

因为 0100 既是 01001 ，又是 01000 的前缀码

可用2叉树产生二元前缀码

设 T 为一棵2叉树，有 t 片树叶

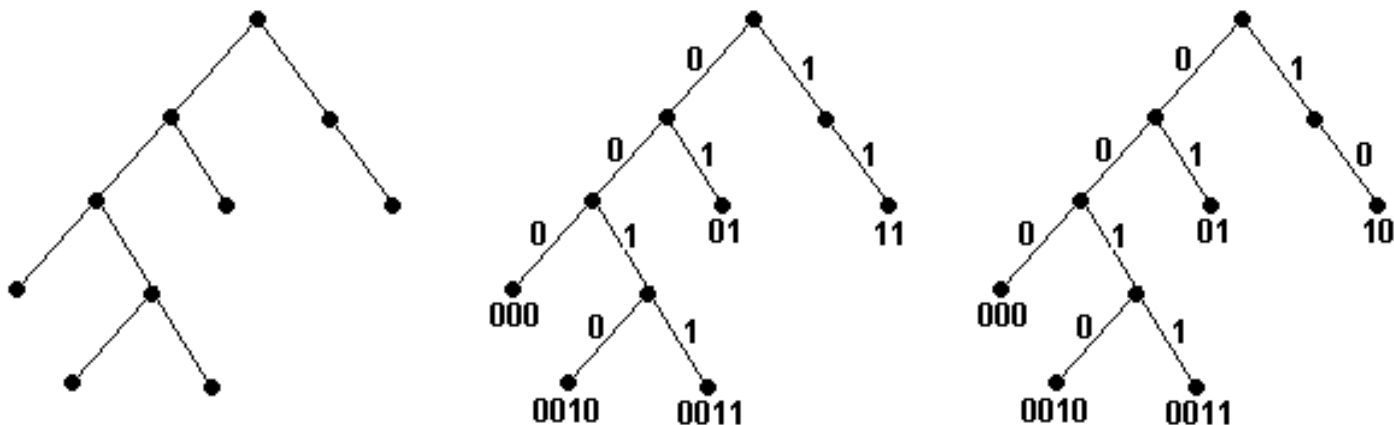
(1) 将 T 的每个分支点关联的两条边，左边标上0，右边标上1

(2) 若某个分支点处只有一个儿子，则对应的边标上0或1均可

(3) 从树根到每片树叶的通路上标注的数字组成一个符号串，记在树叶处

这样，在 t 片树叶处的 t 个符号串组成的集合为一个二元前缀码。

例如：求如下2叉树产生的二元前缀码

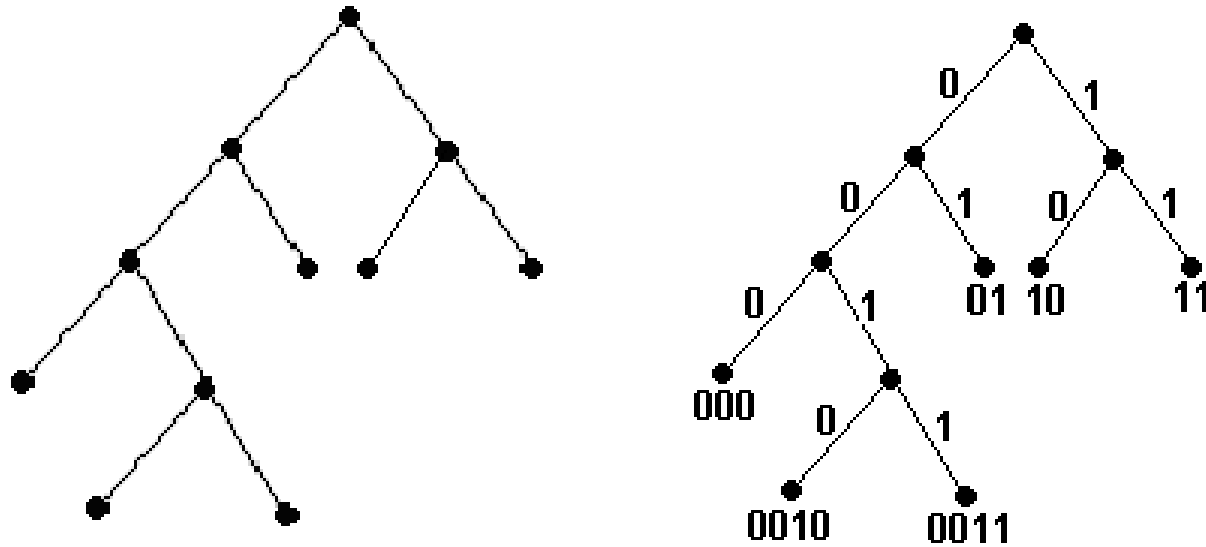


则可以产生二元前缀码

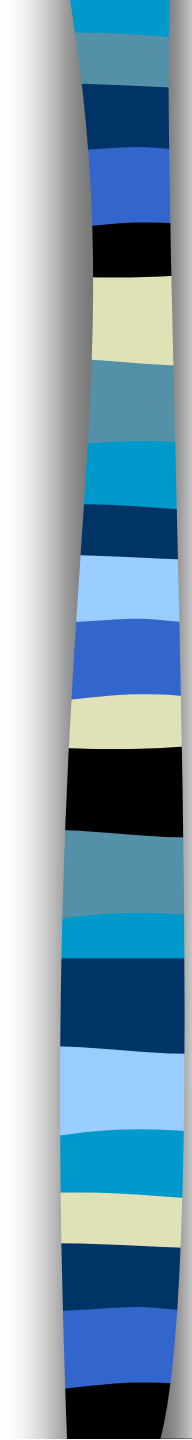
{01, 11, 000, 0010, 0011} 或 {01, 10, 000, 0010, 0011}

定理16.6 一棵2叉正则树产生唯一的一个二元前缀码。

例如： 求如下2叉正则树产生的二元前缀码



仅产生二元前缀码 {01, 10, 11, 000, 0010, 0011}



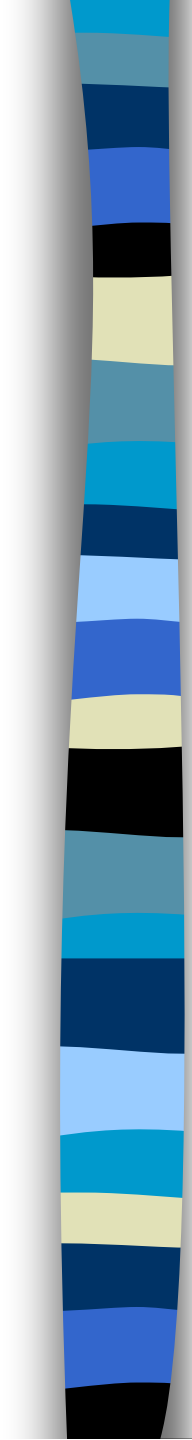
最佳前缀码：传输按一定比例出现的符号所使用的二元前缀码中最省二进制数的二元前缀码。

利用最佳前缀码可以节约符号串和文件的存储量。

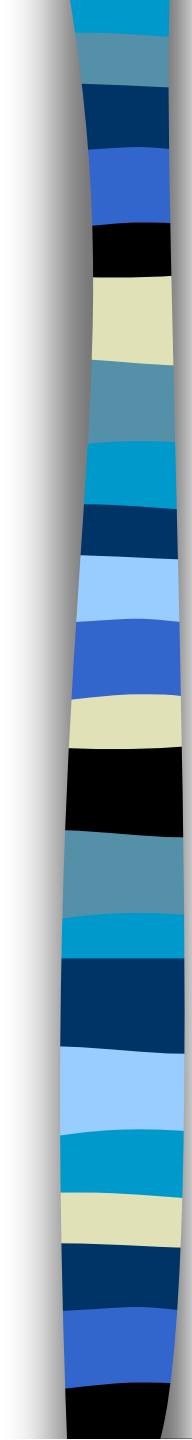
最佳前缀码的应用：数据压缩；数据通信。

如何求最佳前缀码？

利用Huffman算法求最优2叉树，由最优2叉树产生的二元前缀码就是最佳前缀码。 ————Huffman编码



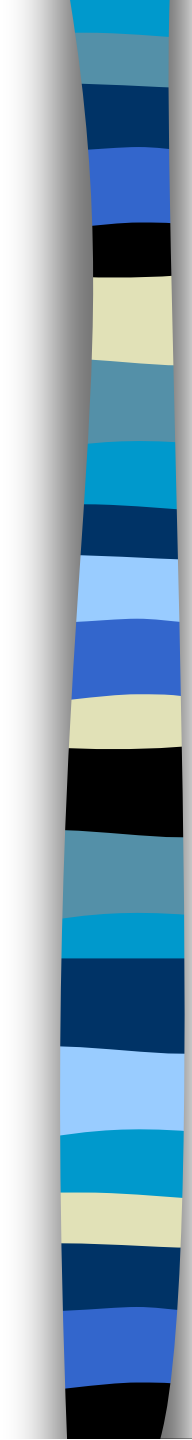
例 求带权2, 2, 3, 3, 5的2叉树产生的最佳前缀码。
最佳前缀码为 {01, 10, 11, 000, 001}



平均编码长度：若共有n个符号，经统计后得知符号i出现的概率为P(i)，且符号i编码后的长度为L(i)比特。

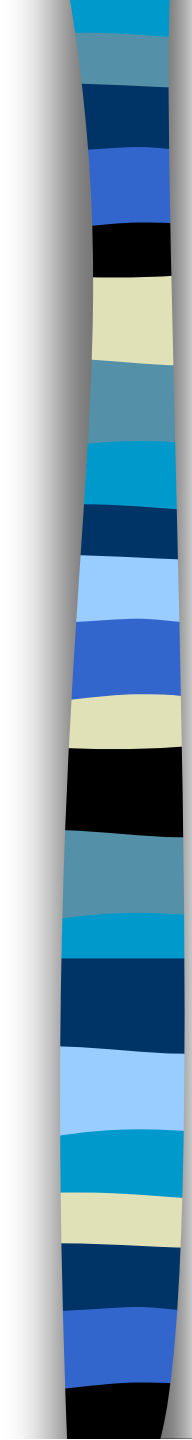
则平均编码长度（比特）为：

$$L = \sum_{i=1}^n P(i) L(i)$$



例 假设共有8个字符a,b,c,d,e,f,g,h，其出现的概率分别为0.4、0.2、0.15、0.1、0.07、0.04、0.03、0.01，编码分别为1、01、001、0001、00001、000001、0000001、00000000，易见编码长度分别为1、2、3、4、5、6、7、8。

平均编码长度为：
$$L = \sum_{i=1}^8 P(i) \times l(i) = 2.53bit$$



例 设有7个符号的信源 $A=\{a,b,c,d,e,f,g\}$,
若其概率分布为 $P=\{0.2, 0.19, 0.18, 0.17, 0.15, 0.1, 0.01\}$ 。

(1) 对信源A其进行Huffman编码。

(2) 求传输 10^n ($n \geq 2$) 个按上述比例出现的7个符号需要多少比特？

(3) 对于 (2) 若是用等长码进行传输至少需要多少比特？

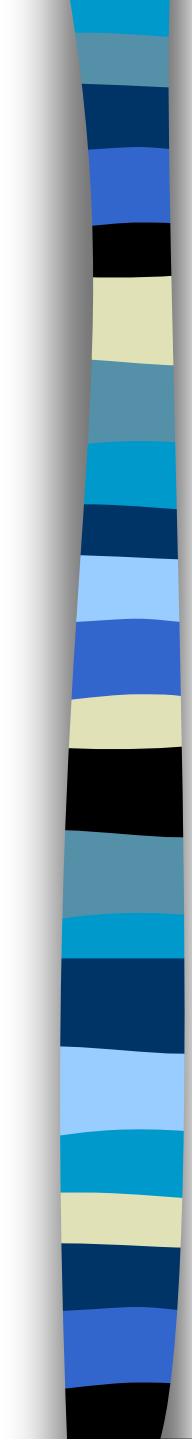


原数据	出现概率	编码	码长
a	0.20	11	2
b	0.19	10	2
c	0.18	011	3
d	0.17	010	3
e	0.15	001	3
f	0.10	0001	4
g	0.01	0000	4

信源A = {a,b,c,d,e,f,g}的编码为：{11, 10, 011, 010, 001, 0001, 0000}

计算平均编码长度：

$$L = \sum_{i=1}^8 P(i) \times l(i) = 2.72 \text{ bit}$$

- 
- (2) 传输 10^n ($n \geq 2$) 个按上述比例出现的7个符号
需要 2.72×10^n 比特;
 - (3) 用等长 (长为3) 的编码传输需要 3×10^n 比特。



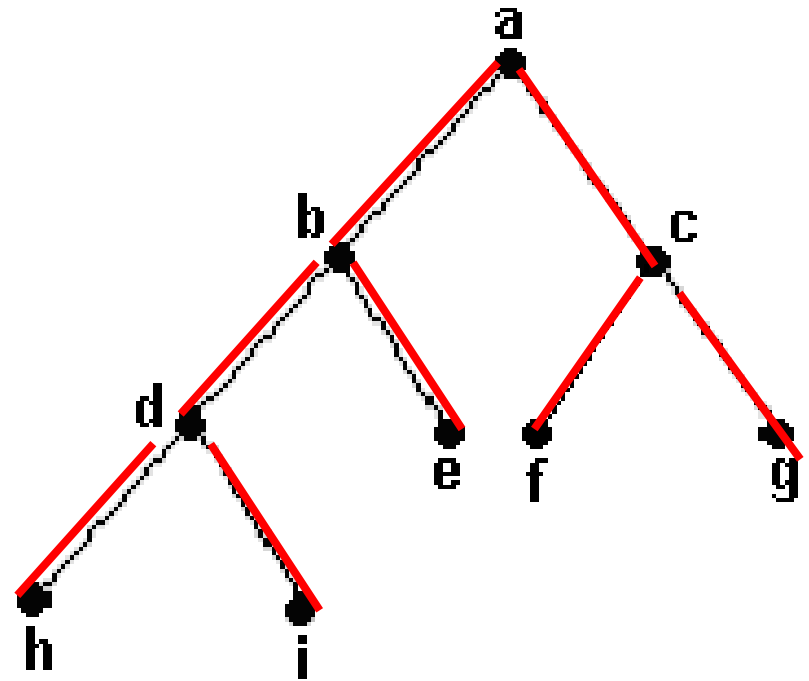
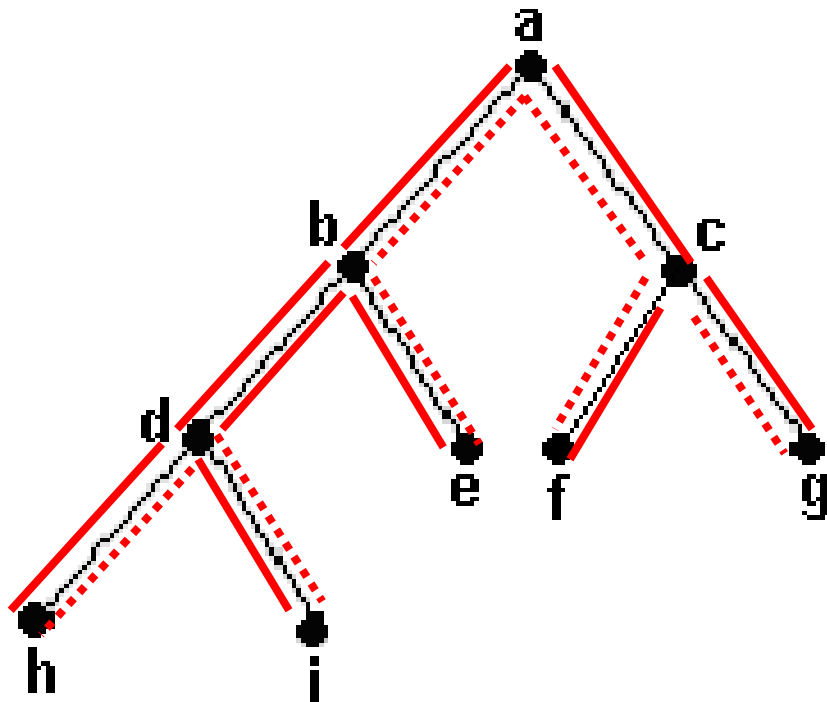
二叉树的遍历



2叉树的遍历

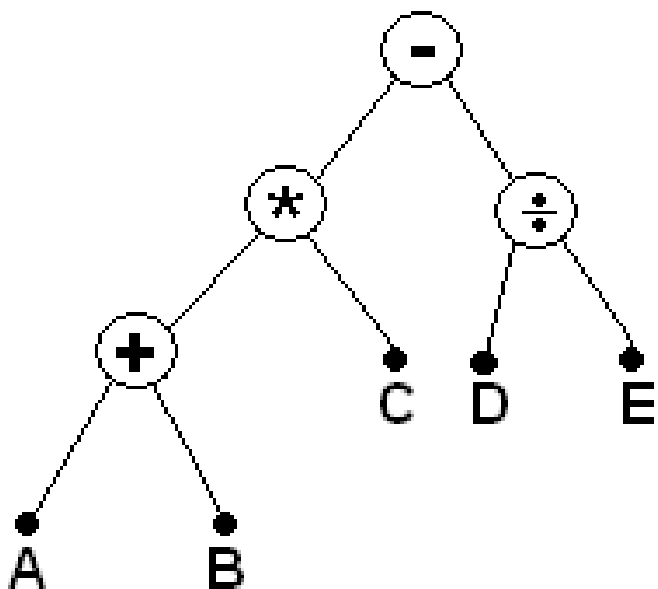
- 按照图的遍历方法：
- (1) 深度优先搜索 DFS (*Depth First Search*)
- (2) 广度优先搜索 BFS (*Breadth First Search*)

2叉树的遍历



图的遍历方法的局限性

- 树，尤其是2叉树，常用于存储一些有序的数据，如下面的2叉有序树存储了一个四则运算的算式 $(A+B) * C - D \div E$ 。



由于深度优先和广度优先搜索法在遍历一个图时，起点以及遍历过程中所经过的顶点具有一定的随机性，因此使用深度优先和广度优先搜索法从下面的2叉树中恢复算式 $(A+B) * C - D \div E$ 显然是不合适的。



2叉树的遍历有三种不同的方法：

(1) **中序遍历法**（中根遍历法）

访问的次序为：左子树，树根，右子树

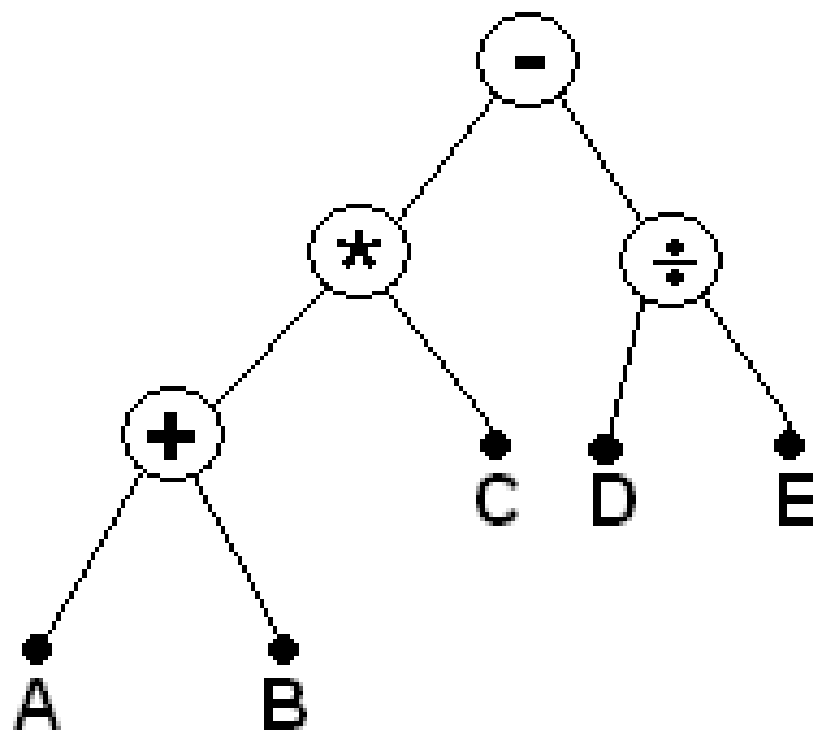
(2) **前序遍历法**（前根遍历法）

访问的次序为：树根，左子树，右子树

(3) **后序遍历法**（后根遍历法）

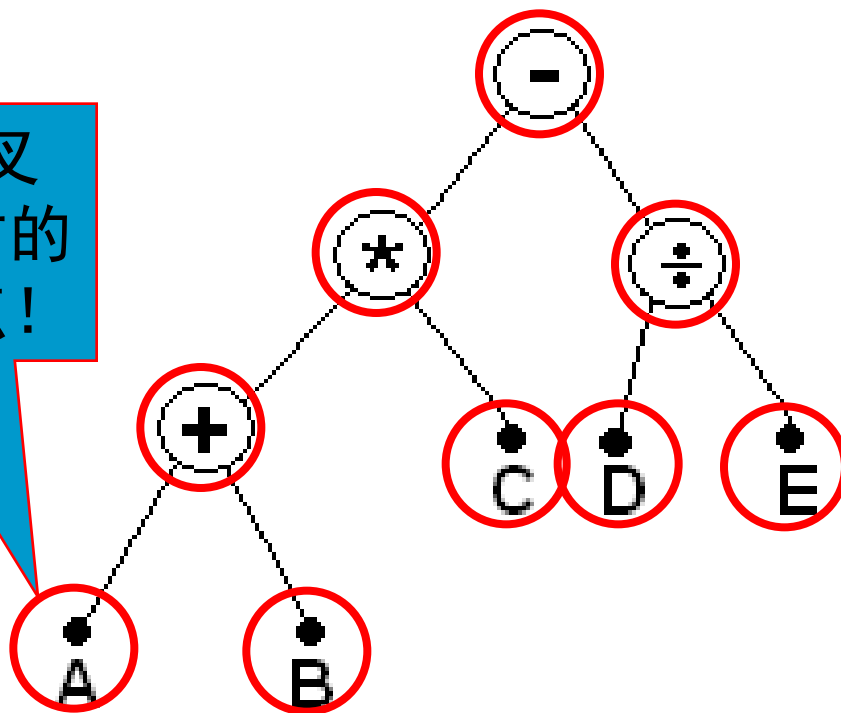
访问的次序为：左子树，右子树，树根

例 分别使用中序遍历法、前序遍历法以及后序遍历法遍历下面的2叉树。



例 分别使用 **中序遍历法**、前序遍历法以及后序遍历法遍历下面的2叉树。

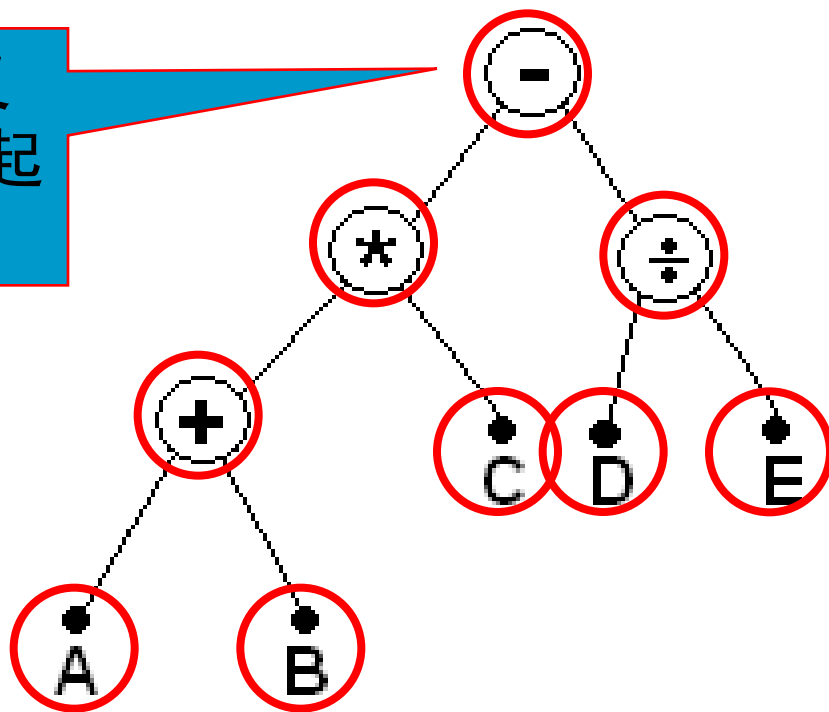
中序遍历以2叉树的最左下方的树叶作为起点！



中序遍历结果为： $((A+B)*C)-(D\div E)$

例 分别使用中序遍历法、前序遍历法以及后序遍历法遍历下面的2叉树。

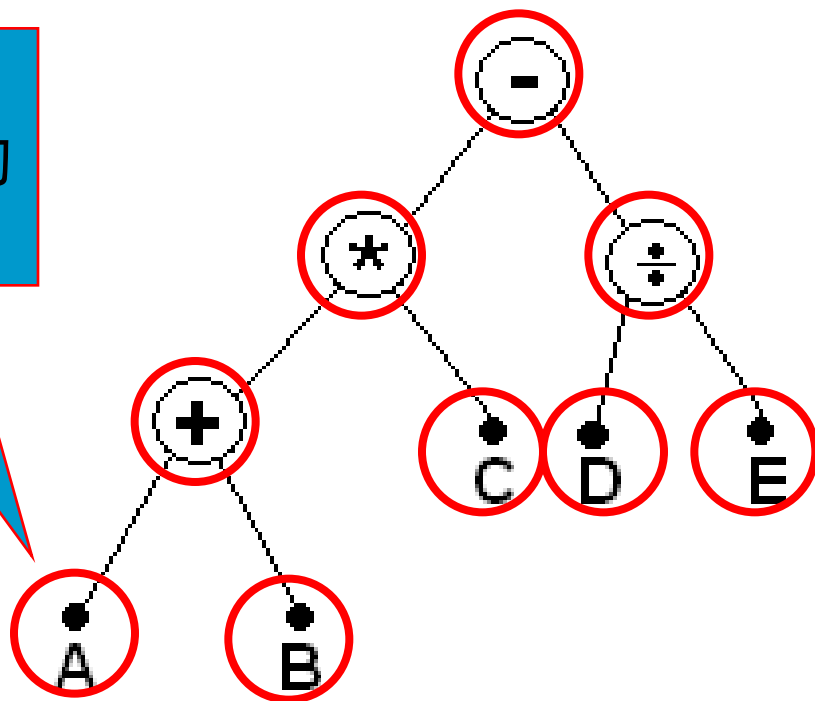
前序遍历以2叉树的树根作为起点！



前序遍历结果为： $-(+AB)C)(\div DE)$

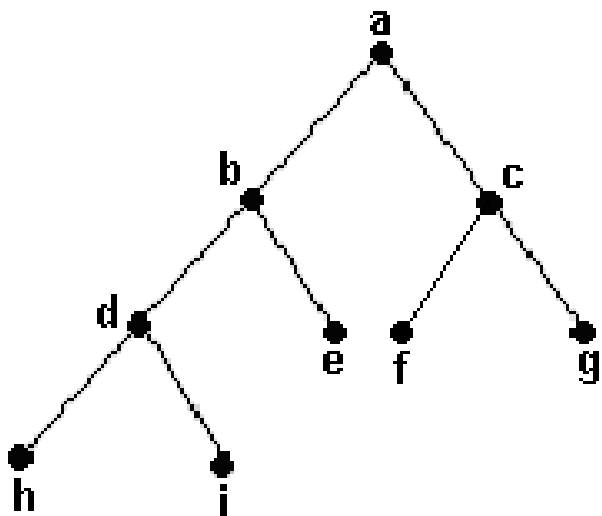
例 分别使用中序遍历法、前序遍历法以及后序遍历法遍历下面的2叉树。

后序遍历以2叉树的最左下方的树叶作为起点！



后序遍历结果为：((AB+)C*)(DE÷)-

练习 分别使用中序遍历法、前序遍历法以及后序遍历法遍历下面的2叉树。



中序行遍结果为：

((h d i) b e) a (f c g)

前序行遍结果为：

a (b (d h i) e) (c f g)

后序行遍结果为：

((h i d) e b) (f g c) a

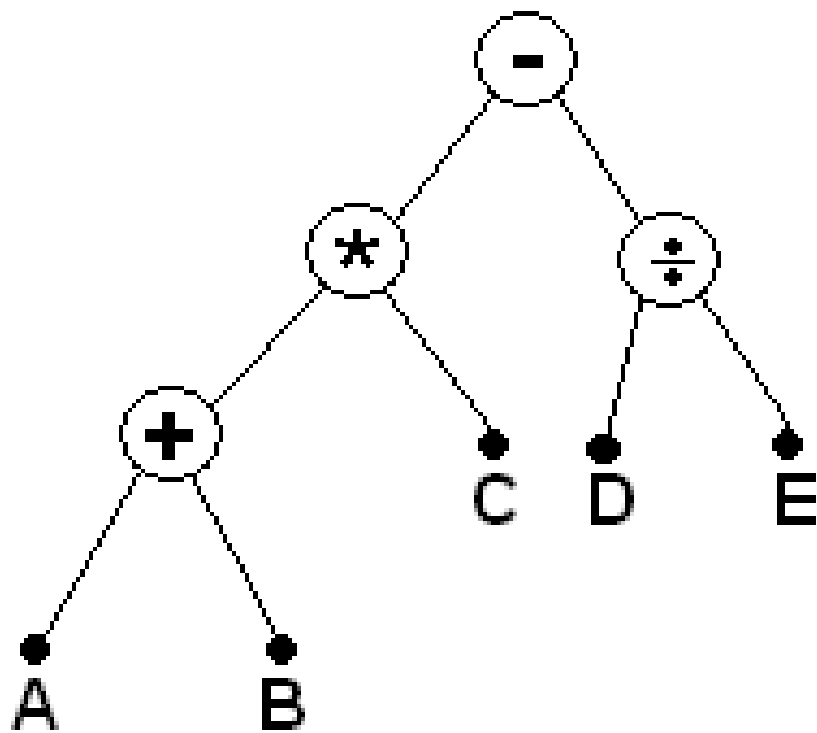
对于上例中的结果：

(1) 中序遍历结果 $((A+B)*C)-(D\div E)$ ，称为**中缀表达式**或**还原算式**。

(2) 前序遍历结果（去掉括号） $-*+ABC\div DE$ ，称为**前缀表达式**或**波兰算式**，规定每个运算符与它后面紧邻的两个数进行运算。

(3) 后序遍历结果（去掉括号） $AB+C*DE\div -$ ，称为**后缀表达式**或**逆波兰算式**，规定每个运算符与它前面紧邻的两个数进行运算。

使用2叉树存储算式的基本方法



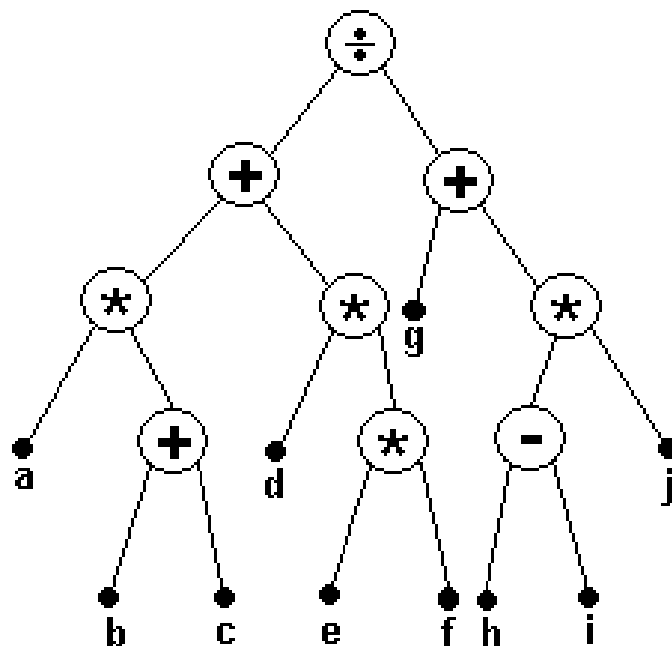
$$((A+B)*C)-(D\div E)$$

最高层次（优先级最低）的运算符放在树根上，然后依次地将运算符（按优先级从低到高的顺序）放在根子树的树根上，参加运算的数放在树叶上，并规定被除数、被减数放在左子树树叶上。

练习

使用2叉树存储下面的算式，并使用三种遍历法分别遍历生成的2叉树。

$$(a*(b+c)+d*e*f) \div (g+(h-i)*j)$$



中序行遍结果为：

$$((a * (b + c)) + (d * (e * f))) \div (g + ((h - i) * j))$$

前序行遍结果为：

$$\div (+ (* a (+ b c)) (* d (* e f))) (+ g (* (- h i) j))$$

后序行遍结果为：

$$((a (b c +) *) (d (e f *) *) +) (g ((h i -) j *) +) \div$$