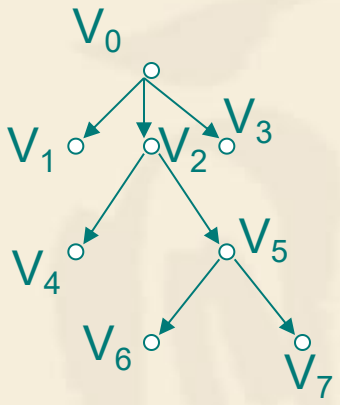
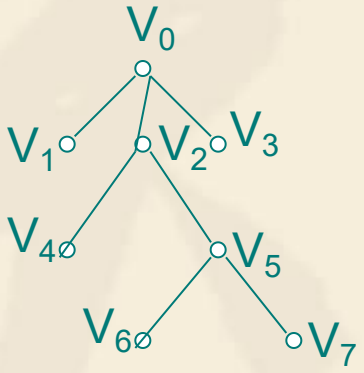
有向树

一个有向图D，如果略去有向边的方向所得的无向图为一颗无向树，则称D为**有向树**。

根树

- 一颗平凡的有向树，如果有一个顶点的入度为0,其余顶点的入度均为1，则称此有向树为**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、出度大于0的顶点称为**内点**，内点和树根统称为**分支点**。





左图为一棵根树。V_0为树根，V_1，V_4,V_3,V_6,V_7为树叶，V_2,V_5为内点，V_0,V_2,V_5均为分支点。由于在根树中有向边的方向均一致（向下），故可省掉其方向，用右图 代替。

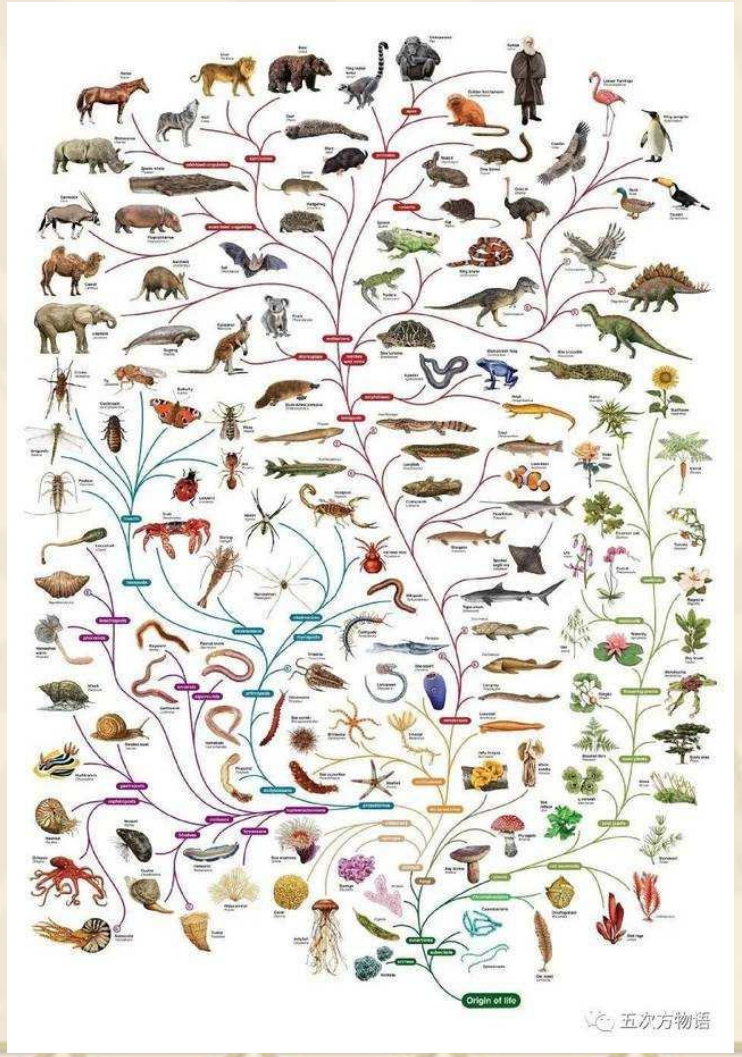
在根树中，从树根到任意顶点v的通路长度称为v的**层数**，记为l(v)．称层数相同的顶点在同一层上。层数最大的顶点的层数称为**树高**．根树T的树高记为h(T)．

家族树

一棵根树可以看成一棵家族树：

- （1）若顶点a邻接到顶点b，则称b为a的儿子，a为b的父亲；
- （2）若b，c同为a的儿子，则称b，c为兄弟；
- （3）若a≠d，而a可达d，则称a为d的祖先，d为a的后代。

举例：进化树



根子树

- 定义设T为一棵根树，a为T中一个顶点，且a不是树根，称a及其后代导出的子图T'为T的以a为根的子树，简称**根子树**.

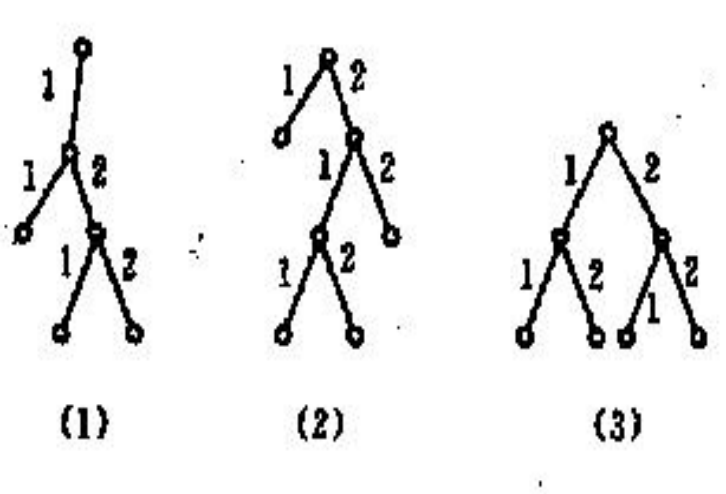
定义

- 如果将根树每一层上的顶点都规定次序，这样的根树称为有序树.
- 次序可排在顶点处，也可以排在边上。次序常常是从左向右，不一定是连续的.

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元树；
- （2）若T的每个分支点都恰好有r个儿子，则称T为r元正则树；
- （3）若r元树T是有序的，则称T是r元有序树 .

- （4）若r元正则树T是有序的，则称T是r元有序正则树；
- (5)若T是r元正则树，且所有树叶的层数相同，都等于树高，则称T为r元完全正则树；
- （6）若r元完全正则树T是有序树，则称T是r元有序完全正则树。



图（1）为2元有序树，（2）为2元有序正则树，（3）为2元有序完全正则树。

最优2元树

设2元树T有t片树叶,分别带权为

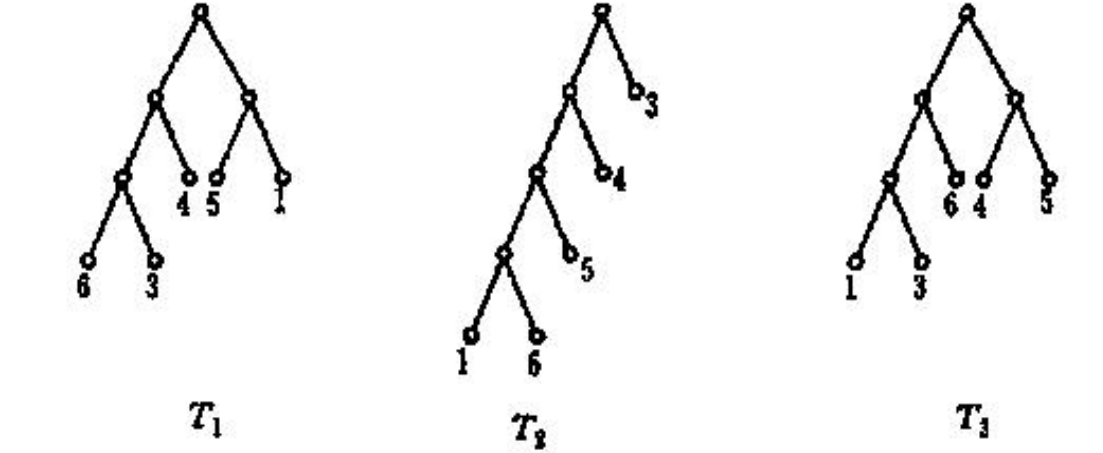
$w_1, w_2, \cdots, w_i, \cdots, w_t$ (w_i为实数, $i = 1,2,\cdots,t$,)

称

$$w(T) = \sum_{i=1}^{t} w_i L(w_i)$$

为T的权, 其中$L(W_i)$为带权w_i的树叶v_i的层数. 在所有的带权$w_1, w_2, \cdots, w_t$的2元树中, 带权最小的2元树称为**最优2元树**.

图16.7



图中所示是带权1，3，4，5，6的2元树

Huffman算法

- 给定实数$w_1,w_2,\cdots,w_t$,且$w_1\le w_2\le\cdots\le w_t$.
- (1)连接w_1,w_2为权的两片树叶,得一分支点,其权为w_1+w_2;
- (2)在$w_1+w_2,w_3,\cdots,w_t$中选出两个最小的权,连接它们对应的顶点(不一定都是树叶),得分支点及所带的权;
- (3)重复(2),直到形成$t-1$个分支点,t片树叶为止.

例16.2

- (1)求带权为1,3,4,5,6的最优2元树;
- (2)求带权为2,3,5,7,8,9的最优2元树.

- 解 (1)图16.3给出了求带权1,3,4,5,6的最优2元树的过程.
- (2)由Huffman算法求出的带权为2,3,5,7,8,9的最优树T为图16.4所示.

图16.3

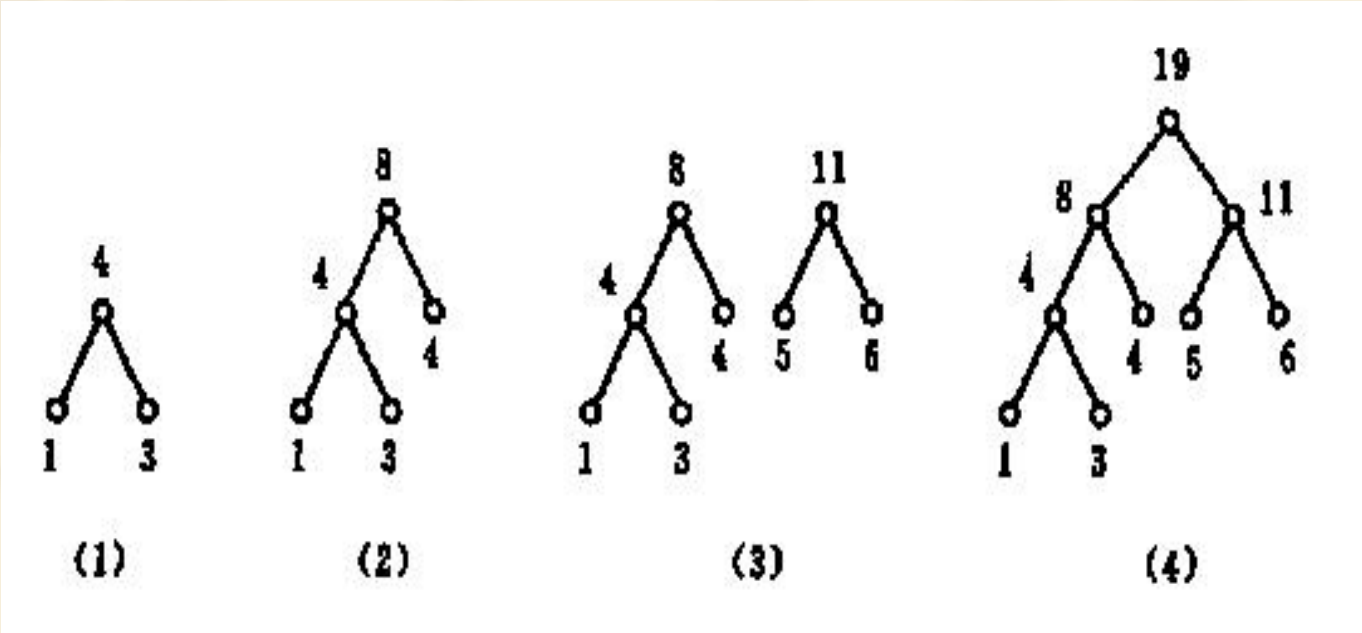
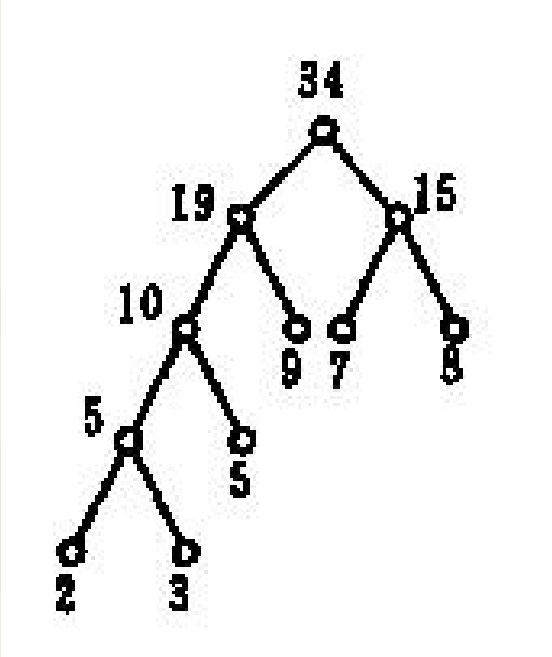


图16.4



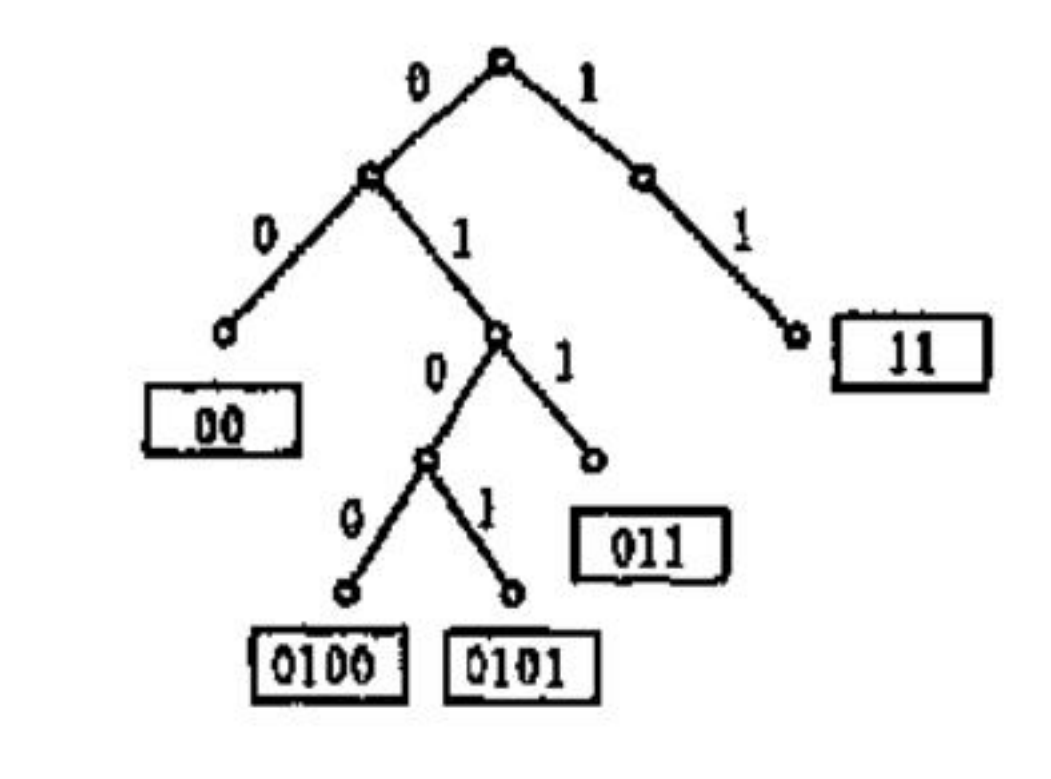
前缀

- 设$\beta=\alpha_1,\alpha_2\cdots\alpha_{n-1},\alpha_n$为长度是n的符号串,称其子串$\alpha_1,\alpha_1\alpha_2,\cdots,\alpha_1\alpha_2\cdots\alpha_{n-1}$分别为$\beta$的长度为$\alpha_1,\alpha_2,\cdots,\alpha_{n-1}$的**前缀**.

前缀码

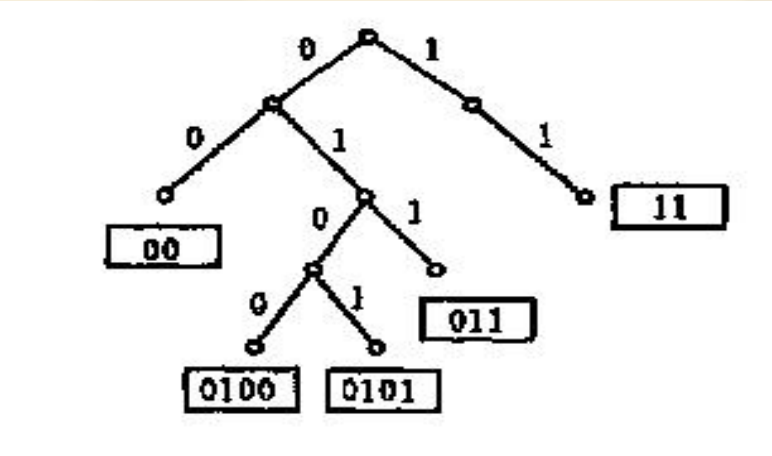
- 设$B=\{\beta_1,\beta_2,\cdots,\beta_m\}$为一个符号串集合，若对于任意的$\beta_i, \beta_j \in B, i \neq j$，i与j互不为前缀，则称B为**前缀码**.
- 若$\beta_i (i=1,2,\cdots,m)$中只出现2个符号(如0,1)，则称β 为**2元前缀码**.

图16.5



- 图16.5所示的2元树产生的前缀码为

{11, 00, 011,0100,0101}.



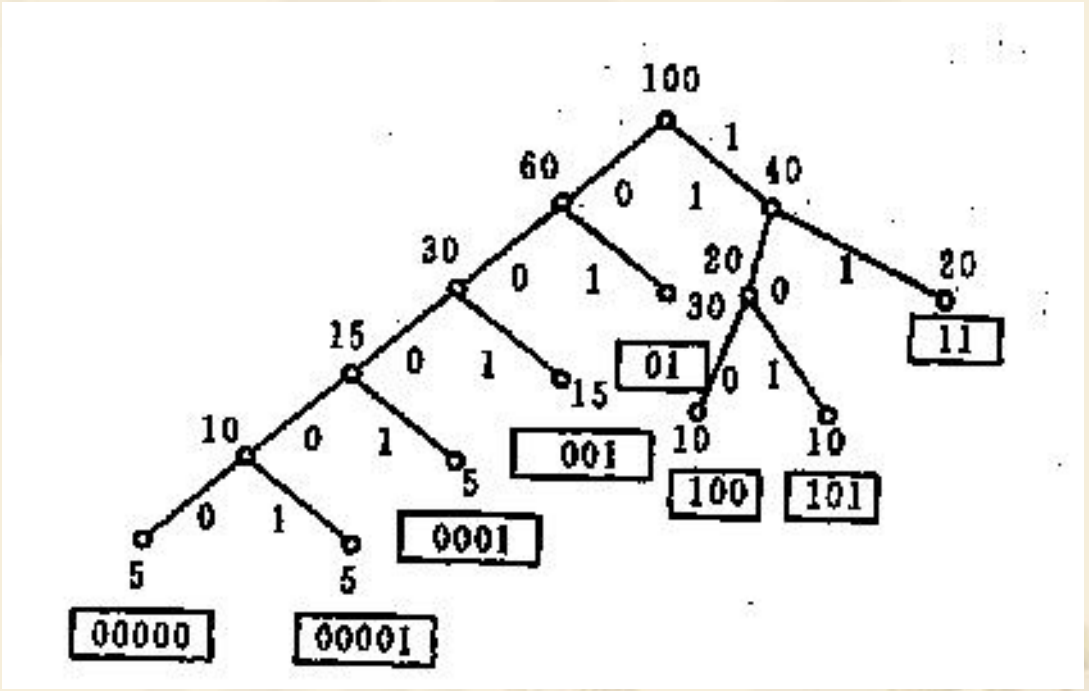
最佳前缀码

当要传输按着一定比例出现的符号时，需要寻找传输它们最省二进制数字的前缀码，这就是**最佳前缀码**.

- 例16.6 在通信中,0,1,2,…,7出现的频率如下:
- 0:30%, 1:20%,
- 2:15%, 3:10%,
- 4:10%, 5:5%,
- 6:5%, 7:5%.

求传输它们的最佳前缀码.

- 所求2元树T


100
1
60
0
1
40
30
0
1
20
1
20
15
0
1
30
0
01
11
10
0
1
15
0
1
001
10
10
5
100
101
0
1
0001
5
5
00000
00001

行遍

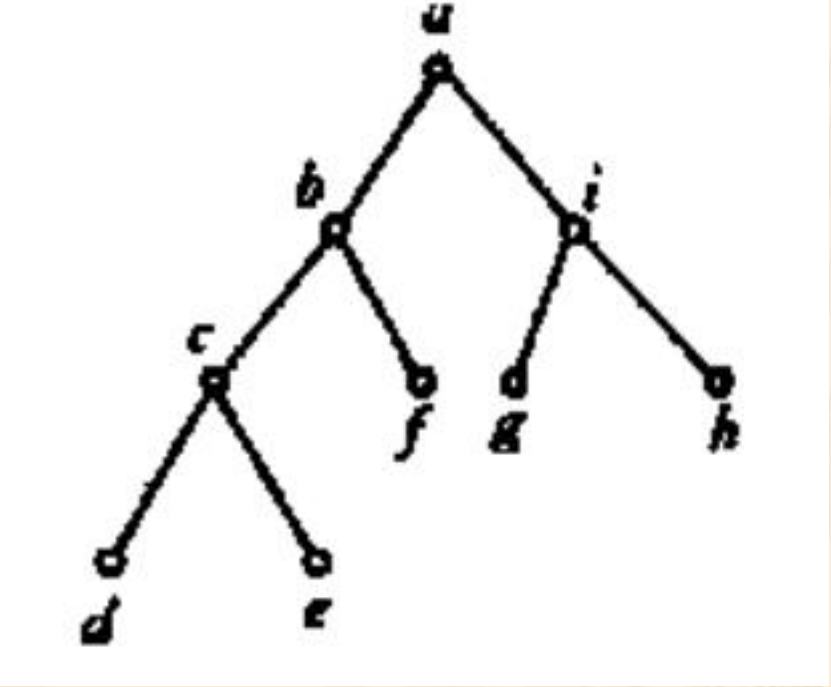
- 对于一棵根树的每个顶点都访问一次且仅一次称为可行遍或周游一棵树.

行遍法

对于2元有序正则树主要有以下3种行遍方法.

- (1)中序行遍法 其访问次序为:左子树,树根,右子树.
- (2)前序行遍法 其访问次序为:树根,左子树,右子树.
- (3)后序行遍法 其访问次序为:左子树,右子树,树根.

图16.7



- 对于图16.7所示的根树,按中序行遍法,其行遍结果为

  ((dce)bf)a(gih).

- 按前序行遍法行遍结果为

  a(b(cde)f)(igh).

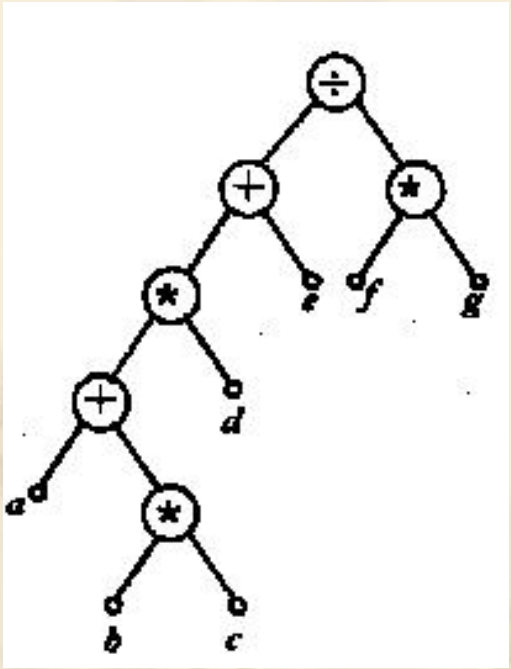
- 按后序行遍法行遍结果为

  ((dec)fb)(ghi)a.

- 例16.4
- (1)用2元有序树表示下面算式:

  ((a+(b*c))*d+e)÷(f*g).
- (2)用3种行遍法访问(1)中根树,写出行遍结果.

（1）所得树T如图所示



前缀符号法

- ÷ ＋ * ＋ a*bcde*fg ④
- 在④中规定,每个运算符与它后面紧邻的两数进行运算,其计算结果是正确的.
- 在此种算法中,因为运算符在参加运算的两数的前面,因而称为**前缀符号法**或**波兰符号法**.

后缀符号法

- abc* + d*e + fg* ÷ . ⑤
- 在⑤中规定,每个运算符与它前面紧邻的两数进行运算,其计算结果也是正确的.在这种算法中,因为运算符在参加运算两数的后面,因而称为**后缀符号法**或**逆波兰符号法**.

波兰表示法的背景

- 卢卡西威茨(Lukasiewicz，波兰，1878—1956)于20世纪20年代首先使用的一种不需用括号表示一个式子的方法。可用来写出算术表达式、代数表达式和逻辑表达式，其中所有运算符号位于进行这些运算的数据(或变量)之后。在求表达式的值时，它能给出需要求出的全部中间结果，而且在求复杂表达式的值时，特别方便。
- 前缀表达式主要是用于学术研究方面。其体系经常在编译器构造的概念教学中首先使用。
- 逆波兰表示法在许多基于堆栈的程序语言（如PostScript）中使用，以及是一些计算器（特别是惠普）的运算原理。

课后习题

P342-343:

37;　　41;