图论

Graph Theory

内容提要

1. 图的基本概念
2. 图的连通性
3. 图的矩阵表示
4. 欧拉图与哈密顿图
5. 无向树与根树
6. 平面图

内容提要

1. 图的基本概念
2. 图的连通性
3. 图的矩阵表示
4. 欧拉图与哈密顿图
5. 无向树与根树
6. 平面图

5、无向树与根树

概念：

无向树，生成树，最小生成树，Kruskal

根树，m叉树，最优二叉树，Huffman算法

有向树与根树的定义

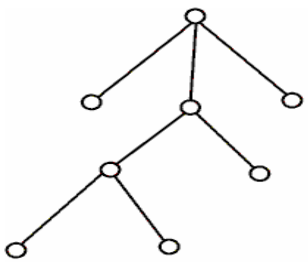
根树（Rooted Tree）

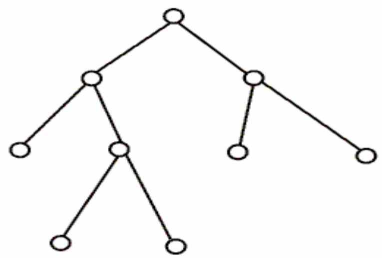
基图为无向树的有向图称为有向树；若有向树中一个顶点入度为0，其余的入度均为1，则称之为根树。

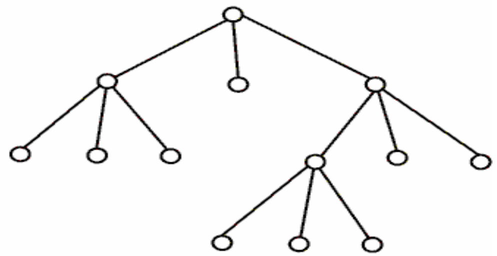
- 树根——入度为0的顶点
- 树叶——入度为1，出度为0的顶点
- 内点——入度为1，出度不为0的顶点
- 分支点——树根与内点的总称
- 顶点 v 的层数——从树根到 v 的通路长度
- 树高——T 中层数最大顶点的层数
- 平凡根树——平凡图

根树实例

根树的画法——树根放上方，省去所有有向边上的箭头







家族树与根子树

定义 T 为非平凡根树

(1) 父亲(Parent)与儿子(Child)：假若从a到b有一条边，则结点b称为a的“儿子”，或称a为b的“父亲”。

(2) 祖先(Ancestor)与后代(Descendant)：假若从a到c有一条单向通路，称a为c的“祖先”或 c是a的“后裔”。

(3) 兄弟(Sibling)：同一个分枝点的“儿子”称为“兄弟”。

设v为根树T中任意一顶点，称v及其后代的导出子图为以v为根的根子树。

根树的分类

(1) T 为有序根树——同层上顶点标定次序的根树

(2) 分类

① r 叉树——每个分支点至多有r 个儿子

② r 叉有序树——r 树是有序的

③ r 叉正则树——每个分支点恰有r 个儿子

④ r 叉正则有序树

⑤ r 叉完全正则树——树叶层数相同的r叉正则树

⑥ r 叉完全正则有序树

最优二叉树

最优二叉树

设2叉树T有t片树叶$v_1, v_2, \ldots, v_t$，权分别为$w_1, w_2, \ldots, w_t$，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。

在所有有t片树叶，带权$w_1, w_2, \ldots, w_t$的2叉树中，权最小的2叉树称为最优2叉树

Huffman算法

给定实数$w_1, w_2, \ldots, w_t$，且$w_1 \leq w_2 \leq \ldots \leq w_t$。

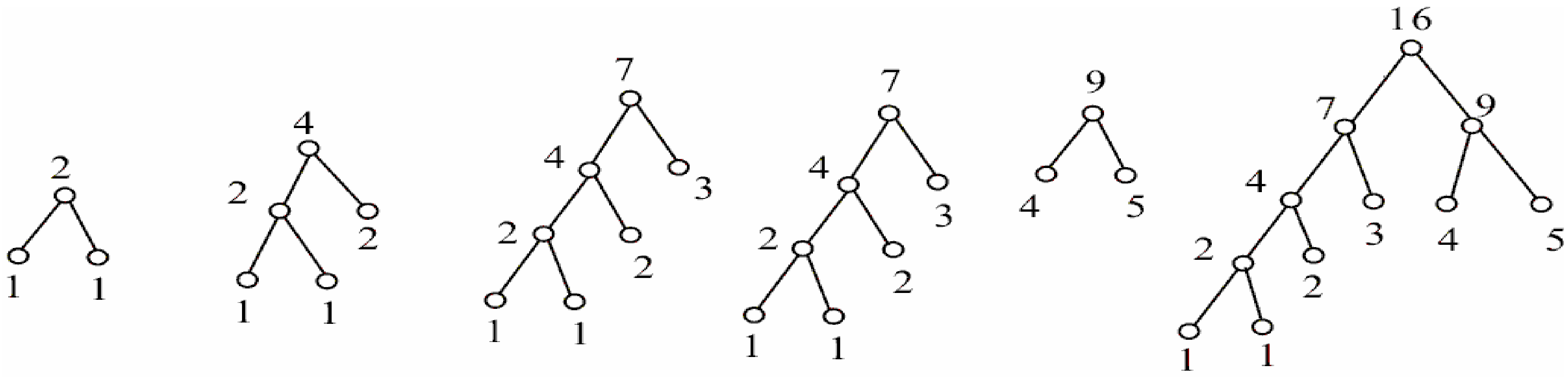
(1) 连接权为w_1, w_2的两片树叶，得一个分支点，其权为w_1+w_2；

(2) 在$w_1+w_2, w_3, \ldots, w_t$中选出两个最小的权，连接它们对应的顶点(不一定是树叶)，得新分支点及所带的权；

(3) 重复(2)，直到形成 $t-1$个分支点，t片树叶为止.。

例: 求带权为1, 1, 2, 3, 4, 5的最优树。

解: 过程由下图给出，$W(T)$=38



最优二叉树的应用：前缀码

前缀码 Prefix code

设α_1，α_2，…，α_{n-1}，α_n是长度为 n 的符号串

(1) 前缀——α_1，$\alpha_1\alpha_2$，…，$\alpha_1\alpha_2\cdots\alpha_{n-1}$

(2) 前缀码——$\{\beta_1, \beta_2, \cdots, \beta_m\}$中任何两个元素互不为前缀

(3) 二元前缀码——β_i（i=1，2，…，m）中只出现两个符号，如0与1

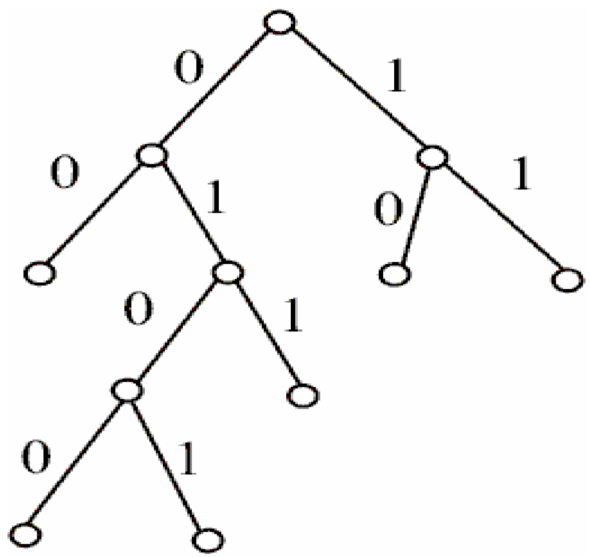
如何产生二元前缀码？

（1）一棵2叉树产生一个二元前缀码；

（2） 一棵正则2叉树产生惟一的前缀码。

（按左子树标0，右子树标1）

图所示二叉树产生的前缀码为　　{ 00, 10, 11, 011, 0100, 0101 }



用Huffman算法产生最佳前缀码

例：在通信中，八进制数字出现的频率如下：

0：25%	1：20%
2：15%	3：10%
4：10%	5：10%
6：5%	7：5%

求传输它们的最佳前缀码，并求传输10^n（$n\geq 2$）个按上述比例出现的八进制数字需要多少个二进制数字？

若用等长的（长为3）的码字传输需要多少个二进制数字？

求最佳前缀码

解： 用100个八进制数字中各数字出现的个数，即以100乘各频率为权，并将各权由小到大排列，得w_1=5, w_2=5, w_3=10, w_4=10, w_5=10, w_6=15, w_7=20, w_8=25。用此权产生的最优树如图所示。

01-----0	11-----1
001-----2	100-----3
101-----4	0001-----5
00000-----6	00001-----7

$W(T)$=285，

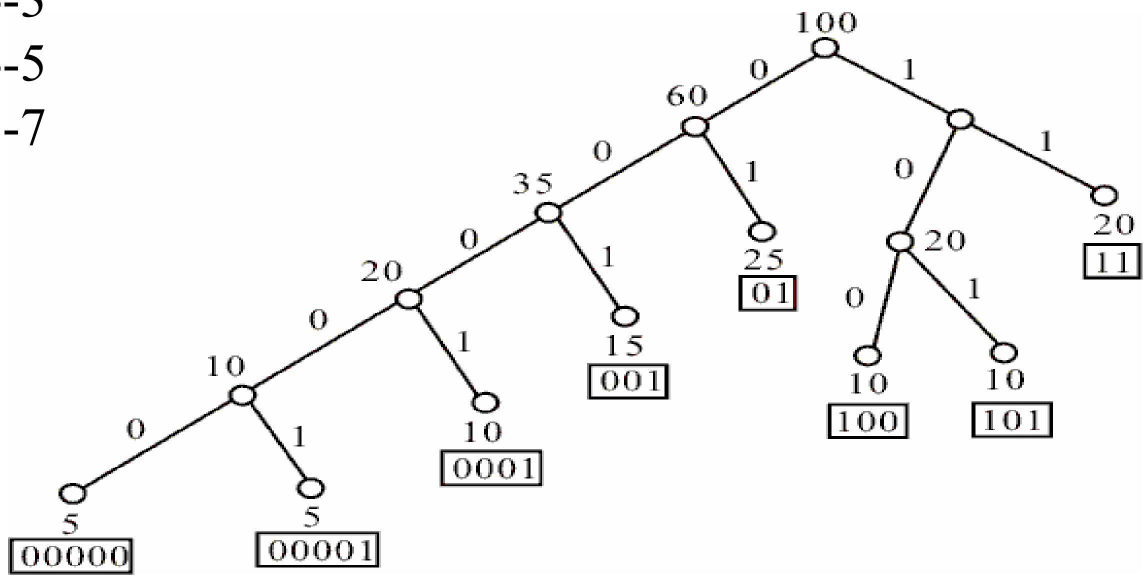
传10^n(n≥2)个

用二进制数字需

2.85×10^n个，

用等长码需

3×10^n个数字.



根树的周游与（逆）波兰表示法

根树的周游（遍历）

行遍或周游根树 T——对 T的每个顶点访问且仅访问一次。

对2叉有序正则树的周游方式：

① 中序行遍法——次序为：左子树、根、右子树

② 前序行遍法——次序为：根、左子树、右子树

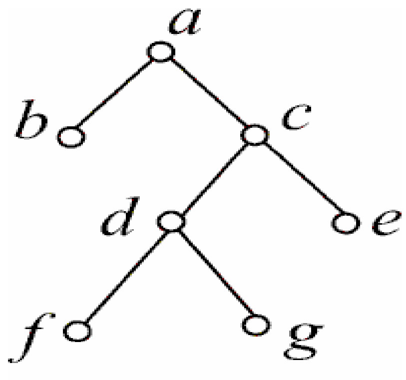
③ 后序行遍法——次序为：左子树、右子树、根

对图所示根树按中序、前序、后序行遍法访问结果分别为：

$b\ \underline{a}\ (f\ \underline{d}\ g)\ \underline{c}\ e$，

$\underline{a}\ b\ (\underline{c}\ (\underline{d}\ f\ g)\ e)$，

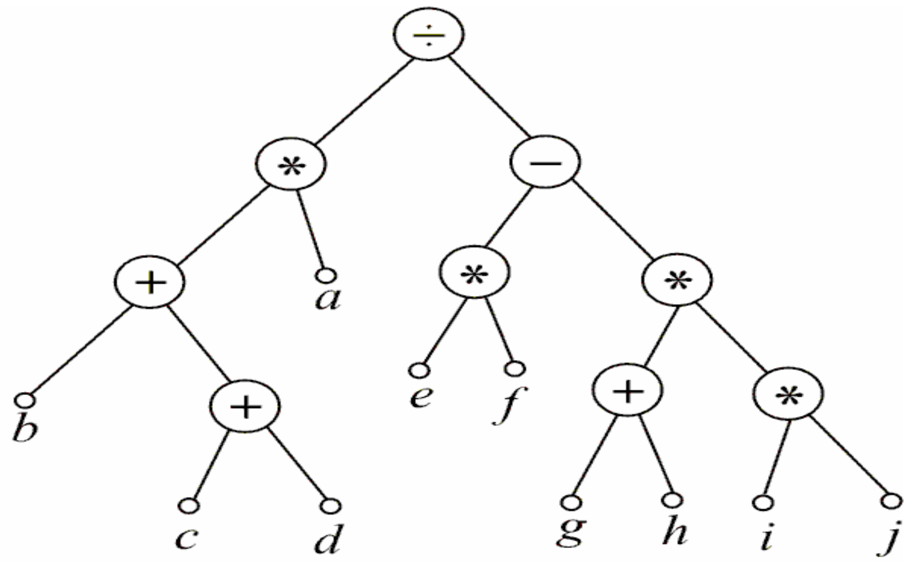
$b\ ((f\ g\ \underline{d})\ e\ \underline{c})\ \underline{a}$



用2叉有序正则树存放算式

存放规则

- 最高层次运算放在树根
- 后依次将运算符放在根子树的根上
- 数放在树叶上
- 规定：被除数、被减数放在左子树树叶上



算式 $((b+(c+d))*a)\div((e*f)-(g+h)*(i*j))$ 存放在图所示2叉树上。

波兰符号法

波兰符号法(Polish Notation)

按前序行遍法访问存放算式的2叉有序正则树，其结果不加括号，规定每个运算符号与其后面紧邻两个数进行运算，运算结果正确。称此算法为波兰符号法或前缀符号法。对上图的访问结果为

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

逆波兰符号法(Reverse Polish Notation)

按后序行遍法访问，规定每个运算符与前面紧邻两数运算，称为逆波兰符号法或后缀符号法。对上图的访问结果为

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