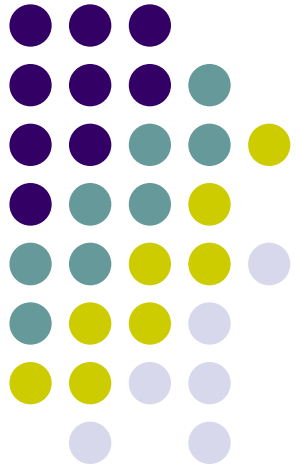
树的应用

离散数学—树

南京大学计算机科学与技术系

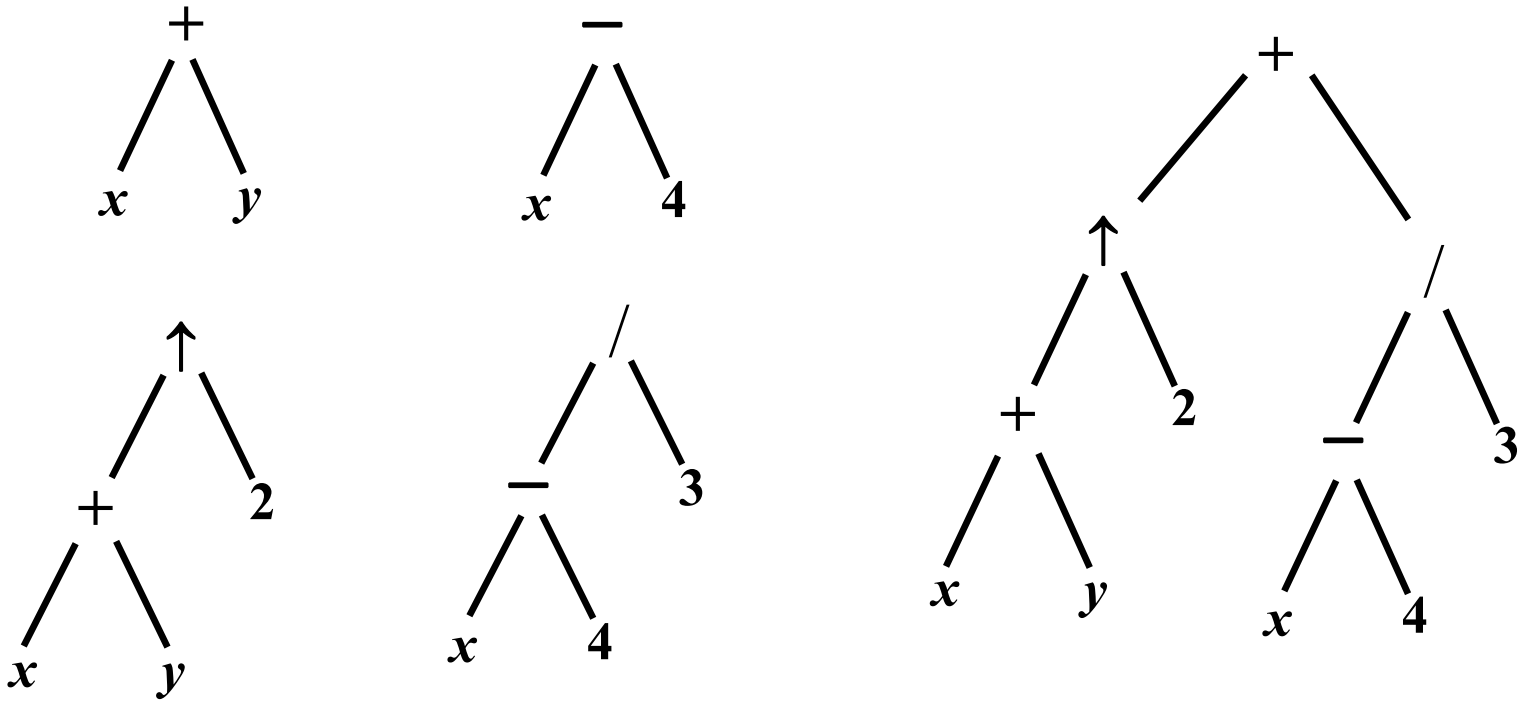


内容提要

- 表达式的（逆）波兰记法
- 二叉搜索树
- 决策树
- 前缀码
- **Huffman编码（算法）**

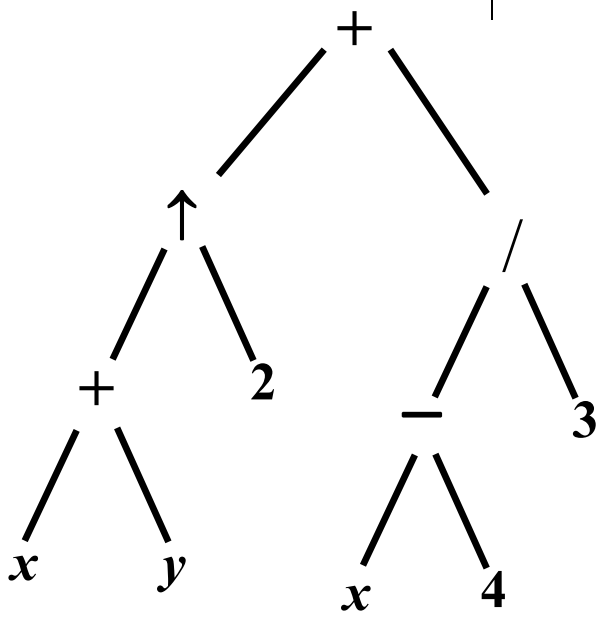
表达式的根树表示

- 用根树表示表达式：内点对应于运算符，树叶对应于运算分量。
- 举例：((x+y)↑2+ ((x-4)/3)



表达式的（逆）波兰表示法

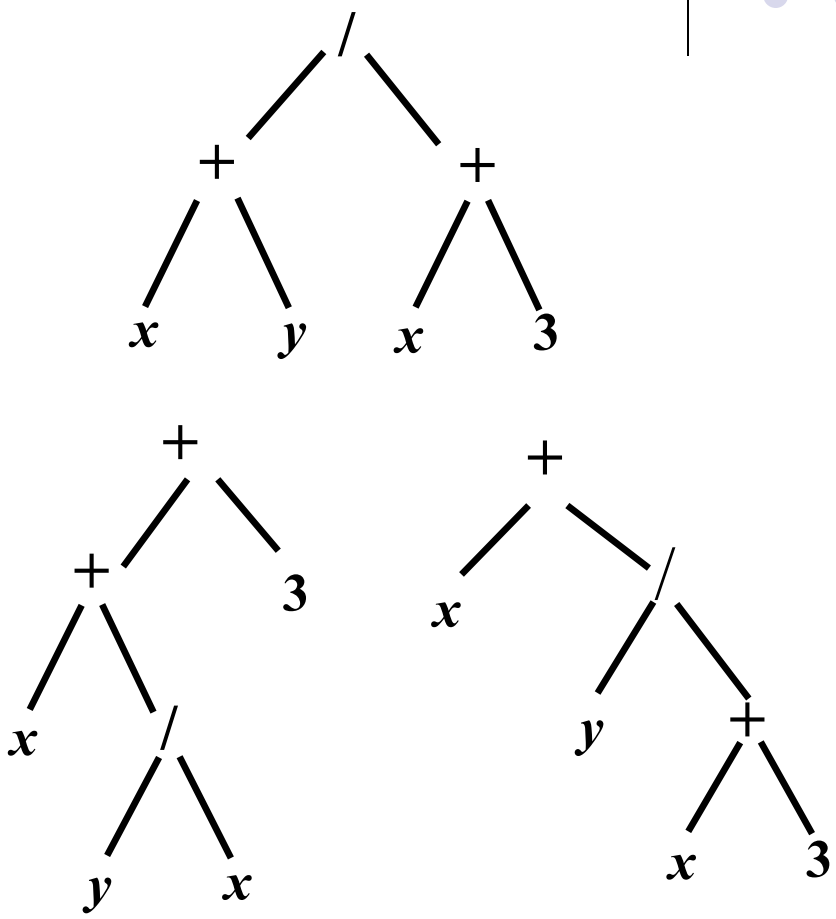
- **(*x*+*y*)↑2+ ((*x*-4)/3)**
- 前缀形式（波兰表示法）
  - **+↑+*xy*2 /-*x*4 3**
- 后缀形式（逆波兰表示法）
  - ***xy*+2↑ *x*4- 3/+**
- 中缀形式
  - ***x*+*y*↑2+ *x*-4/3**



中缀表示法的缺陷

- 中缀形式：*x*+*y*/*x*+3
  - 有3种解释:
  - (*x*+*y*)/(*x*+3)
  - *x*+*y*/*x*+3
  - *x*+*y*/(*x*+3)

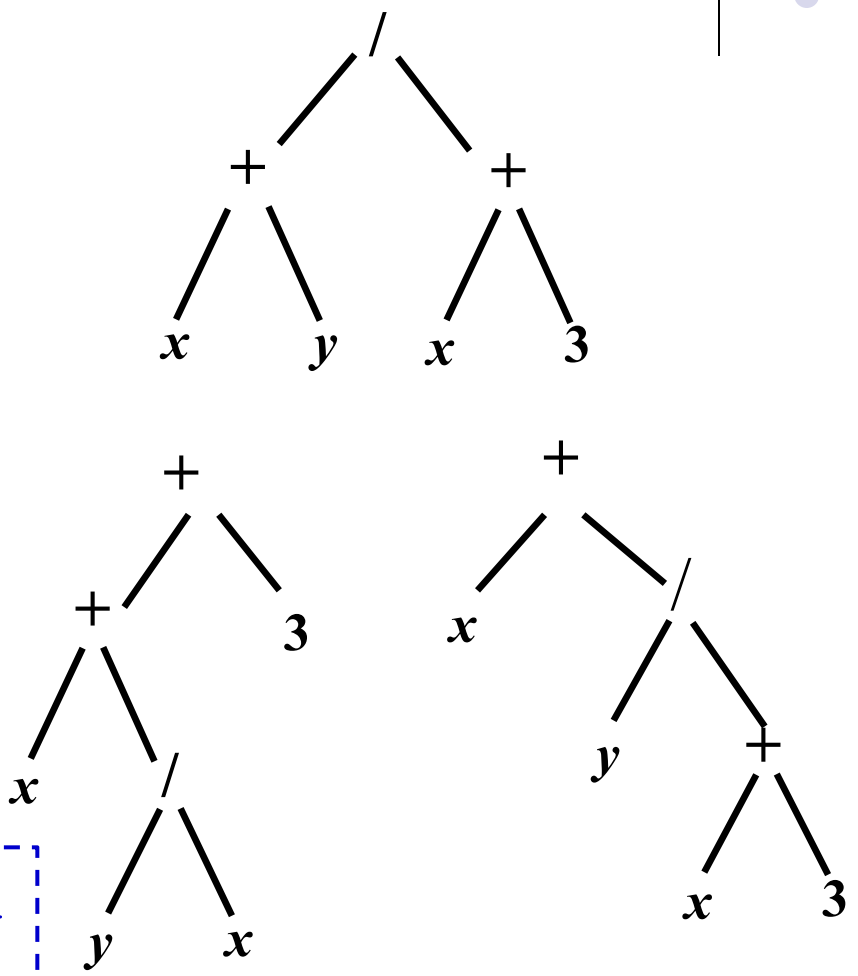
不同的根树有相同的中缀形式。



前缀与后缀则有一定的唯一性。(p. 565: 26-27)

前缀表示法（波兰表示法）

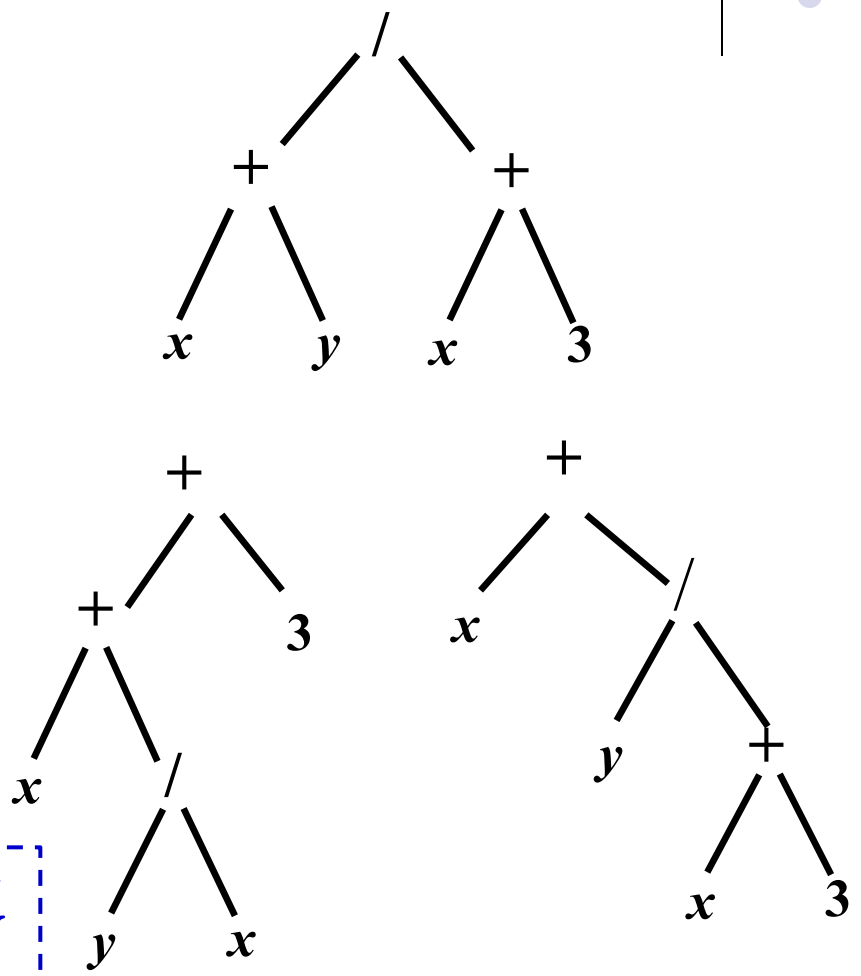
- (*x*+*y*)/(*x*+3)
  - /+*xy*+*x*3
- *x*+*y*/*x*+3
  - ++*x*/*yx*3
- *x*+*y*/(*x*+3)
  - +*x*/*y*+*x*3



从右向左，遇到运算符，对右边紧接着的2个运算对象进行运算

后缀表示法（逆波兰表示法）

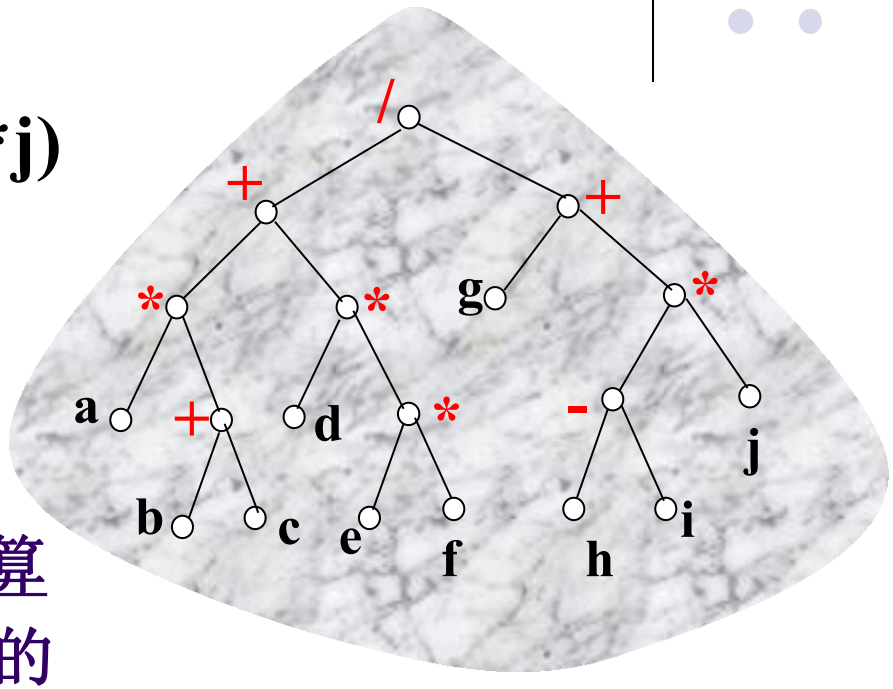
- $(x+y)/(x+3)$
  - $xy+x3+/$
- $x+y/x+3$
  - $xyx/+3+$
- $x+y/(x+3)$
  - $xyx3+/+$



从左向右，遇到运算符，对左边紧接着的2个运算对象进行运算

后缀表示法（逆波兰表示法）

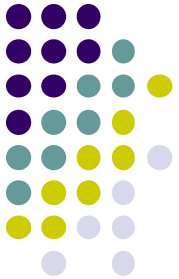
- **(a*(b+c)+d*(e*f))/(g+(h-i)*j)**
- 逆波兰表示：
  - **abc+*def**+ghi-j*+/**



从左往右，遇到运算符，根据运算符所需运算分量个数确定前面的元素作为运算分量。

不需要括弧唯一地表示计算顺序。

后缀表达式求值



7 2 3 * - 4 ↑ 9 3 / +

7 6 - 4 ↑ 9 3 / +

1 4 ↑ 9 3 / +

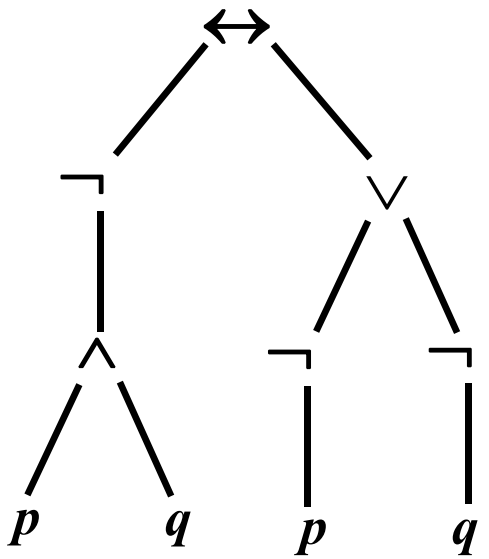
1 9 3 / +

1 3 +

4

复合命题的根树表示

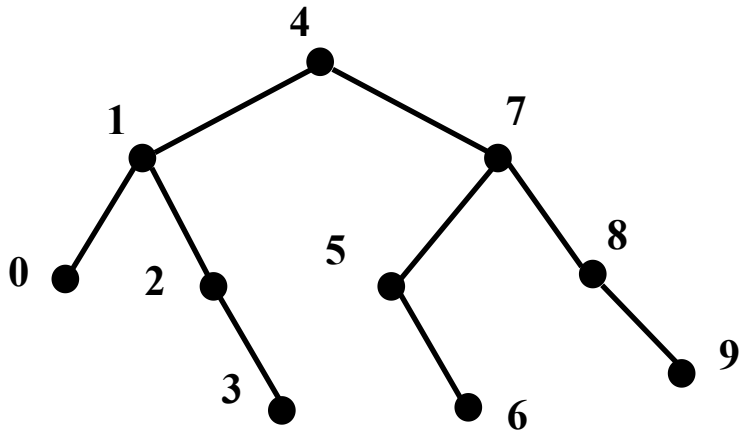
命题：$(\neg(p \wedge q)) \leftrightarrow (\neg p \vee \neg q)$



后缀形式：$pq\wedge\neg p\neg q\neg\vee\leftrightarrow$

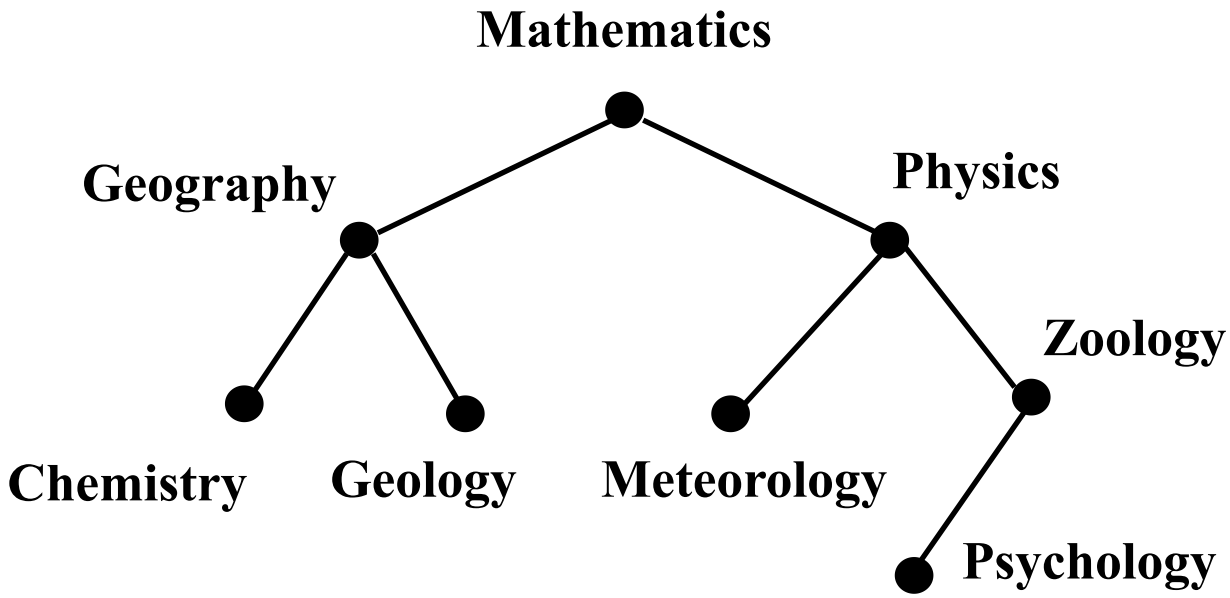
二叉搜索树

- 二叉搜索树满足下列条件
  - 二叉树，各顶点的子女非左即右，左或右都不超过一个。
  - 每个顶点有一个唯一的标号，该标号取自一个全序集。
  - 若*u*是树中任意的顶点，则：
    - *u* 的左子树中任意顶点的标号小于*u* 的标号。
    - *u* 的右子树中任意顶点的标号大于*u* 的标号。



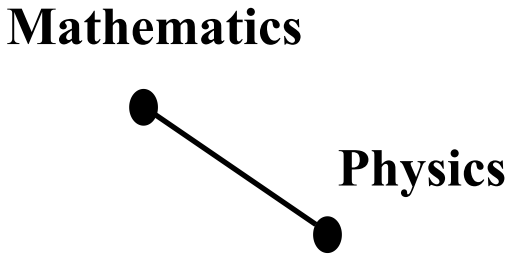
构造二叉搜索树（举例）

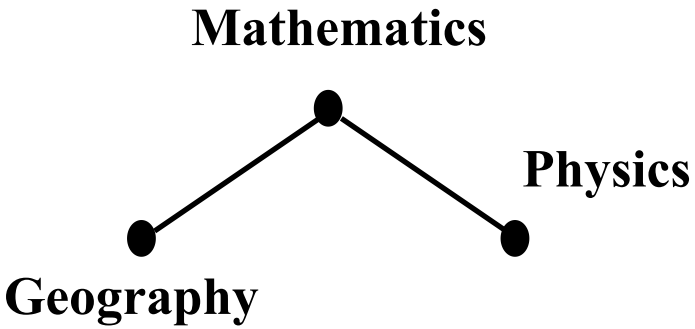
mathematics, physics, geography, zoology, meteorology, geology, psychology, chemistry

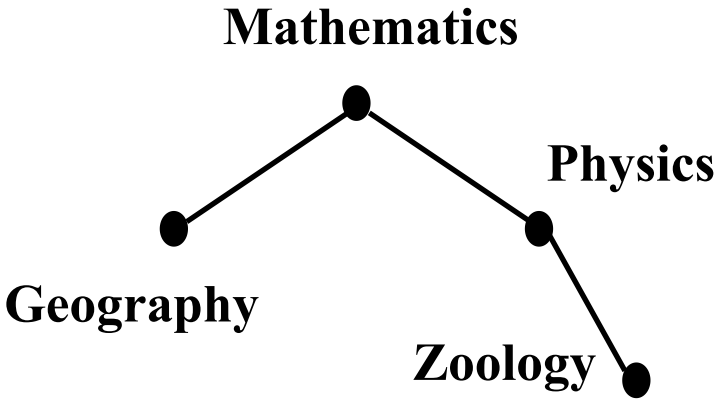


构造二叉搜索树（举例）

mathematics, physics, geography, zoology, …







二叉搜索树算法

Procedure insertion(T: binary search tree, *x*:item) //定位或添加

v:=root of T //*v*可能为null

if *v*=null **then** add a vertex to the tree and label it with *x*

while *v* !=null and label(*v*) != *x* {

if *x* < label(*v*) **then**

if left child of *v* !=null then *v*:= left child of *v*

else add *new vertex* as a left child of *v* and set *v*:=null

else

if right child of *v* !=null then *v*:= right child of *v*

else add *new vertex* as a right child of *v* and set *v*:=null

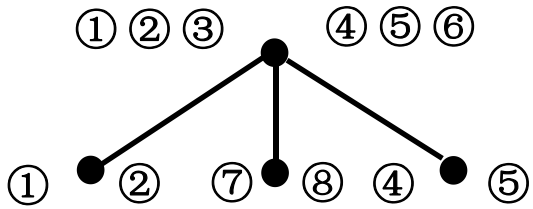
}

if *v* is null **then** label *new vertex* with *x* and let *v* be this *new vertex*

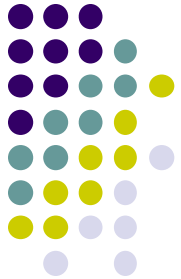
return *v*

决策树

- 这样的根树，每个内点对应一次决策，子树对应于该决策的后果。根到树叶的通路为一个解。
- 举例：8枚硬币，其中7个等重，一个重量较轻的是伪币，使用天平找出伪币，至少多少次称重？
  - 3元树，至少2次称重才能确保找到。

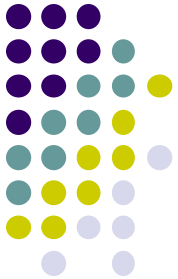


决策树



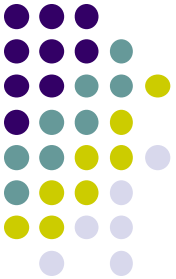
- 以决策树为模型，排序算法最坏情形复杂性的下界。
- 基于二叉比较的排序算法至少需要$\lceil \log n! \rceil$次比较。
  - $n!$个树叶，其二叉树的高度至少为$\lceil \log n! \rceil$
  - $\Omega(n \log n)$

编码



- 如何从信号流中识别字符
  - 等长度编码 vs. 不等长度编码
- 例子：对包含{a(45),b(13),c(12),d(16),e(9),f(5)}6个字符的10万个字符的数据文件编码，每个字符后面的数字表示该字符出现的频率(%)。
  - 编码方案一：a(000), b(001), c(010), d(011), e(100), f(101); 则文件总长度30万字位。
  - 编码方案二：a(0), b(101), c(100), d(111), e(1101), f(1100); 则文件总长度22.4万字位，空间节省四分之一。

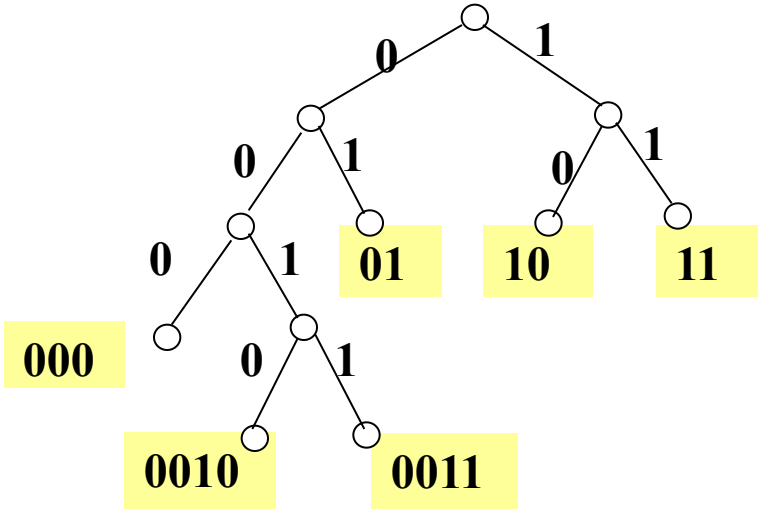
不等长编码的分隔



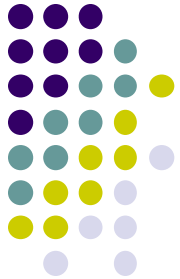
- 如何从信号流中识别不等长编码表示的字符
  - 显式表示长度：专用位或特定结束信号
  - 匹配的唯一性（比如，前缀码）
- 如果符号串α可以表示成符号串β_1和β_2的并置，则β_1称为α的一个前缀。(注意：β_1和β_2可以是空串。)
- 设$A=\{\beta_1,\beta_2,\ldots,\beta_m\}$是符号串的集合，且对任意$\beta_i,\beta_j \in A$, 若$i \neq j$, β_i与β_j互不为前缀，则称A为前缀码。
- 若A中的任意串β_i只含符号0, 1, 则称A是二元前缀码。

用二叉树生成二元前缀码

- 生成方法
  - 给边标号：对内点，对其出边标上号，左为0，右为1。
  - 给叶编号：从根到每个树叶存在唯一的通路，构成该通路的边的标号依次并置，所得作为该树叶的编号。
- 给定一棵完全二叉树，可以产生唯一的二元前缀码。



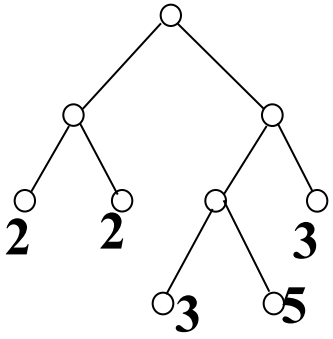
最优前缀码



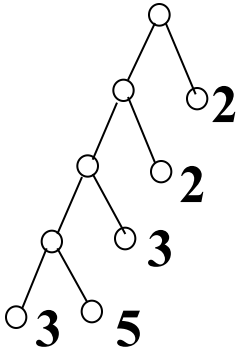
- 问题：二进前缀码$A=\{\beta_1,\beta_2,\ldots,\beta_m\}$表示m个不同的字母，如果各字母使用频率不同，如何设计编码方案可以使总传输量最少。
- 基本思想：使用频率高的字母用尽量短的符号串表示。
- 问题的解：若用频率(相对值)作为树叶的权，最佳二元前缀码对应的二叉树应该是最优二叉树。

最优二叉树

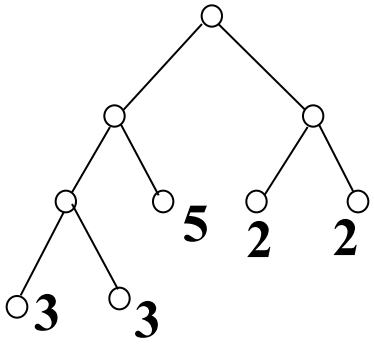
- 若T是二叉树，且每个叶$v_1,v_2,\ldots,v_t$带数值权$w_1,w_2,\ldots w_t$, 则二叉树T的权$W(T)$定义为： $\Sigma^t_{i=1} w_i l(v_i)$, 其中：$l(v_i)$表示$v_i$的层数。
- 具有相同权序列的二叉树中权最小的一棵树称为最优二叉树。



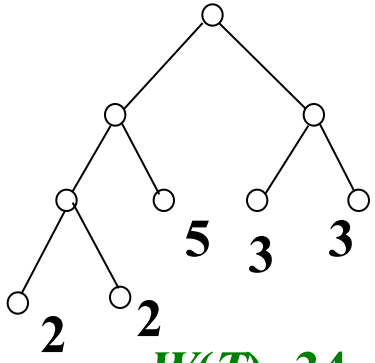
$W(T)=38$



$W(T)=47$



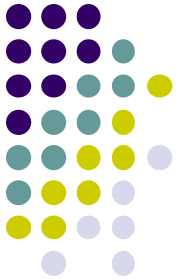
$W(T)=36$



$W(T)=34$

注意：最优二叉树一定是完全二叉树（t≥2）

Huffman编码（算法）

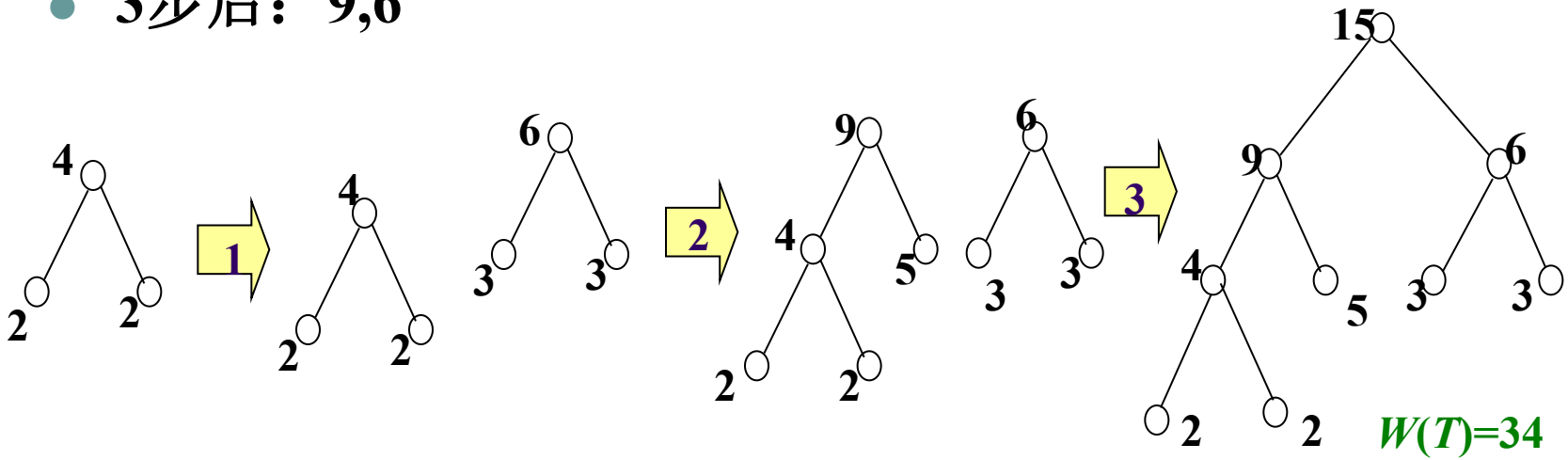


- 输入：正实数序列$w_1,w_2,\ldots,w_t$。
- 输出：具有t个树叶，其权序列为$w_1,w_2,\ldots,w_t$的最优二叉树。
- 过程：
  - T棵根树（森林），其根的权分别是$w_1,w_2,\ldots,w_t$。
  - 选择根权最小的两棵树，以它们为左、右子树（合并）生成新的二叉树，其根权等于2棵子树的根权之和。
  - 重复第2步，直至形成一棵树。

注意：结果可能不唯一(如果“当前”权最小顶点对不唯一)。

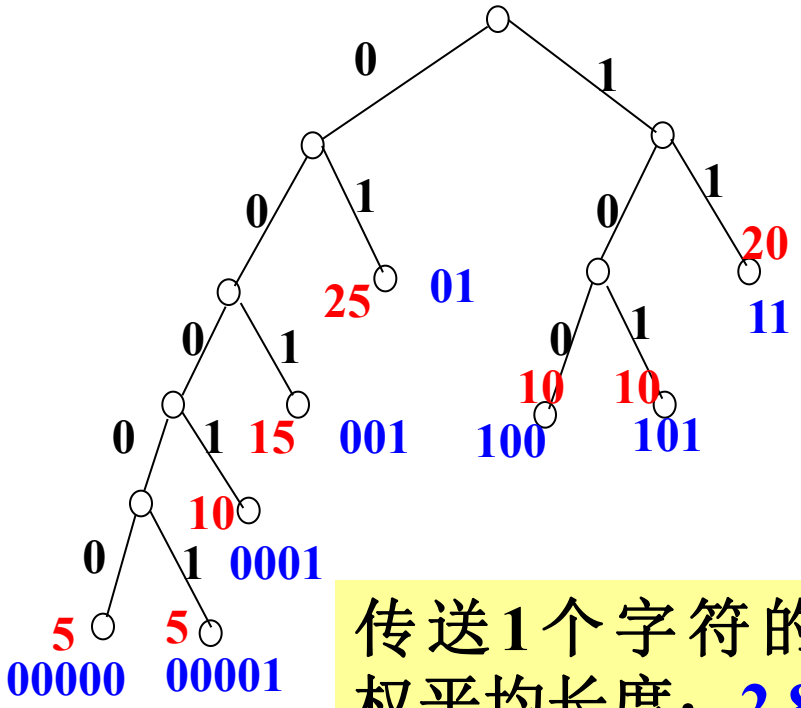
Huffman编码（算法）：举例

- 例子：开始序列：2,2,3,3,5
  - 1步后：4,3,3,5
  - 2步后：4,6,5
  - 3步后：9,6



一个应用示例

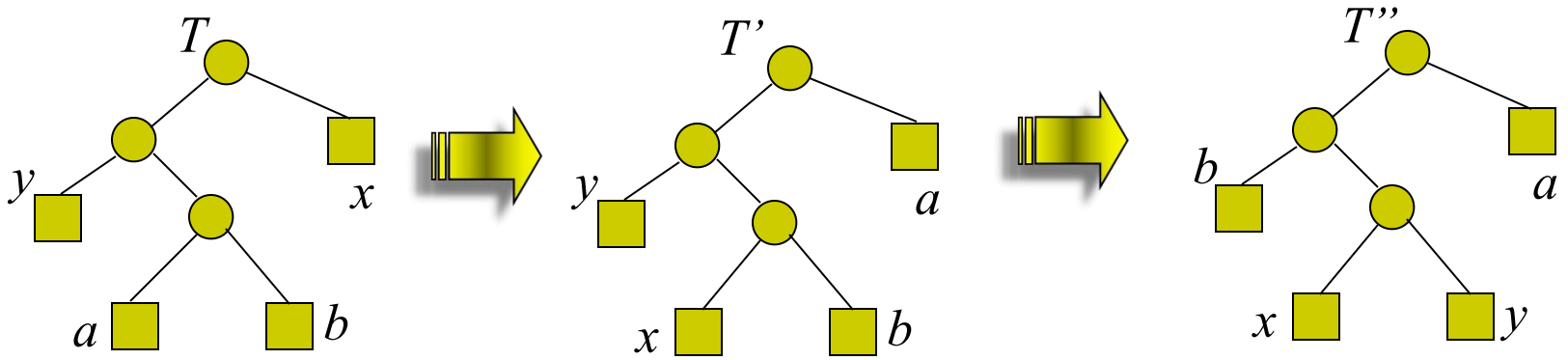
- 八个字符的传输问题
  - 假设八个字符的频率如下：
    - 0：25%　1: 20%
    - 2: 15%　3: 10%
    - 4: 10%　5: 10%
    - 6: 5%　7: 5%
  - 则对应的权为：
    - 5,5,10,10,10,15,20,25



传送1个字符的加权平均长度：2.85

Huffman算法的正确性

设C是字母表，其中每个字符c的频率为$f[c]$。若x,y是两个频率最小的字符，则必存在C的一种最优前缀码，使得x,y的编码仅有最后一位不同。



T为任意最优前缀码

在上图的变换中，二叉树的权保持不变，即：$W(T) \geq W(T') \geq W(T'') \geq W(T)$

保持权不变的变换

不妨假设$f[a] \le f[b], f[x] \le f[y]$; 于是$f[x] \le f[a], f[y] \le f[b]$

$$W(T) - W(T') = \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c)$$

$$= f[x]d_T(x) + f[a]d_T(a) - f[x]d_{T'}(x) - f[a]d_{T'}(a)$$

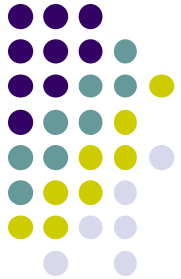
$$= f[x]d_T(x) + f[a]d_T(a) - f[x]d_T(a) - f[a]d_T(x)$$

$$= (f[a] - f[x])(d_T(a) - d_T(x)) \ge 0$$

$\therefore W(T) \ge W(T')$; 同理, $W(T') \ge W(T'')$; 但$W(T)$最小

$\therefore W(T) = W(T') = W(T'')$

Huffman算法的正确性（续）



C是字母表，$f[c]$为字符c的频率，x,y是两个频率最小的字符。令C'=C-{x,y}∪{z}, $f[z]=f[x]+f[y]$, 若T'是C'的最优二叉树，则将顶点z替换为分支点，并以x,y为其子女，所得T是C的一棵最优二叉树。

$$d_T(x) = d_T(y) = d_{T'}(z) + 1,$$

因此, $f[x]d_T(x) + f[y]d_T(y) = (f[x] + f[y])(d_{T'}(z) + 1)$

$= f[z]d_{T'}(z) + (f[x] + f[y])$

于是, $\mathrm{W}(T) = W(T') + (f[x] + f[y])$

如果存在 T'' 满足 $W(T'') < W(T)$, 不失一般性，x与y在T''中为siblings.

将x, y连同它们的父结点替换 为一叶结点z, 并令$f[z] = f[x] + f[y]$,

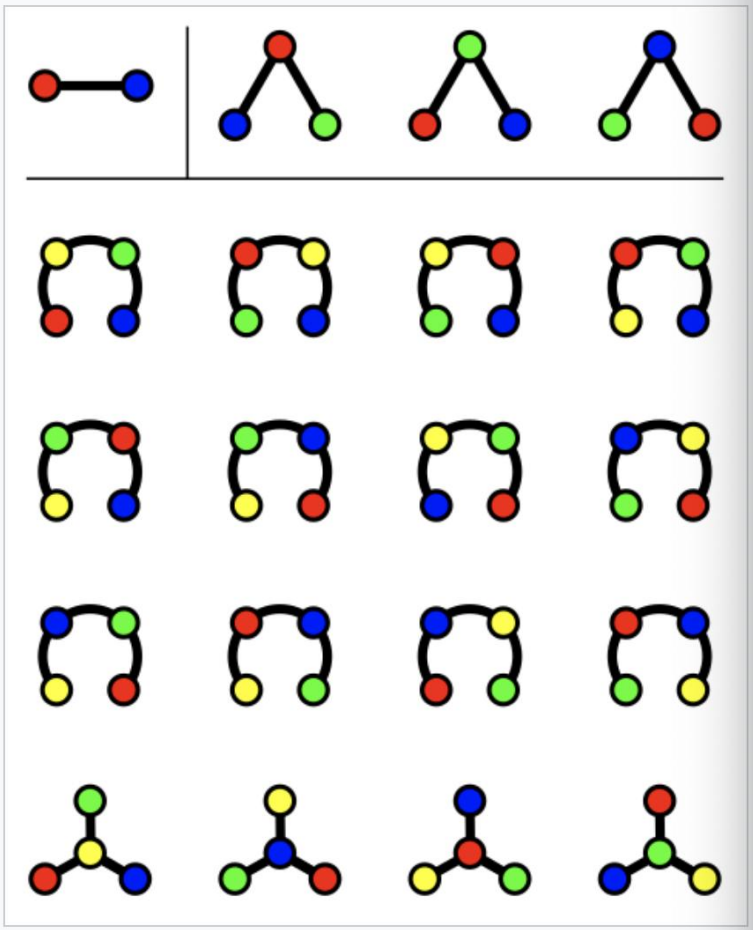
设得到的新树为 T''', 则：

$W(T''') = W(T'') - f[x] - f[y] < W(T) - f[x] - f[y] = W(T')$, 矛盾。

N个不同点，可以构成多少个不同的树？

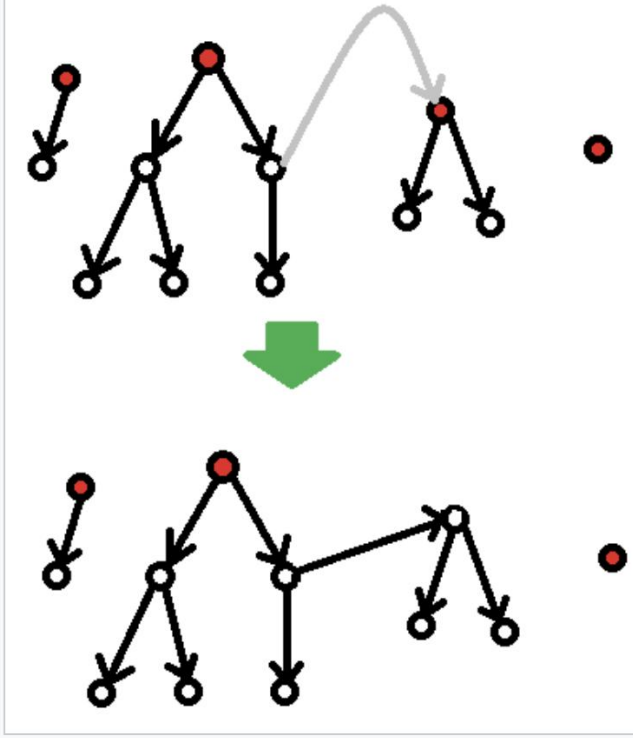
What is the number T_n of different trees that can be formed from a set of n distinct vertices? Cayley's formula gives the answer $T_n = n^{n-2}$. Aigner & Ziegler (1998) list four proofs of this fact; they write of the fourth, a double counting proof due to Jim Pitman, that it is "the most beautiful of them all."

Pitman's proof counts in two different ways the number of different sequences of directed edges that can be added to an empty graph on n vertices to form from it a rooted tree. One way to form such a sequence is to start with one of the T_n possible unrooted trees, choose one of its n vertices as root, and choose one of the $(n-1)!$ possible sequences in which to add its $n-1$ (directed) edges. Therefore, the total number of sequences that can be formed in this way is $T_n n(n-1)! = T_n n!$.



Cayley's formula implies that there is
$1 = 2^{2-2}$ tree on two vertices,
$3 = 3^{3-2}$ trees on three vertices, and
$16 = 4^{4-2}$ trees on four vertices.

Another way to count these edge sequences is to consider adding the edges one by one to an empty graph, and to count the number of choices available at each step. If one has added a collection of $n-k$ edges already, so that the graph formed by these edges is a rooted forest with k trees, there are $n(k-1)$ choices for the next edge to add: its starting vertex can be any one of the n vertices of the graph, and its ending vertex can be any one of the $k-1$ roots other than the root of the tree containing the starting vertex. Therefore, if one multiplies together the number of choices from the first step, the second step, etc., the total number of choices is



Adding a directed edge to a rooted forest

$$\prod_{k=2}^{n} n(k-1) = n^{n-1}(n-1)! = n^{n-2}n!.$$

$$T_n n! = n^{n-2} n! \qquad T_n = n^{n-2}.$$