树/Trees

11.1 树的概念/Introduction of Trees

11.2 树的应用/Applications of Trees

11.3 树的遍历/Tree Traversal

11.4 生成树/Spanning Trees

11.5 最小生成树/ minimum Spanning Trees

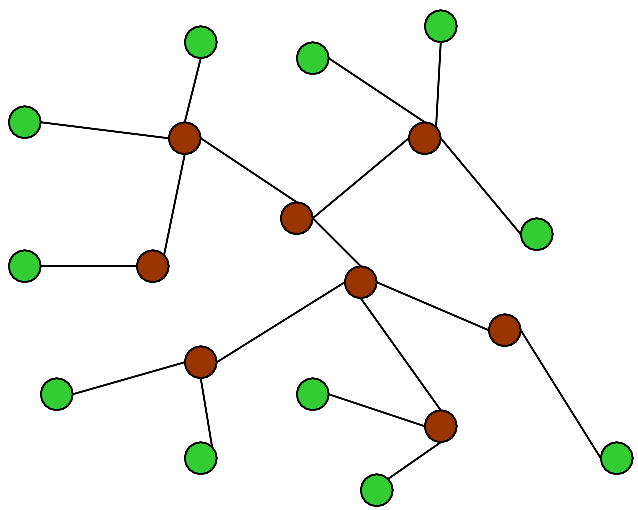
§11.1: Introduction to Trees
树的概念

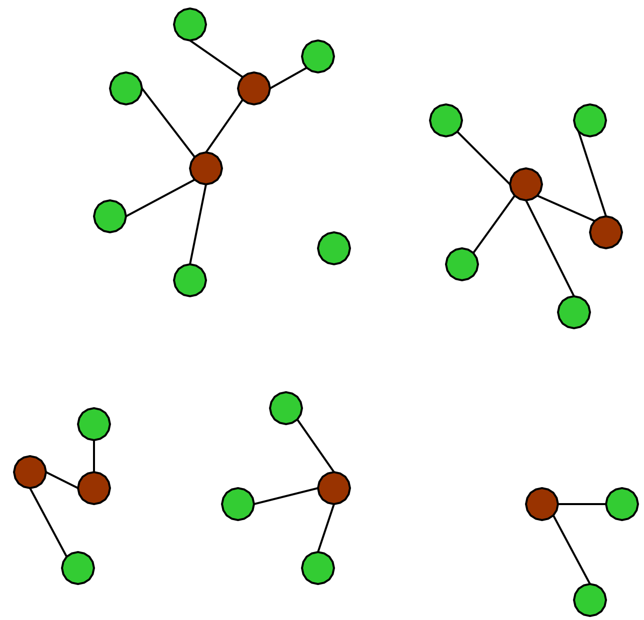
- A *tree* is a connected undirected graph with no simple circuits.
  - **Theorem:** There is a unique simple path between any two of its nodes.
- An undirected graph without simple circuits is called a *forest*.
  - You can think of it as a set of trees having disjoint sets of nodes.

Tree and Forest Examples

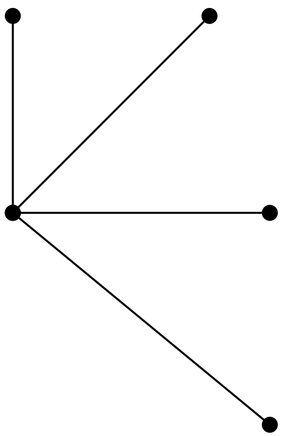
- A Tree:
- A Forest:

Leaves in green, internal nodes in brown.

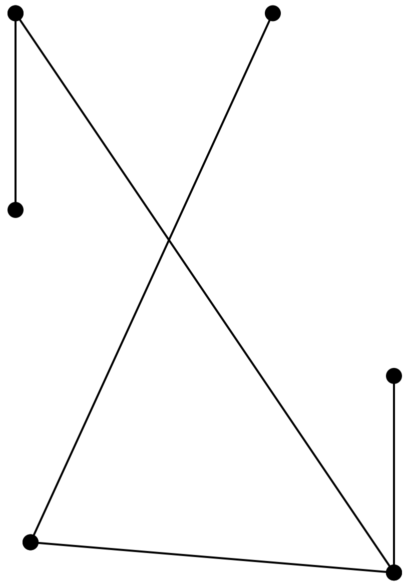




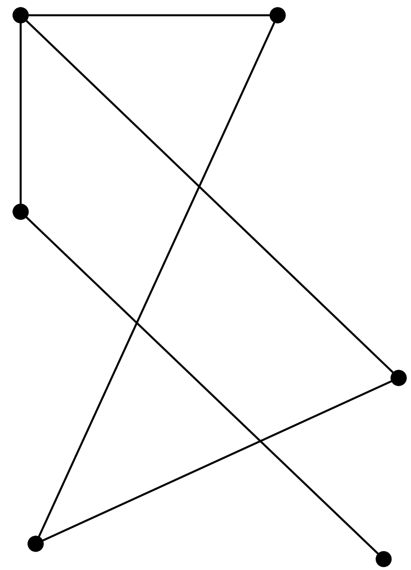
Examples of Trees and Graphs that are not Trees



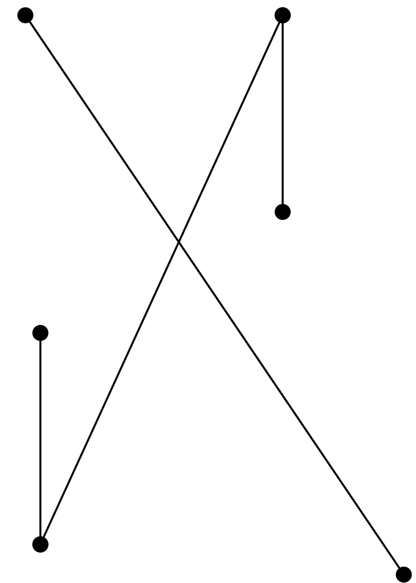
G1



G2



G3



G4

给定图T,以下关于树的定义是等价的:

- 无回路的连通图;
- 无回路且$e = v - 1$;
- 连通且$e = v - 1$;
- 无回路,但增加一条新边,得一个且仅一个回路;
- 连通但删去1条边后便不连通;
- 每一对结点间有一条且仅有一条通路。

树的基本性质

- 树是平面图。
- 树中任二顶点间都有唯一道路。
- 删除树的任一边，成了森林。
- 在树上任添一条边，产生唯一回路。
- 设T＝(V，E)，|V|＝v，|E|＝e，则$v = e + 1$。

- 树是平面图，且无回路，区域数（面数）$r = 1$，满足欧拉公式$v - e + r = 2$，
  - 即$v = e + 1$。
  - 也可用强归纳法证：（对v归纳）

① $\mathrm{v} \leq 2$，显然成立

② 假设$v < p$时成立，$v = p$时，先移去一条边，成了两棵树T_1和T_2，有$v_1 = e_1 + 1$，$v_2 = e_2 + 1$，且$e_1 + e_2 = e - 1$，故$v = v_1 + v_2 = e_1 + 1 + e_2 + 1 = (e_1 + e_2 + 1) + 1 = e + 1$。

Rooted Trees
有根树

- A *rooted tree* is a tree in which one node has been designated the *root*.
  - Every edge is (implicitly or explicitly) directed away from the root.

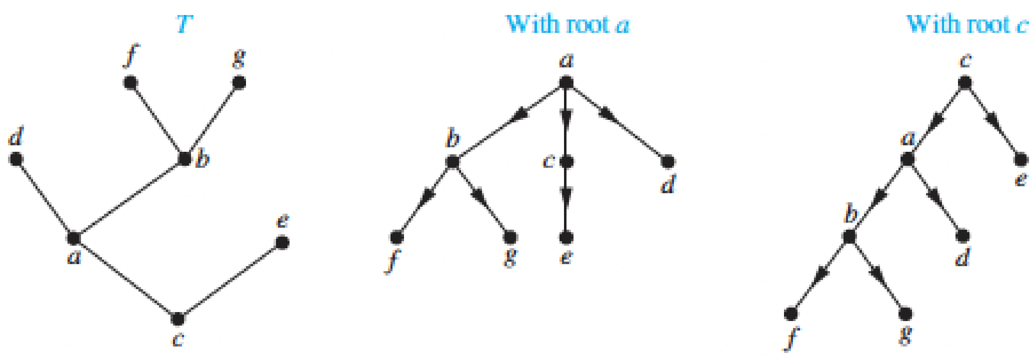
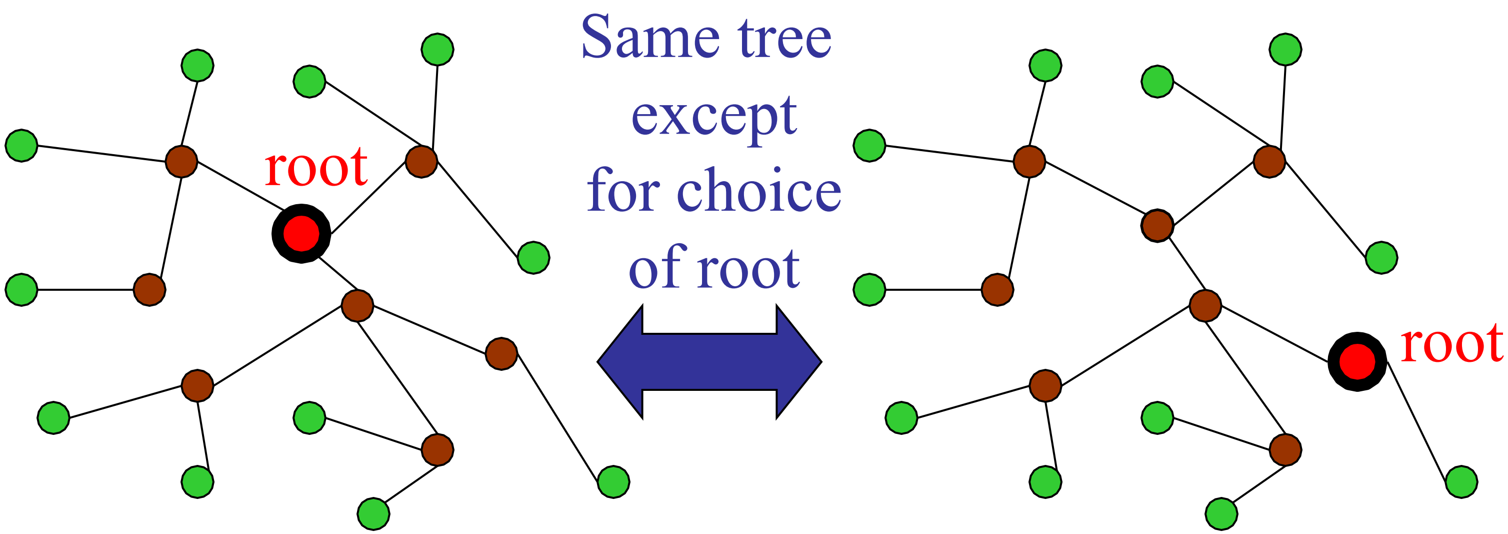


FIGURE 4 A tree and rooted trees formed by designating two different roots.

Rooted Tree Examples

- Note that a given unrooted tree with *n* nodes yields *n* different rooted trees.



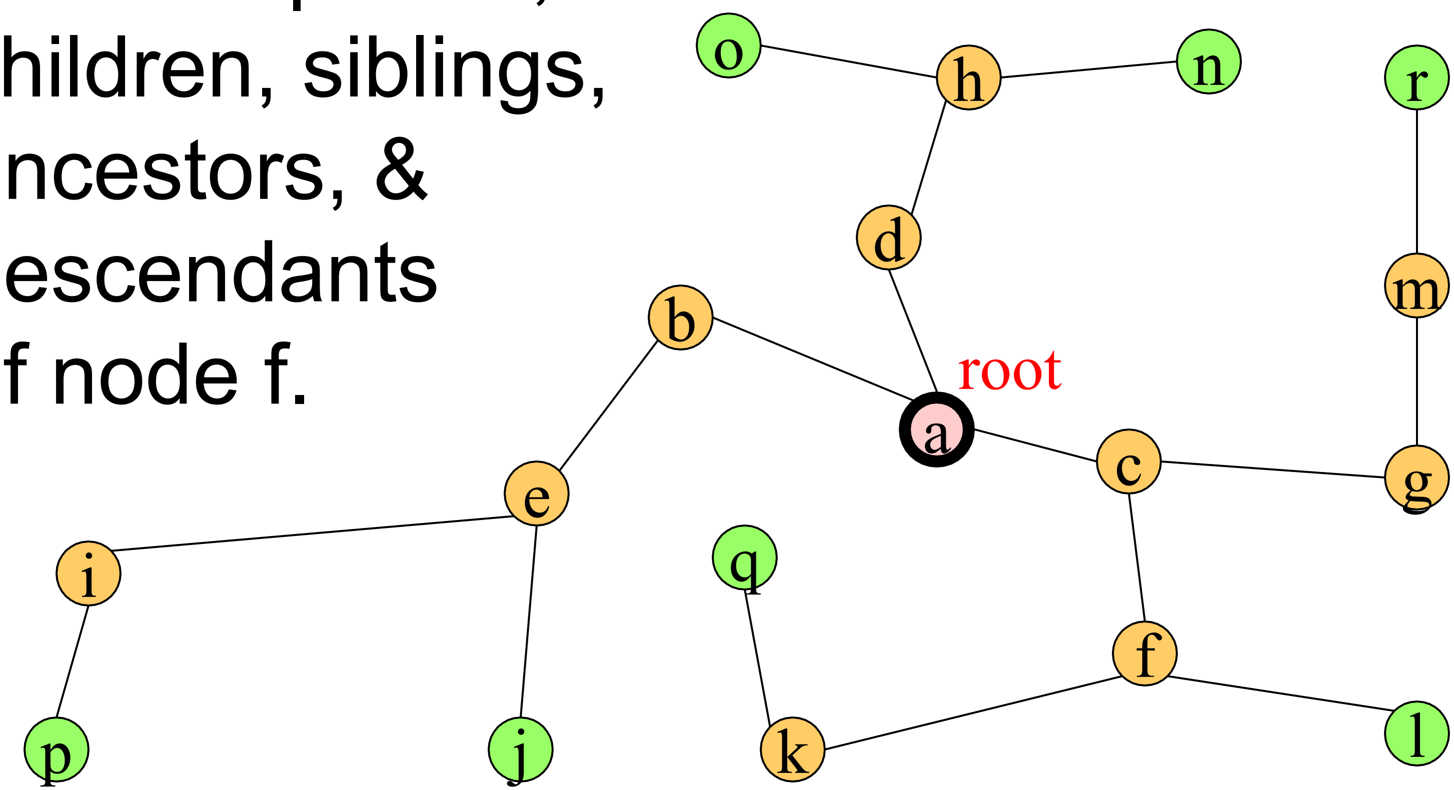
Rooted-Tree Terminology
有根树术语

- You should know the following terms about rooted trees:
  - Parent, child, siblings, ancestors, descendents, leaf, internal node（内点）, subtree.

- Parent, child, siblings：
  - rooted trees： $G=(V, E)$，if $(a, b) \in E$， then *a is the* parent of b， *b* is the child of a。
  - If $(a, b_1) \in E$， $(a, b_2) \in E$， then b_1 and b_2 are siblings。
- ancestors, descendents：

Rooted-Tree Terminology Exercise

- Find the parent, children, siblings, ancestors, & descendants of node f.



example 2

In the rooted tree T (with root a) shown in Figure 5, find the parent of c, the children of g, the
siblings of h, all ancestors of e, all descendants of b, all internal vertices, and all leaves. What
is the subtree rooted at g?

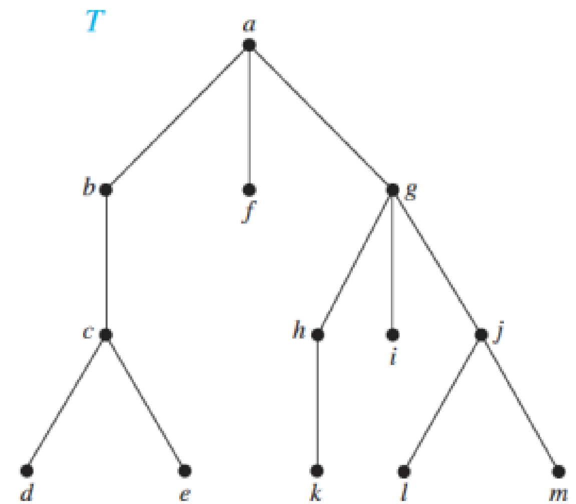


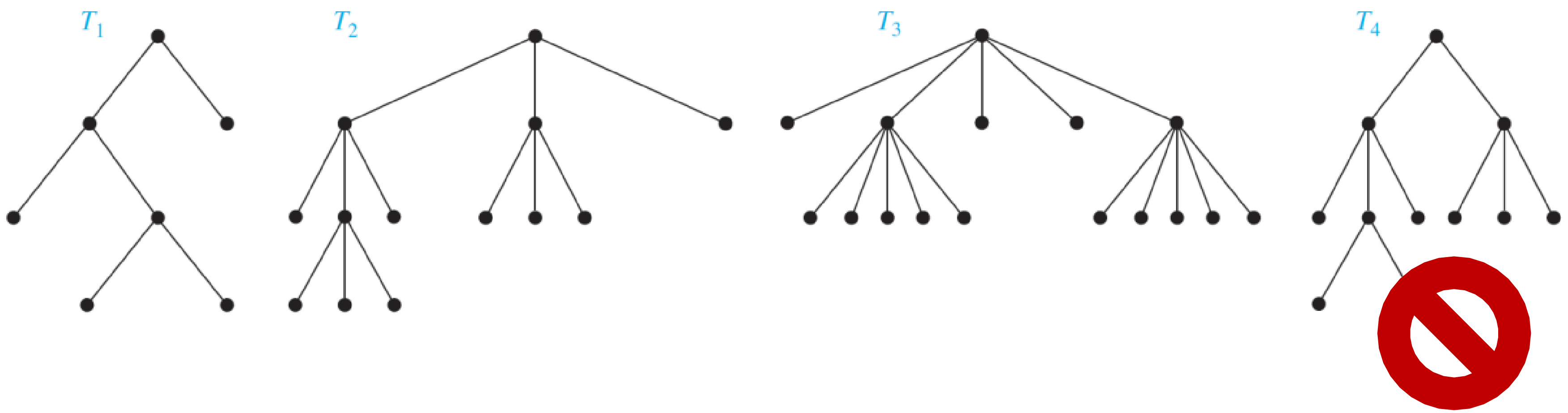
FIGURE 5 A rooted tree *T*.

Definition 3

- A rooted tree is called an m-ary tree if every internal vertex has no more than m children.

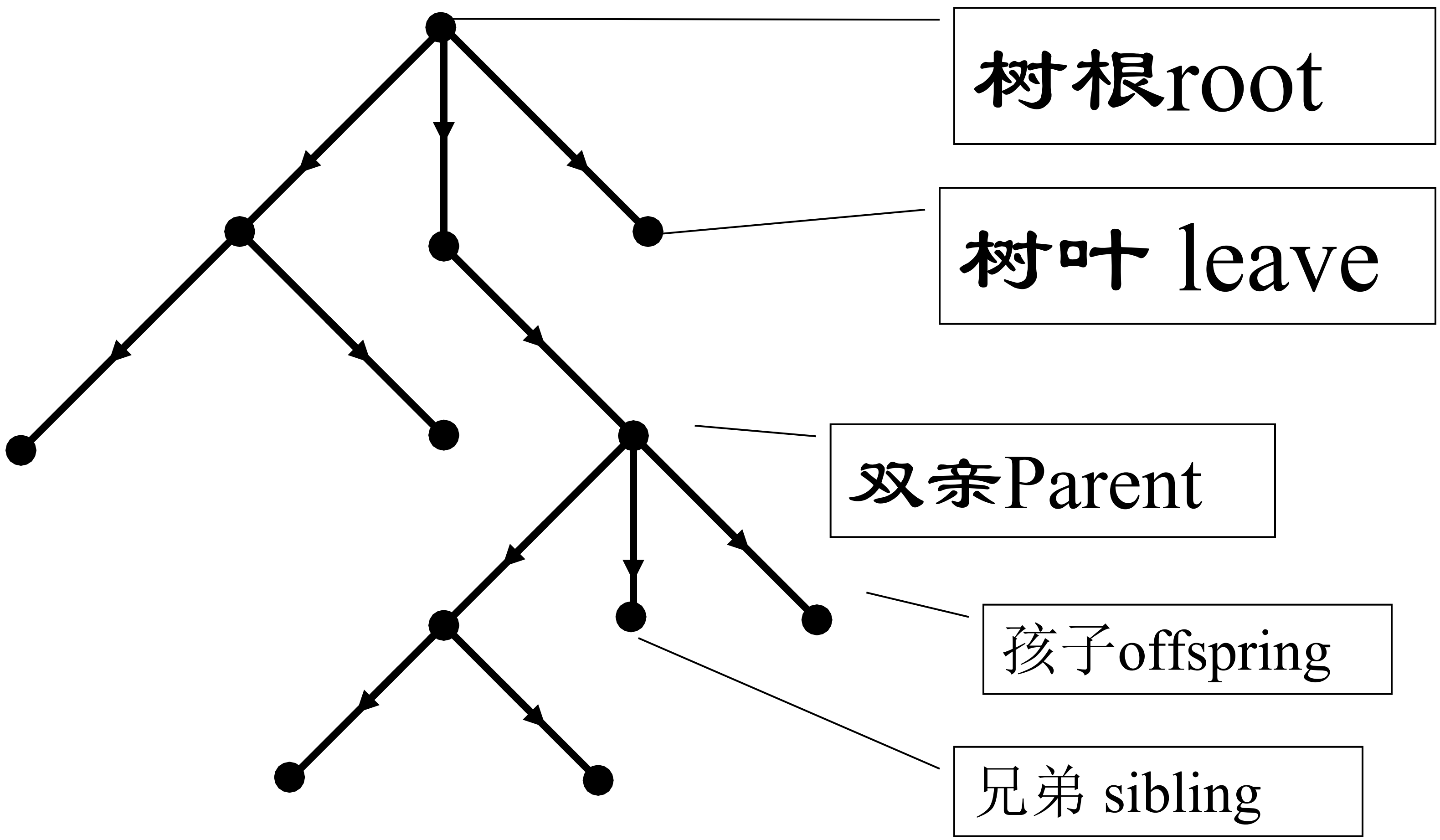
- The tree is called a full m-ary tree if every internal vertex has exactly m children. An m-ary tree with m = 2 is called a binary tree.

Example 3



Ordered Rooted Tree
有序根树

- A rooted tree where the children of each internal node（内点/枝点） are ordered.
- In ordered binary trees, we can define:
  - left child, right child
  - left subtree, right subtree
- For *n*-ary trees with *n*>2, can use terms like “leftmost”, “rightmost,” etc.


树根root
树叶 leave
双亲Parent
孩子offspring
兄弟 sibling

Example 4

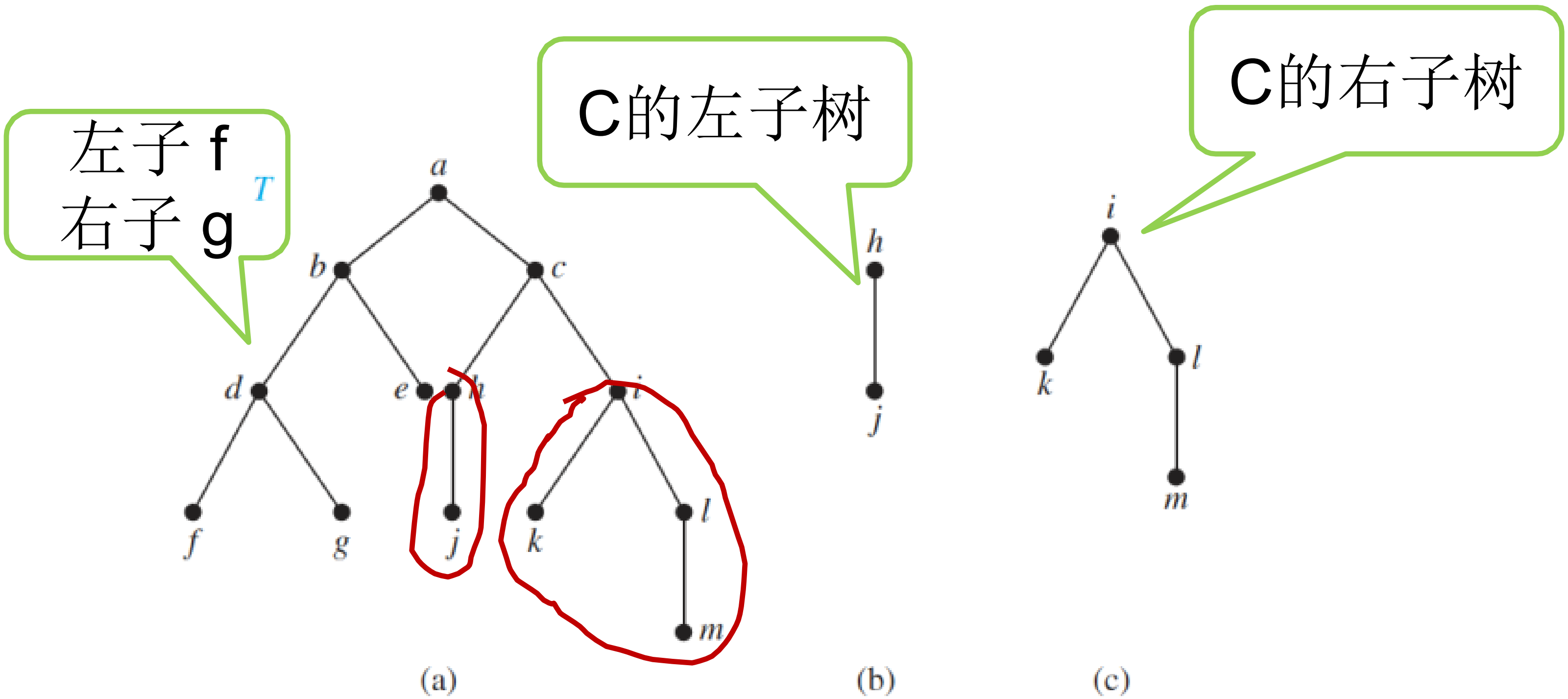
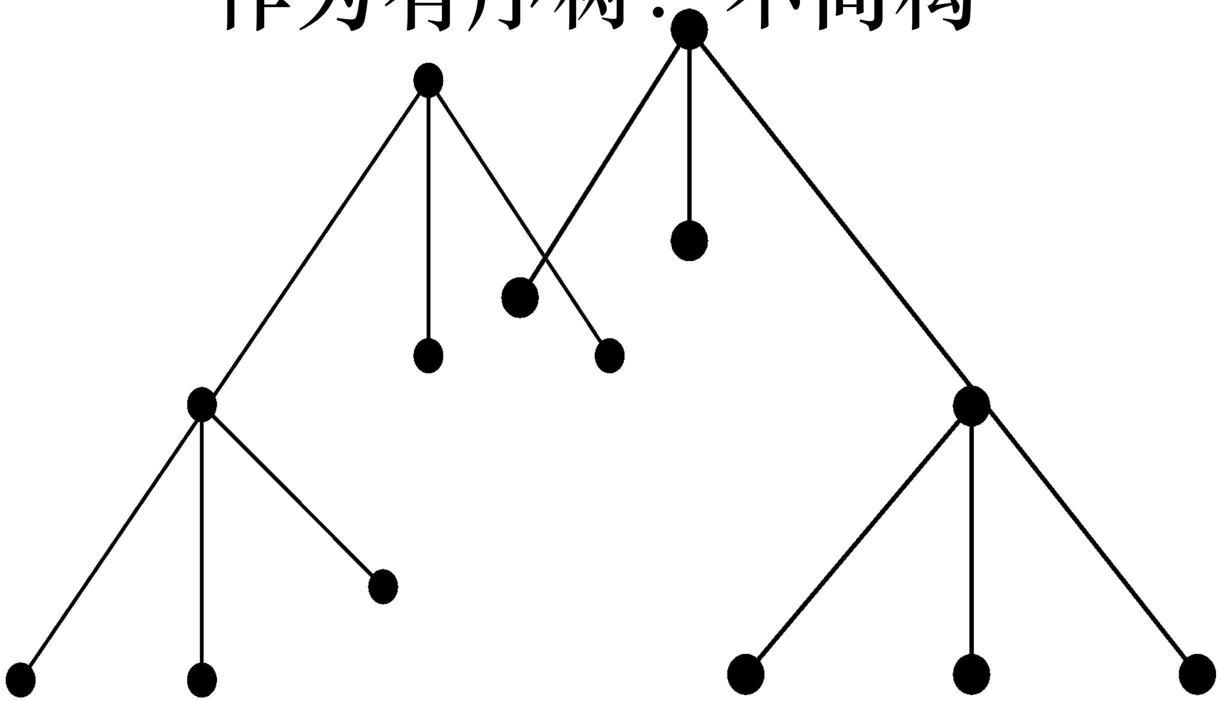


FIGURE 8 **A Binary Tree *T* and Left and Right Subtrees of the Vertex *c*.**

如下图

作为有根树：同构

作为有序树：不同构

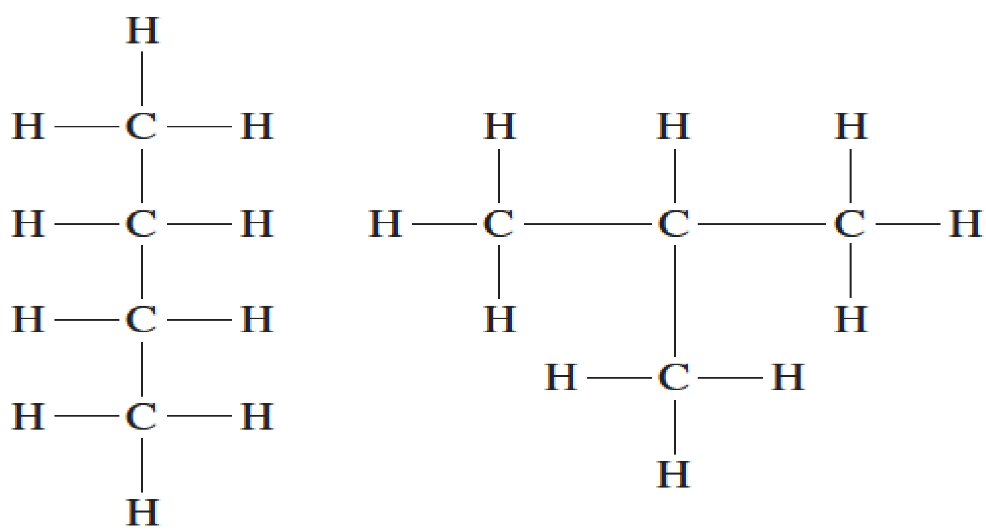


Trees as Models
树模型

- Can use trees to model the following:
  - Saturated hydrocarbons（饱和烃）
  - Organizational structures（组织架构）
  - Computer file systems（计算机文件系统）
- In each case, would you use a rooted or a non-rooted tree?

Example 5

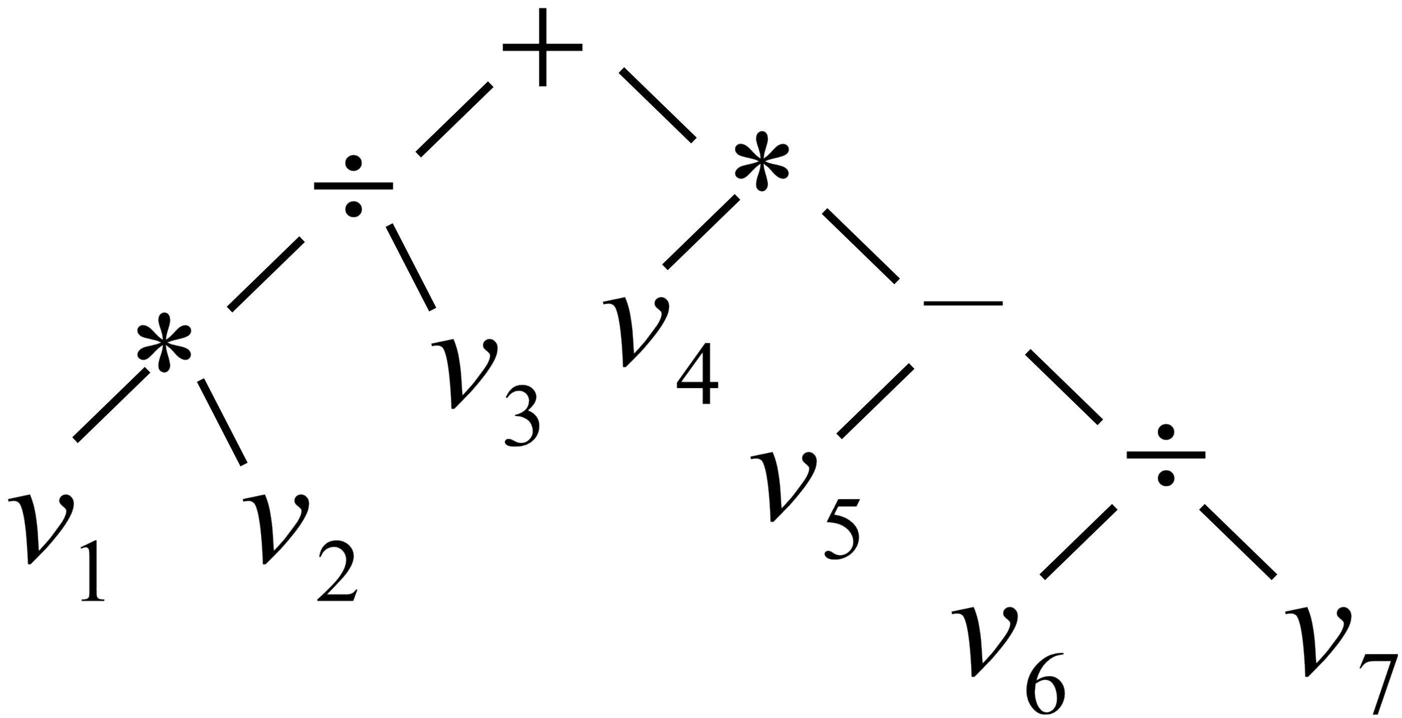
Saturated Hydrocarbons and Trees Graphs can be used to represent molecules, where atoms are represented by vertices and bonds between them by edges



Butane　　　　Isobutane

FIGURE 9 **The two isomers of butane.**

$$v_1 v_2 / v_3 + v_4 (v_5 - v_6 / v_7)$$



Trees as Models

- Organizational structures

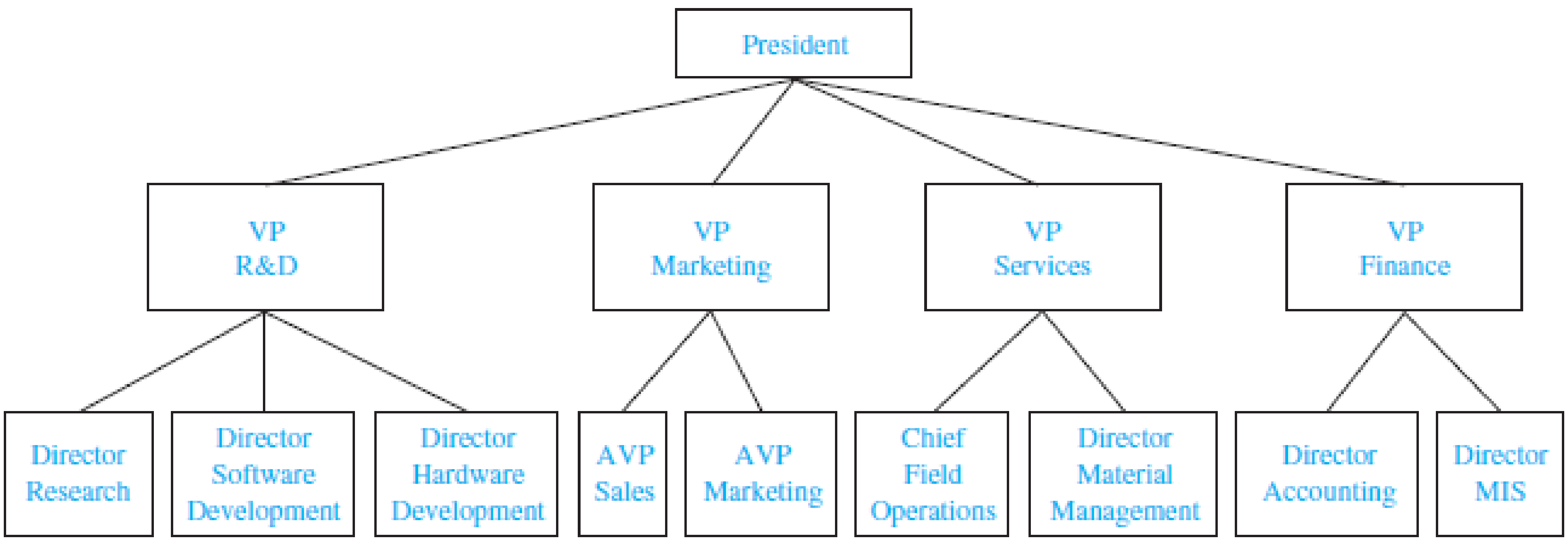


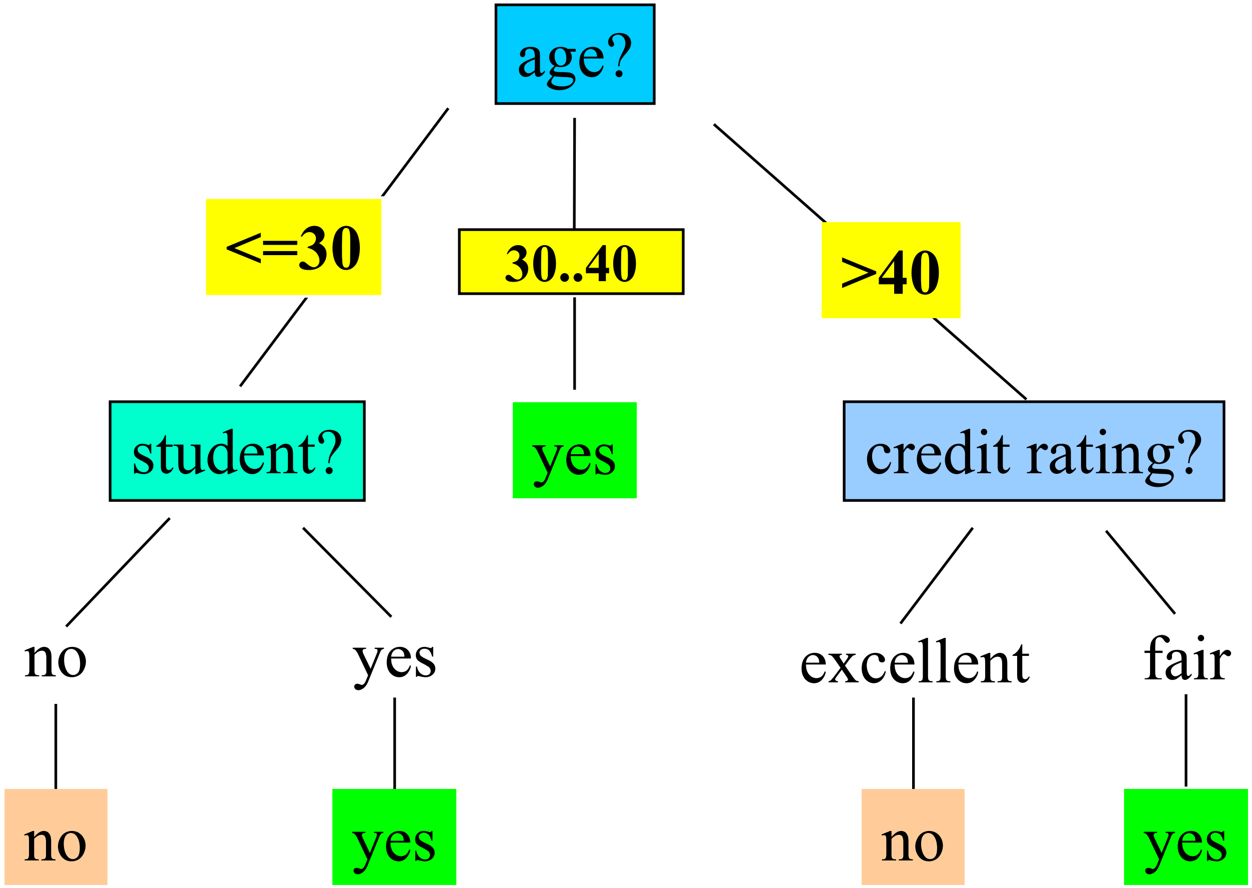
FIGURE 10 An Organizational Tree for a Computer Company.

Training Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



北京邮電大学
Beijing University of Posts and Telecommunications

Some Tree Theorems

- 2：A tree with n nodes has $n-1$ edges.
- 3：A full m-ary tree with i internal nodes has $n=mi+1$ nodes, and $\ell=(m-1)i+1$ leaves.
  - Proof: There are mi children of internal nodes, plus the root. And, $\ell = n-i = (m-1)i+1$. □

Some Tree Theorems

- 4： Thus, when m is known and the tree is full, we can compute all four of the values e, i, n, and ℓ, given any one of them.
  - n vertices： $i=(n-1)/m$, $l=[(m-1)n+1]/m$
  - i internal vertices： $n=mi+1$ ， $l=(m-1)i+1$
  - l leaves， ： $n=(ml-1)/(m-1)$, $i=(l-1)/(m-1)$

Example 9

- Suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to four other people. Some people do this, but others do not send any letters.

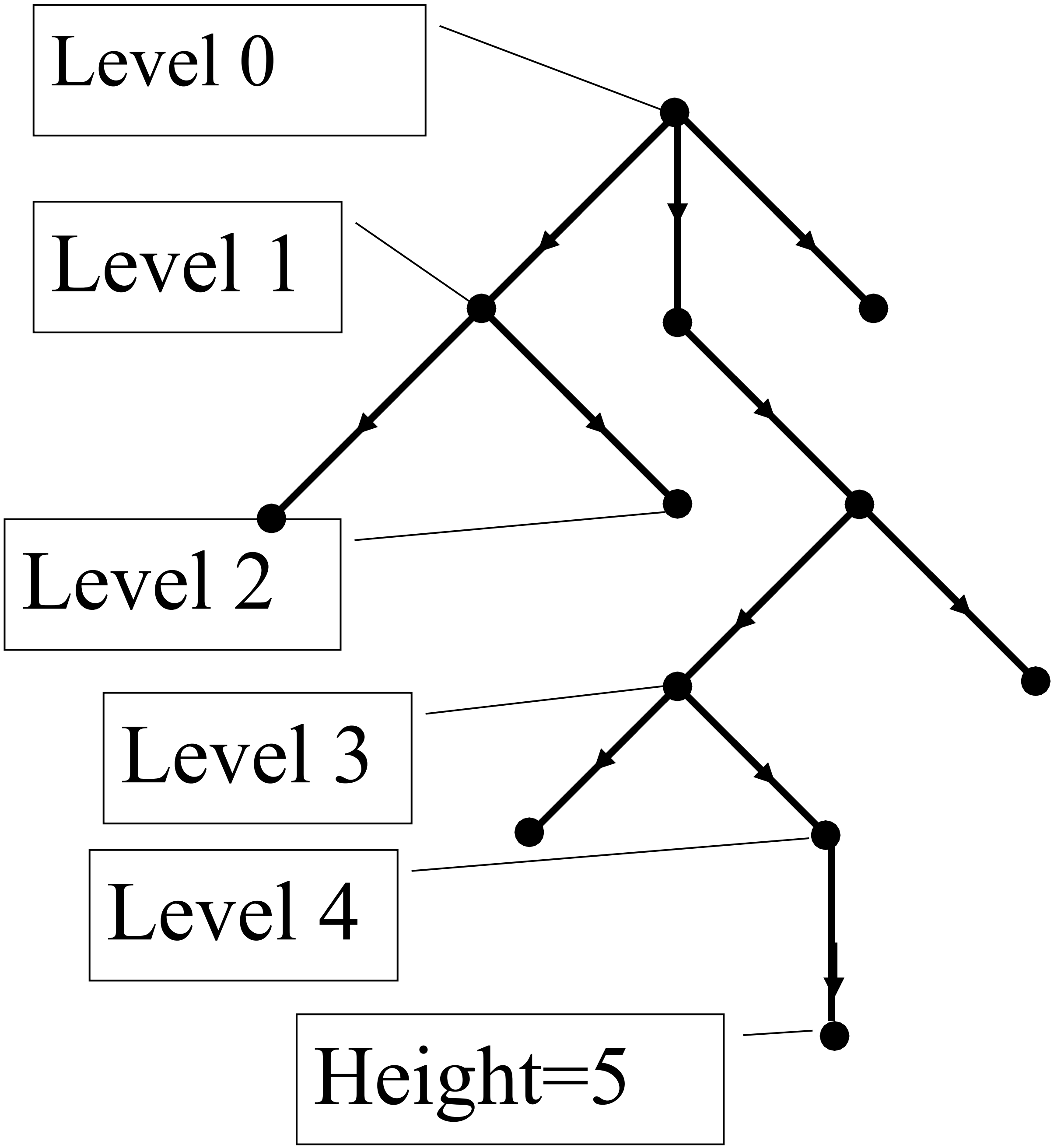
SOL: $n=(ml-1)/(m-1)$, $i=(l-1)/(m-1)$ (by theorem 4)

$l=100$, $m=4$

- $n=(4*100-1)/(4-1)$, $i=133$,
- $133-100=33$, 33 people sent out letter.

Some More Tree Theorems

- **Definition:** The *level* of a node is the length of the simple path from the root to the node.
  – The *height* of a tree is maximum node level.


Level 0
Level 1
Level 2
Level 3
Level 4
Height=5

Some More Tree Theorems

A rooted *m*-ary tree with height *h* is called *balanced* (平衡树)if all leaves are at levels *h* or *h*−1.

- Example 11

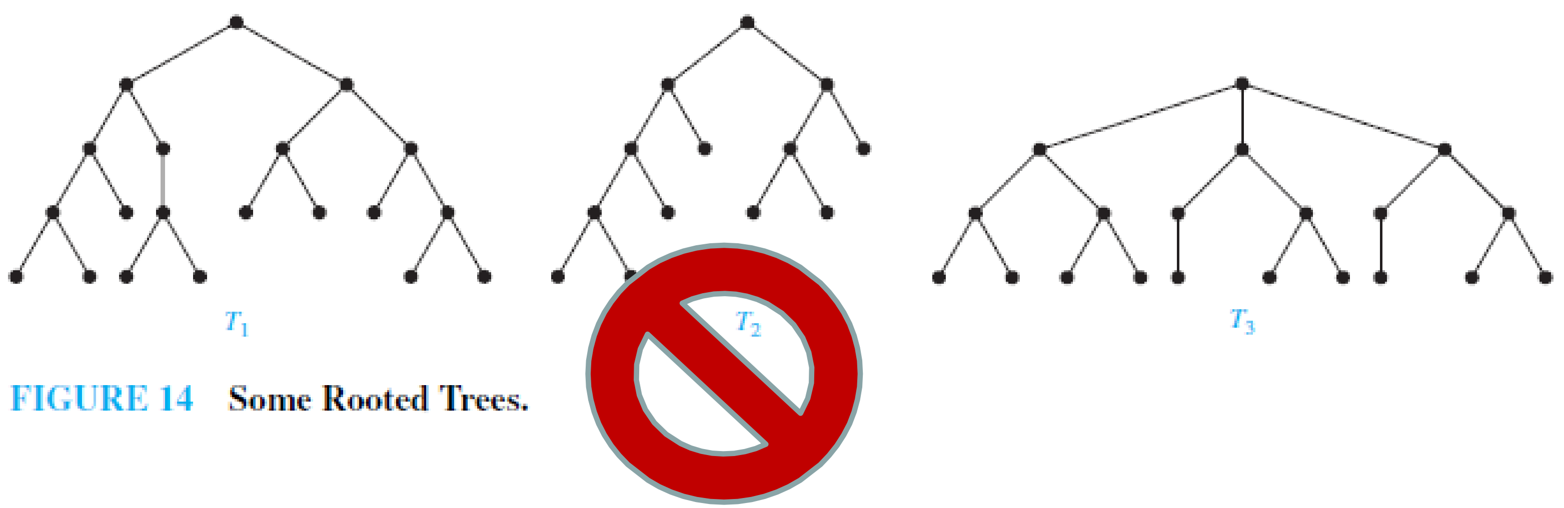
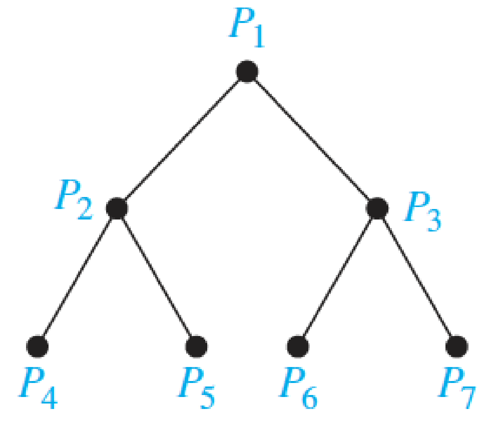


FIGURE 14 Some Rooted Trees.

- **Theorem5:** There are at most m^h leaves in an m-ary tree of height h.
  - **Corollary:** An m-ary tree with ℓ leaves has height $h \geq \lceil \log_m \ell \rceil$. If m is full and balanced then $h = \lceil \log_m \ell \rceil$.



例：28盏灯拟用一个电源插座，问需要多少块四插座接线板？

$$(m-1)i = l-1$$

$$(4-1)i = 28-1 \qquad i = 9$$

需要9个接线板。

§11.2: Applications of Trees

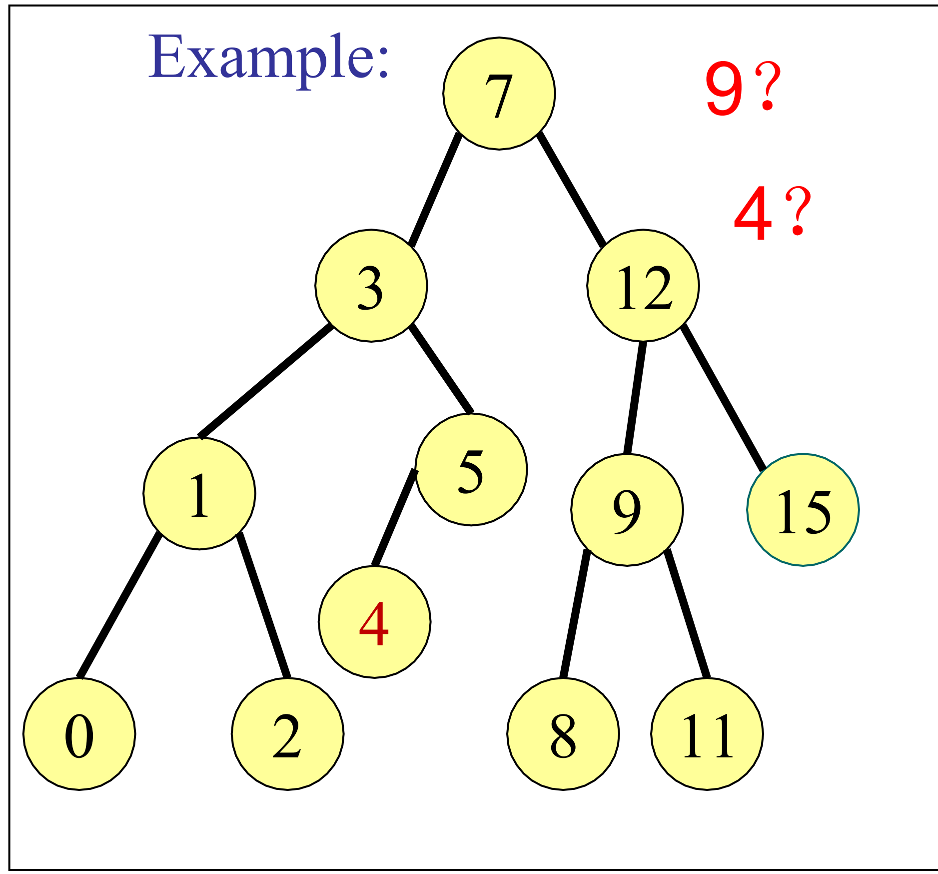
- Binary search trees（二叉搜索树）
- Decision trees(决策树)
  - Minimum comparisons in sorting algorithms
  - 8硬币问题
- Prefix codes（前缀码）
  - Huffman coding
- Game trees

Binary Search Trees

- A representation for sorted sets of items.
  - Supports the following operations in Θ(log n) average-case time:
    - Searching for an existing item.
    - Inserting a new item, if not already present.
  - Supports printing out all items in Θ(n) time.
- Note that inserting into a plain sequence a_i would instead take Θ(n) worst-case time.

Binary Search Tree Format

- Items are stored at individual tree nodes.
- We arrange for the tree to always obey this invariant:
  - For every item *x*,
    - Every node in *x*'s left subtree is less than *x*.
    - Every node in *x*'s right subtree is greater than *x*.



Decision Trees

- A decision tree represents a *decision-making process*.
  - Each possible "decision point" or situation is represented by a node.
  - Each possible choice that could be made at that decision point is represented by an edge to a child node.
- In the extended decision trees used in *decision analysis*, we also include nodes that represent random events and their outcomes.

Decision trees
决策树

- Minimum comparisons in sorting algorithms

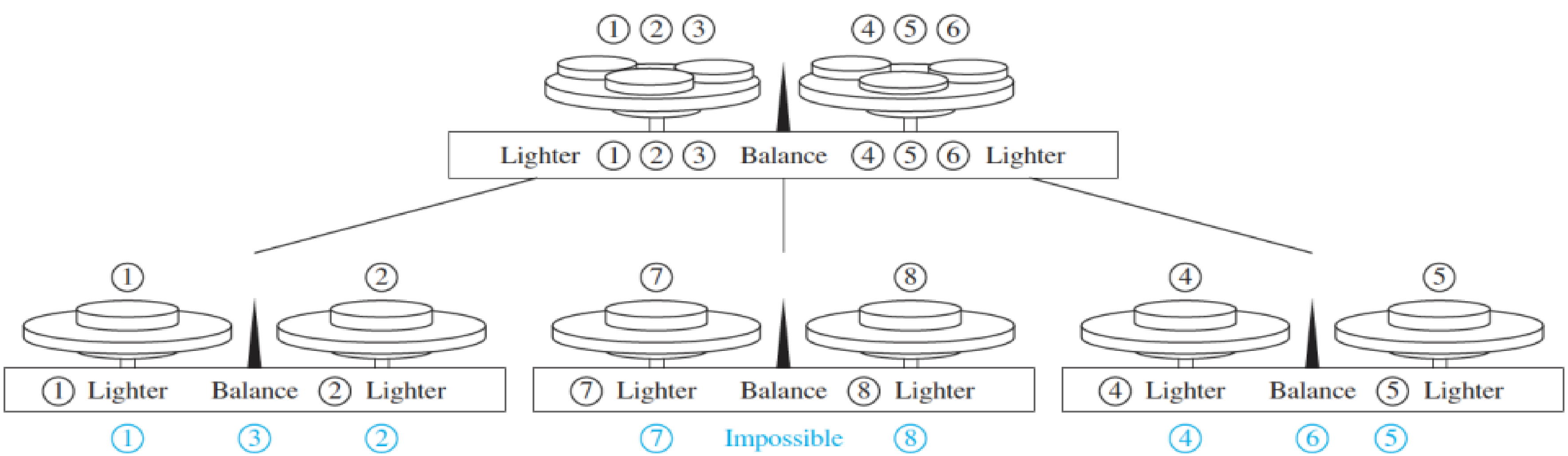
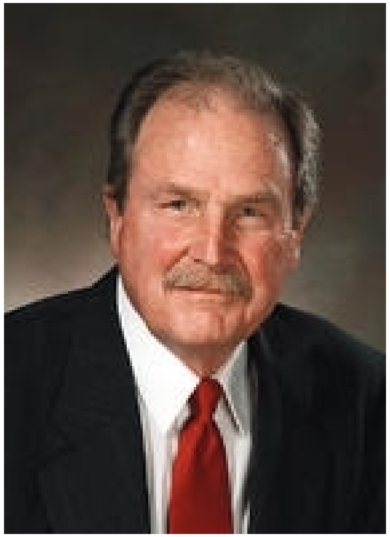


FIGURE 3 A Decision Tree for Locating a Counterfeit Coin. The counterfeit coin is shown in color below each final weighing.

Prefix Codes

- Huffman Coding
- was developed by David Huffman in 1951
-



哈夫曼（1925-1999）

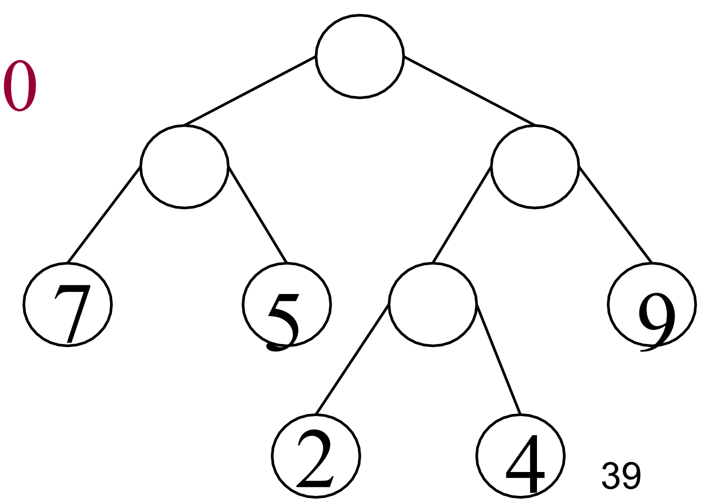
赫夫曼树（Huffman tree）及其应用

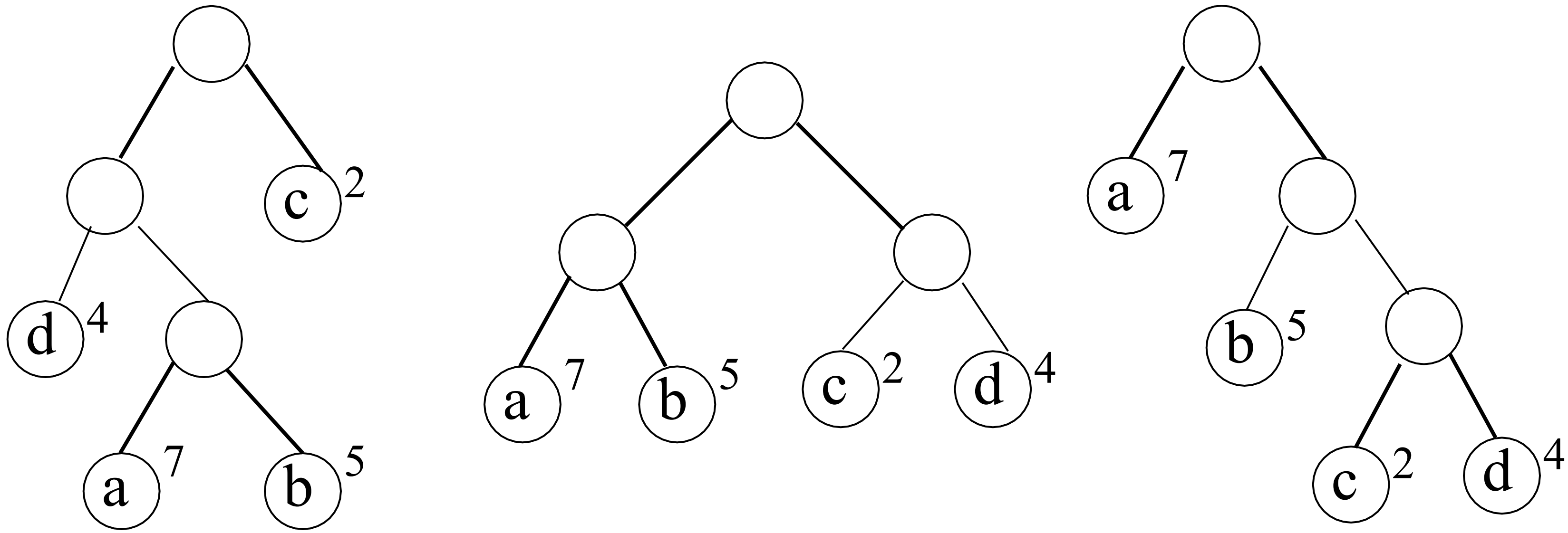
- 叶子结点的权值：对叶子结点赋予一个有意义的数量值。
- 树的带权路径长度(Weighted Path Length, WPL)：从根结点到各个叶子结点的路径长度与相应叶子结点权值的乘积之和。

WPL= $7\times2+5\times2+2\times3+4\times3+9\times2=60$

赫夫曼树:

带权路径长度最小的二叉树，也叫做最优二叉树。





(a) (b) (c)

带权路径长度：

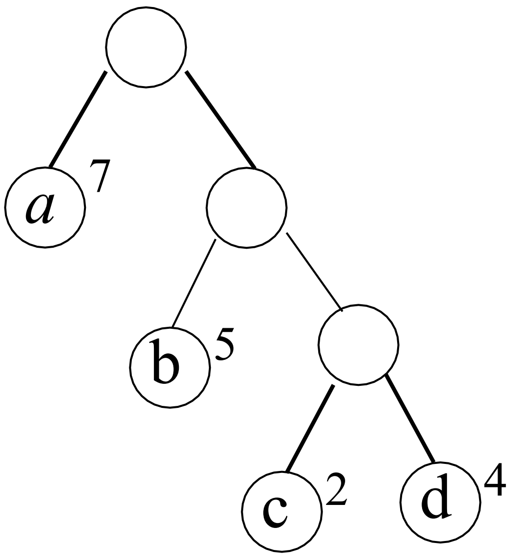
（a） WPL=7×3+5×3+2×1+4×2=46

（b） WPL=7×2+5×2+2×2+4×2=36

（c） WPL=7×1+5×2+2×3+4×3=35

赫夫曼树的特点

- 只有出度为0和2的结点，无出度为1的结点；
- n 个叶子的赫夫曼树共有 2n-1 个结点。
- 权值大的结点离根结点近；
- 赫夫曼树的形态不唯一，但WPL是相同的



赫夫曼树的构造

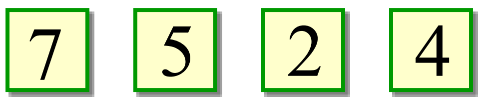
已知：一组权值给定的叶子结点$\{w_1, w_2, \cdots, w_n\}$，如何构造一棵哈夫曼树?

只要让权值最大的叶子结点离根最近，权值最小的叶子结点离根最远，就能使带权路径长度最小。

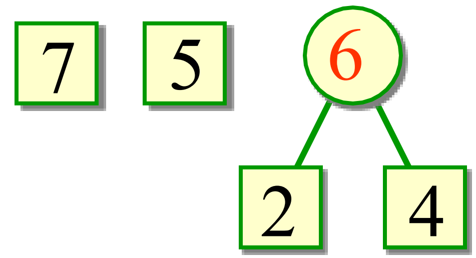
赫夫曼树的构造

1. 从这 n 结点中选取两个权值最小的结点组成新结点；
2. 将这两个结点的权值之和作为新结点的权值；
3. 将这两个结点从删去，把新结点加入；
4. 原来的 n 个结点减少为 n-1 结点；
5. 重复步骤 1——4，直到 n=1 即为所求。

例： 有4个结点，权值分别为7，5，2，4，构造赫夫曼树。

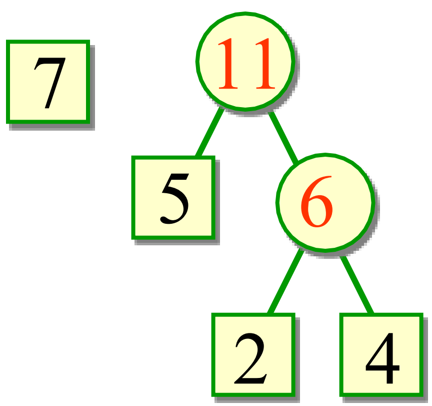


初始

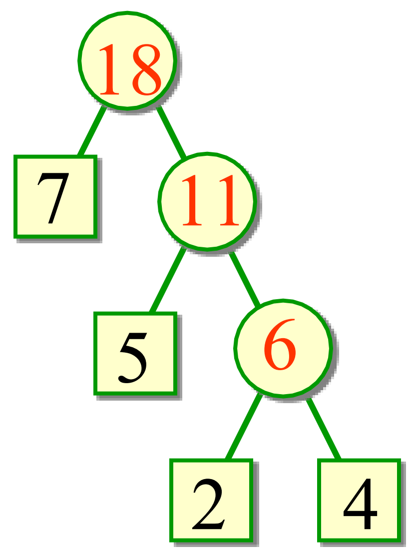


合并2 4

注意：结点有规律的放置问题(全部左小右大或全部左大右小)

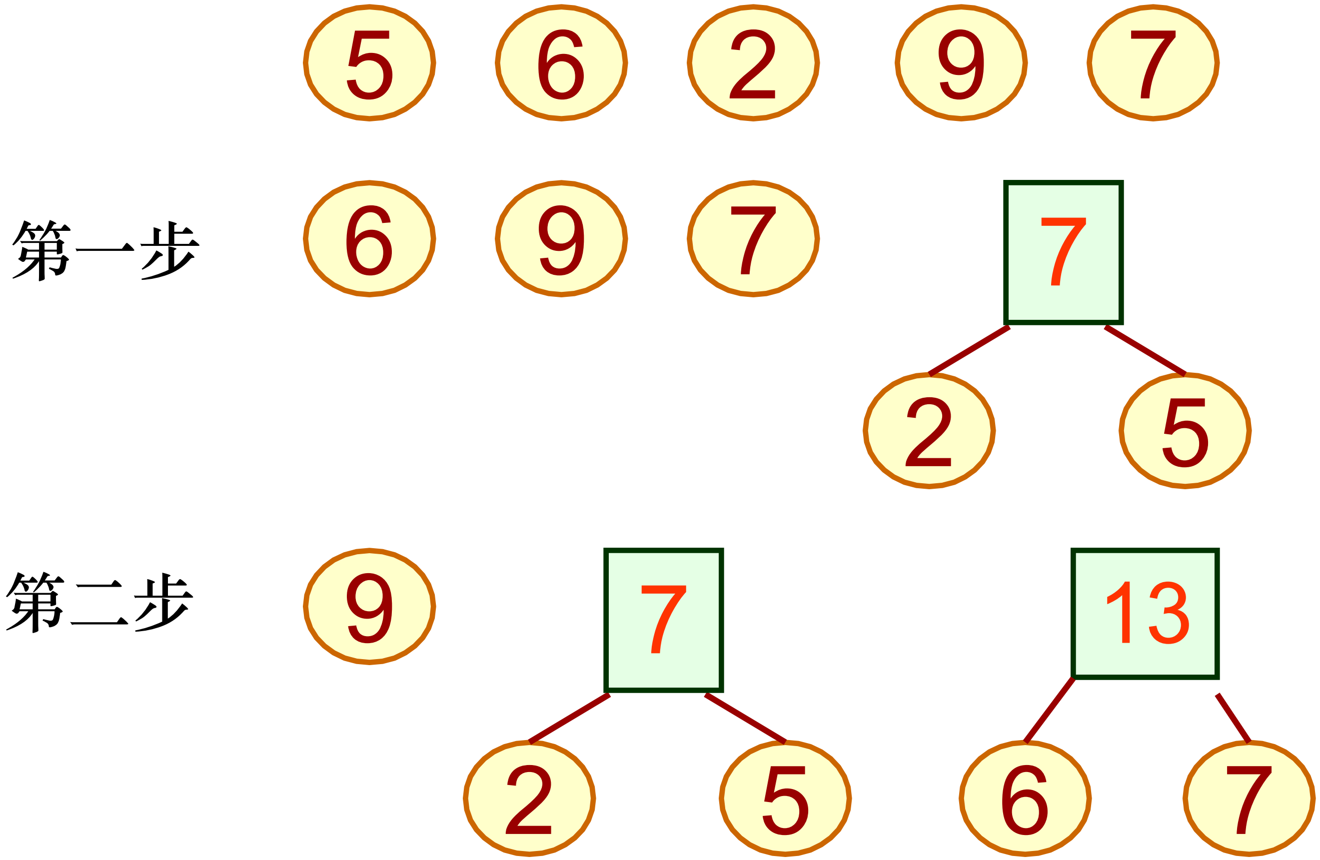


合并5 6

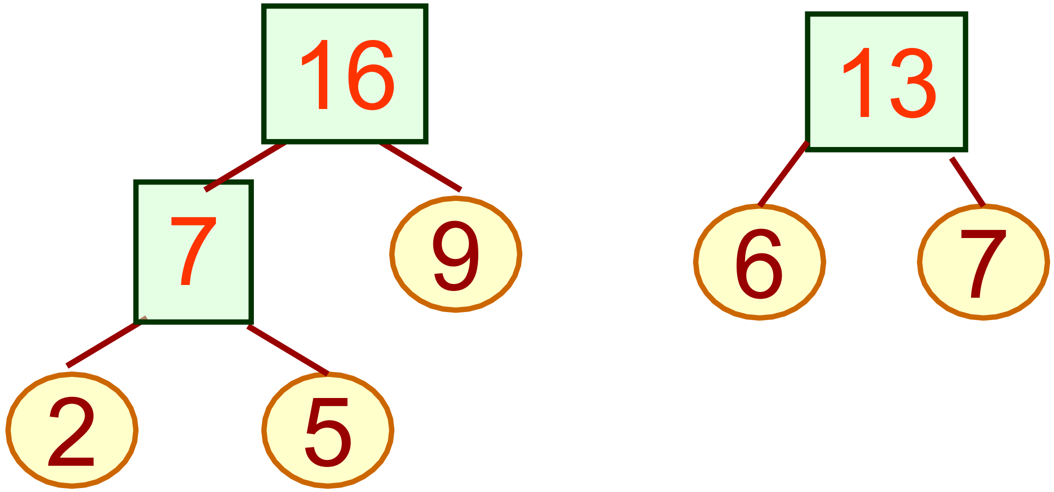


合并7 11

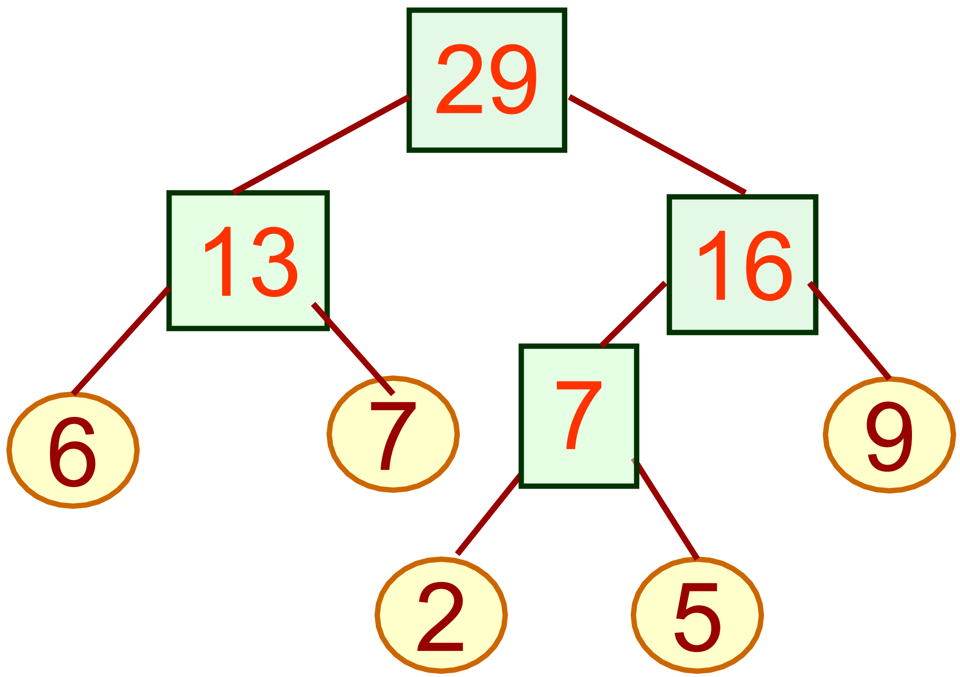
例如：已知权值 W={ 5, 6, 2, 9, 7 }



第三步:

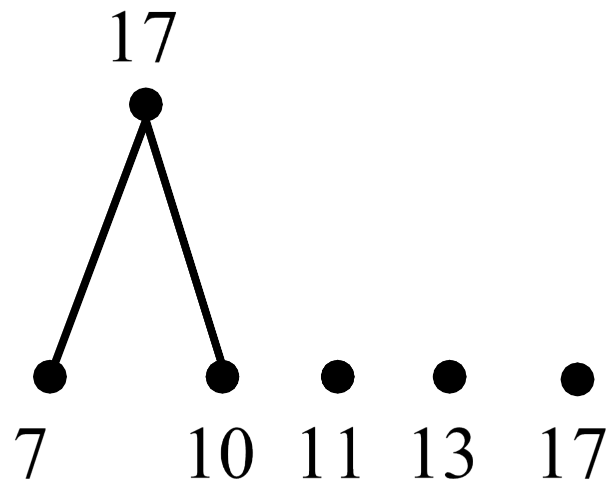


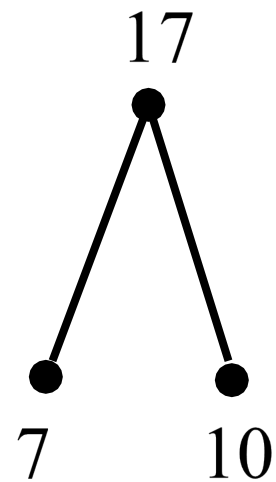
第四步:

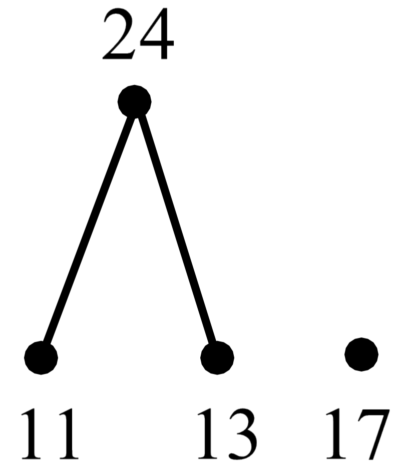


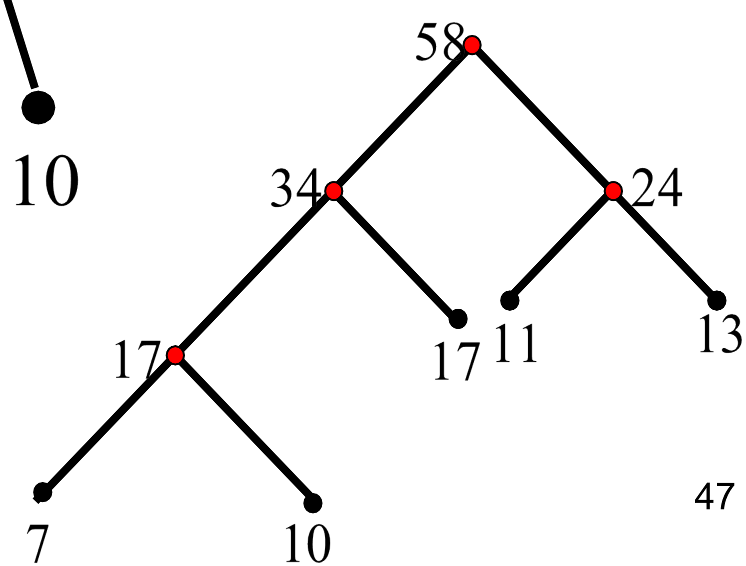
例题：求带权7,10,11,13,17的最优二叉树。

解：求解过程如下

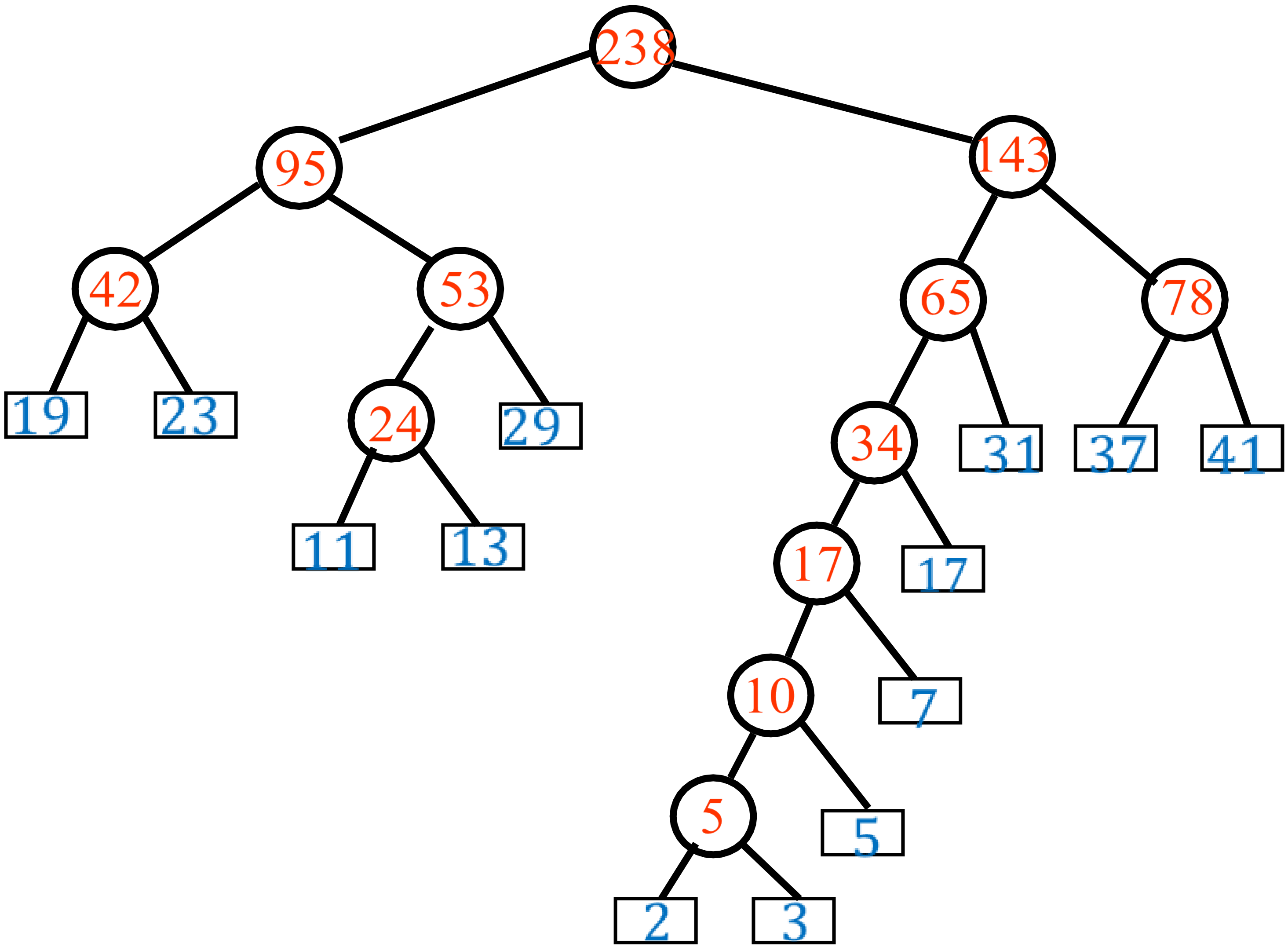








例：有权 2，3，5，7，11，13，17，19，23，29，31，37，41求相应的最优树。



赫夫曼编码

- 编码分类

  等长编码：每个字符编码长度相等。

  不等长编码：字符的编码长度不相等，Huffman编码。

- 例：3位二进制

  固定长度：{000,001,010,011,100,101,111};

  不固定长度：{0,1,

  00,01,10,11,

  000,001,010,011,100,101,111};

- 不等长编码的关键：解码时的唯一性

A:0

B:1　　　　B:10

C:10　　　C:11

让任意编码都不是其他编码的前缀

编码：AABAABC→00100110

解码：AA?

编码：AABAABC→0010001011

解码：AABAABC

- 赫夫曼编码思想（不等长编码）

  编码长度不固定;

  使用频率高的字符用较少的位（长度短）；

  利用不等长编码，可以使报文总长度较短，这也是文件压缩技术的核心。

beep boop beer

- 前缀编码：任一字符的编码都不是另一字符的前缀。

[定义]前缀：a，b，c均为二进制序列，如果b＝（a并列c），c不是空序列，则称a是b的前缀/prefix。即a是b的前面一部分。

例： $a=011, c=010, b=011010$

- Defintions： prefix code（前缀码）： the bit string for a letter never occurs as the first part of the bit string for another letter. Codes with this property are called prefix codes

二元前缀码

设 $\alpha_1\alpha_2\cdots\alpha_{n-1}\alpha_n$是长为$n$的符号串，$\alpha_1,\alpha_1\alpha_2,\cdots,\alpha_1\alpha_2\cdots\alpha_{n-1}$,均为该符号串的前缀，它们的长度分别为 1,2, … ,n-1。

$\beta=\alpha_1\alpha_2\cdots\alpha_n$

$\beta_1=\alpha_1$

$\beta_2=\alpha_1\alpha_2$

$\beta_{n-1}=\alpha_1\alpha_2\cdots\alpha_{n-1}$

前n-1位中取任意位数形成的串——前缀

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

若符号串$\beta_i(i=1,2,\cdots,m)$中只出现0,1两个符号，则称A为二元前缀码。

$$B_1 = \{0,10,110,1111\} \quad \surd$$

$$B_2 = \{1,01,001,000\} \quad \surd$$

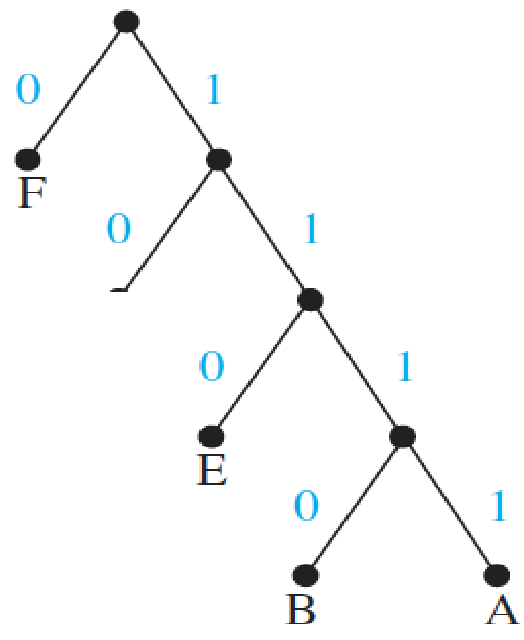
$$B_3 = \{1,11,101,001,0011\} \quad \times$$

$$B_4 = \{b,c,aa,ac,aba,abb,abc\} \quad \surd$$

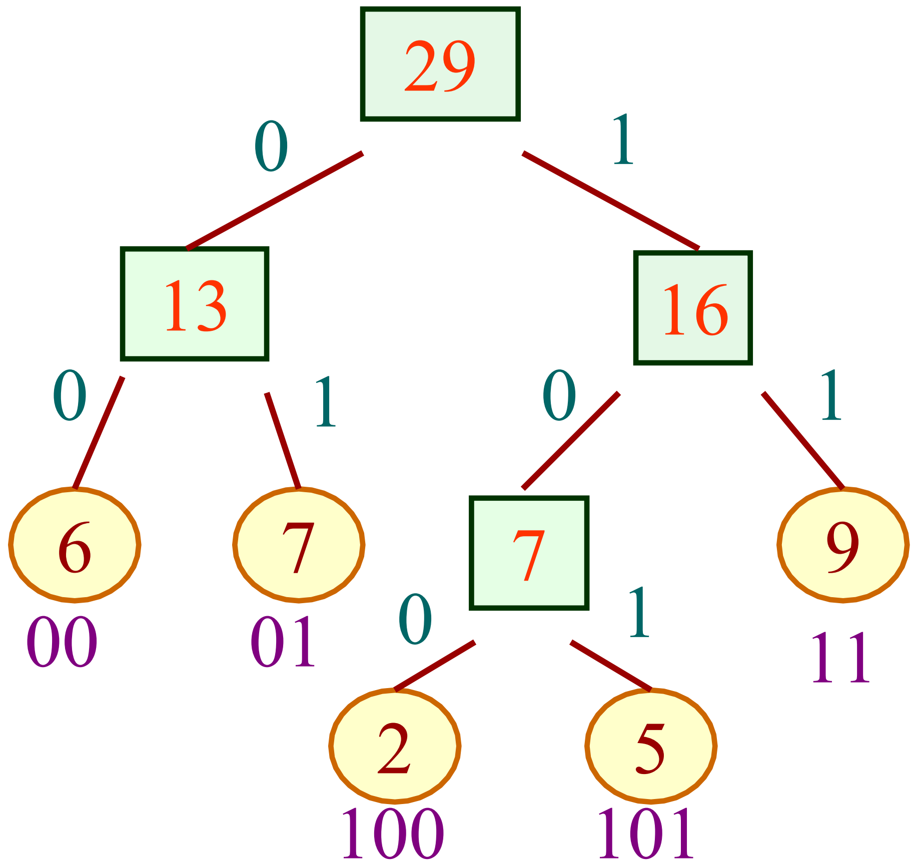
$$B_5 = \{b,c,a,aa,ac,aba,abb,abc\} \quad \times$$

定理

- 任意一棵二叉树都可产生一个前缀码。
- 任何一个前缀码都对应一棵二叉树。



- Huffman编码是利用赫夫曼树设计的最优前缀编码.
  - 约定
    左分支：0
    右分支：1
  - 叶结点编码
    从根到叶子的路径



- 哈夫曼编码的性质：
  ①前缀编码。
  ②高效编码。

北京郵電大學
Beijing University of Posts and Telecommunications

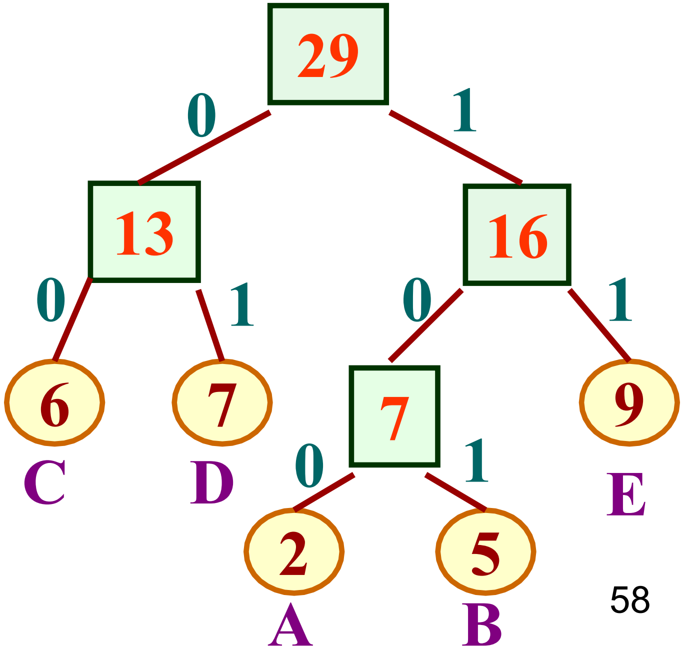
- Huffman编码的解码

  - 解码过程

    1)从左到右读取编码串

    2)从根结点开始,如果是0,则选择左支;如果是1,则选择右支;直到叶结点。

000100111011011110 1
CDCEBBEB

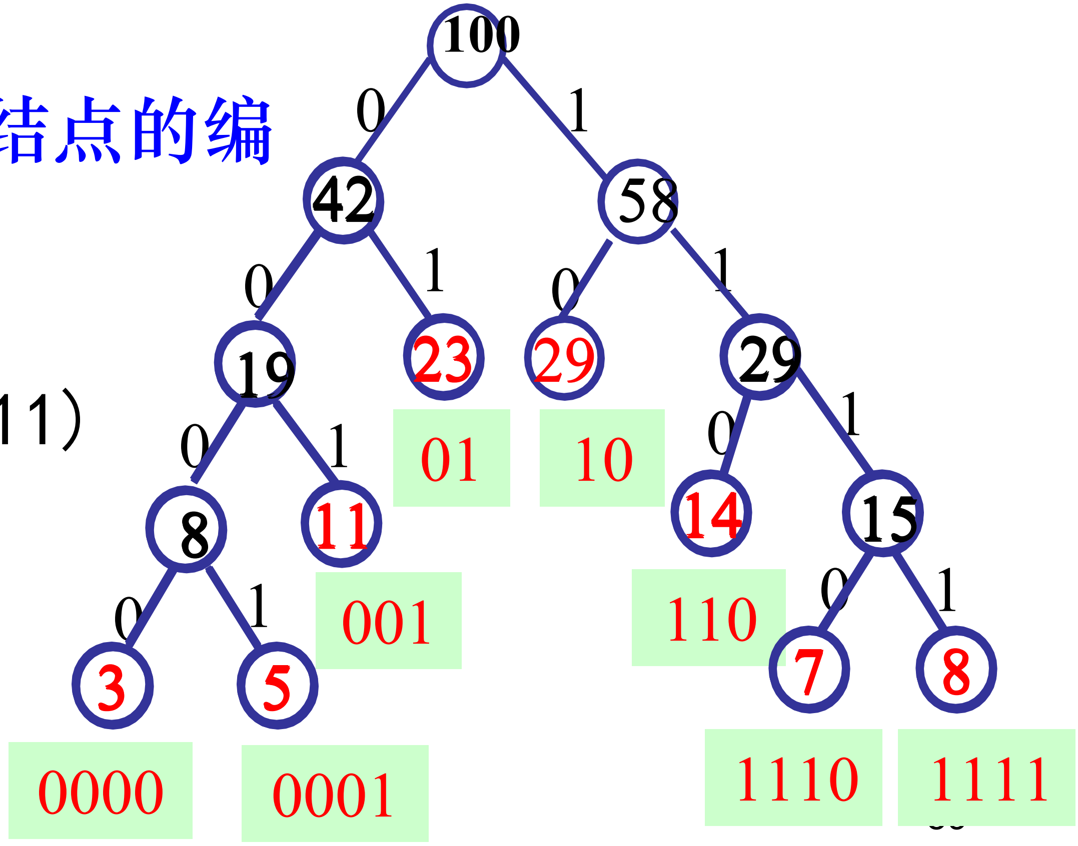


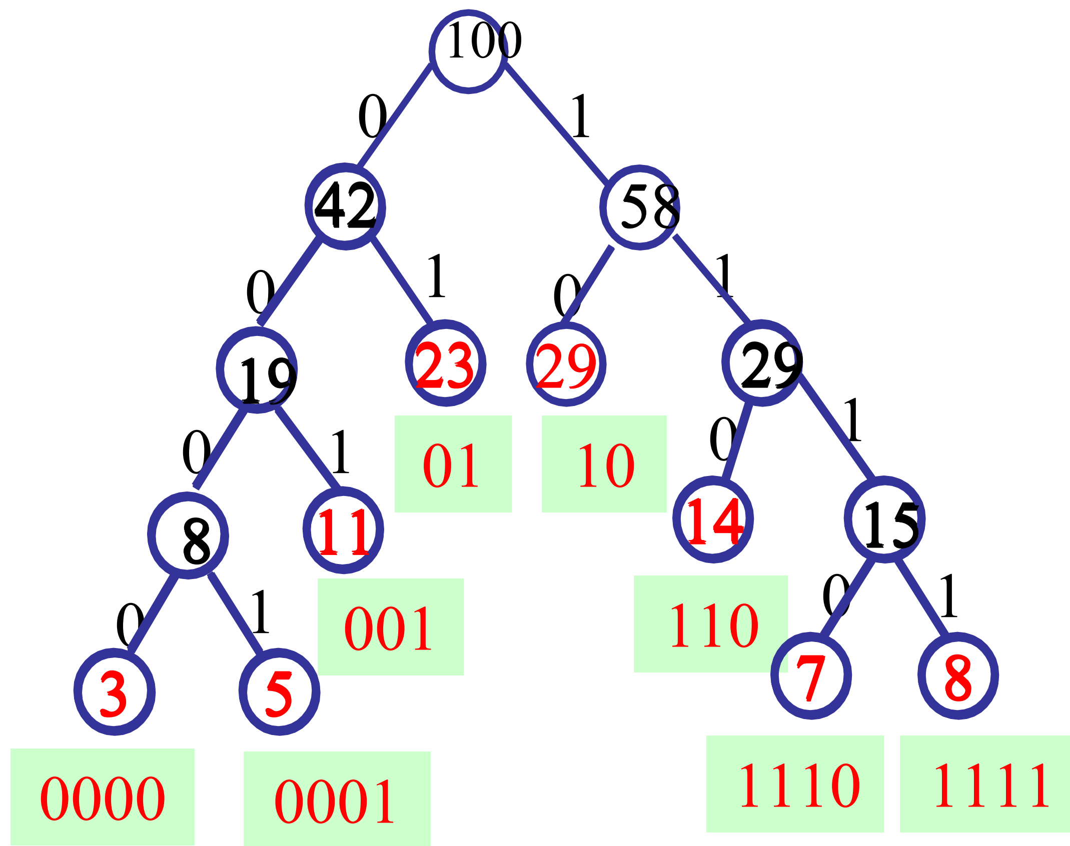
已知某系统在通信联络中只可能出现八种字符，其概率分别为0.05(A)，0.29(B)，0.07(C)，0.08(D)，0.14(E)，0.23(F)，0.03(G)，0.11(H)，试设计赫夫曼编码,比使用等长编码的电文总长压缩多少?

1.构造赫夫曼树

2.在赫夫曼树上求叶子结点的编码

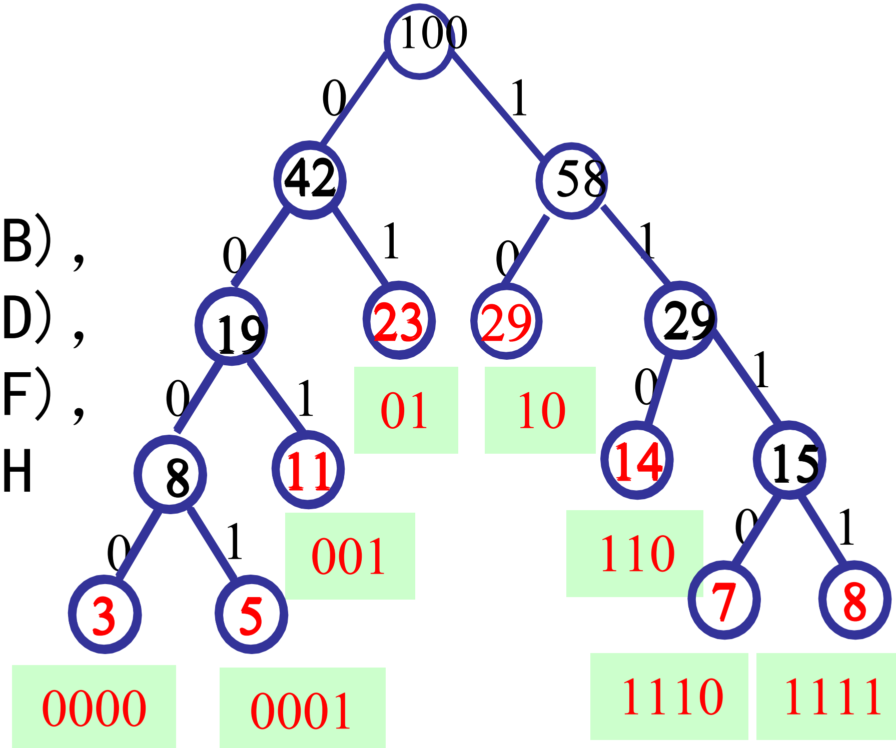
w=(5, 29, 7, 8, 14, 23, 3, 11)




100
0
1
42
58
0
1
0
1
19
23
29
29
0
1
01
10
0
1
8
11
14
15
0
1
001
110
0
1
3
5
7
8
0000
0001
1110
1111

A	0001
B	10
C	1110
D	1111
E	110
F	01
G	0000
H	001

0. 05 (A)， 0. 29 (B)，
0. 07 (C)， 0. 08 (D)，
0. 14 (E)， 0. 23 (F)，
0. 03 (G)， 0. 11 (H



WPL= 4×(0.03+0.05+0.07+0.08)+ 3×(0.11+0.14)+ 2×(0.23+0.29) = 2.71

等长编码长度：3

(3−2.71)/3 = 9.7%

例：已知字母A，B，C，D，E，F，G出现的频率如下：

A—35% B—20 % C—15 % D—10 %
E－10% F－5% G－5%

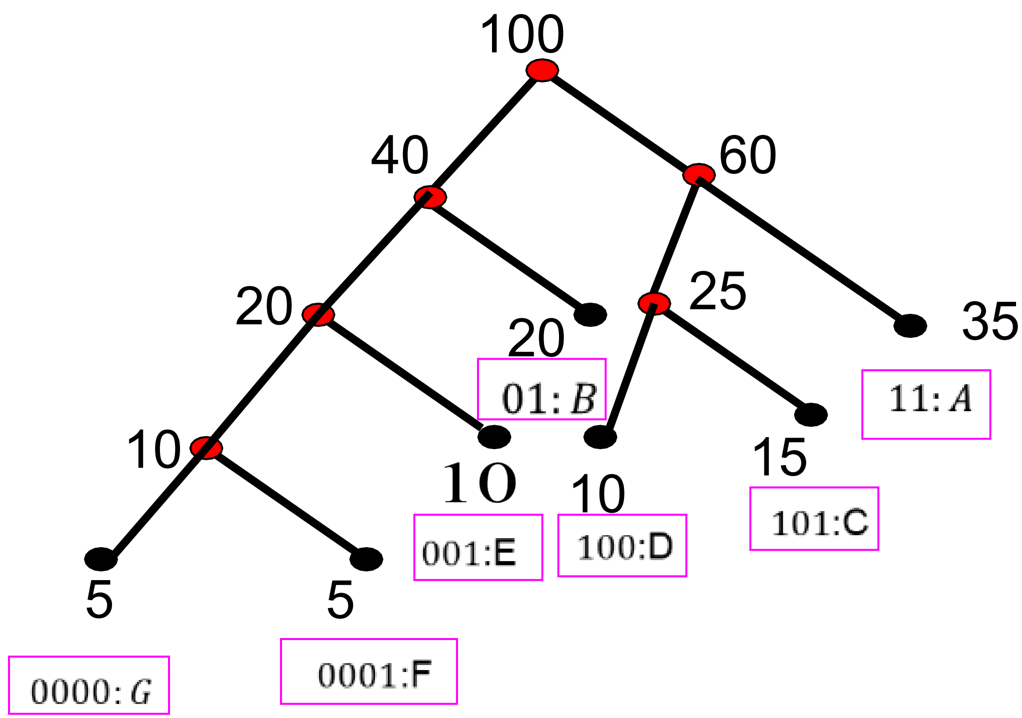
1) 求带权5,5,10,10,15,20,35的最优二叉树。

2) 求T所对应的前缀码

3) 设树叶v_i带权$w_i \times 100$

求v_i处的符号串表示出现频率为w%的字母

1)



2)前缀码= {01,11,001,100,101,0000,0001}

3）11 A; 01 B; 101 C; 100 D;

001 E; 0001 F; 0000 G

Game Trees

- Nim game

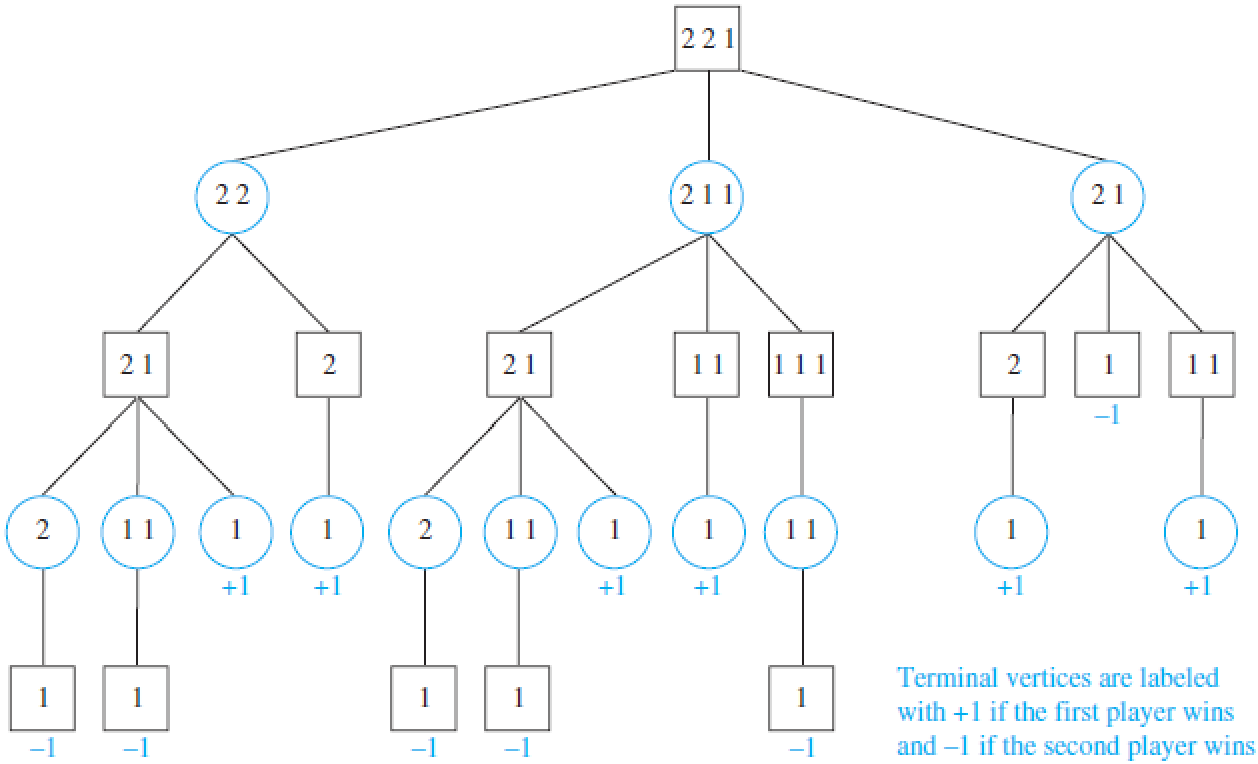


FIGURE 7 The Game Tree for a Game of Nim.

Game Trees

- The strategy where the first player moves to a position represented by a child with maximum value and the second player moves to a position of a child with minimum value is called the **minmax strategy**.

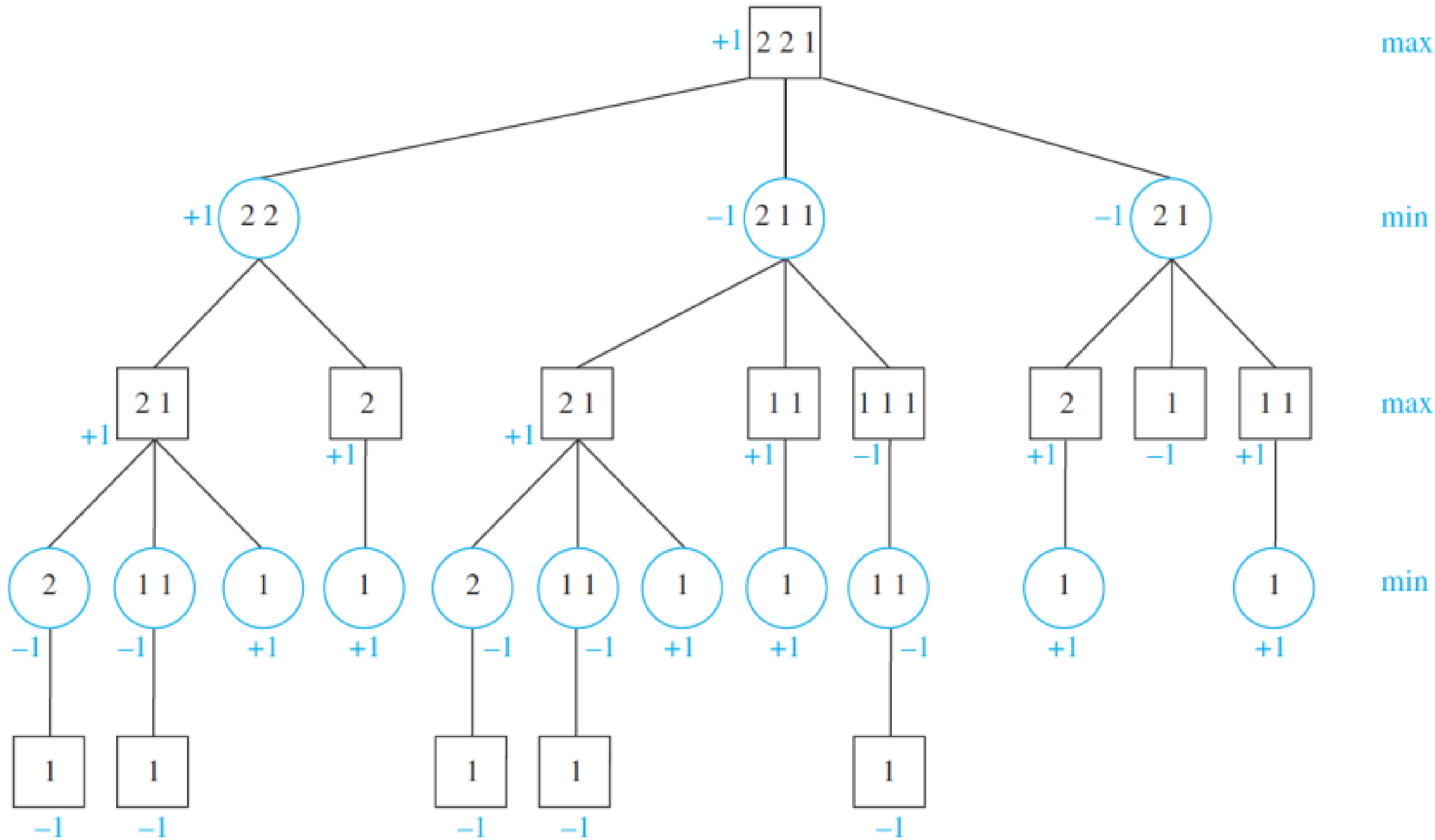
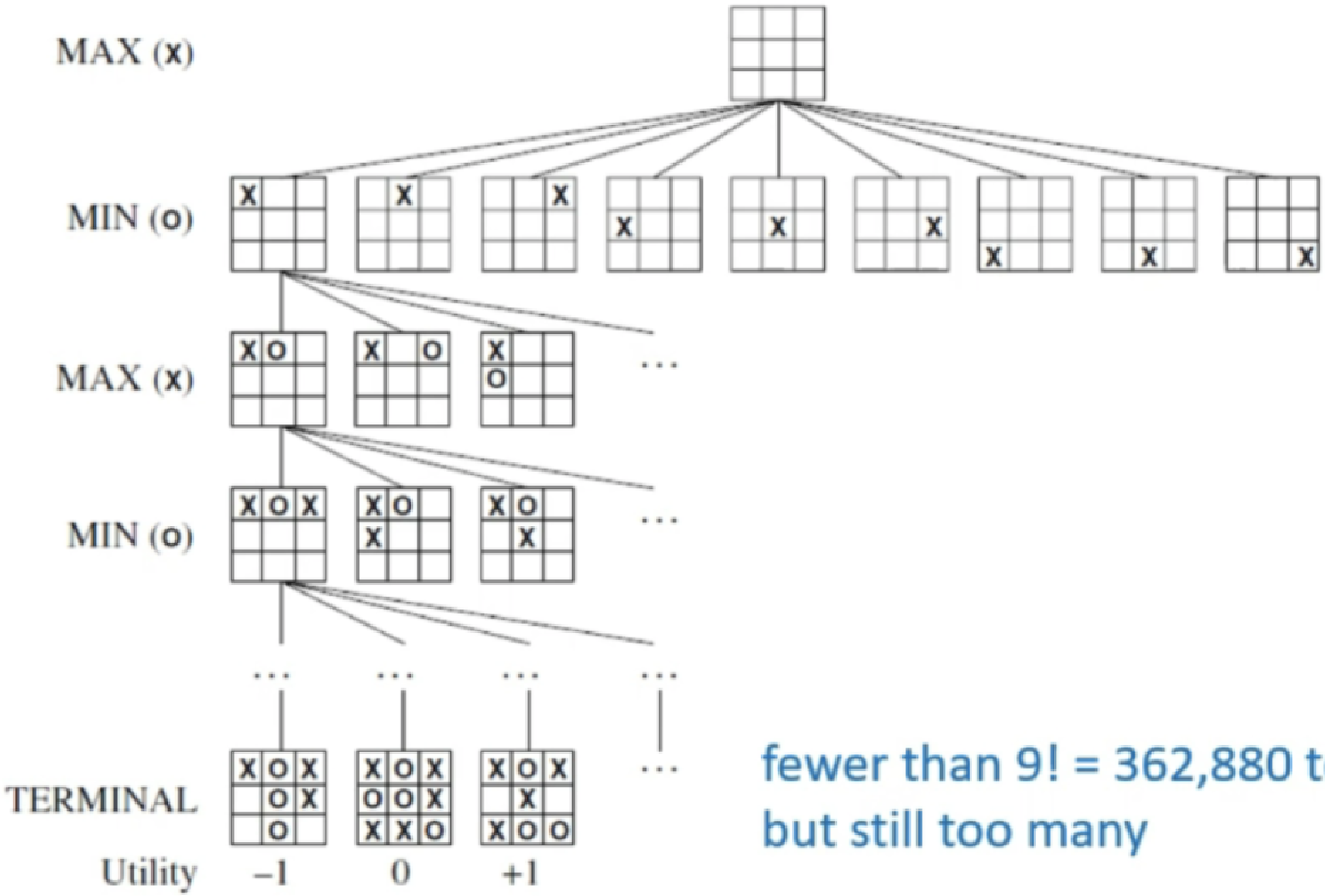


FIGURE 9 **Showing the Values of Vertices in the Game of Nim.**

Max：=max（孩子1得分，孩子2得分，…）

Min: =min(孩子1得分，孩子2得分，…)

Tic-tac-toe



fewer than 9! = 362,880 terminal nodes, but still too many

作业

- §11.1 – 4, 16, 20, 28, 44
- §11.2 – 6, 16, 30