



福州大学至诚学院  
FUZHOU UNIVERSITY ZHICHENG COLLEGE

# 高级语言程序设计 (C语言与数据结构)

杨雄

83789047@qq.com





## 6.树和二叉树

01

树的定义和基本术语

02

二叉树和遍历二叉树

03

哈夫曼树及应用



# 学习目的

目的



**熟练掌握二叉树的性质**



**熟练掌握二叉树的各种遍历策略**



**掌握建立哈夫曼编码的方法**

# Part.1

## 6.1 树的定义和基本术语

## 6.1 树的定义和基本术语

**树** (Tree) 是  $n$  ( $n \geq 0$ ) 个结点的**有限集**。

若  $n = 0$ ，称为空树；

若  $n > 0$ ，则它满足如下两个条件：

- (1) **有且仅有一个**特定的称为**根** (Root) 的结点；
- (2) 其余结点可分为  $m$  ( $m \geq 0$ ) 个**互不相交**的有限集  $T_1, T_2, T_3, \dots, T_m$ ，其中每一个集合本身又是一棵树，并称为根的**子树** (SubTree)。

显然，树的定义是一个**递归**的定义。

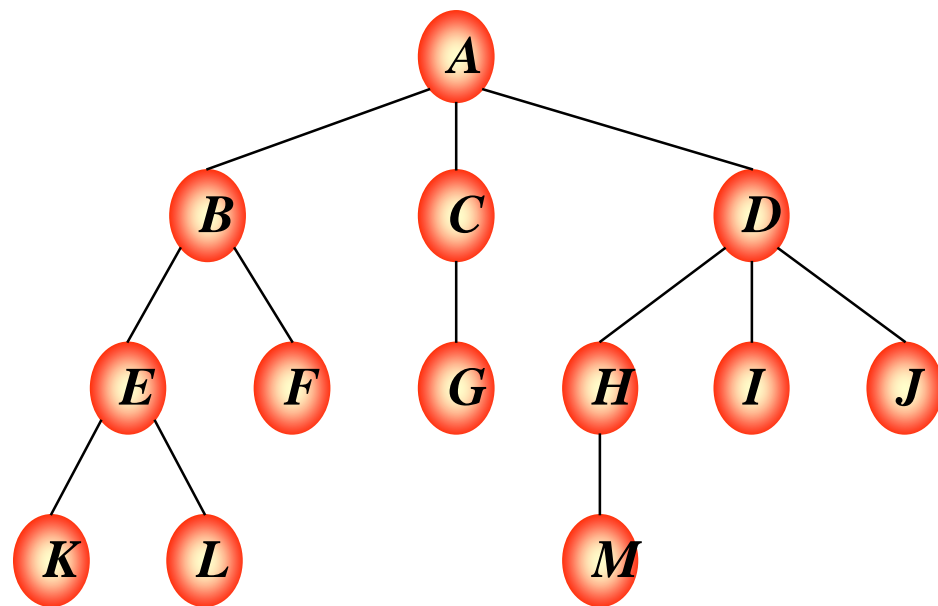
## 6.1 树的定义和基本术语

### ■ 树的表示形式

#### 1. 树形表示法

$\phi$   
空树

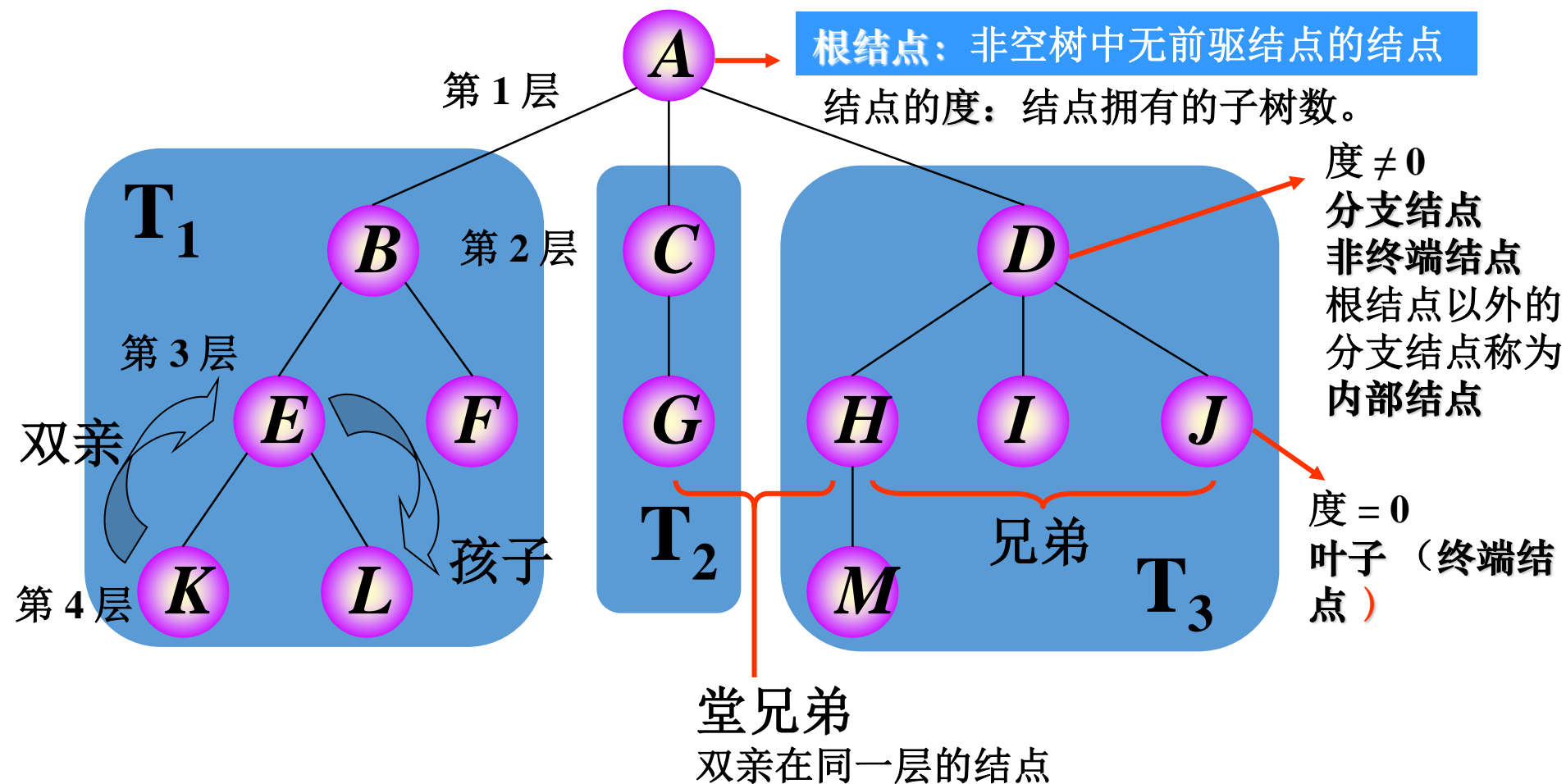
  
仅含有根结点的树





## 6.1 树的定义和基本术语

### 基本术语



## 6.1 树的定义和基本术语

**树的度**：树内各结点的度的最大值。

**树的深度**：树中结点的最大层次。

**结点的层次**：根为第一层，根的孩子为第二层。

**孩子、双亲**：结点的子树的根称为该结点的孩子，相应的，该结点称为孩子的双亲。

**兄弟**：同一个双亲的孩子之间互称兄弟。

**祖先**：从根到该结点所经分支上的所有结点

**子孙**：以某结点为根的子树中的任一结点都称为该结点的子孙。

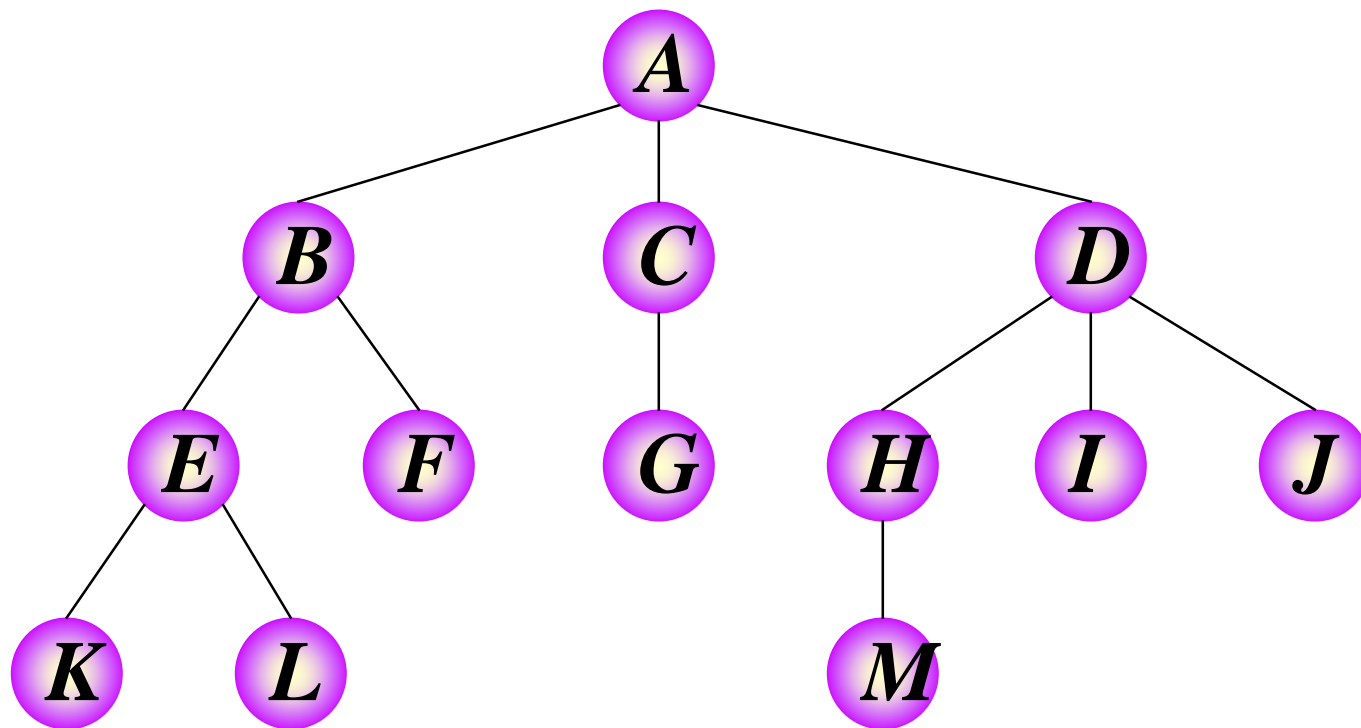
**有序树**：树中结点的各子树从左至右有次序（最左边的为第一个孩子）。

**无序树**：树中结点的各子树无次序。



## 6.1 树的定义和基本术语

### ■ 练习



结点A的度：

结点B的度：

结点M的度：

树的度：

叶子结点：

分支结点：

结点A的子孙：

树的深度（高度）：

# Part.2

## 6.2 二叉树

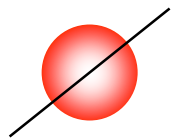
## 6.2 二叉树

二叉树是  $n$  ( $n \geq 0$ ) 个结点的有限集，它或者是空集 ( $n = 0$ )，或者由一个根结点及两棵互不相交的分别称作这个根的左子树和右子树的二叉树组成。

- 二叉树定义：度不超过2的有序树。
- 二叉树的特点
  - 每个结点最多有俩孩子 (二叉树中不存在度大于 2 的结点)。
  - 子树有左右之分，其次序不能颠倒。
  - 二叉树可以是空集合，根可以有空的左子树或空的右子树。

## 6.2 二叉树

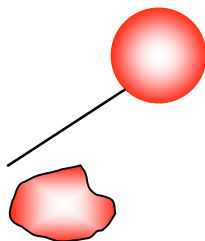
### • 二叉树的五种基本形态：



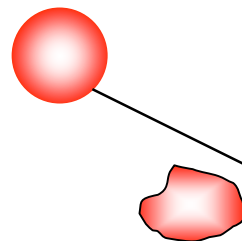
(a)  
空二叉树



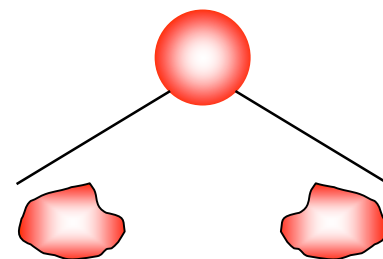
(b)  
根和空的  
左右子树



(c)  
根和左子树



(d)  
根和右子树



(e)  
根和左右子树

注：二叉树与树概念不同，但有关树的基本术语对二叉树都适用。

### ■ 二叉树的性质

**性质 1:** 在二叉树的第  $i$  层上至多有  $2^{i-1}$  个结点 ( $i \geq 1$ )。

证：采用归纳法证明此性质。

归纳基：当  $i = 1$  时，只有一个根结点， $2^{i-1} = 2^0 = 1$ ，命题成立。

归纳假设：设对所有的  $j$  ( $1 \leq j < i$ )，命题成立，即第  $j$  层上至多有  $2^{j-1}$  个结点。那么可以证明  $j = i$  时命题也成立。

归纳证明：由归纳假设可知，第  $i-1$  层上至多有  $2^{i-2}$  个结点。  
由于二叉树每个结点的度最大为 2，故在第  $i$  层上最大结点数为第  $i-1$  层上最大结点数的 2 倍，即：  
 $2 \times 2^{i-2} = 2^{i-1}$ 。 证毕。

## 6.2 二叉树

**性质 2:** 深度为  $k$  的二叉树至多有  $2^k - 1$  个结点 ( $k \geq 1$ )。

证：由性质 1 可知，深度为  $k$  的二叉树的最大结点数为：

$$\sum_{i=1}^k (\text{第 } i \text{ 层上的最大结点数}) = \sum_{i=1}^k 2^{i-1} = 2^k - 1$$

证毕。



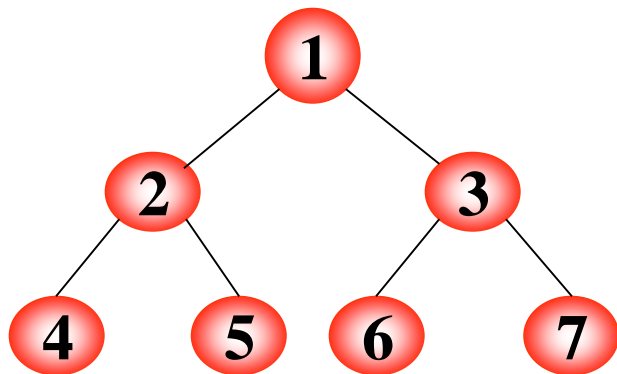
## 6.2 二叉树

### ■ 满二叉树

一棵深度为  $k$  且有  $2^k - 1$  个结点的二叉树称为**满二叉树**。

**特点：**每一层上的结点数都达到最大；  
叶子全部在最底层。

**编号规则：**从根结点开始，自上而下，自左而右。

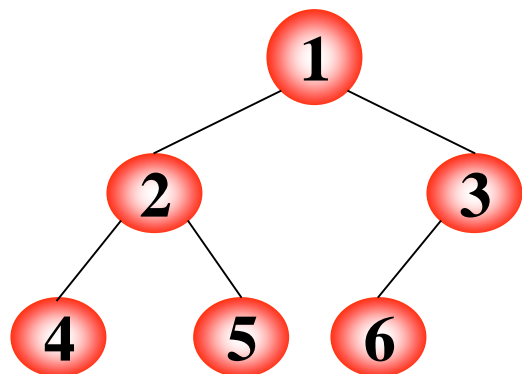


## 6.2 二叉树

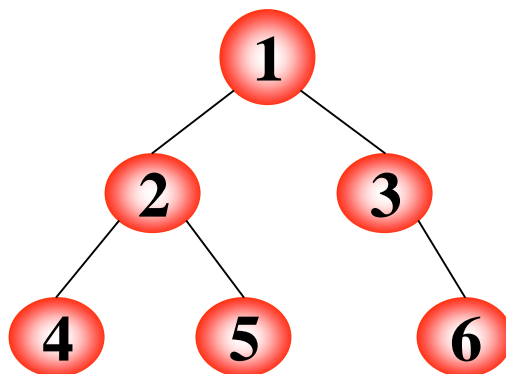
### ■ 完全二叉树

深度为  $k$  的具有  $n$  个结点的二叉树，当且仅当其每一个结点都与深度为  $k$  的满二叉树中编号为  $1 \sim n$  的结点一一对应时，称之为**完全二叉树**。

特点：即至多只有最下面的2层上结点的度数可以小于2，并且最底层的结点都集中在该层最左边。



完全二叉树



非完全二叉树

满二叉树  
↓  
一定是  
完全二叉树  
↑  
是定一不

## 6.2 二叉树

■ **性质 3**：对任何一棵二叉树  $T$ ，如果其叶子数为  $n_0$ ，  
度为 2 的结点数为  $n_2$ ，则  $n_0 = n_2 + 1$ 。

证：设  $n_1$  为二叉树  $T$  中度为 1 的结点数。因为二叉树中所有结点均  $\leq 2$ ，所以其结点总数为：

$$n = n_0 + n_1 + n_2$$

再看二叉树中的分支数，除根结点外，其余结点都有一个分支进入，设  $B$  为分支总数，则有：

$$n = B + 1$$

由于这些分支都是由度为 1 和 2 的结点射出的，所以有：

$$B = n_1 + 2n_2$$

于是有：

$$n = B + 1 = n_1 + 2n_2 + 1$$

所以有：

$$n_0 + n_1 + n_2 = n_1 + 2n_2 + 1$$

即：

$$n_0 = n_2 + 1 \quad \text{证毕。}$$

### ■ 完全二叉树的性质

■ **性质 4**：具有  $n$  个结点的完全二叉树的深度为  $\lfloor \log_2 n \rfloor + 1$ 。

证：假设此二叉树的深度为  $k$ ，则根据性质 2 及完全二叉树的

定义得到：
$$2^{k-1} - 1 < n \leq 2^k - 1$$

或
$$2^{k-1} \leq n < 2^k$$

取对数得：
$$k - 1 \leq \log_2 n < k$$

因为  $k$  是整数，所以有：

$$k = \lfloor \log_2 n \rfloor + 1$$

- **性质 5** : 如果对一棵有  $n$  个结点的**完全二叉树** (深度为  $\lfloor \log_2 n \rfloor + 1$ ) 的结点按层序编号 (从第 1 层到第  $\lfloor \log_2 n \rfloor + 1$  层, 每层从左到右), 则对任一结点  $i$  ( $1 \leq i \leq n$ ), 有 :
- (1) 如果  $i = 1$ , 则结点  $i$  是二叉树的根, 无双亲; 如果  $i > 1$ , 则其双亲是结点  $\lfloor i/2 \rfloor$ 。
  - (2) 如果  $2i > n$ , 则结点  $i$  为叶子结点, 无左孩子; 否则, 其左孩子是结点  $2i$ 。
  - (3) 如果  $2i + 1 > n$ , 则结点  $i$  无右孩子; 否则, 其右孩子是结点  $2i + 1$ 。

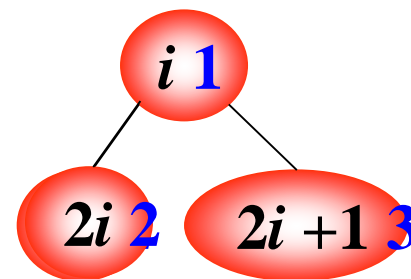
## 6.2 二叉树

证：(1) 可以从 (2) 和 (3) 推出，所以先证明 (2) 和 (3)。

对于  $i=1$ ，由完全二叉树的定义，其左孩子是

结点  $2 = 2i$ ，若  $2 = 2i > n = 1$ ，即不存在结点 2，此

时，结点  $i$  无左孩子。 (2) 得证。



结点  $i$  的右孩子也只能是结点  $3 = 2i + 1$ ，若

$3 = 2i + 1 > n$ ，即不存在结点 3，此时结点  $i$  无右孩子。

(3) 得证。



## 6.2 二叉树

对于  $i > 1$  , 可分为两种情况 :

(1) 设第  $j$  ( $1 \leq j \leq \lfloor \log_2 n \rfloor$ ) 层的

第一个结点的编号为  $i$  (由二叉树的

定义和性质 2 知  $i = 2^{j-1}$ ), 则其左孩

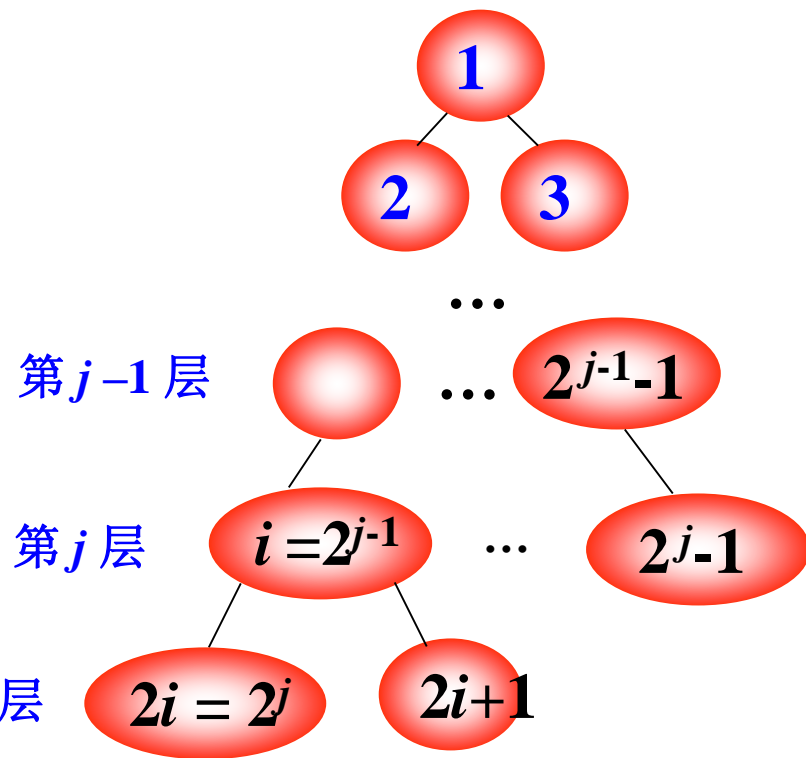
子必为第  $j+1$  层的第一个结点, 其

编号为  $2^j = 2 \times 2^{j-1} = 2i$ 。

如果  $2i > n$ , 则无左孩子。 (2) 得证。

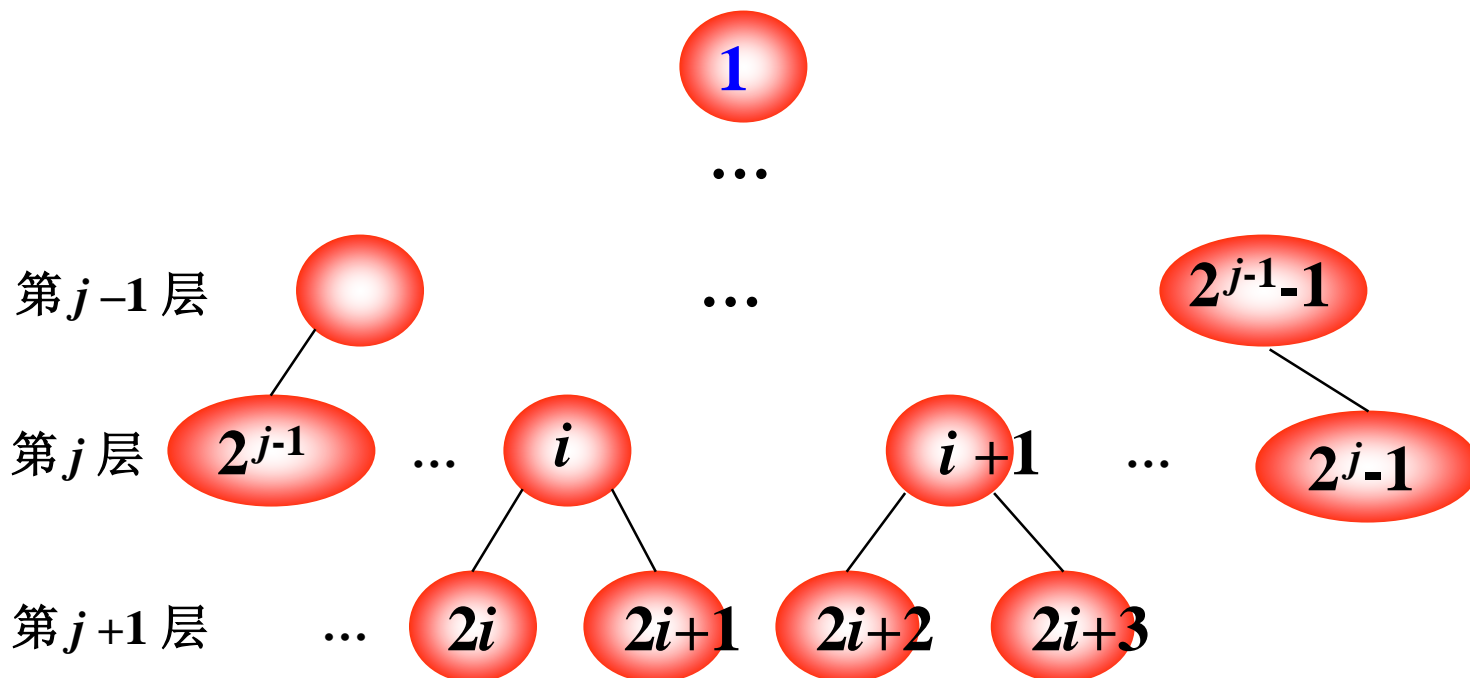
其右孩子必定为第  $j+1$  层的第二个结点,

编号为  $2i+1$ 。若  $2i+1 > n$ , 则无右孩子。 (3) 得证。



## 6.2 二叉树

(2) 设第  $j$  ( $1 \leq j \leq \lfloor \log_2 n \rfloor$ ) 层的某个结点的编号为  $i$  ( $2^{j-1} \leq i < 2^j$ ), 且  $2i+1 < n$ , 其左孩子为  $2i$ , 右孩子为  $2i+1$ 。则编号为  $i+1$  的结点是编号为  $i$  的结点的右兄弟或堂兄弟。若它有左孩子, 则其编号必定为:  $2i+2 = 2(i+1)$ , (2) 得证。若它有右孩子, 则其编号必定为:  $2i+3 = 2(i+1)+1$ 。(3) 得证。



## 6.2 二叉树

下面证明 (1)。

当  $i=1$  时：此结点就是根，因此无双亲。

当  $i>1$  时：

$i$  为偶数

如果  $i$  为左孩子，且  $i$  的双亲为  $p$ ，则有  $i=2p$ ，

$p=i/2=\lfloor i/2 \rfloor$ ，即  $\lfloor i/2 \rfloor$  是  $i$  的双亲；

$i$  为奇数

如果  $i$  为右孩子，且  $i$  的双亲为  $p$ ，则有  $i=2p+1$ ，

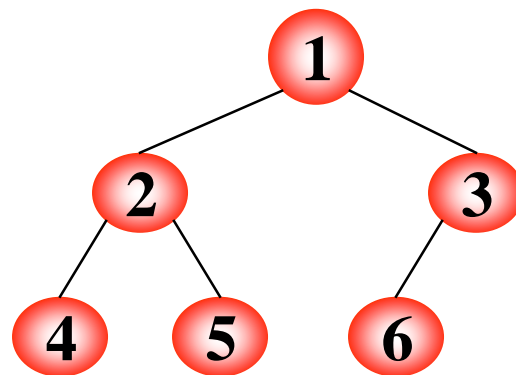
$p=(i-1)/2=i/2-1/2=\lfloor i/2 \rfloor$ ，即  $\lfloor i/2 \rfloor$  是  $i$  的双亲。证毕。

## 6.2 二叉树

### ■ 二叉树的存储结构

#### □ 顺序存储结构

**完全二叉树：**用一组地址连续的存储单元依次**自上而下、自左至右**存储结点元素，即将编号为  $i$  的结点元素存储在一维数组中下标为  $i-1$  的分量中。

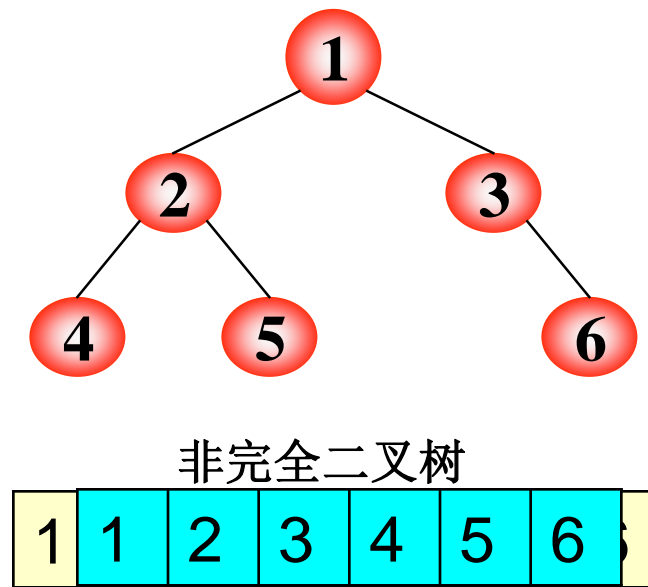


完全二叉树

1	2	3	4	5	6
---	---	---	---	---	---

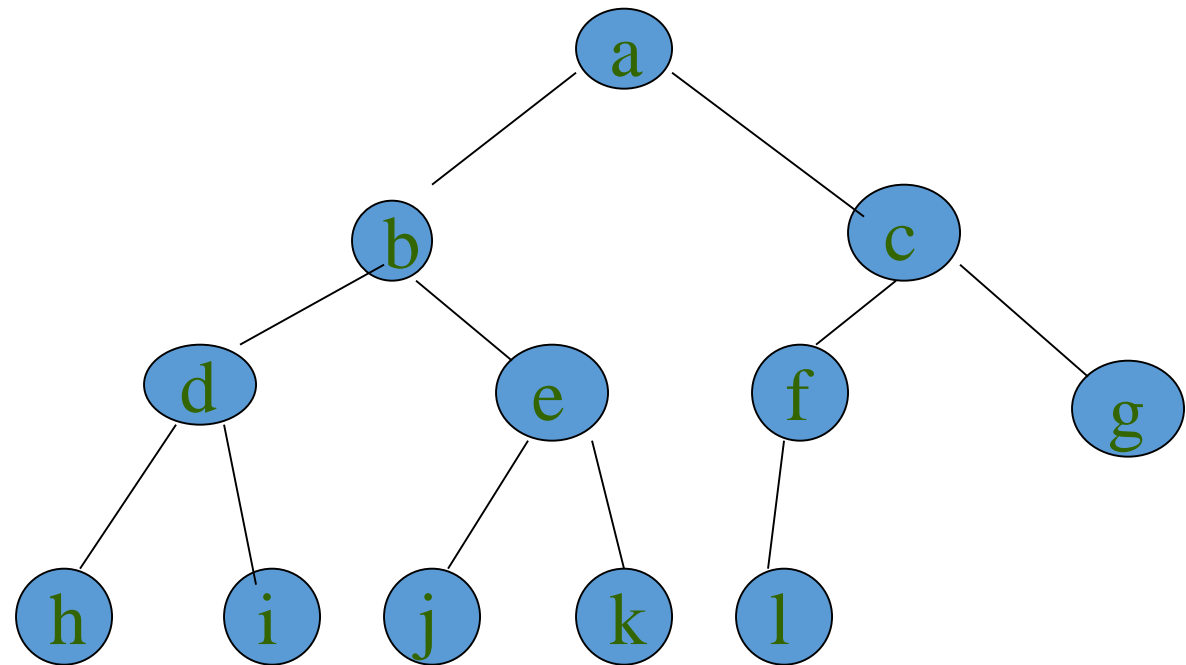
## 6.2 二叉树

**一般二叉树：**将其每个结点与完全二叉树上的结点相对照，存储在一维数组的相应分量中。



## 6.2 二叉树

### 完全二叉树



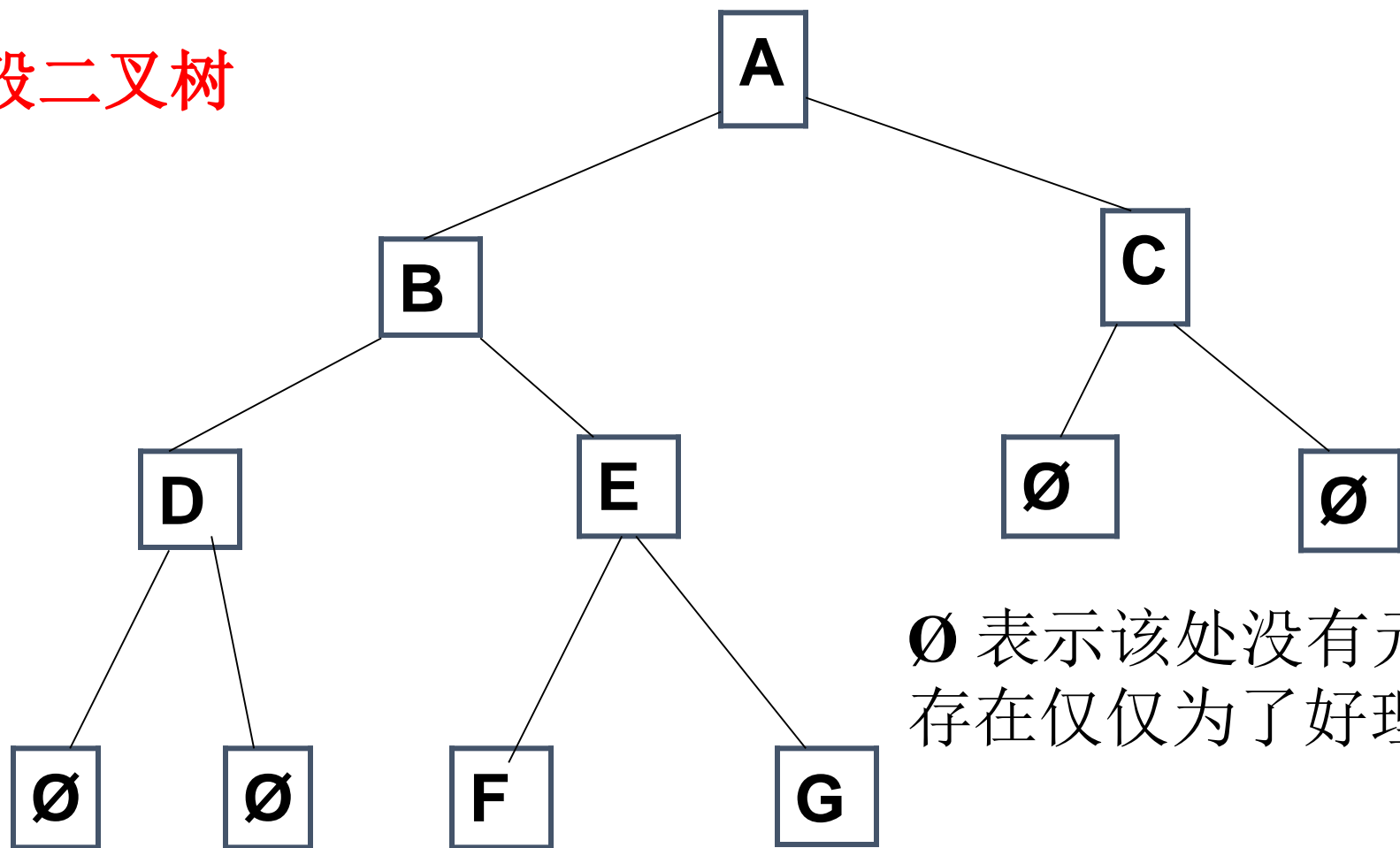
1   2   3   4   5   6   7   8   9   10   11   12

<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>	<b>f</b>	<b>g</b>	<b>h</b>	<b>i</b>	<b>j</b>	<b>k</b>	<b>l</b>
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------



## 6.2 二叉树

### 一般二叉树

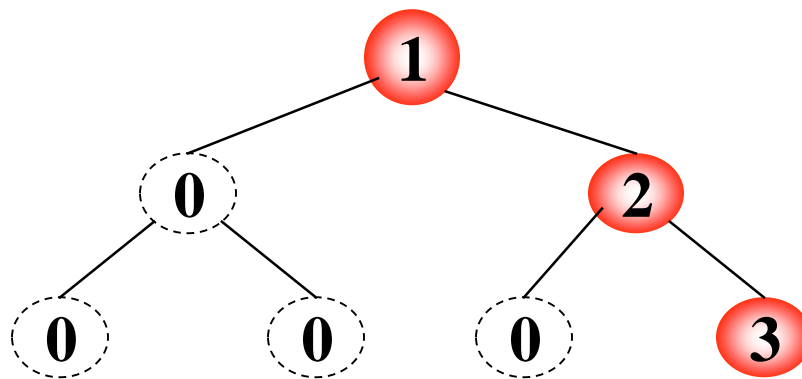


∅ 表示该处没有元素  
存在仅仅为了好理解

A	B	C	D	E	∅	∅	∅	∅	F	G
---	---	---	---	---	---	---	---	---	---	---

## 6.2 二叉树

**最坏情况：**深度为  $k$  的且只有  $k$  个结点的右单支树需要长度为  $2^k - 1$  的一维数组。



右单支树

1	0	2	0	0	0	3
---	---	---	---	---	---	---

这种顺序存储结构仅适用于完全二叉树

## 6.2 二叉树

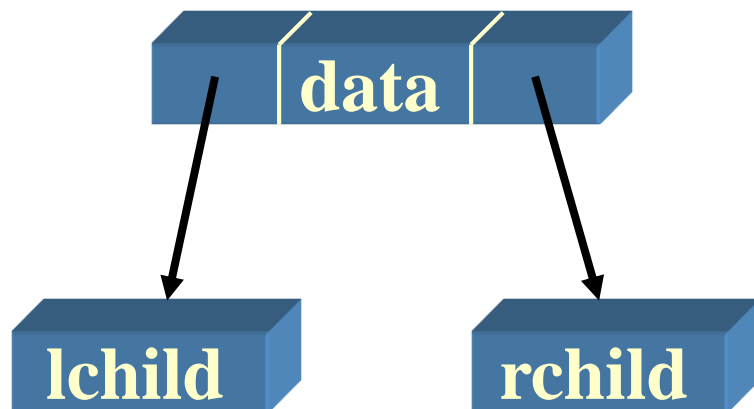
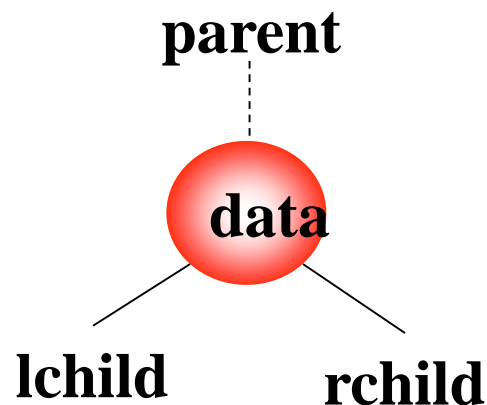
### ■ 二叉树的存储结构

#### □ 链式存储结构

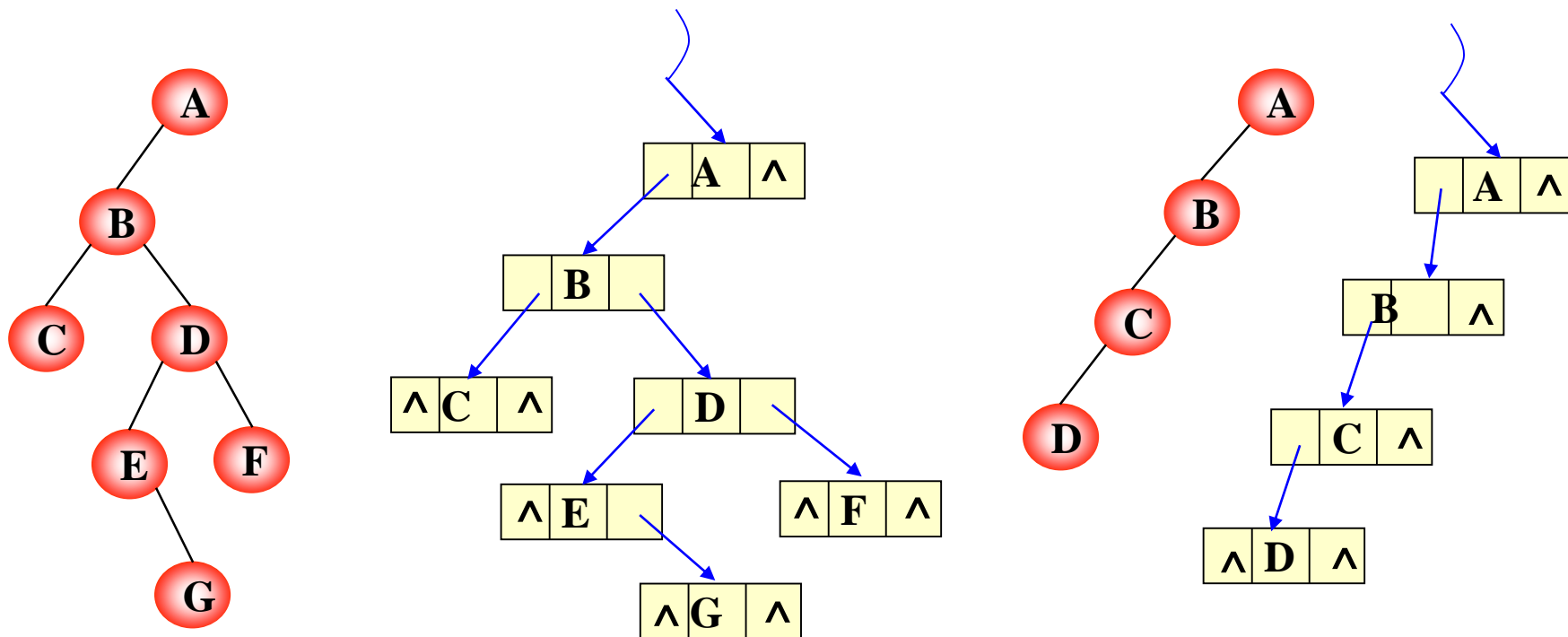
- 二叉树结点的特点
- 存储方式



二叉链表结点结构



## 6.2 二叉树



二叉链表

在  $n$  个结点的二叉链表中，有  $n + 1$  个空指针域。

## 6.2 二叉树

C 语言的类型描述如下：

```
typedef struct BiTNode { // 结点结构  
  
    TElemType    data;  
  
    struct BiTNode *lchild, *rchild;    // 左右孩子指针  
  
} BiTNode, *BiTree;
```

# Part.3

## 6.3 遍历二叉树



## 6.3 遍历二叉树

### ■ 遍历二叉树

#### □ 遍历概念

顺着某一条搜索路径（**次序**）巡访二叉树中的结点，使得每个结点**均被访问一次**，而且**仅被访问一次**。

“**访问**”的含义很广，可以是对结点作各种处理，如：输出结点的信息、修改结点的数据值等，但要求这种访问**不破坏原来的数据结构**。

## 6.3 遍历二叉树

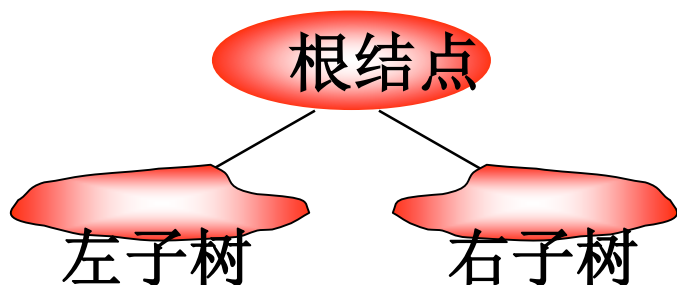
□ 遍历目的：得到树中所有结点的一个线性排列。

□ 二叉树的遍历方法（三条搜索路径）

□ 先上后下按层次遍历

□ 先左(子树)后右(子树)遍历

□ 先右(子树)后左(子树)遍历



依次遍历二叉树中的三个组成部分，便是遍历了整个二叉树

假设：L：遍历左子树    D：访问根结点    R：遍历右子树

则遍历整个二叉树方案共有：

**DLR、LDR、LRD、DRL、RDL、RLD 六种。**

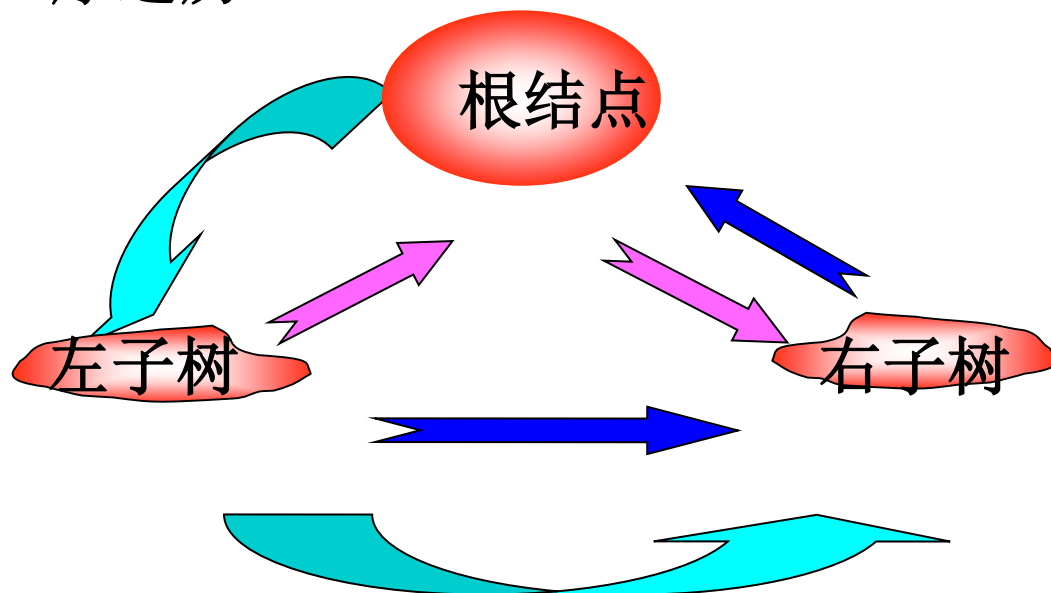
## 6.3 遍历二叉树

若规定先左后右，则只有前三种情况：

**DLR** —— 先（根）序遍历，

**LDR** —— 中（根）序遍历，

**LRD** —— 后（根）序遍历。

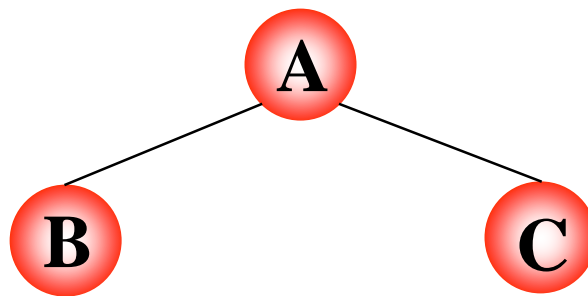


## 6.3 遍历二叉树

### ● 先序遍历二叉树的操作定义：

若二叉树为空，则空操作；否则

- (1) 访问根结点；
- (2) 先序遍历左子树；
- (3) 先序遍历右子树。



先序遍历的顺序为：ABC

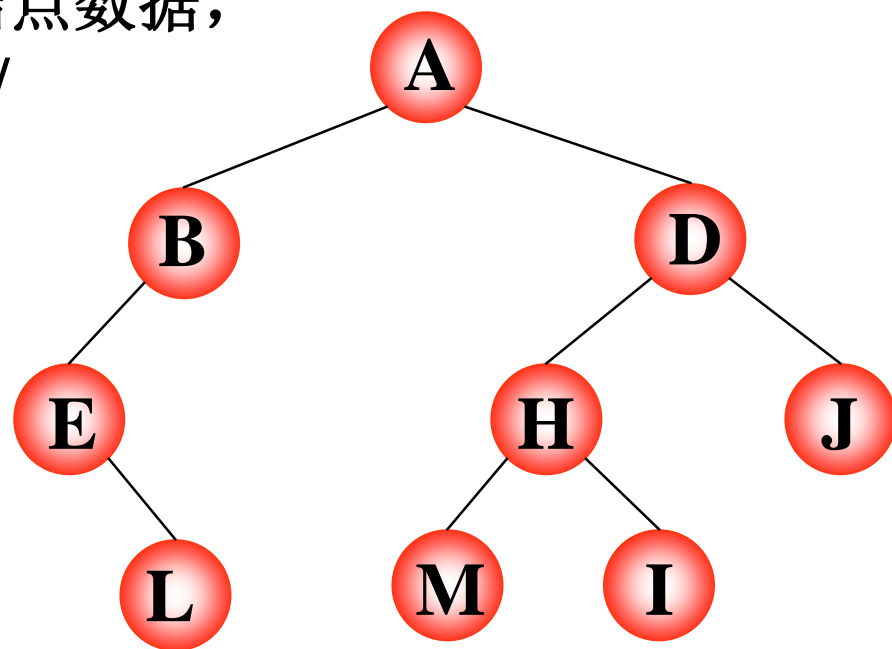
## 6.3 遍历二叉树

先序遍历二叉树基本操作的递归算法在二叉链表上的实现：

```
void PreOrderTraverse (BiTree T)
{
    if (T == NULL)
        return;
    printf("%c", T->data); /*显示结点数据,
    可以更改为其他对结点的操作*/
    //再先序遍历左子树
    PreOrderTraverse(T->lchild);
    //最后先序遍历右子树
    PreOrderTraverse(t->rchild);
} // PreOrderTraverse
```

先序遍历的顺序为：

**ABELDHMIJ**



## 6.3 遍历二叉树

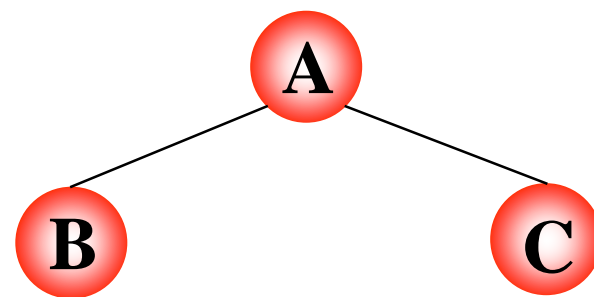
### ● 中序遍历二叉树的操作定义：

若二叉树为空，则空操作；否则

(1) 中序遍历左子树；

(2) 访问根结点；

(3) 中序遍历右子树。



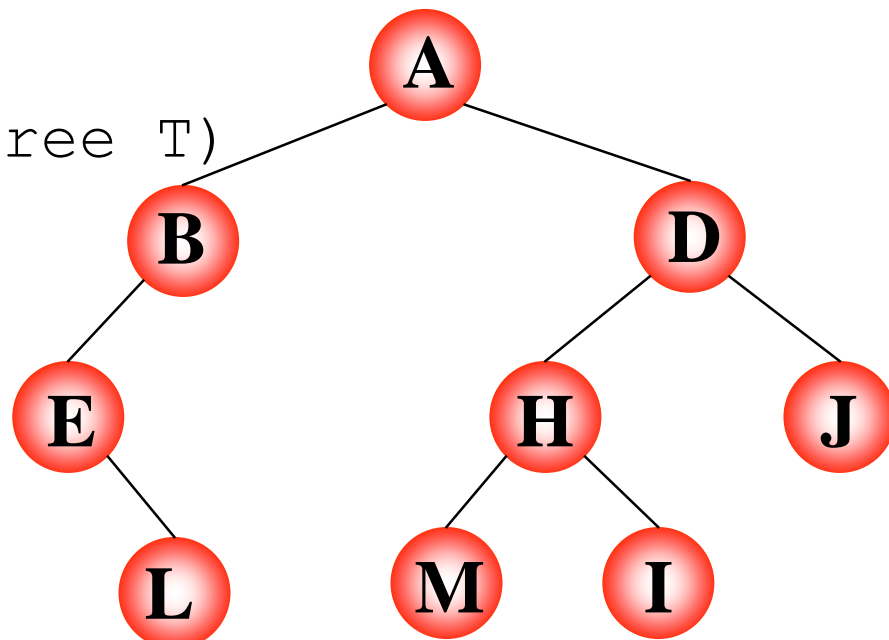
中序遍历的顺序为：BAC

## 6.3 遍历二叉树

### 中序遍历二叉树的递归算法

```
void InOrderTraverse( BiTree T)
{
    if (T == NULL)
        return;

    //中序遍历左子树
    InOrderTraverse(T->lchild);
    //显示结点数据
    printf("%c", T->data);
    //最后中序遍历右子树
    InOrderTraverse(T->rchild);
} //InOrderTraverse
```



**中序遍历的顺序为：**

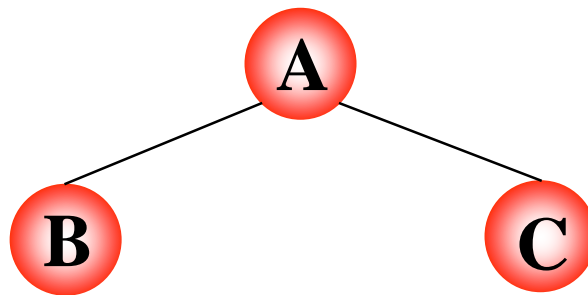
**ELBAMHIDJ**

## 6.3 遍历二叉树

### ● 后序遍历二叉树的操作定义：

若二叉树为空，则空操作；否则

- (1) 后序遍历左子树；
- (2) 后序遍历右子树；
- (3) 访问根结点。



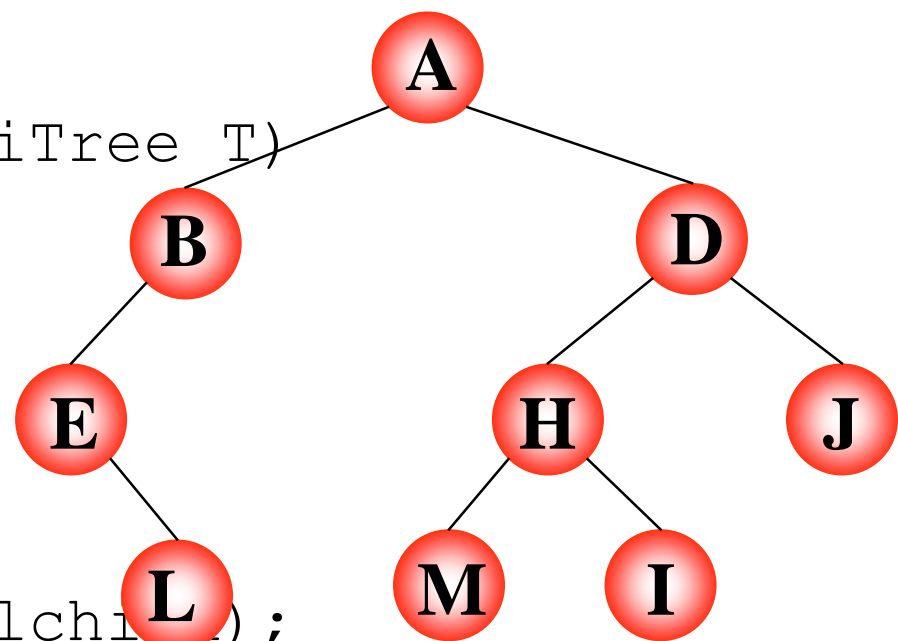
后序遍历的顺序为：BCA



## 6.3 遍历二叉树

### 后序遍历二叉树的递归算法

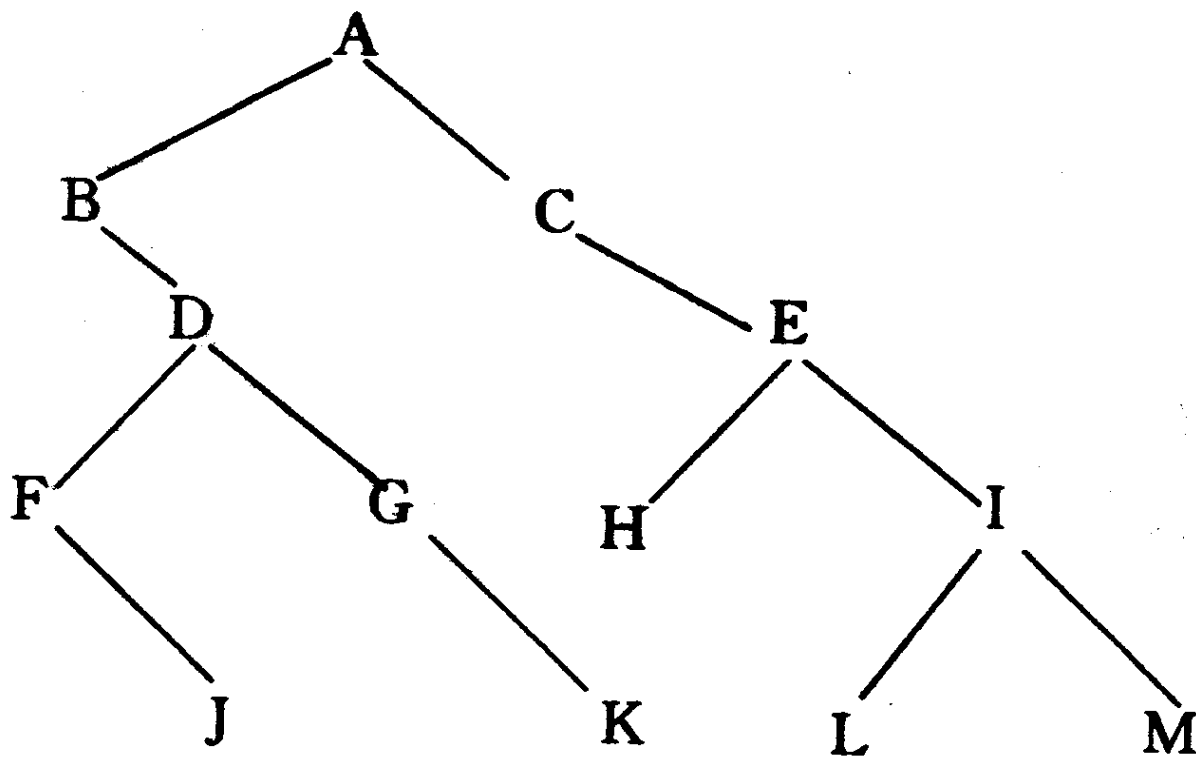
```
void PostOrderTraverse( BiTree T)
{
    if (T == NULL)
        return;
    //先后序遍历左子树
    PostOrderTraverse(T->lchild);
    //再后序遍历右子树
    PostOrderTraverse(T->rchild);
    //显示结点数据
    printf("%c", T->data);
} //PostOrderTraverse
```



**后序遍历的顺序为：**  
**LEBMIHJDA**

## 6.3 遍历二叉树

例：写出如图所示的二叉树分别按先序、中序、后序遍历时得到的结点序列。

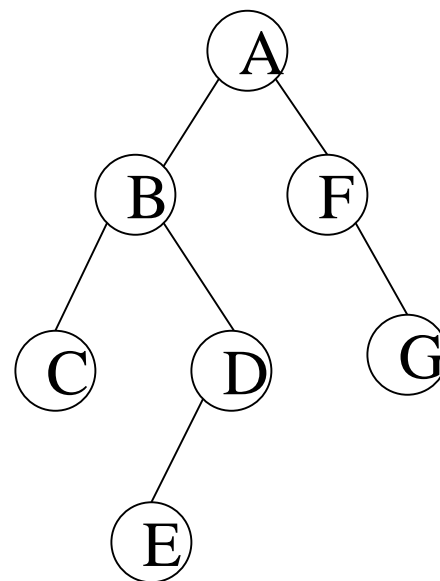
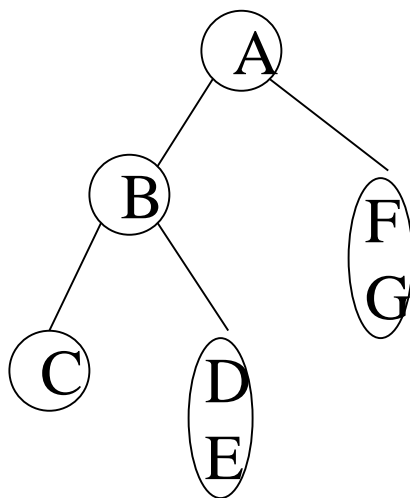
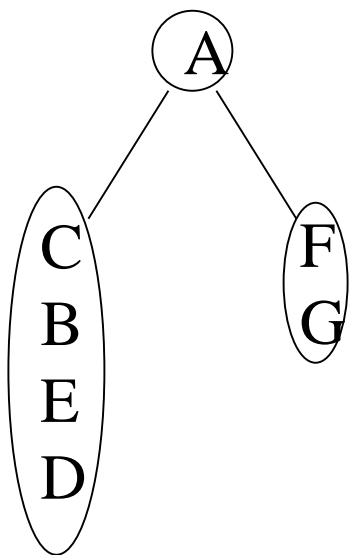


## 6.3 遍历二叉树

例：已知结点的先序序列和中序序列，求整棵二叉树。

- 先序序列：A B C D E F G

- 中序序列：C B E D A F G



## Part.4

### 6.4 哈夫曼树及应用及应 用

## 6.4 哈夫曼树及应用

### □ 最优二叉树（哈夫曼树）

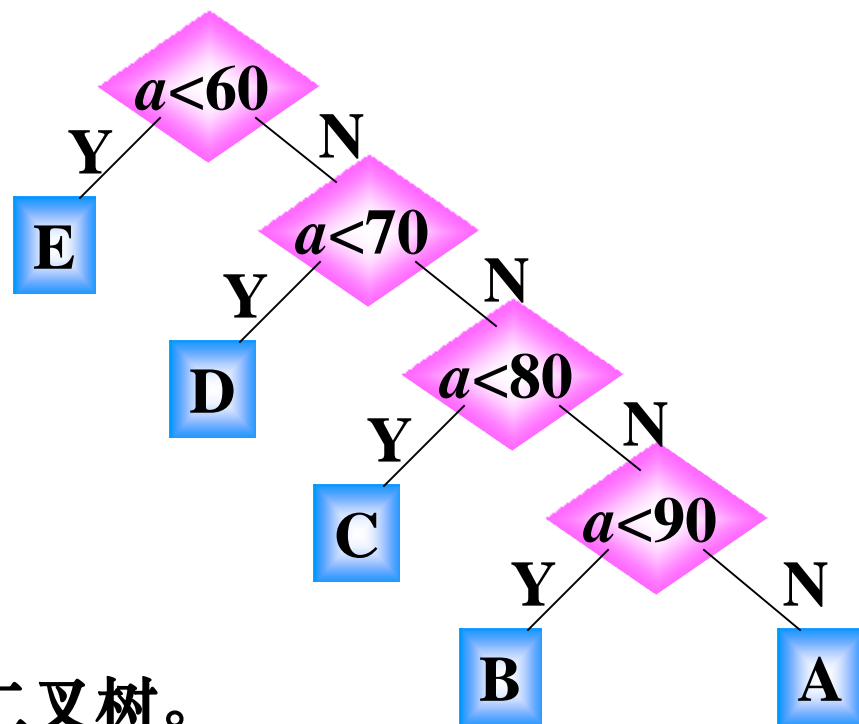
**问题：**什么是哈夫曼树？

例：将学生的百分制成绩转换为五分制成绩： $\geq 90$  分：A，  
80~89分：B，70~79分：C，60~69分：D， $< 60$ 分：E。

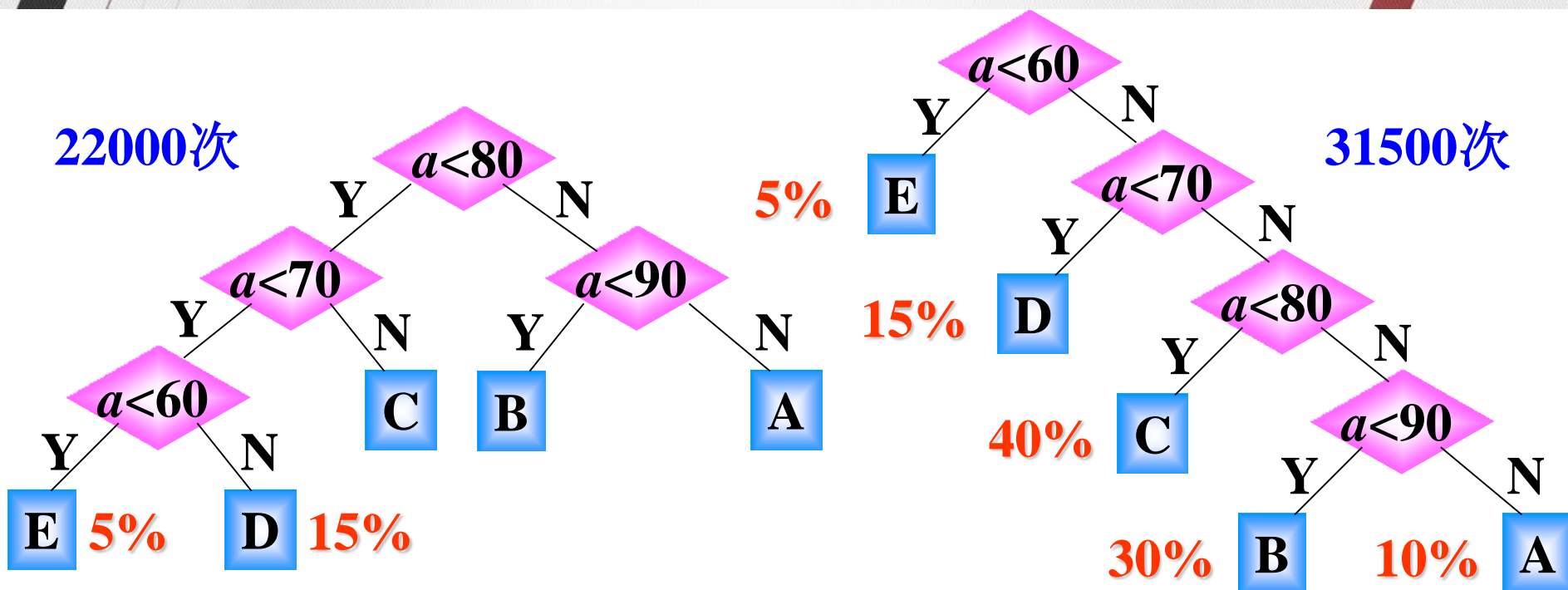
转换程序可用条件语句实现：

```
if ( $a < 60$ )  $b == 'E'$ ;  
    else if ( $a < 70$ )  $b == 'D'$ ;  
        else if ( $a < 80$ )  $b == 'C'$ ;  
            else if ( $a < 90$ )  $b == 'B'$ ;  
                else  $b == 'A'$ ;
```

**判别树：**用于描述分类过程的二叉树。



## 6.4 哈夫曼树及应用



显然：两种判别树的效率是不一样的。

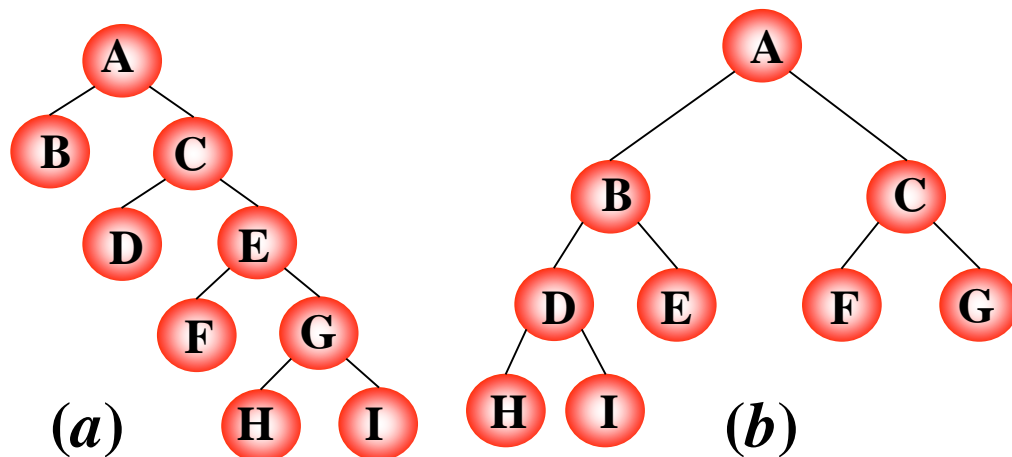
问题：能不能找到一种效率最高的判别树呢？ **哈夫曼树**

## 6.4 哈夫曼树及应用

### 基本概念：

**路径：**从树中一个结点到另一个结点之间的分支构成这两个结点间的路径。

**结点的路径长度：**两结点间路径上的分支数。



(b) 从 A 到 B, C, D, E, F, G, H, I 的路径长度分别为 1, 1, 2, 2, 2, 2, 3, 3。

**树的路径长度：**从树根到每一个结点的路径长度之和。记作：TL

$$TL(a) = 0 + 1 + 1 + 2 + 2 + 3 + 3 + 4 + 4 = 20$$

$$TL(b) = 0 + 1 + 1 + 2 + 2 + 2 + 2 + 3 + 3 = 16$$

完全二叉树是路径长度最短的二叉树。

## 6.4 哈夫曼树及应用

**权：**将树中结点赋给一个有着某种含义的数值，则这个数值称为该结点的权。

**结点的带权路径长度：**从根结点到该结点之间的路径长度与该结点的权的乘积。

**树的带权路径长度：**树中所有叶子结点的带权路径长度之和。

记作：

$$\text{WPL} = \sum_{k=1}^n w_k l_k$$

权值

结点到根的路径长度



## 6.4 哈夫曼树及应用

哈夫曼树：  
最优树

**带权路径长度 (WPL) 最短的树**

**注**

“带权路径长度最短”是在“度相同”的树中比较而得的结果，因此有最优二叉树、最优三叉树之称等等。

哈夫曼树：  
最优二叉树

**带权路径长度 (WPL) 最短的二叉树**

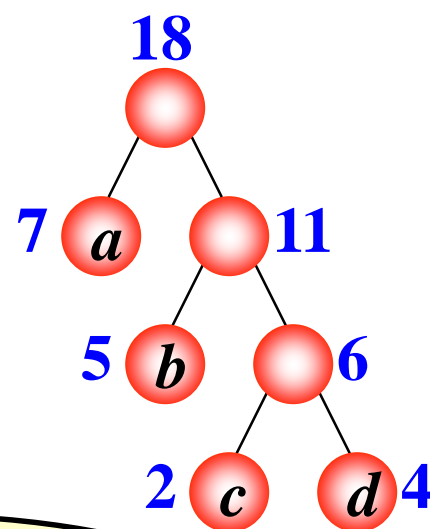
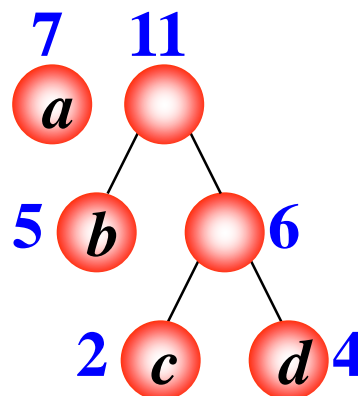
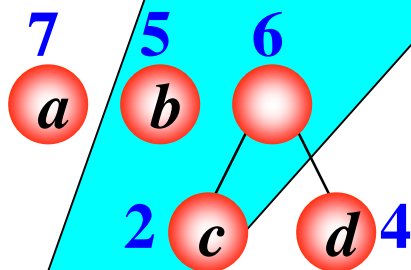
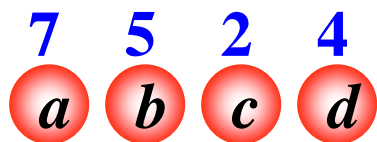
因为构造这种树的算法是由哈夫曼于 **1952** 年提出的，所以被称为哈夫曼树，相应的算法称为**哈夫曼算法**。

## 6.4 哈夫曼树及应用

哈夫曼算法口诀

包含  $n$  棵树的森林要经过  $n-1$  次合并才能形成哈夫曼树，共产生  $n-1$  个新结点。

例：有4个结点



由二叉树的性质 3

包含  $n$  个叶子结点的哈夫曼树中共有  $2n - 1$  个结点。

哈夫曼树的结点的度数为 0 或 2，没有度为 1 的结点。

## 6.4 哈夫曼树及应用

### ● 哈夫曼算法（构造哈夫曼树的方法）

- (1)、根据  $n$  个给定的权值  $\{w_1, w_2, \dots, w_n\}$  构成  $n$  棵二叉树的森林  $F=\{T_1, T_2, \dots, T_n\}$ , 其中  $T_i$  只有一个带权为  $w_i$  的根结点。

构造森林全是根

- (2)、在  $F$  中选取两棵根结点的权值最小的树作为左右子树，构造一棵新的二叉树，且置新的二叉树的根结点的权值为其左右子树上根结点的权值之和。 选用两小造新树

- (3)、在  $F$  中删除这两棵树，同时将新得到的二叉树加入森林中。

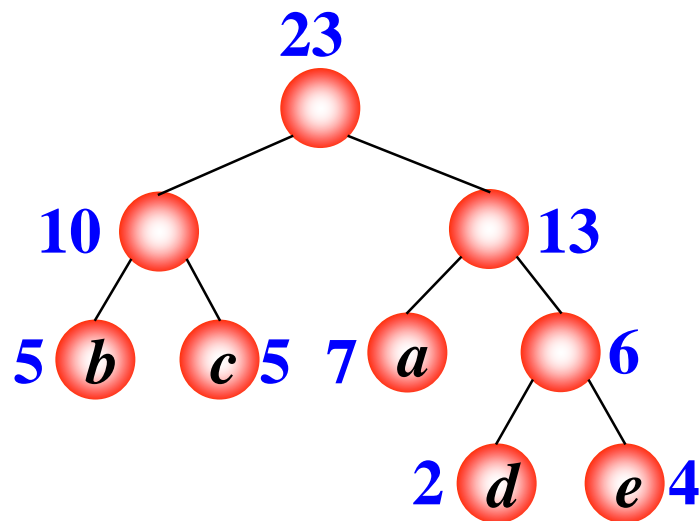
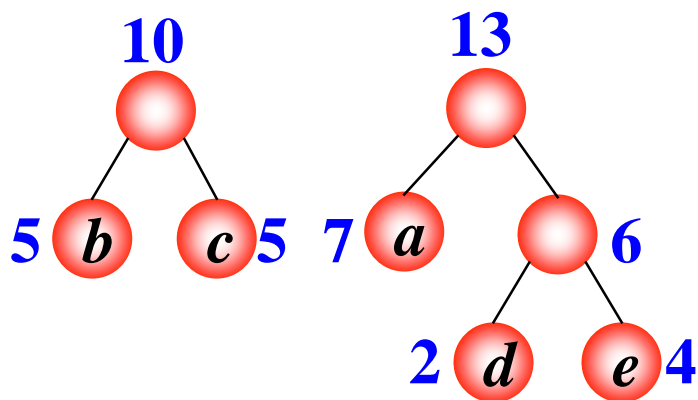
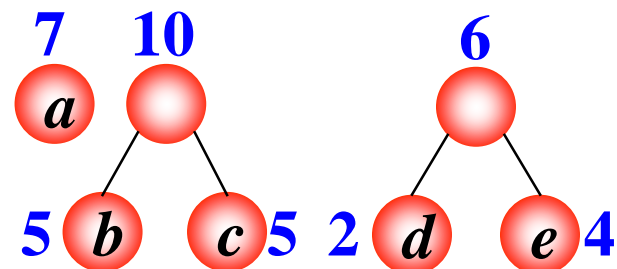
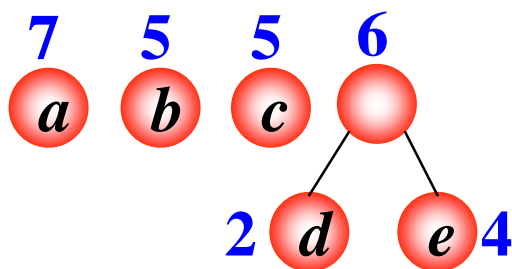
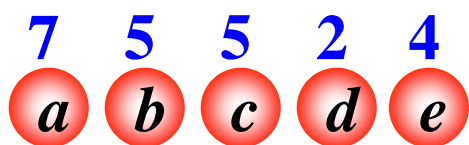
删除两小添新人

- (4)、重复 (2) 和 (3)，直到森林中只有一棵树为止，这棵树即为哈夫曼树。

重复 2、3 剩单根

## 6.4 哈夫曼树及应用

例：有5个结点  $a, b, c, d, e$ ，权值分别为 7, 5, 5, 2, 4，构造哈夫曼树。



## 6.4 哈夫曼树及应用

### ● 哈夫曼编码

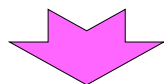
哈夫曼树的应用很广，哈夫曼编码就是其在电讯通信中的应用之一。在电讯通信业务中，通常用二进制编码来表示字母或其他字符，并用这样的编码来表示字符序列。

例：如果需传送的电文为 ‘ABACCDA’，它只用到四种字符，用两位二进制编码便可分辨。假设 A, B, C, D 的编码分别为 00, 01, 10, 11，则上述电文便为 ‘00010010101100’（共 14 位），译码员按两位进行分组译码，便可恢复原来的电文。

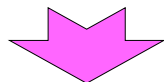
**能否使编码总长度更短呢 ?**

## 6.4 哈夫曼树及应用

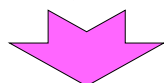
实际应用中各字符的出现频度不相同



用短（长）编码表示频率大（小）的字符



使得编码序列的总长度最小，使所需总空间量最少



### 数据的最小冗余编码问题

在上例中，若假设 A, B, C, D 的编码分别为 0, 00, 1, 01,

则电文 ‘ABACCDA’ 便为 ‘000011010’（共 9 位）。

但此编码存在多义性：可译为 ‘BBCCDA’、‘ABACCDA’、  
‘AAAACCACA’ 等。

## 6.4 哈夫曼树及应用

### ➤ 译码的惟一性问题

要求任一字符的编码都不能是另一字符编码的前缀！

这种编码称为**前缀编码**（其实是非前缀码）。

数据的**最小冗余编码问题**  
在编码过程要考虑两个问题 {  
译码的**惟一性问题**

**利用最优二叉树可以很好地解决上述两个问题**



## 6.4 哈夫曼树及应用

### ➤ 用二叉树设计二进制前缀编码

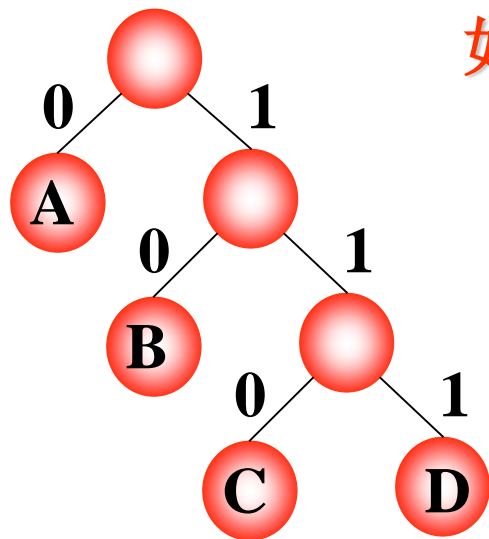
以电文中的字符作为叶子结点构造二叉树。

将二叉树中结点引向其左孩子的分支标 ‘0’，

引向其右孩子的分支标 ‘1’；

每个字符的编码即为从根到每个叶子的路径上得到的 0, 1 序列。如此得到的即为二进制前缀编码。

例：



如此得到的即为二进制前缀编码。

编码： A: 0

B: 10

C: 110

D: 111

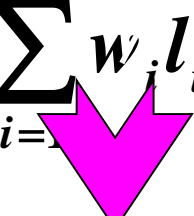
任意一个叶子  
结点都不可能  
在其它叶子结  
点的路径中。



## 6.4 哈夫曼树及应用

### ➤ 用哈夫曼树设计总长最短的二进制前缀编码

假设各个字符在电文中出现的**次数**（或频率）为  $w_i$ ，其**编码长度**为  $l_i$ ，电文中只有  $n$  种字符，则**电文编码总长**为：

$$\text{WPL} = \sum_{i=1}^n w_i l_i$$


从根到叶子的路径长度

叶子结点的权

设计电文总长最短的编码

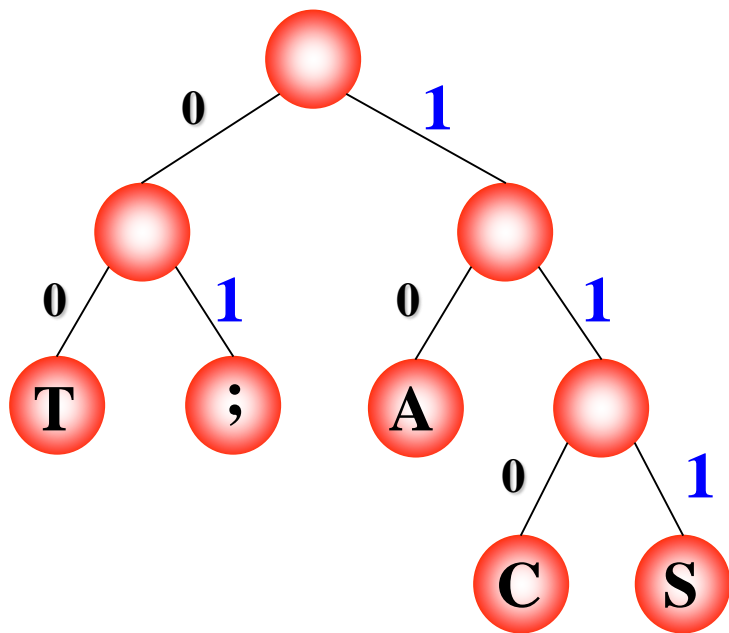
设计哈夫曼树（以  $n$  种字符出现的频率作权）

由哈夫曼树得到的二进制前缀编码称为**哈夫曼编码**

## 6.4 哈夫曼树及应用

### 译码

从哈夫曼树根开始，对待译码电文逐位取码。若编码是“0”，则向左走；若编码是“1”，则向右走，一旦到达叶子结点，则译出一个字符；再重新从根出发，直到电文结束。



电文为 “1101000”

译文只能是 “CAT”

# Part.5

总结

# 总 结

- 1.掌握二叉树的概念、性质。
- 2.熟练二叉树的递归遍历及算法实现。
- 3.掌握哈夫曼树、哈夫曼编码。