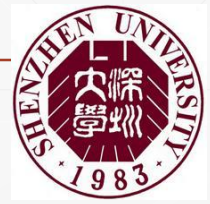
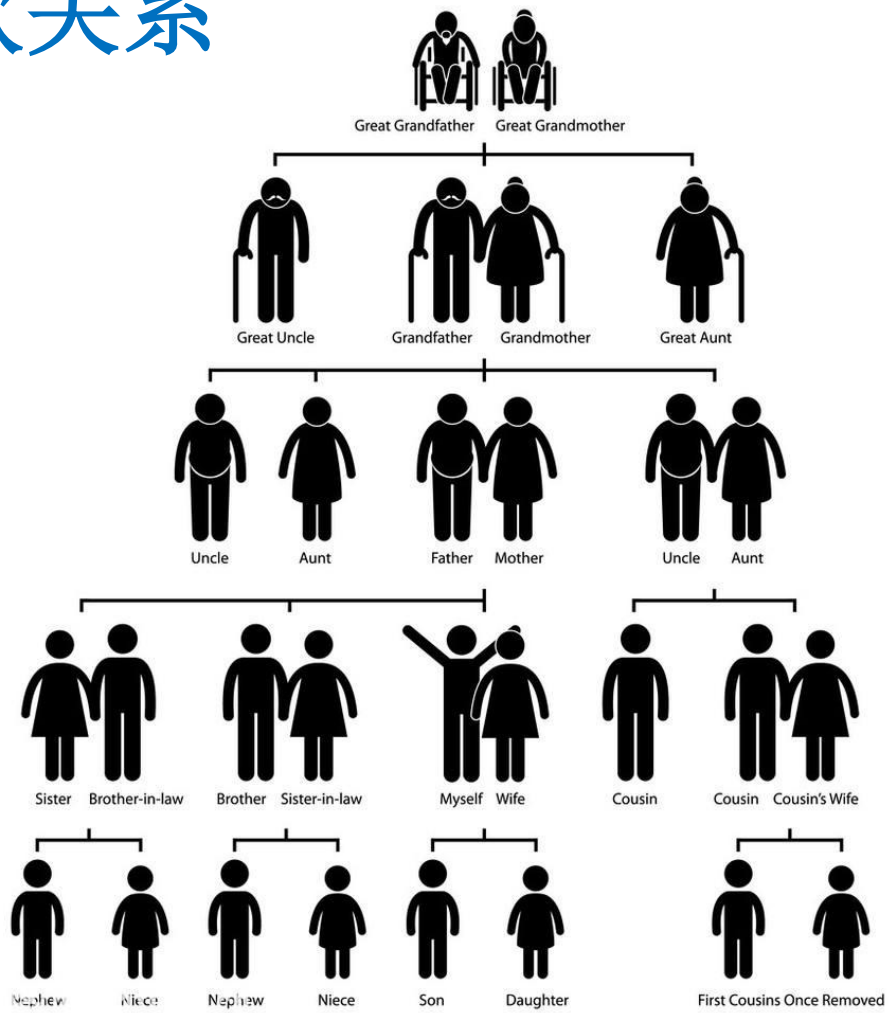
第五章 树与二叉树



客观世界中许多事物存在层次关系

- 社会组织结构
- 人类社会家谱
- 图书信息管理

分层次组织在管理上具有更高的效率!



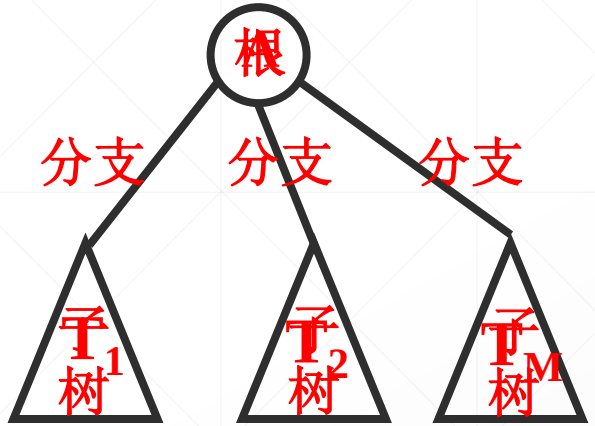
本章学习目标

- 树型结构是一种非线性结构，反映了结点之间的层次关系
- 掌握树(森林)和二叉树的定义及其相关的术语
- 重点掌握二叉树的结构、性质，存储表示和遍历算法
- 了解树的结构性质、存储表示方法和遍历算法；
- 掌握森林(树)与二叉树的对应关系和相互转换方法；
- 了解树型结构的应用，重点掌握赫尔曼树的概念、构造方法，以及哈夫曼编码的原理及实现方法。

主要内容

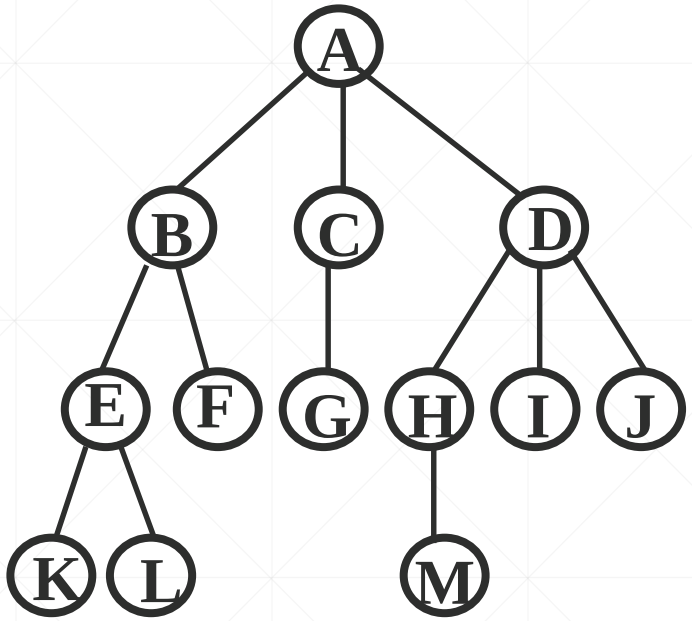
- 3.1 树的定义与基本术语
- 3.2 二叉树
- 3.3 遍历二叉树
- 3.4 树与森林
- 3.5 赫尔曼树及其应用

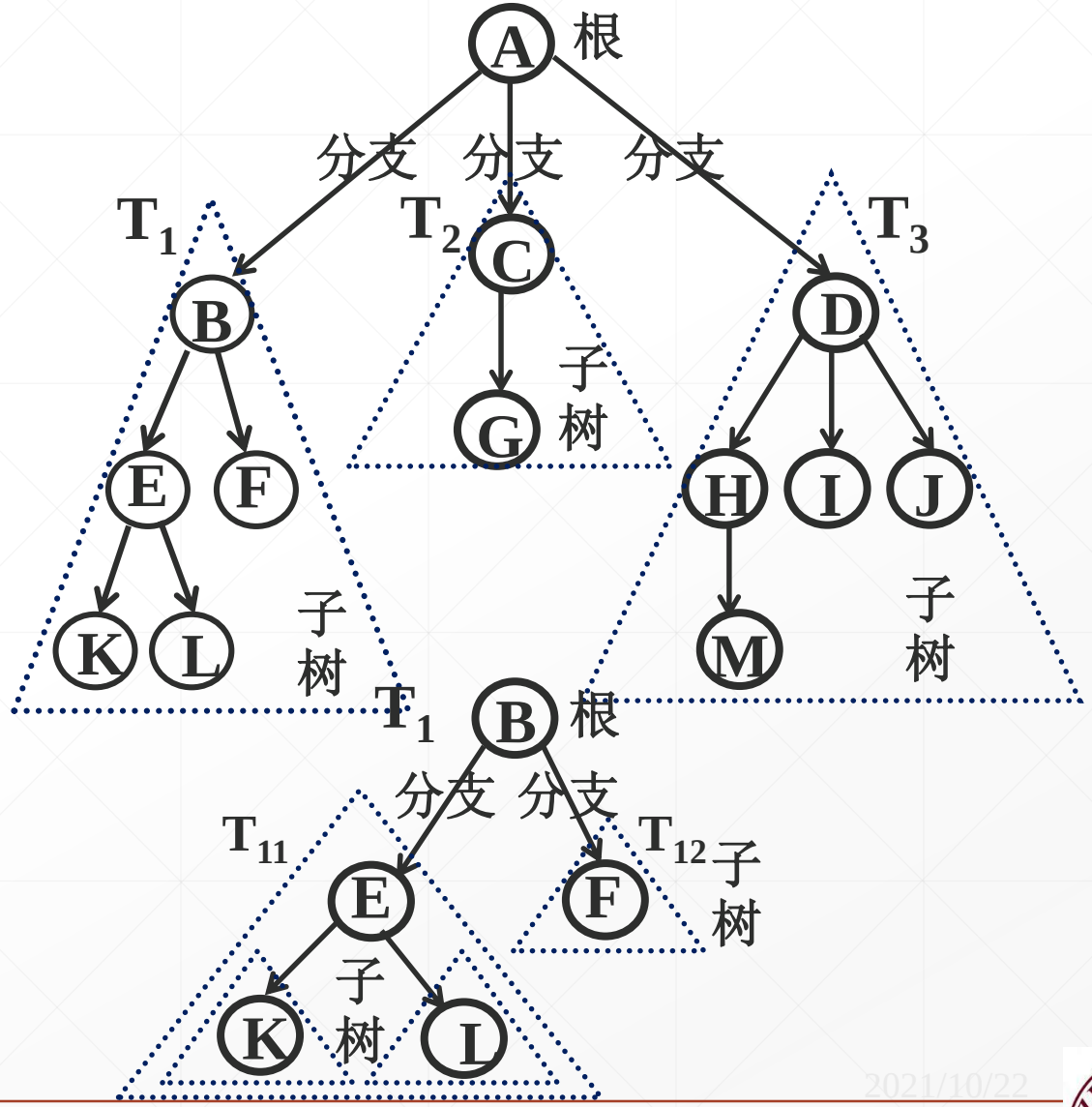
树（Tree）的概念与基本术语



- 树是有$n(n \geq 0)$个结点的有限集合
- 如果$n = 0$，称为空树；
- 如果$n > 0$，称为非空树，对于非空树，有且仅有一个特定的称为根（Root）的结点（无直接前驱）
- 如果$n > 1$，则除根以外的其他结点划分为$m(m > 0)$个互不相交的有限集$T_1, T_2, \cdots, T_m$，其中每个集合本身又是一颗树，并且称为根的子树（SubTree）
- 每个结点都有唯一的直接前驱，但可能有多个后驱

[例]树的定义

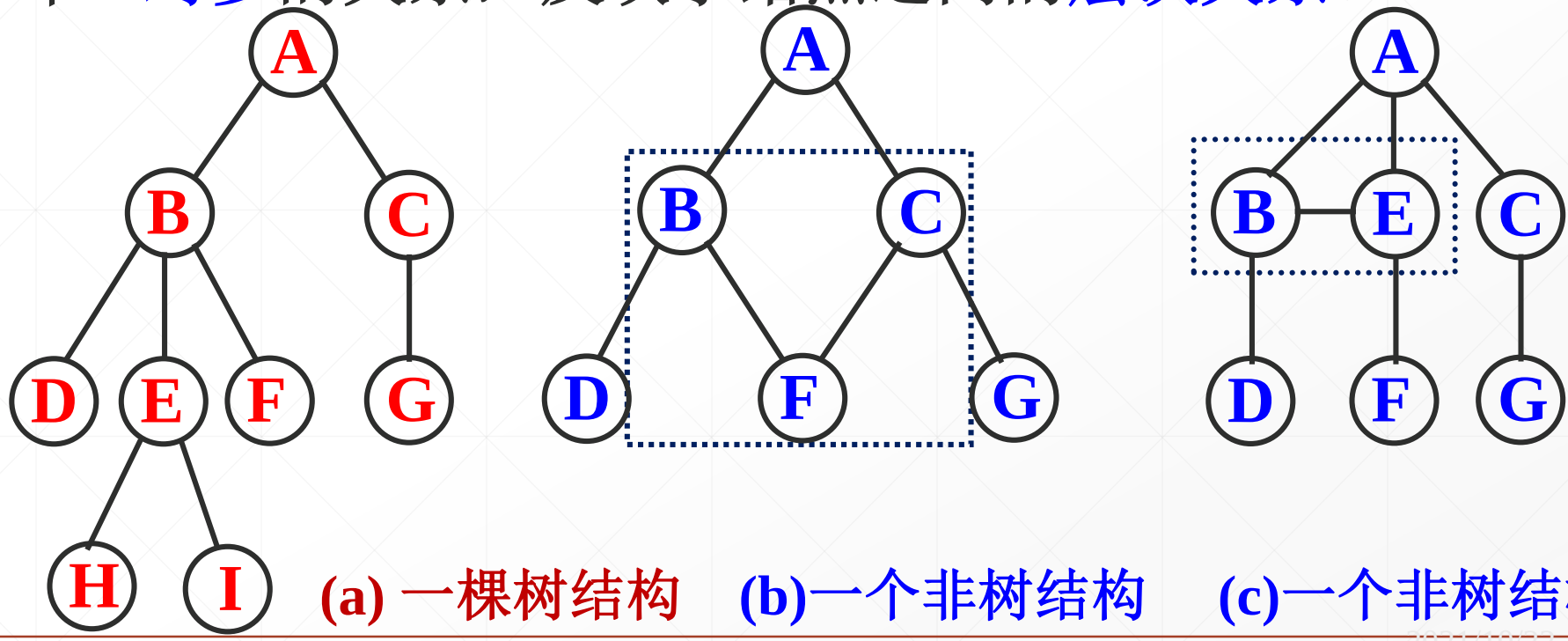




- A是根
- 其余结点分成三个互不相交的子集
- T_1, T_2, T_3都是根A的子树，且本身也是一颗树

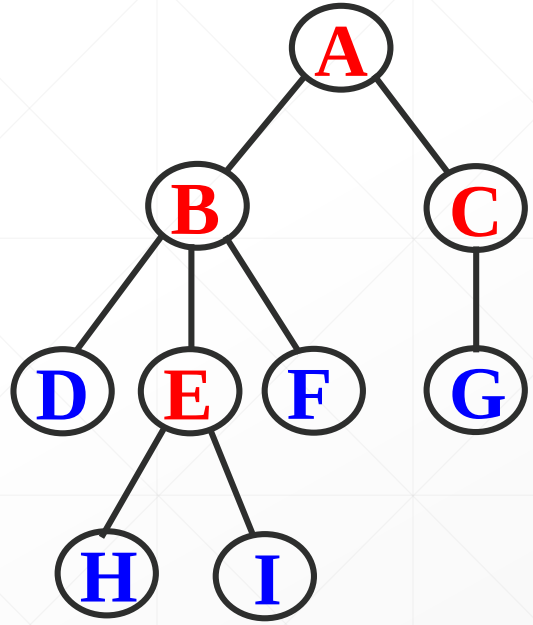
树的逻辑结构特点

- 除根结点之外，每棵子树的根结点有且仅有一个直接前驱，但可以有0个或多个直接后继。
- 即一对多的关系，反映了结点之间的层次关系。



(a) 一棵树结构　(b)一个非树结构　(c)一个非树结构

树的基本术语



- 结点：包含一个数据元素及若干指向其子树的分支
- 结点的度（Degree）：结点所具有的子树的个数
- 树的度：树内各结点的度的最大值
- 叶结点：度为 0 的结点［没有子树的结点］，也称为终端结点
- 分支节点：度不为 0 的结点[包括根结点]，

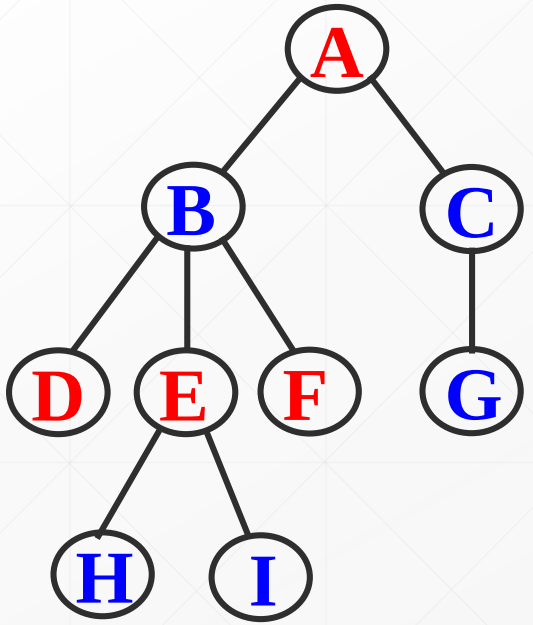
  也称为非终端结点。

  除根外称为内部结点

树的基本术语（cont.）

- 孩子（Child）：结点的子树的根［直接后继，可能有多个］
- 双亲（Parent）：孩子的直接前驱［最多只能有一个］
- 兄弟（Sibling）：同一双亲的孩子
- 子孙：以某结点为根的树中的所有结点
- 祖先：从根到该结点所经分支上的所有结点

[即在树中，如果有一条路径从结点x到结点y，那么x就称为y的祖先，而y称为x的子孙。]

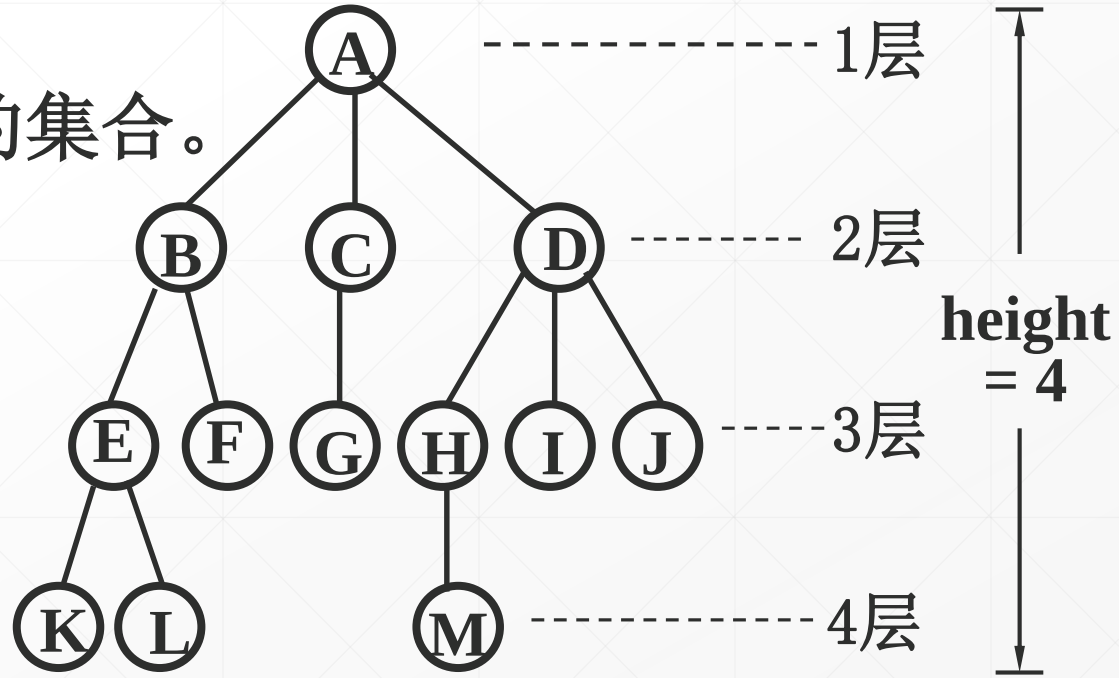


树的基本术语（cont.）

- 层次：根结点为第一层，其孩子为第二层，以此类推
- 深度：树中所有结点的最大层次
- 森林：*m (m≥0)* 棵互不相交的树的集合。

[对树中每个结点而言，

其子树的集合即为森林]



树型结构和线性结构的比较

线性结构

- 第一个数据元素 –无前驱
- 最后一个数据元素 –无后继
- 其它数据元素 –

一个前驱, 一个后继

“一对一”

vs.

树型结构

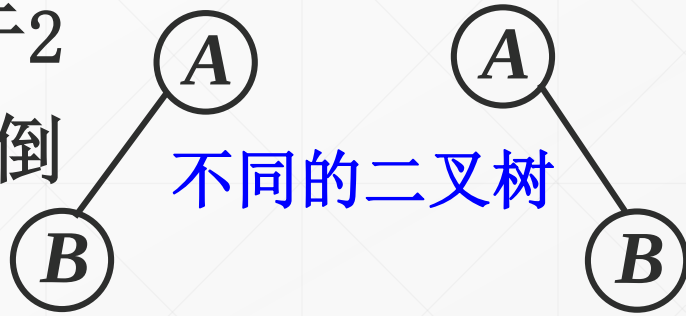
- 根结点（只有一个）–无双亲
- 叶子节点(可以有多个)–无孩子
- 其它结点 –

一个双亲, 多个孩子

“一对多”

二叉树（Binary Tree）

- 二叉树一个是$n(n \geq 0)$个结点的有限集合
- 该集合或者为空（称为空二叉树）；或者是由一个根结点和两棵互不相交的、分别称为左子树和右子树的二叉树组成。
- 结构特点：
  - 每个结点最多有2颗子树，即结点的度不大于2
  - 子树有左右之别，子树的次序(位置)不能颠倒
  - 即使某结点只有一棵子树，也有左右之分

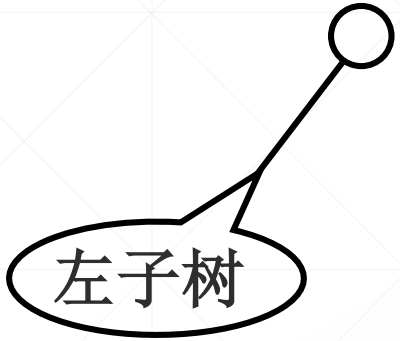


二叉树的基本形态：

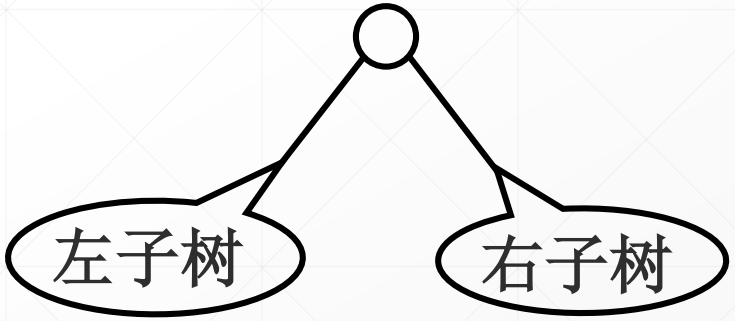
空二叉树

只有一个根结点

根结点只有右子树

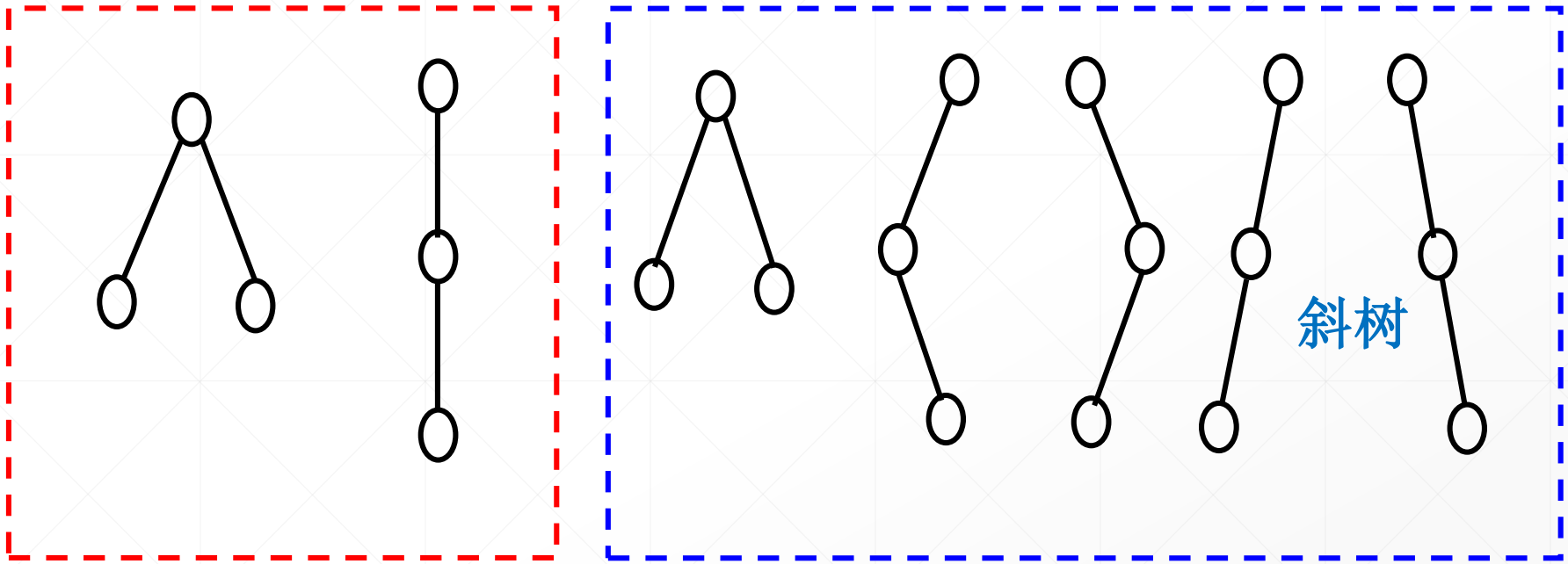


根结点只有左子树



根结点同时有左右子树

具有3个结点的树和二叉树



树的不同构形态　　二叉树的不同构形态

二叉树的性质-1

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

［证明］（数学归纳法）

1. 当$i = 1$时，第1层只有一个根结点，$2^{i-1} = 2^0 = 1$，结论成立。

2. 假定$i=k$ $(1 \leq k < i)$ 时结论成立，即第k 层上至多有2^{k-1}个结点，则$i=k+1$时，因为第$k+1$层上的结点是第k 层上结点的孩子，而二叉树中每个结点最多有2个孩子，故在第$k+1$ 层上最大结点个数为第k 层上的最大结点个数的两倍，即 $2 * 2^{k-1} = 2^k$ 。结论成立。 □

二叉树的性质-2

- 深度为 k 的二叉树至多有 2^k-1个结点

[证明]

1. 由性质1，已知第 i 层上的结点数至多有 2^{i-1}个

2. 深度为 k 的二叉树的最大结点数为：

$\sum_{i=1}^{k}$(第 i 层上的最大结点数) $=\sum_{i=1}^{k}2^{i-1}=2^0+2^1+2^2+\cdots+2^{k-1}=2^k-1$ □

- 另外，每一层至少要有一个结点，因此，深度为 k 的二叉树至少有 k个结点。

二叉树的性质-3

- 如果二叉树终端结点数为 n_0，度为2的结点数为 n_2，则 $n_0 = n_2+1$，
  而与度数为1的结点数 n_1无关。

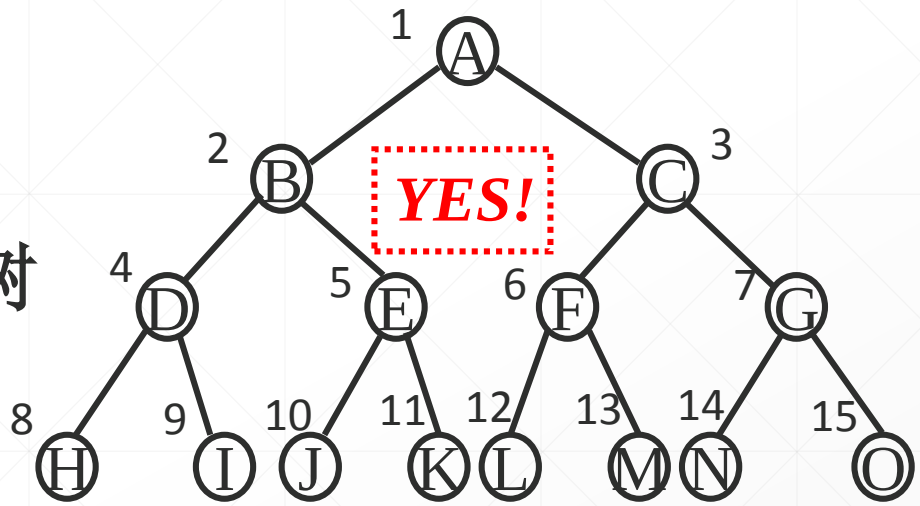
[证明]

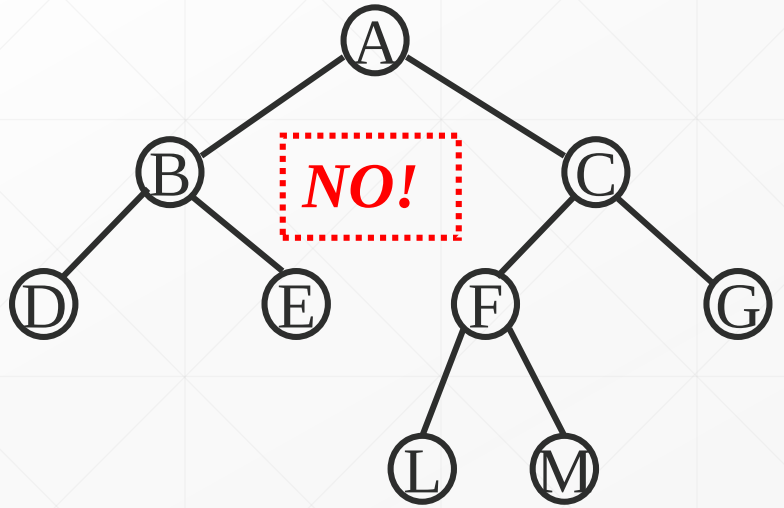
1. 设 n_i 为度为 i 的结点，则总结点数为 $n = n_0+n_1+n_2$
2. 设 B 为二叉树的分支数，在n 个结点的二叉树中，共有n-1条分支，
   故 $B = n-1$；
3. 在这些分支中，度为1和度为2的结点分别提供1条和2条分支，
   故 $B = n_1 + 2 * n_2$；所以有$n-1 = n_1 +2 * n_2$；

因此可以得到 $n_0 = n_2+1$ □

满二叉树

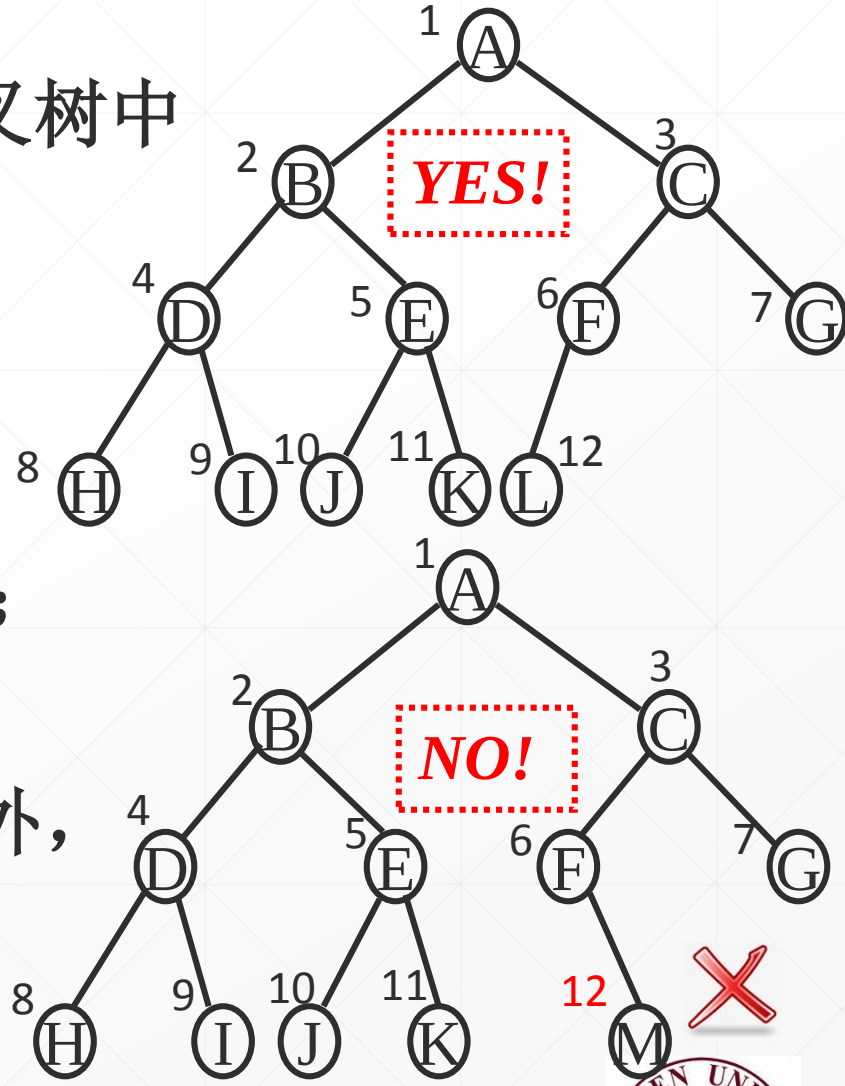
- 一个深度为 k 且有 2^k-1个结点的二叉树
- 每层上的结点数都是最大数
- 结构特点：
  - 分支结点都有两棵子树
  - 叶子结点都在最后一层
- 可以自上而下、自左至右连续编号
- 满二叉树在相同高度的二叉树中，结点数、分支结点数和叶结点数都是最多的。



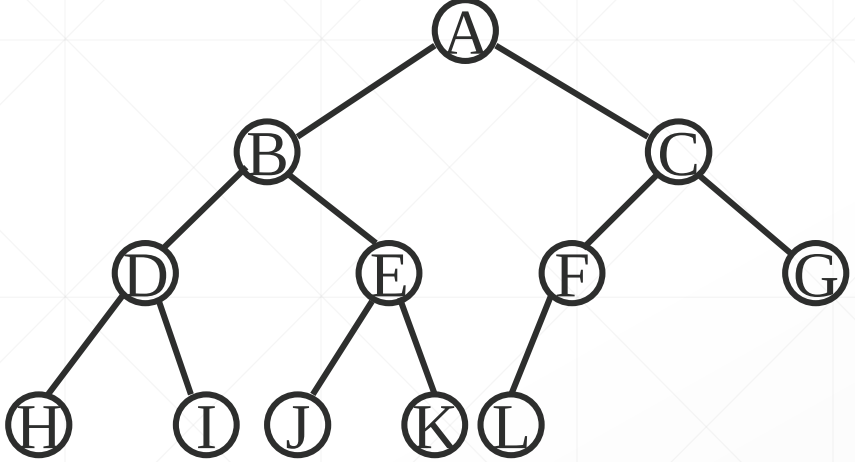


完全二叉树

- 当且仅当每一个结点都与深度相同的满二叉树中编号从 1 到 n 的结点一一对应的二叉树
- 假设深度为 k
  - 所有的叶结点都出现在 k或 $k-1$层；
  - $k-1$层的所有叶都在非终结结点的右边；
  - 除了$k-1$层的最右非终结结点可能有一个（只能是左分支）或两个分支之外，其余非终结结点都有两个分支。



完全二叉树（cont.）



- 结构特点：
  - 叶子结点只能出现在最下两层，且最下层的叶子结点都集中在二叉树的左部；
  - 左子树深度与右子树深度相等或大于 1
  - 完全二叉树中如果有度为1的结点，只可能有一个，且该结点只有左孩子。
  - 深度为 k 的完全二叉树的前k-1层一定是满二叉树。

完全二叉树的性质

- 性质4:具有 n 个结点的完全二叉树，其深度为$\lfloor \log_2 n \rfloor + 1$
- 设 k 为深度，由二叉树的性质2，已知

$$2^{k-1} - 1 < n \leq 2^k - 1$$

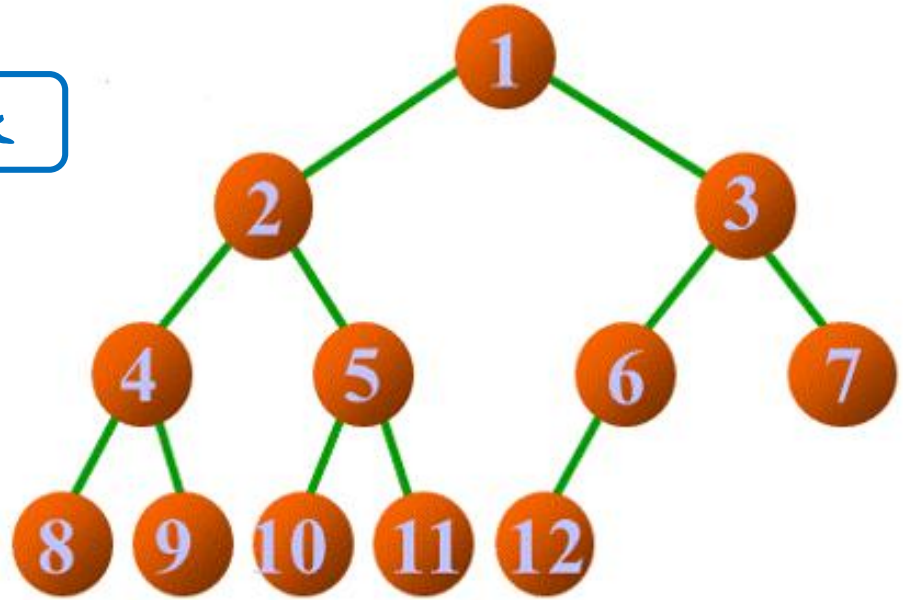
前k-1层最多结点数　　前k层最多结点数

即

$$2^{k-1} \leq n < 2^k$$

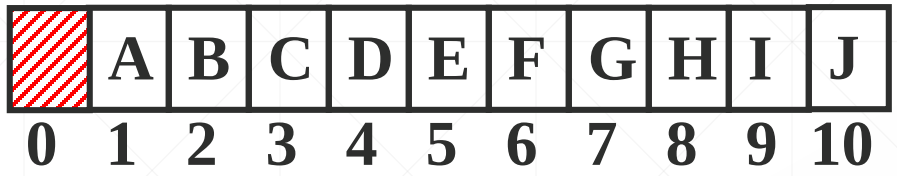
$$k - 1 \leq \log_2 n < k$$

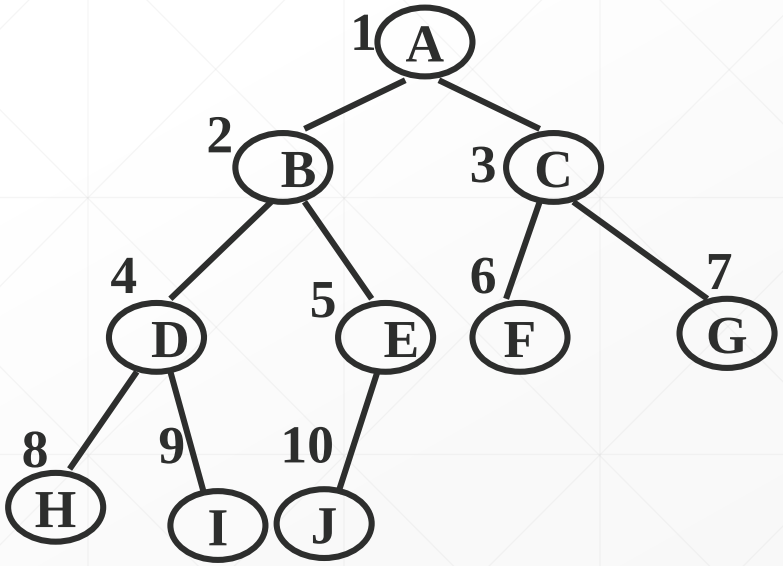
$$k = \lfloor \log_2 n \rfloor + 1$$



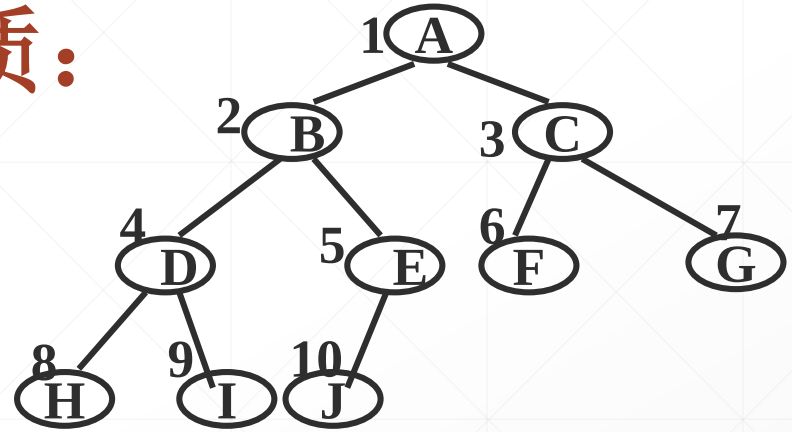
完全二叉树的顺序存储结构：

- 完全二叉树的 n个结点自上而下、同一层自左向右连续编号（$1, 2, \ldots, n$），则这种存储表示方法称为完全（满）二叉树的顺序存储结构。





完全二叉树的顺序存储结构的性质：

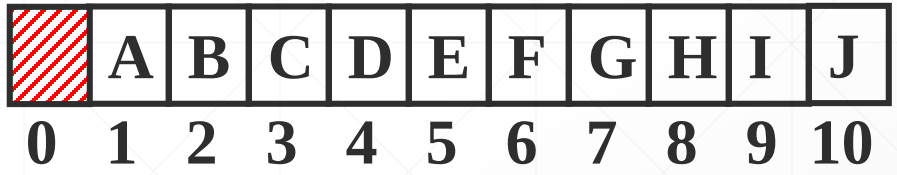


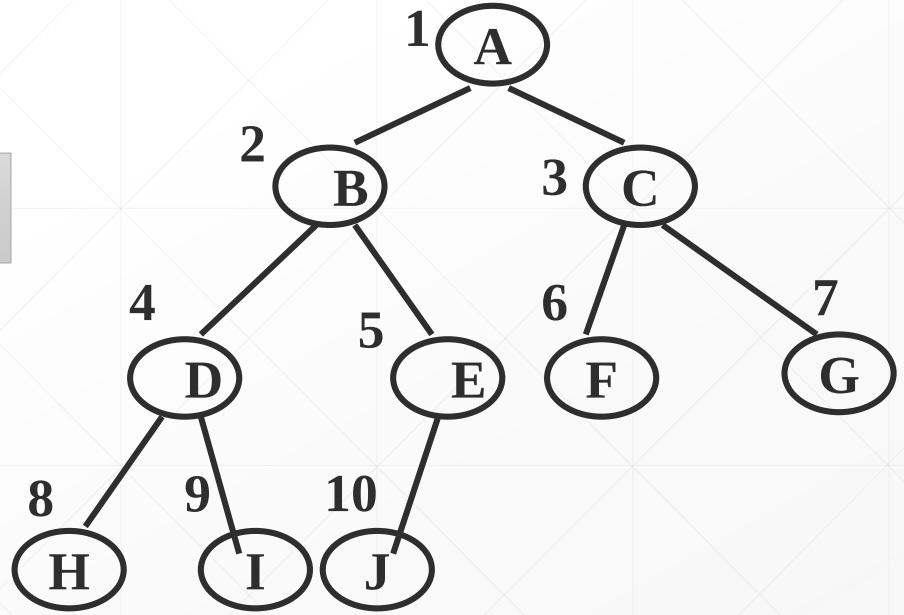
- 性质5：对任一结点i（$1 \quad i \le n$）
  - ✓若$i = 1$，则 i 是根结点，无父结点；
    若$i > 1$，则 i 的父结点为$\lfloor i/2 \rfloor$
  - ✓若$2 * i \le n$, 则 i 有左孩子且为 $2 * i$；否则， i 无左孩子
  - ✓若$2 * i + 1 \le n$, 则 i 有右孩子且为 $2 * i + 1$；否则， i 无右孩子
  - ✓若 i 为偶数，且$i < n$，则有右兄弟，且为 $i + 1$
  - ✓若 i 为奇数，且$i \le n$ && $i != 1$，则有左兄弟，且为$i - 1$

二叉树的顺序存储结构

- 完全（满）二叉树的顺序存储结构
  - 采用一维数组，按层序顺序依次存储二叉树的每一个结点。如下图所示：

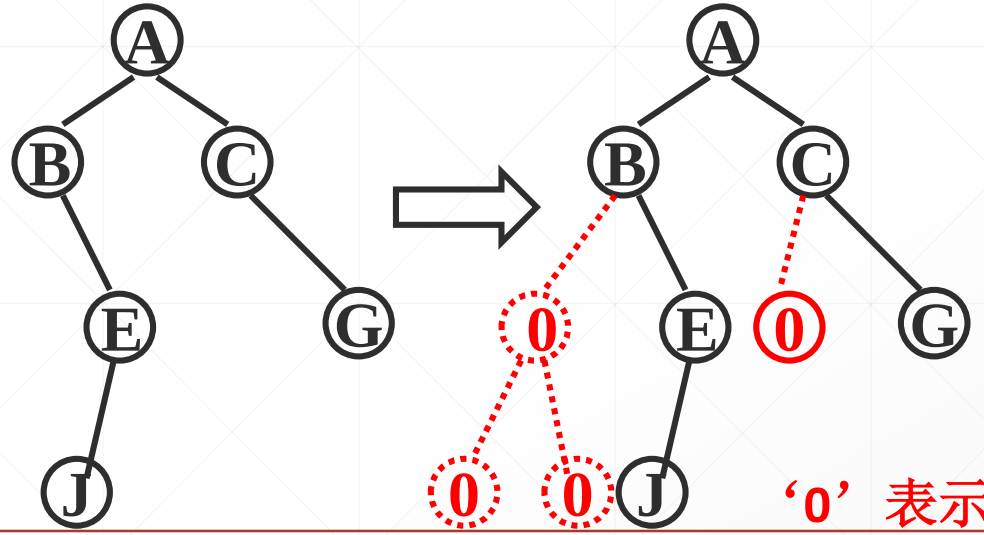
自上而下、同一层自左向右





二叉树的顺序存储结构（cont.）

- 一般二叉树的顺序存储结构
  - 实现：按完全二叉树的结点层次编号，依次存放二叉树中的数据元素
  - 特点：结点间关系蕴含在其存储位置中

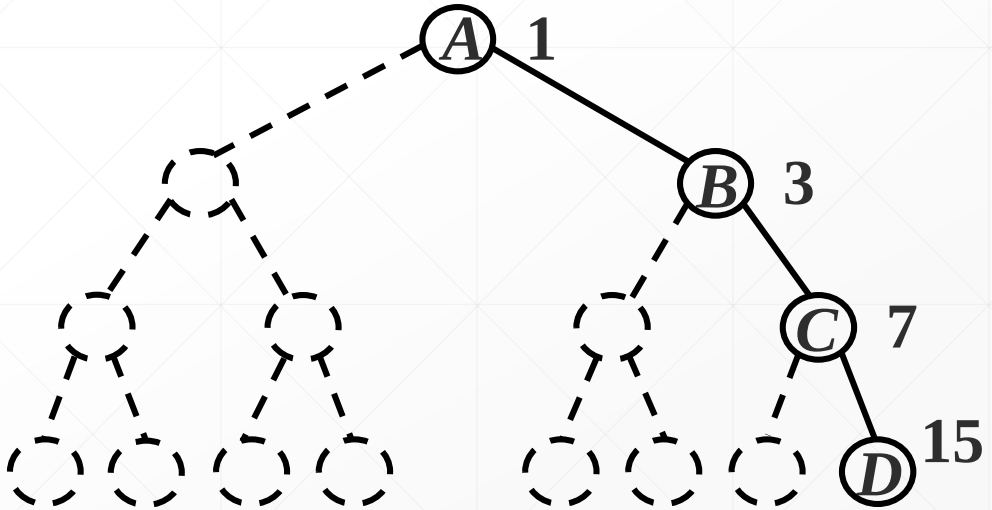


	A	B	C	0	E	0	G	0	0	J
0	1	2	3	4	5	6	7	8	9	10

‘0’表示不存在此结点

二叉树的顺序存储结构（cont.）

- 一般二叉树的顺序存储结构
  - 一棵斜树（树中不存在度为2的结点）的顺序存储会怎样呢？
    - 深度为 k 的树需分配$2^k - 1$个存储单元
    - 需增加很多空结点，造成存储空间的浪费



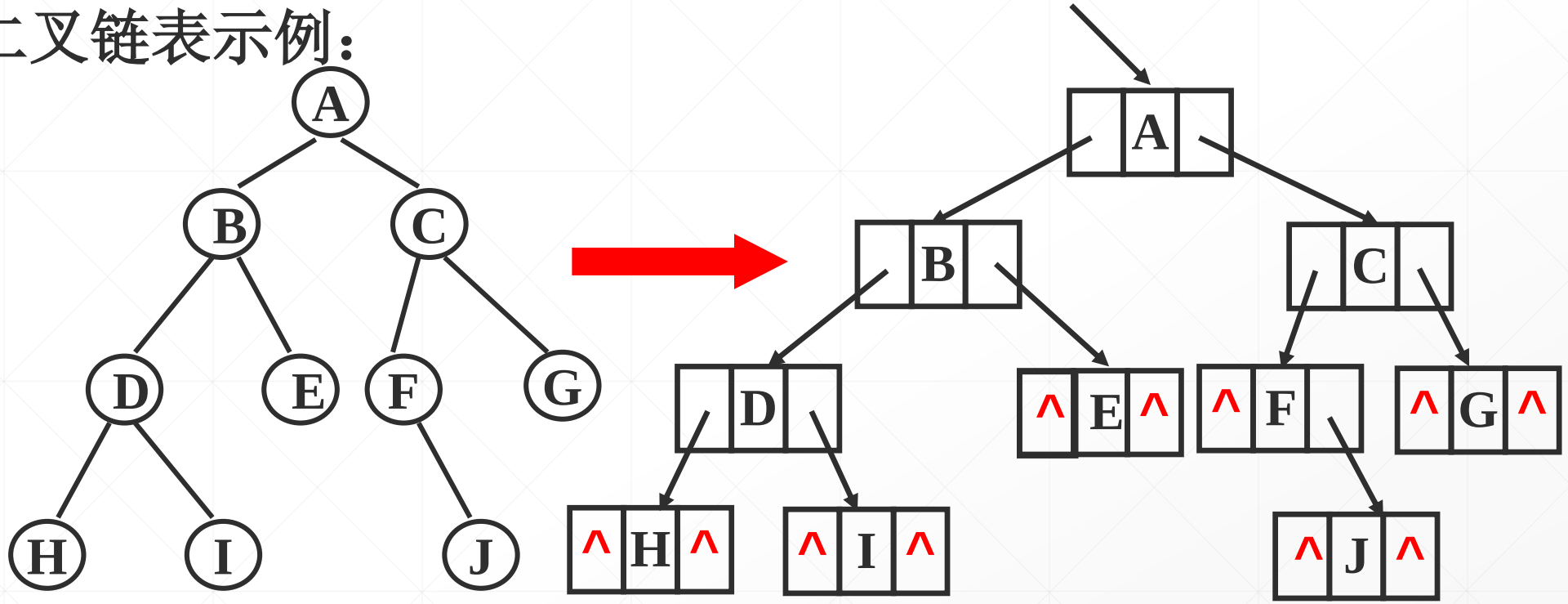
二叉树的链式存储结构

- 二叉链表：每个结点除了存放结点数据信息外，还设置两个指示左、右孩子的指针，如果该结点没有左或右孩子，则相应的指针为空。并用一个指向根结点的指针标识这个二叉树
- 结点结构：

data：数据域，存放该结点的数据信息；
lChild：左指针域，存放指向左孩子的指针；
rChild：右指针域，存放指向右孩子的指针。

二叉树的链式存储结构（cont.）

- 二叉链表示例：

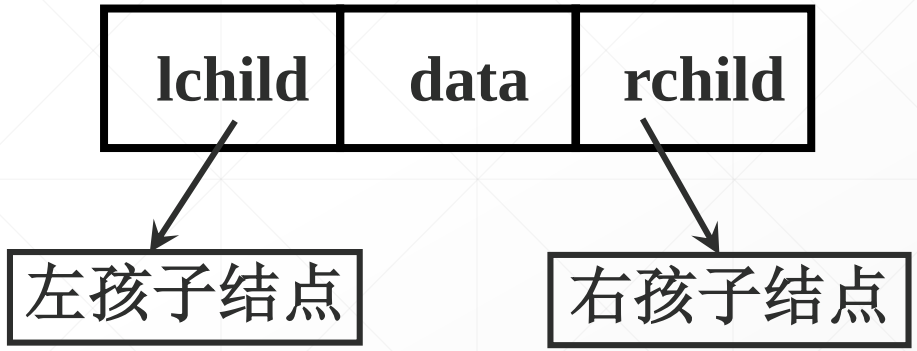


✓ 具有n个结点的二叉链表中，有多少个空指针？有多少指向孩子结点的指针？

二叉树的链式存储结构（cont.）

- 二叉链表的存储结构定义：

```
struct  BiNode  {
    datatype  data ;
    BiNode *lchild, *rchild ;
} ;
BiNode *BiTree ;
```



二叉树的链式存储结构（cont.）

- 三叉链表：

  采用数据域加上左、右孩子指针以及双亲指针

lChild	data	parent	rChild

二叉树的链式存储结构（cont.）

- 三叉链表示例：