第3章 树与二叉树

学习目标

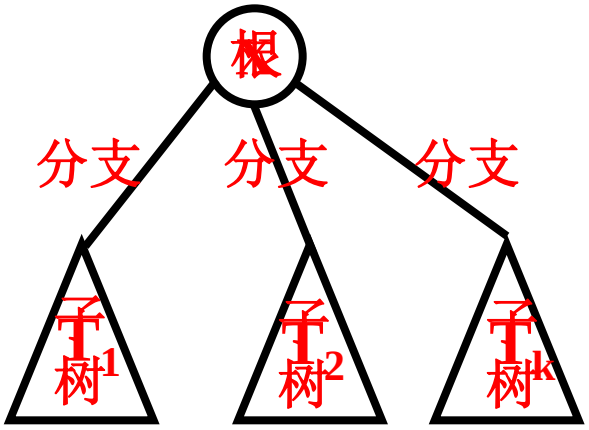
- 树型结构是一种非线性结构，反映了结点之间的层次关系，在计算机科学与软件工程中有着广泛的应用。
- 掌握树(森林)和二叉树的定义及其相关的术语；
- 重点掌握二叉树的结构、性质，存储表示和四种遍历算法；二叉树线索化的实质及线索化的过程；
- 了解树的结构性质、存储表示方法和遍历算法；
- 掌握森林(树)与二叉树的对应关系和相互转换方法；
- 了解树型结构的应用，重点掌握哈夫曼树的概念和构造方法，哈夫曼编码和译码的原理及实现方法。

本章主要内容

- 3.1 树与二叉树的基本术语
- 3.2 二叉树
- 3.3 堆
- 3.4 选择树
- 3.5 树
- 3.6 森林（树）与二叉树的相互转换
- 3.7 树的应用
- 本章小结

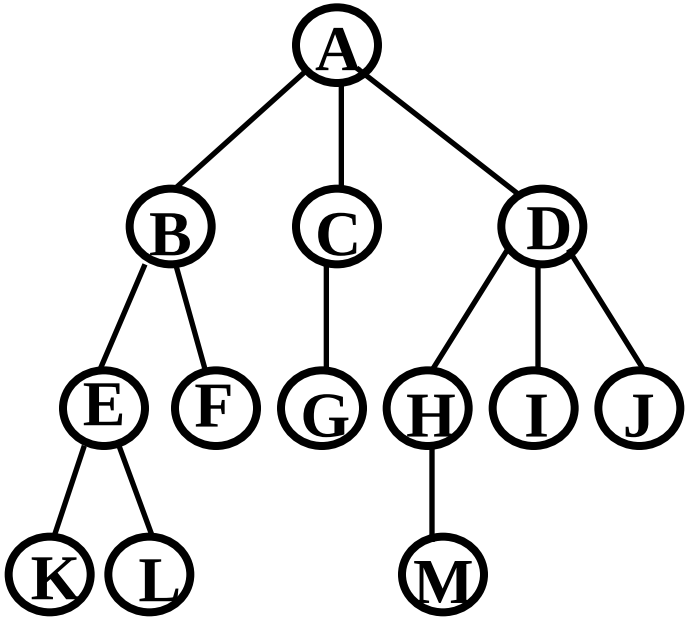
3.1 树与二叉树的基本术语

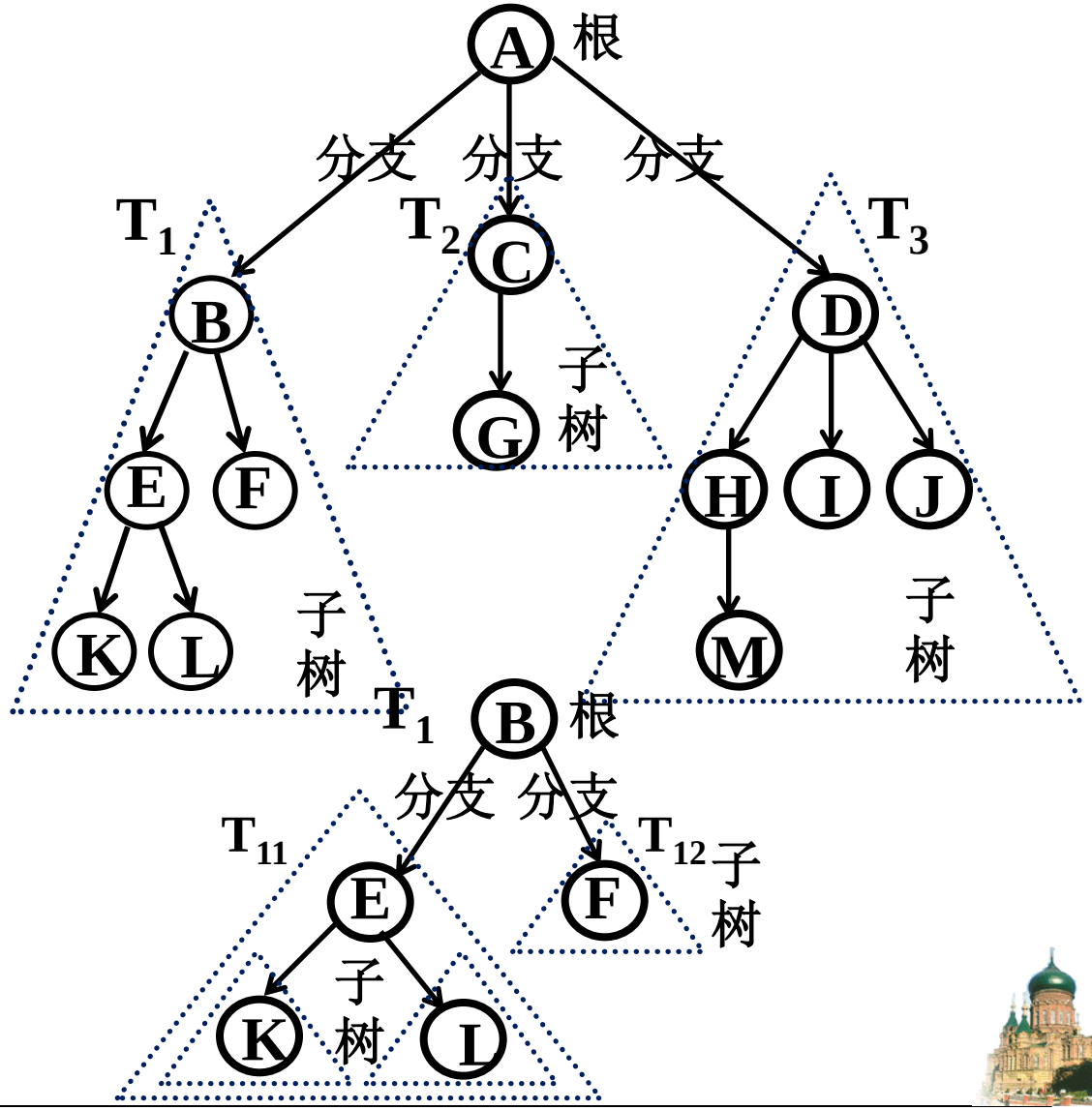
- 树的构造性递归定义：
  - 一个结点 X 组成的集合{ X }是一棵树，这个结点 X 称为这棵树的根（root）。
  - 假设 X 是一个结点，T_1，T_2，…，T_k是 k棵互不相交的树，可以构造一棵新树：令 X 为根，并有k 条边由 X 指向树T_1，T_2，…，T_k。这些边也叫做分支，T_1，T_2，…，T_k称作根为X的树之子树（SubTree）。
    - 说明：
      - 递归定义，但不会产生循环定义；
      - 构造性定义便于树型结构的建立；
      - 一株树的每个结点都是这株树的某株子树的根；



3.1 树与二叉树的基本术语（Cont.）

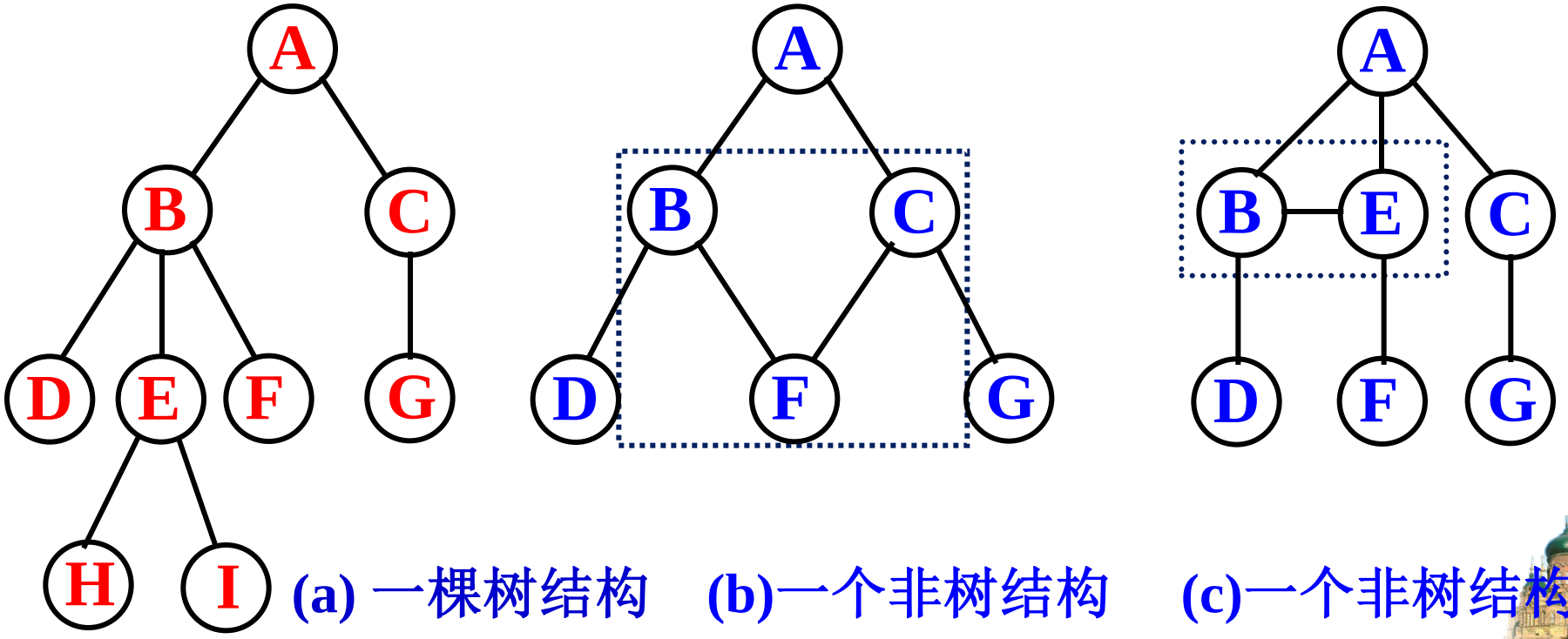
- 树的构造性递归定义：





3.1 树与二叉树的基本术语（Cont.）

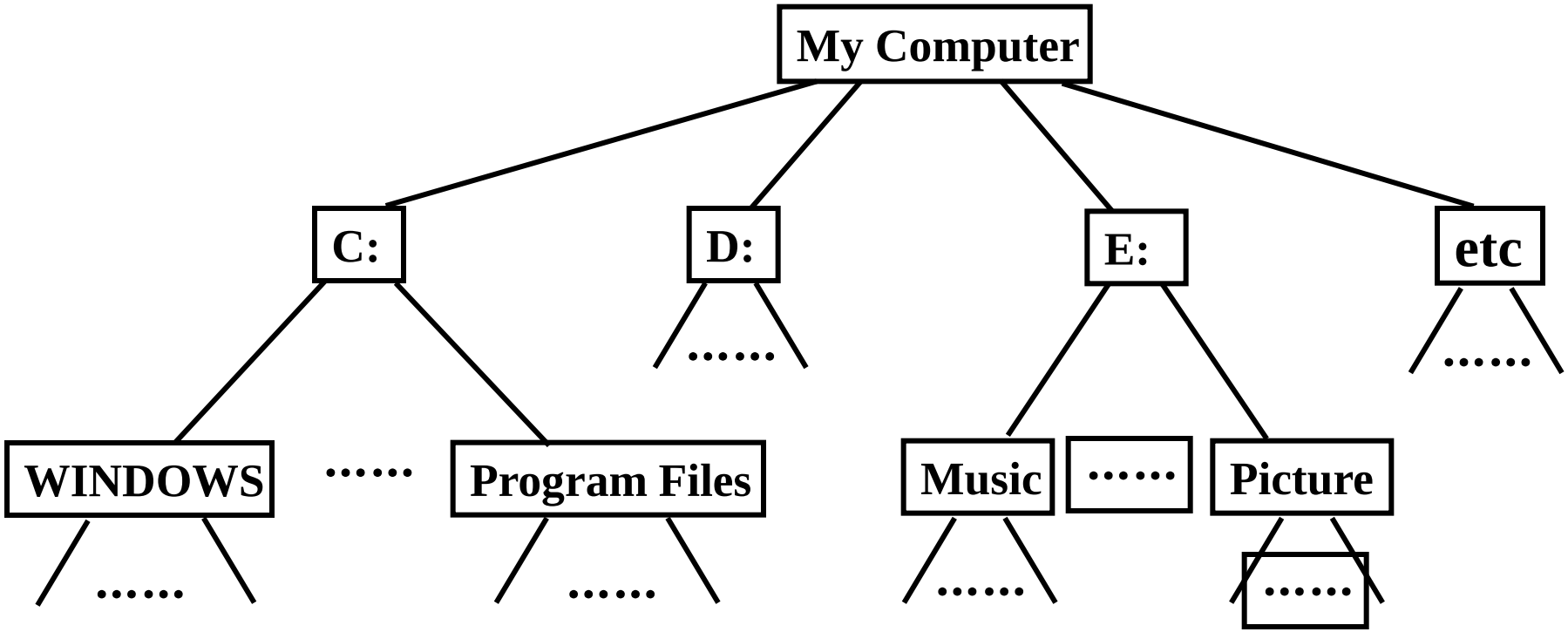
- 树的逻辑结构特点：
  - 除根结点之外，每棵子树的根结点有且仅有一个直接前驱，但可以有0个或多个直接后继。
  - 即一对多的关系，反映了结点之间的层次关系。



(a) 一棵树结构　(b)一个非树结构　(c)一个非树结构

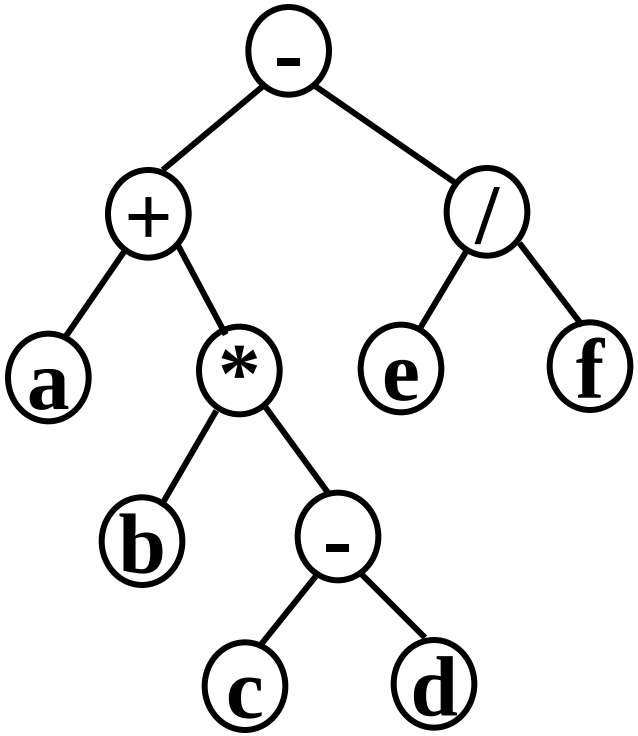
3.1 树与二叉树的基本术语（Cont.）

- 树型结构的应用示例----文件（目录）结构：



3.1 树与二叉树的基本术语（Cont.）

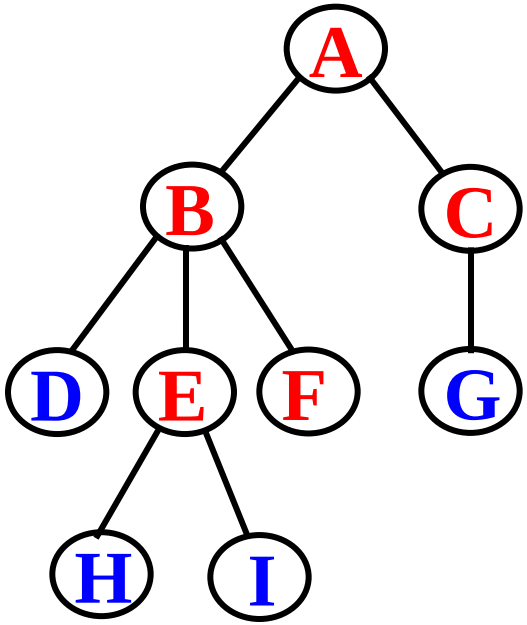
- 树型结构的应用示例----(无公共子式的)表达式的表示：



a+b*(c-d)-e/f

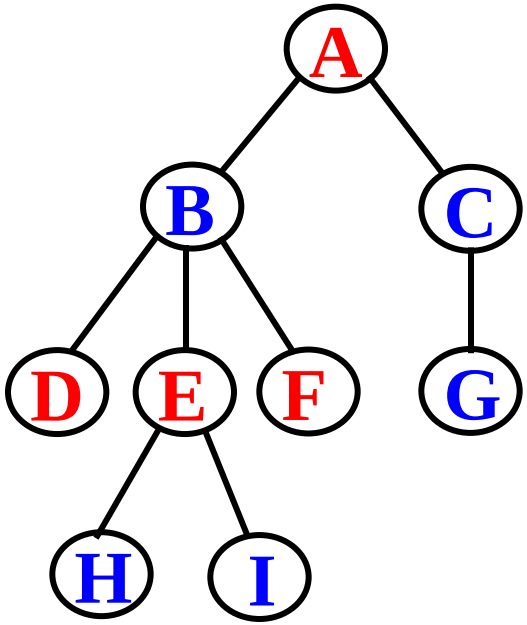
3.1 树与二叉树的基本术语（Cont.）

- 基本术语：
  - 结点的度：结点所具有的子树的个数。
  - 树的度：树中各结点度的最大值。
  - 叶子结点：度为0的结点，也称为终端结点。
  - 分支结点：度不为0的结点，也称为非终端结点。



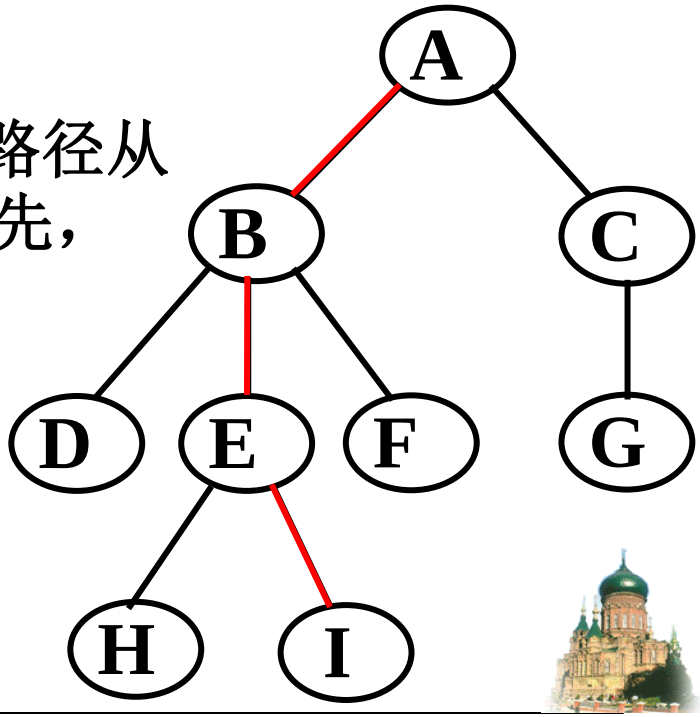
3.1 树与二叉树的基本术语（Cont.）

- 基本术语：
  - 结点孩子、双亲：树中某结点子树的根结点称为这个结点的孩子结点（子结点、儿子），这个结点称为它孩子结点的双亲结点（父结点）；
  - 兄弟：具有同一个双亲的孩子结点互称为兄弟。



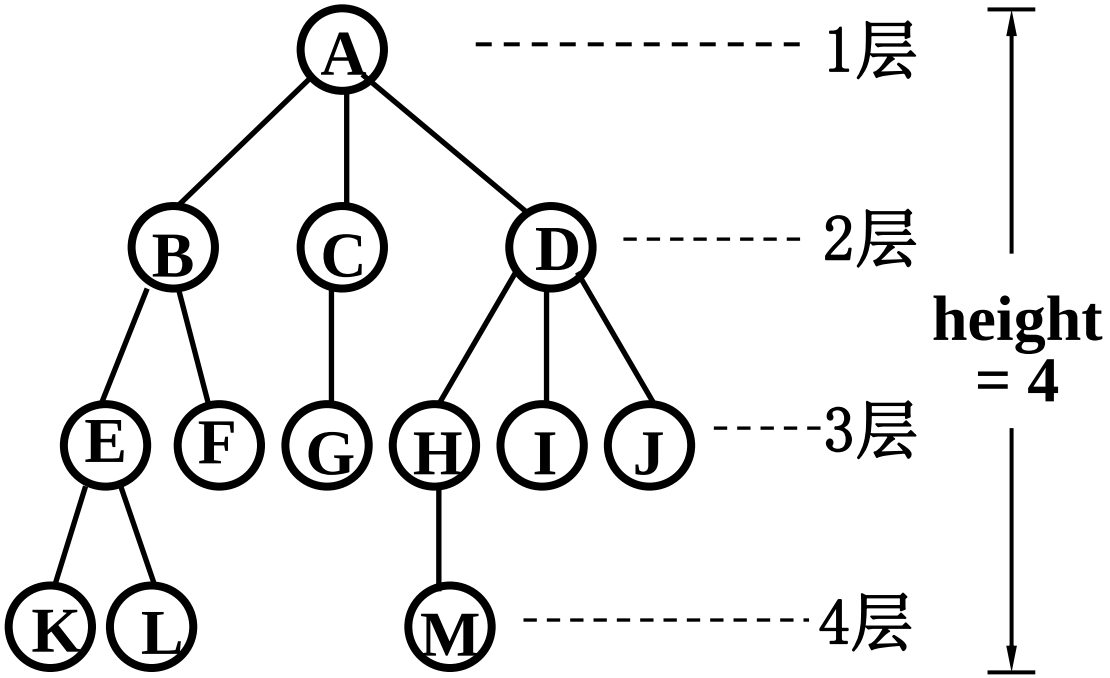
3.1 树与二叉树的基本术语（Cont.）

- 基本术语：
  - 路(径)和路(径) 长度：如果树的结点序列$n_1, n_2, \ldots, n_k$有如下关系：结点n_i是n_{i+1}的双亲（$1<=i<k$），则把$n_1, n_2, \ldots, n_k$称为一条由n_1至n_k的路径；路径上经过的边的个数称为路径长度。
  - 祖先、子孙：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。



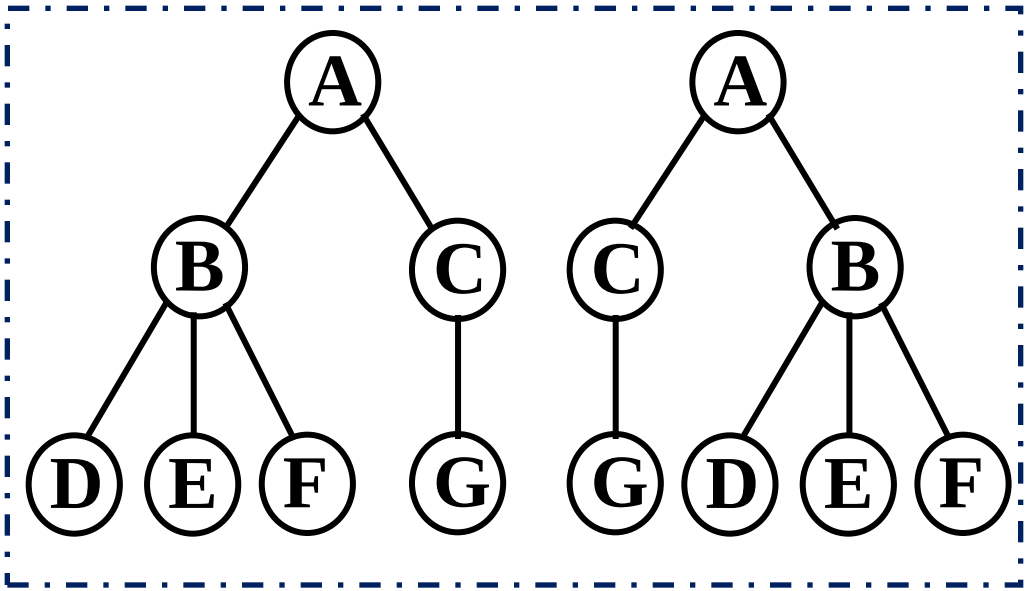
3.1 树与二叉树的基本术语（Cont.）

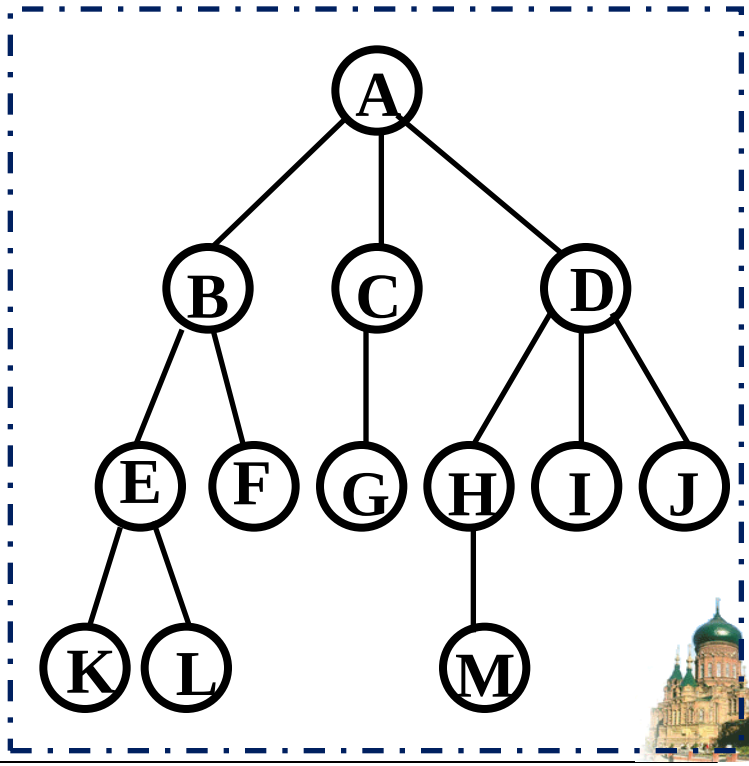
- 基本术语：
  - 结点的层数：根结点的层数为1；对其余任何结点，若某结点在第k层，则其孩子结点在第k+1层。
  - 树的深度：树中所有结点的最大层数，也称高度。



3.1 树与二叉树的基本术语（Cont.）

- 基本术语：
  - 有序树、无序树：如果一棵树中结点的各子树从左到右是有次序的，称这棵树为有序树；反之，称为无序树。
  - 森林：m ($m \geq 0$)棵互不相交的树的集合。





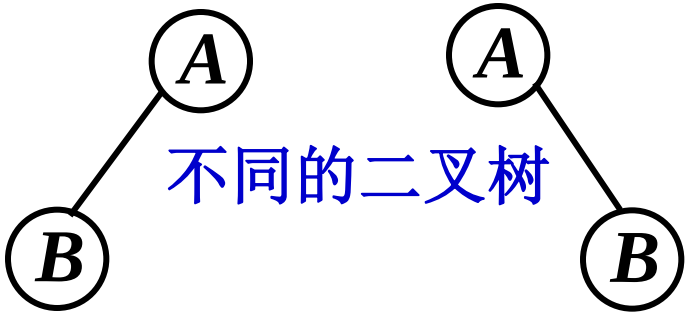
3.1 树与二叉树的基本术语（Cont.）

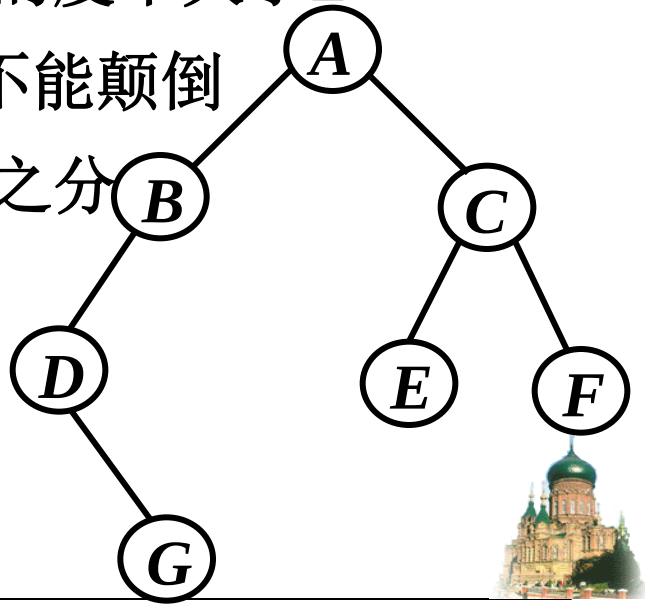
- 树型结构和线性结构的比较

线性结构	树型结构
第一个数据元素	根结点（只有一个）
无前驱	无双亲
最后一个数据元素	叶子结点(可以有多个)
无后继	无孩子
其它数据元素	其它结点
一个前驱,一个后继	一个双亲,多个孩子
一对一	一对多

3.2 二叉树

- 二叉树(Binary Tree)的定义：
  - 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：
  - 每个结点最多只有两棵子树，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



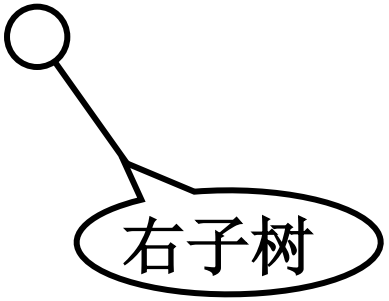


3.2 二叉树（Cont.）

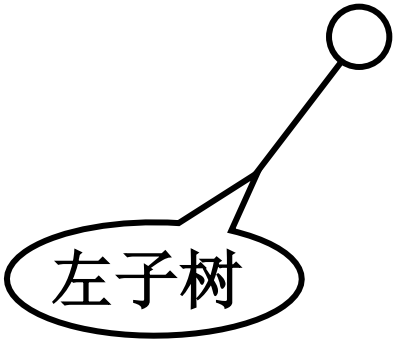
- 二叉树的基本形态：

空二叉树

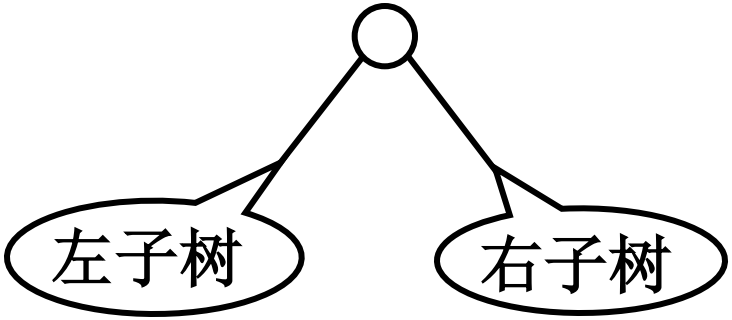
只有一个根结点



根结点只有右子树



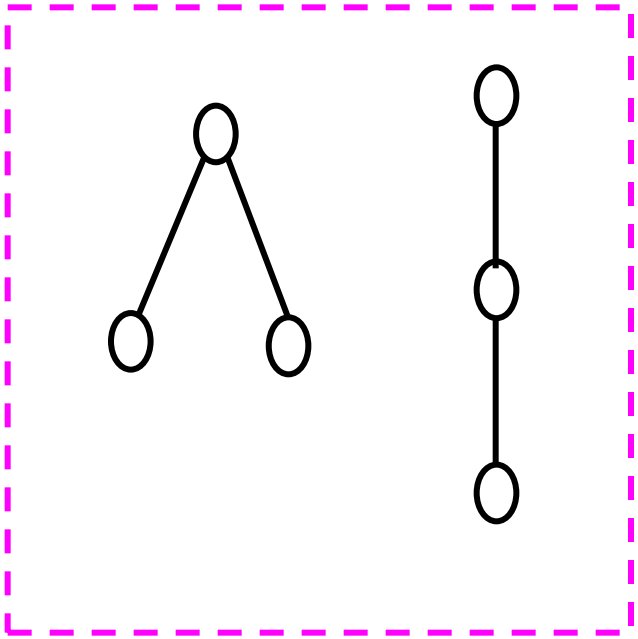
根结点只有左子树



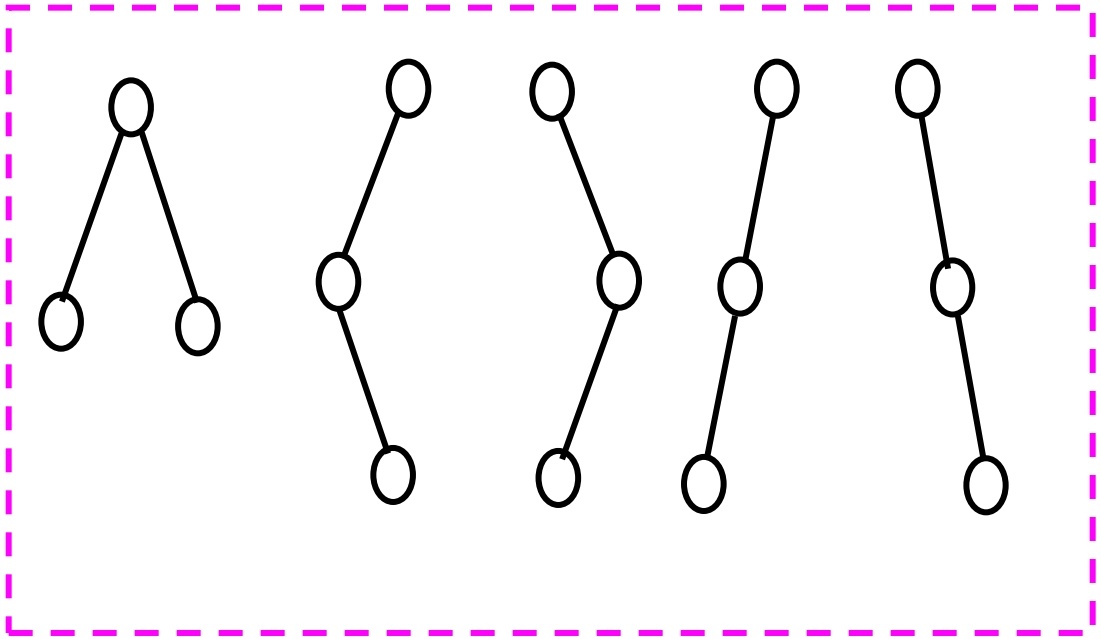
根结点同时有左右子树

3.2 二叉树（Cont.）

- 具有3个结点的树和二叉树的不同构的形态：



树的不同构形态

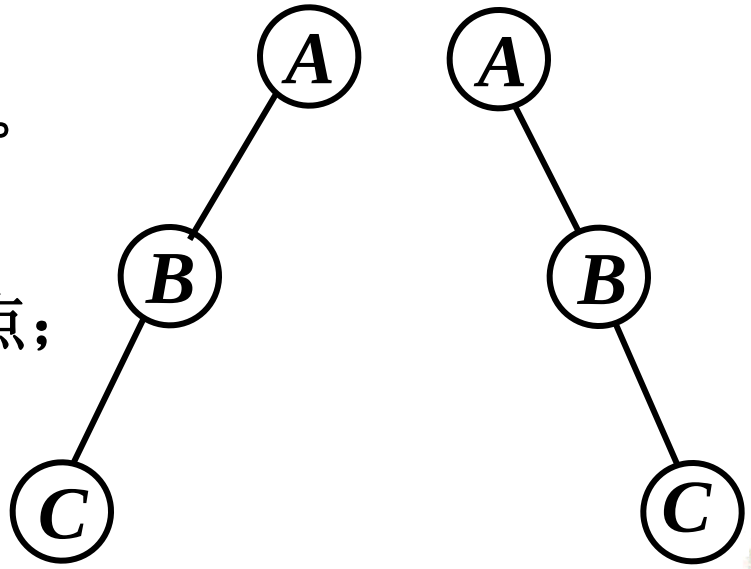


二叉树的不同构形态

3.2 二叉树（Cont.）

特殊的二叉树----斜树

- 左斜树
  - 所有结点都只有左子树的二叉树称为左斜树；
- 右斜树
  - 所有结点都只有右子树的二叉树称为右斜树；
- 斜树：
  - 左斜树和右斜树统称为斜树。



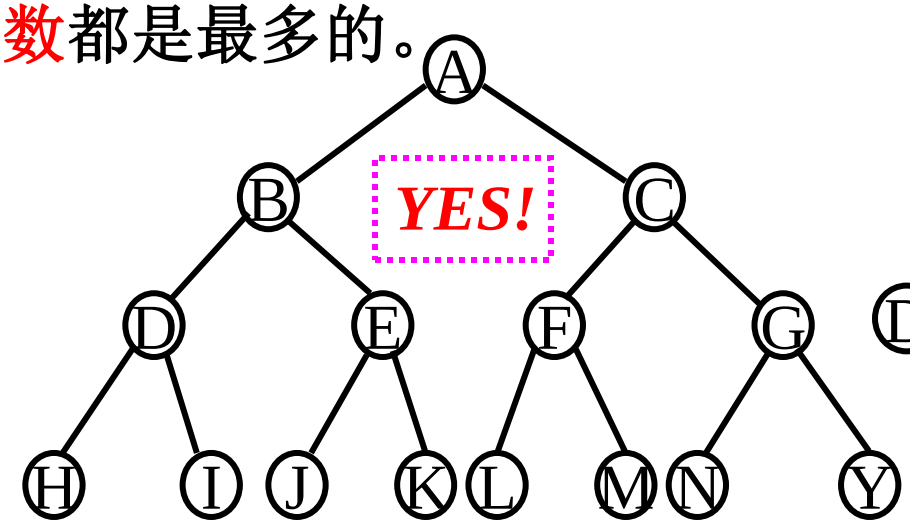
斜树的结构特点：

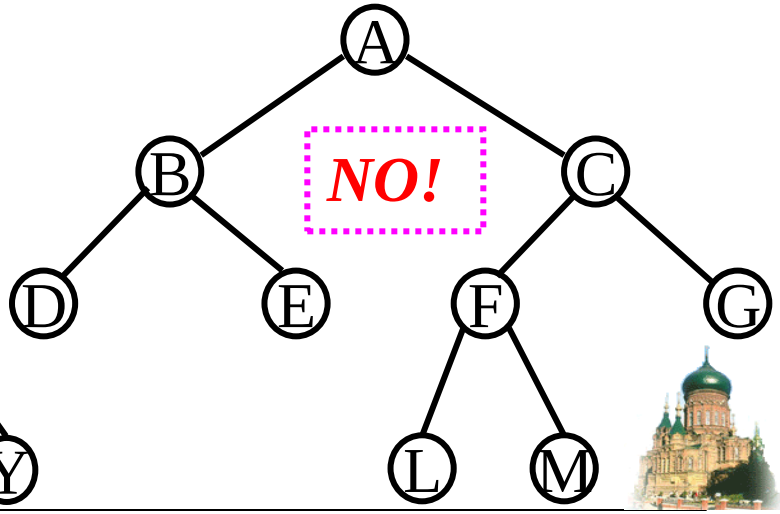
- 在斜树中，每一层只有一个结点；
- 斜树的结点个数与其高度相同。

3.2 二叉树（Cont.）

特殊的二叉树----满二叉树

- 定义：高度为K且有2^K-1个结点的二叉树称为满二叉树。
- 结构特点：
  - 分支结点都有两棵子树
  - 叶子结点都在最后一层
- 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。

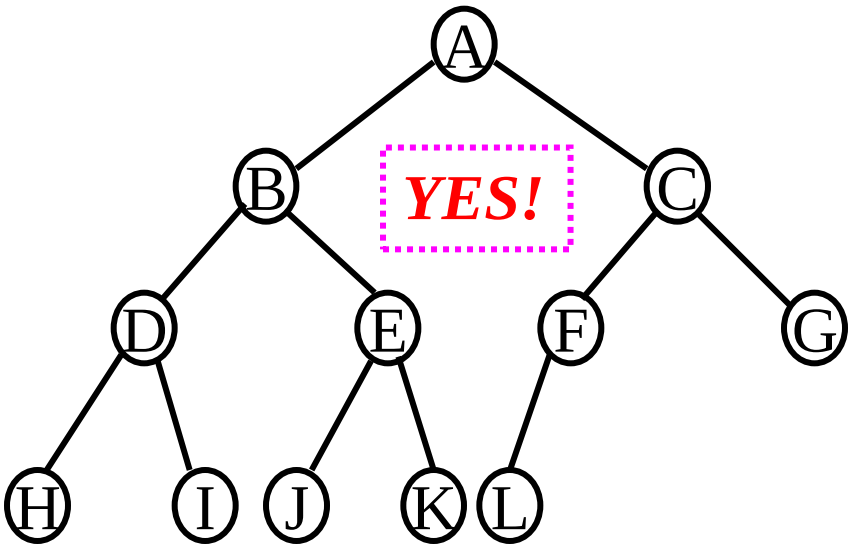




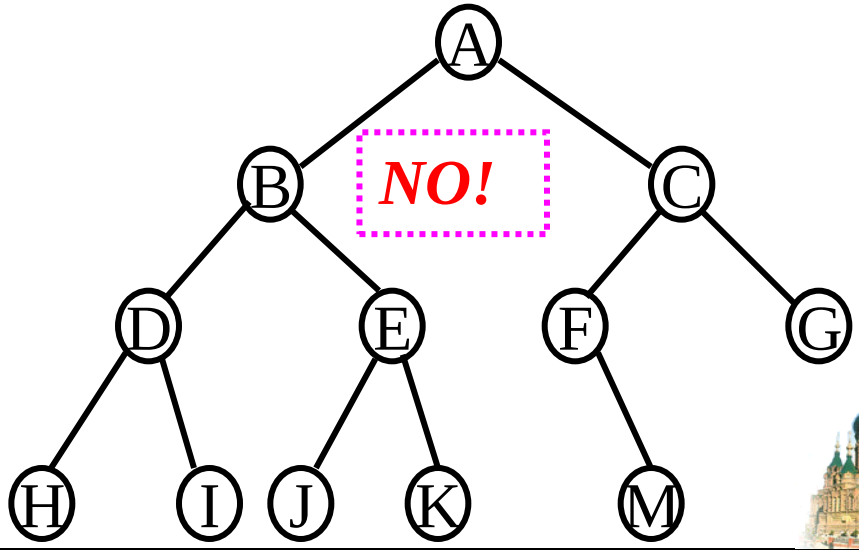
3.2 二叉树（Cont.）

特殊的二叉树----完全二叉树

- 定义：称满足下列性质的二叉树(假设高度为k)为完全二叉树:
  - 1.所有的叶都出现在k或k-1层；
  - 2.k-1层的所有叶都在非终结结点的右边；
  - 3. 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。



YES!

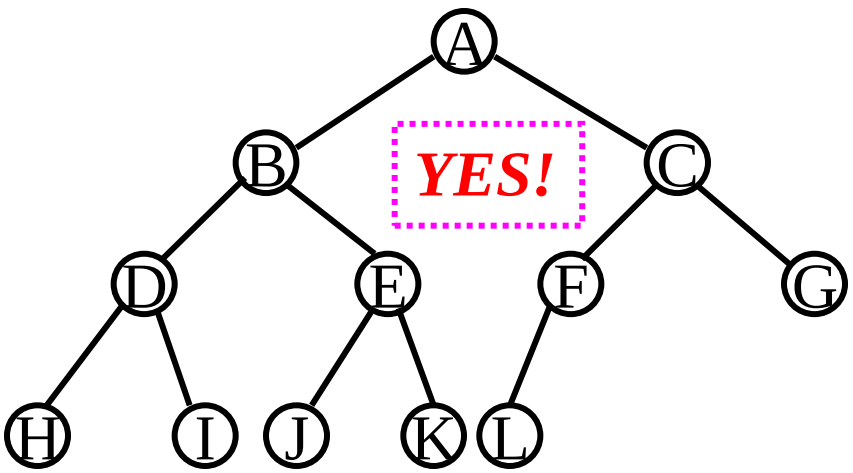


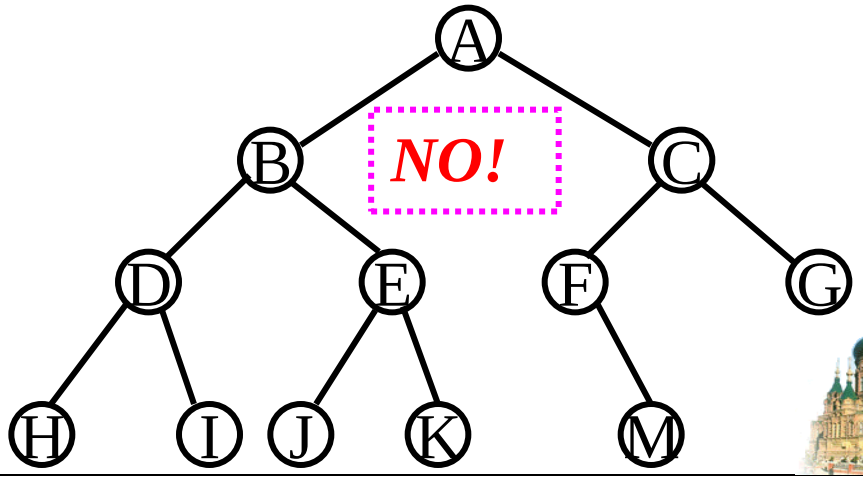
NO!

3.2 二叉树（Cont.）

特殊的二叉树----完全二叉树

- 结构特点：
  - 1. 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；
  - 2. 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。
  - 3. 深度为k的完全二叉树的前k-1层一定是满二叉树。





3.2 二叉树（Cont.）

二叉树的性质

- 性质1:
  - 二叉树的第 i 层最多有 2^{i-1} 个结点。($i \geq 1$)
  - 证明(数学归纳法)：

    当i=1时，第1层只有一个根结点，而

    $2^{i-1}=2^0=1$，

    结论成立。

    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二倍，即$2\times 2^{k-1}=2^k$。结论成立。 □

3.2 二叉树（Cont.）

二叉树的性质

- 性质2:
  - 高度为 k ($k \geq 1$)的二叉树最多有 $2^k - 1$个结点，最少有k个结点。
  - 证明：由性质1可知，高度为k的二叉树中结点个数最多

$$= \sum_{i=1}^{k} (\text{第}i\text{层上结点的最大个数}) = 2^0 + 2^1 + 2^2 + \ldots + 2^{k-1} = 2^k - 1;$$

另外，每一层至少要有一个结点，因此，高度为k的二叉树，至少有k个结点。 □

- 高度为k且具有2^k-1个结点的二叉树一定是满二叉树，
- 高度为k且具有k个结点的二叉树不一定是斜树。

3.2 二叉树（Cont.）

二叉树的性质

- 性质3:
  - 在非空二叉树中，如果叶子结点数为n_0，度为2的结点数为n_2，则有: $n_0=n_2+1$，而与度数为1的结点数n_1无关。
  - 证明：设n为二叉树的结点总数，则有：

$$n=n_0+n_1+n_2$$

在n个结点的二叉树中，共有$n-1$条分支(边)；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。所以有：

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

因此可以得到：$n_0=n_2+1$ 。与度数1的结点数n_1无关。□

- 在有n个结点的满二叉树中，有多少个叶子结点？

3.2 二叉树（Cont.）

完全（满）二叉树的性质

- 性质4: 具有 n ($n \geq 0$) 个结点的完全二叉树的高度为

$$\lceil \log_2(n+1) \rceil$$

证明：设完全二叉树的高度为 k，则根据性质2有：

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

前k-1层最多结点数　　前k层最多结点数

变形　$2^{k-1} < n + 1 \leq 2^k$

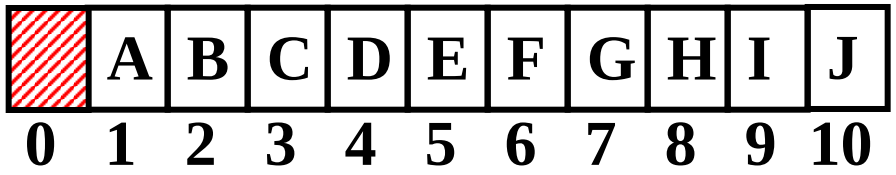
取对数　$k-1 < \log_2(n+1) \leq k$

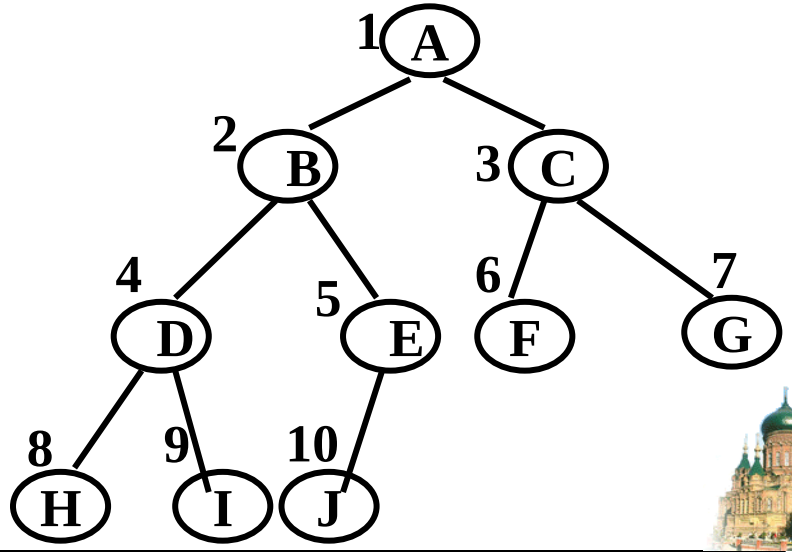
因此有　$\lceil \log_2(n+1) \rceil = k$ □

3.2 二叉树（Cont.）

完全（满）二叉树的性质

- 完全二叉树的顺序存储结构：
  - 如果把一棵完全二叉树的具有*n*个结点自顶向下，同一层自左向右连续编号:1, 2,…,*n*，且使该编号对应于数组的下标，即编号为*i*（1 ≤ *i* ≤ *n*）的结点存储在下标为*i*的数组单元中，则这种存储表示方法称为完全（满）二叉树的顺序存储结构。

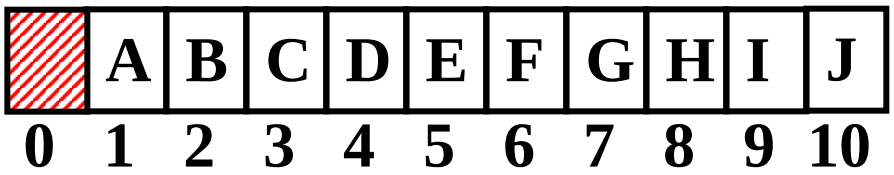


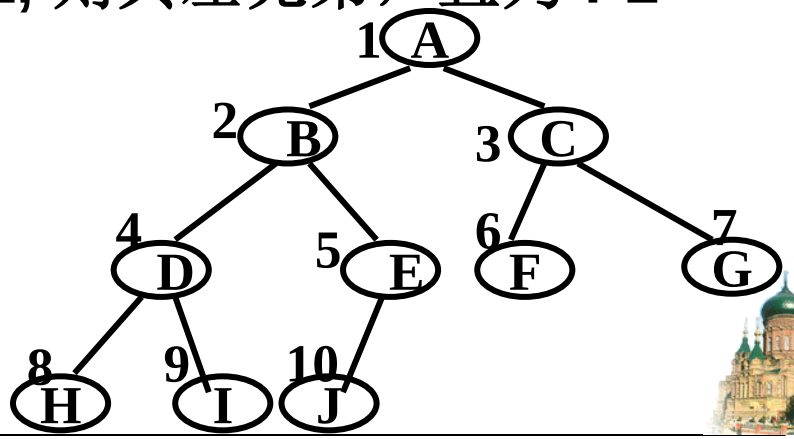


3.2 二叉树（Cont.）

- 完全二叉树的顺序存储结构的性质：
  - 若i = 1, 则 i 是根结点，无父结点；

    若i > 1, 则 i 的父结点为⌊i/2⌋
  - 若 2*i ≤ n, 则 i 有左儿子且为 2*i；否则，i 无左儿子。
  - 若2*i+1 ≤ n, 则 i 有右儿子且为2*i+1；否则，i 无右儿子
  - 若 i 为偶数, 且 i < n , 则有右兄弟，且为 i + 1。
  - 若 i 为奇数, 且 i ≤ n && i != 1, 则其左兄弟，且为 i-1





3.2 二叉树（Cont.）

二叉树的遍历操作

- 遍历的定义：根据某种策略，按照一定的次序访问二叉树中的每一个结点，使每个结点被访问一次且只被访问一次。这个过程称为二叉树的遍历。

二叉树 ┌ 根结点D
　　 ├ 左儿子L
　　 └ 右儿子R

三个结点的可能顺序：

DLR　　LDR　　LRD

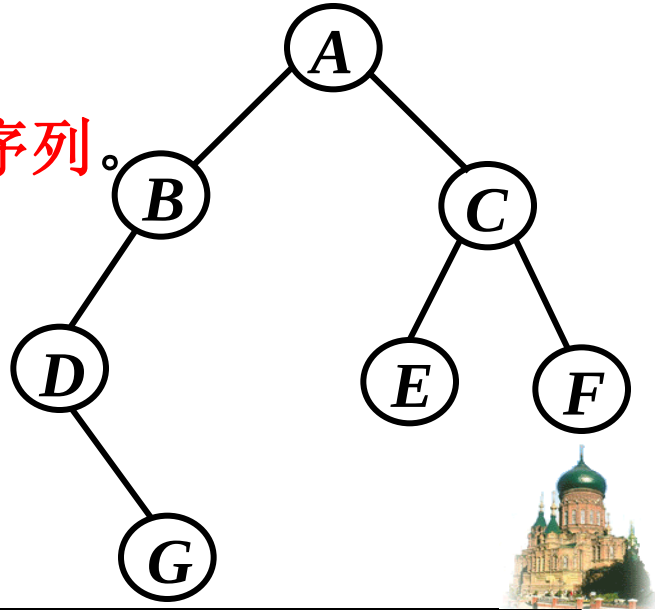
~~DRL~~　　~~RLD~~　　~~RLD~~

- 遍历的结果是二叉树结点的线性序列。非线性结构线性化。
  - 策略：左孩子结点一定要在右孩子结点之前访问
  - 次序：先序（根）遍历、中序（根）遍历、后序（根）遍历和层序（次）遍历
  - 访问：抽象操作，可以是对结点进行的各种处理，这里简化为输出结点的数据。

3.2 二叉树（Cont.）

二叉树遍历的定义

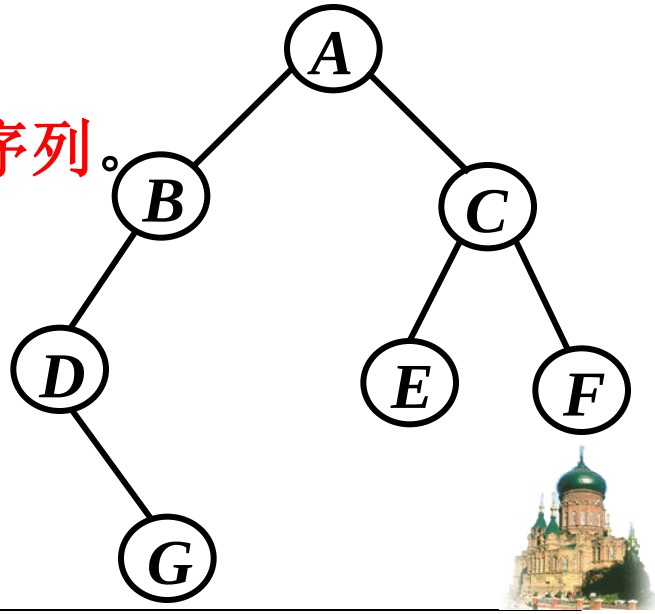
- 先序（根）遍历二叉树
  - 若二叉树为空，则返回；否则，
    - ①访问根结点；
    - ②先序遍历根结点的左子树；
    - ③先序遍历根结点的右子树；
- 所得到的线性序列分别称为先序（根）序列。
- 先序遍历序列为：*A B D G C E F*



3.2 二叉树（Cont.）

二叉树遍历的定义

- 中序（根）遍历二叉树
  - 若二叉树为空，则返回；否则，
    - ①中序遍历根结点的左子树;
    - ②访问根结点；
    - ③中序遍历根结点的右子树;
- 所得到的线性序列分别称为中序（根）序列。
- 中序遍历序列为： ***D G B A E C F***



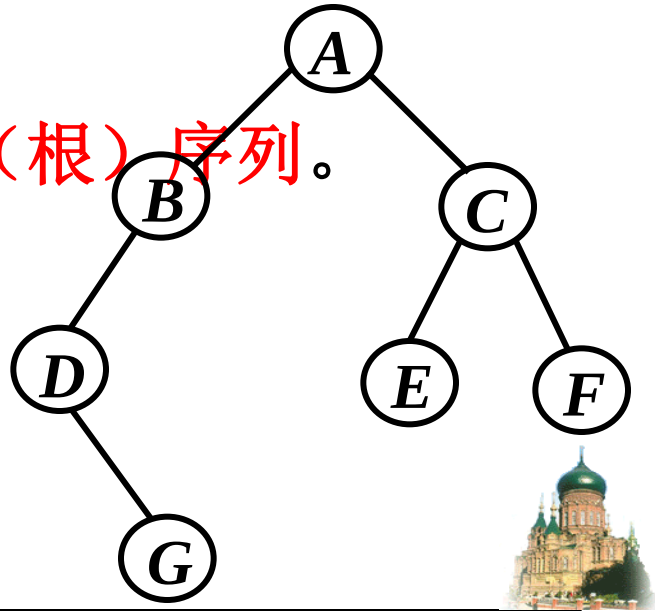
3.2 二叉树（Cont.）

二叉树遍历的定义

- 后序（根）遍历二叉树
  - 若二叉树为空，则返回；否则，
    - ①后序遍历根结点的左子树；
    - ②后序遍历根结点的右子树；
    - ③访问根结点；

    所得到的线性序列分别称为后序（根）序列。

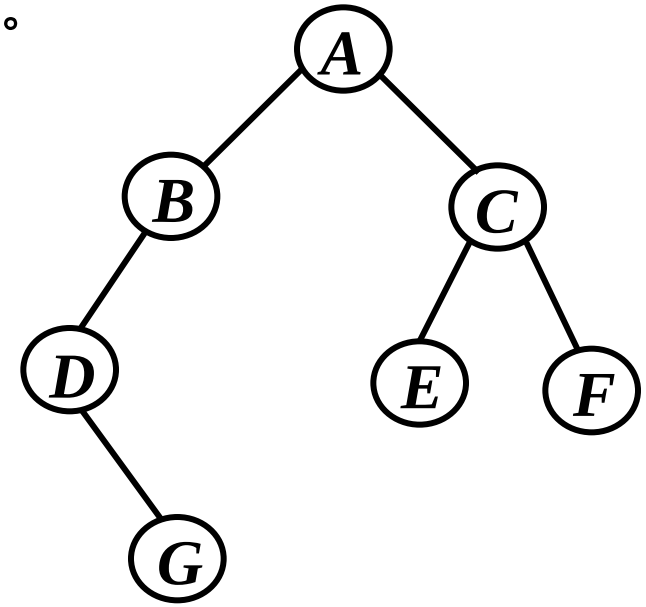
- 后序遍历序列为： ***G D B E F C A***



3.2 二叉树（Cont.）

二叉树遍历的定义

- 层序（次）遍历二叉树
  - 从二叉树的第一层（即根结点）开始，从上至下逐层遍历，在同一层中，则按从左到右的顺序对结点进行访问。
  - 所得到的线性序列分别称为层序序列。



- 层序遍历序列为： *A B C D E F G*

3.2 二叉树（Cont.）

二叉树的基本操作

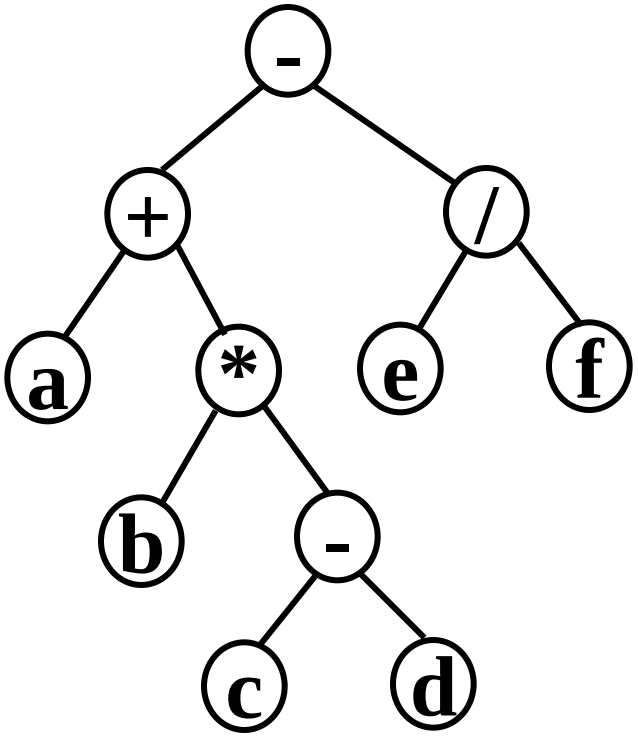
- ① **Empty (BT)：** 建立一株空的二元树。
- ② **IsEmpty (BT)：** 判断二元树是否为空,若是空则返回**TRUE**;否则返回**FALSE**。
- ③ **CreateBT (V, LT , RT)：** 建立一株新的二元树。这棵新二元树根结点的数据域为**V**,其作右子树分别为**LT**，**RT**。
- ④ **Lchild (BT)：** 返回二元树**BT**的左儿子。若无左儿子，则返回空。
- ⑤ **Rchild (BT)：** 返回二元树**BT**的右儿子。若无右儿子，则返回空。
- ⑥ **Data (BT)：** 返回二元树**BT**的根结点的数据域的值。

3.2 二叉树（Cont.）

利用二叉树的基本操作，写出前三种遍历算法的递归形式

- 先序遍历算法

```
void  PreOrder (BTREE BT )
{
    if ( ! IsEmpty ( BT ) )
    {   visit ( Data ( BT ) ) ;
        PreOrder ( Lchild ( BT ) ) ;
        PreOrder ( Rchild ( BT ) ) ;
     }
}
```



a+b*(c-d)-e/f

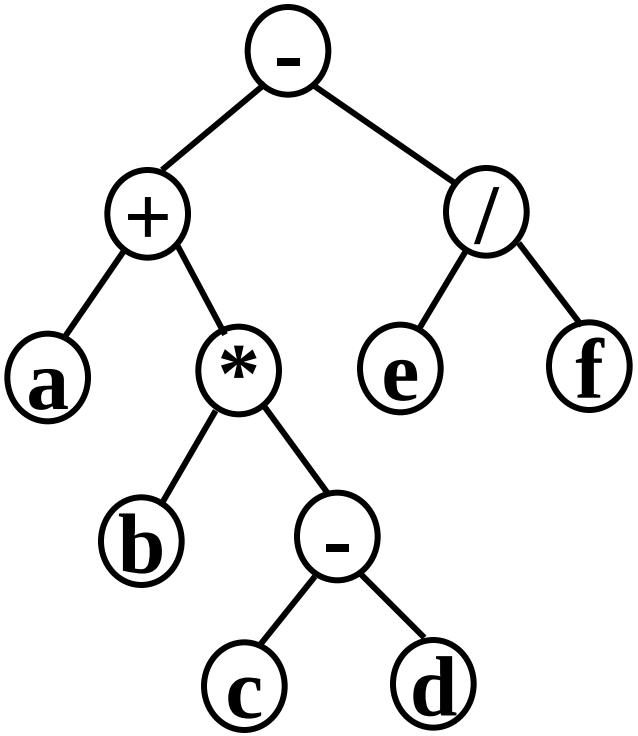
- 先序序列：- + a * b – c d / e f

3.2 二叉树（Cont.）

利用二叉树的基本操作，写出前三种遍历算法的递归形式

- 中序遍历算法

```
void  InOrder (BTREE BT )
{
    if ( ! IsEmpty ( BT ) )
    {   InOrder ( Lchild ( BT ) ) ;
        visit ( Data ( BT ) ) ;
        InOrder ( Rchild ( BT ) ) ;
     }
}
```



a+b*(c-d)-e/f

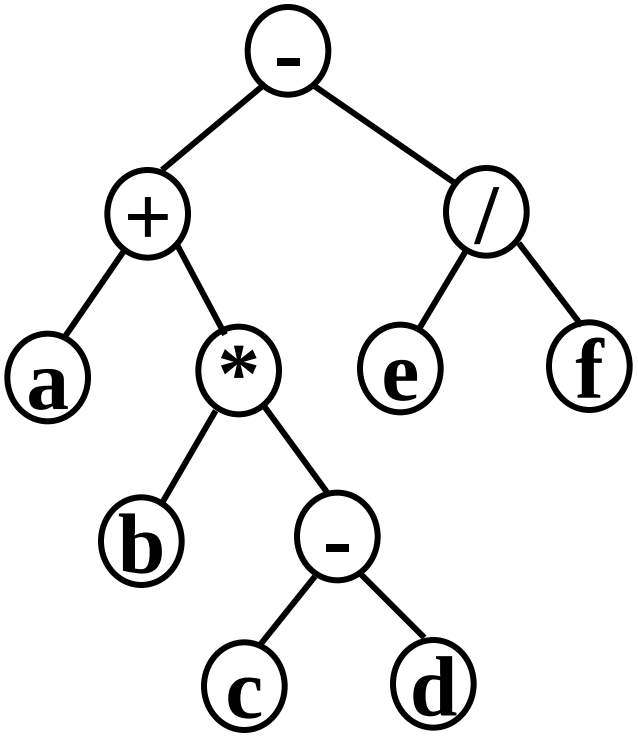
- 中序序列： a + b * c – d – e / f

3.2 二叉树（Cont.）

利用二叉树的基本操作，写出前三种遍历算法的递归形式

- 后序遍历算法

```
void  PostOrder (BTREE BT )
{
    if ( ! IsEmpty ( BT ) )
   {     PostreOrder ( Lchild ( BT ) ) ;
         PostOrder ( Rchild ( BT ) ) ;
         visit ( Data ( BT ) ) ;
    }
}
```



a+b*(c-d)-e/f

- 后序序列：a b c d - * + e f / -