第六章 树和二叉树

6.1 树的定义和基本术语

6.2 二叉树

6.3 遍历二叉树和线索二叉树

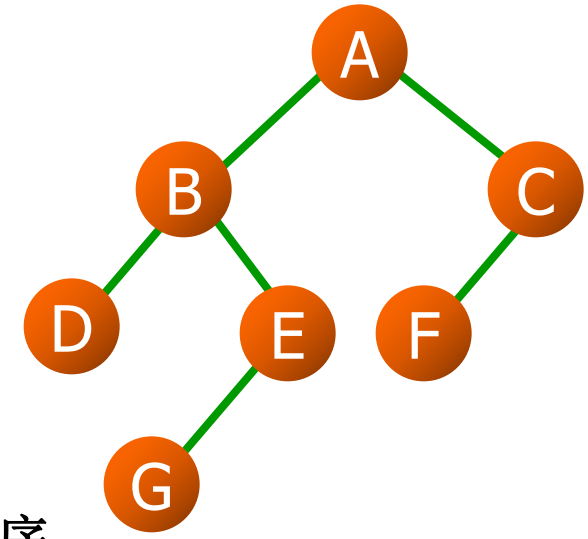
6.4 树和森林

6.6 赫夫曼树及其应用

上节复习

- 二叉树的四种遍历：
  - 访问根结点用D表示，遍历左、右子树用L、R表示
  - 先序、中序、后序、层次
- 二叉树的三种构建：
  - 根据先序遍历，空树用0表示，递归构建
  - 根据数组存储结果，通过补0把普通二叉树转完全二叉树，用性质5构建
  - 根据先序+中序，或中序+后序来推导二叉树逻辑结构

练习



1. 已知二叉树如图所示，求：
   - 左孩子、右孩子、叶子的数量
   - 先序、中序、后序、层次遍历

2. 根据以下遍历结果，求出其他遍历
   - 先序**ABCDEF**，中序**CBDAFE**，求后序
   - 后序**DCBFHGEA**，中序**DCBAFEGH**，求先序
   - 请画出上面两个二叉树的顺序存储结构和链式存储结构

3. 已知二叉树的先序序列的字符串表示为 **AB00CD0E000**，请画出该二叉树，并求中序遍历结果。

4. 已知二叉树的顺序存储结果是 **ABCDEF000G00H**，请画出二叉树，并求后序遍历结果。

6.3 线索二叉树

一. 建立二叉树线索的方法

- 遍历二叉树是按一定的规则将树中的结点排列成一个线性序列，即是对非线性结构的线性化操作
  - 需要找到遍历过程中动态得到的每个结点的直接前驱和直接后继
  - 如何保存这些前驱、后继信息？

6.3 线索二叉树

一. 建立二叉树线索的方法

- **方法一：**增加新指针。

在每个结点中增加前驱(fwd)和后继(bkwd)指针，在做二叉树遍历（前、中、后序），将每个结点的前驱和后继信息添入fwd和bkwd域中。

fwd	lChild	data	rChild	bkwd

! 需要增加额外空间。

6.3 线索二叉树

一. 建立二叉树线索的方法

- 方法二：**利用二叉链表中n+1个空指针**（存放指向结点在某种遍历次序下的前驱和后继结点的地址），**另外，在结点中增加两个标记位**（LTag，RTag）
  - LTag=0，lChild域指示结点的左孩子

    LTag=1，lChild域指示结点的前驱结点
  - RTag=0，lChild域指示结点的右孩子

    RTag=1，lChild域指示结点的后继结点

LTag	lChild	data	rChild	RTag

6.3 线索二叉树

二. 线索二叉树的实现

- 使用方法二，指向结点前驱和后继的指针叫做**线索**，加上线索的二叉树称之为**线索二叉树**。对二叉树以某种次序遍历使其变为线索二叉树的过程称作是**线索化**。
- 线索二叉树的存储表示

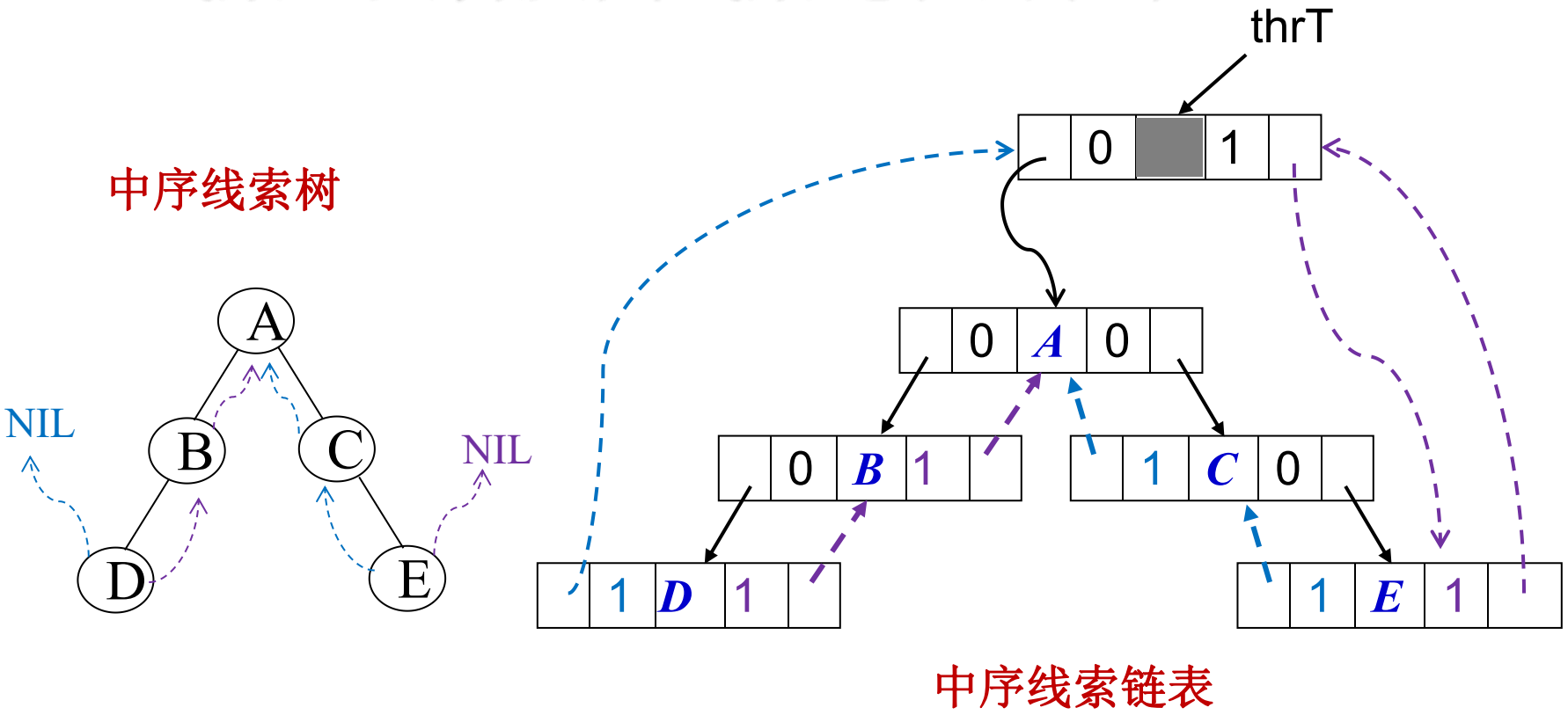
```
typedef enum PointTag { Link, Thread };
            //Link==0：指针，指向孩子结点
            //Thread==1：线索，指向前驱或后继结点
typedef struct BiThrNode
{   ElemType  data;
        struct BiTreeNode *Lchild , *Rchild ;
        PointTag  Ltag , Rtag ;
}BiThrNode ,    *BiThrTree;
```

6.3 线索二叉树

二. 线索二叉树的实现

- 以这种带有线索结构结点构成的二叉链表作为二叉树的存储结构，叫做**线索链表**，便于对二叉树做查找结点的前驱和后继方面的应用。
- **线索化的实质**就是将二叉链表中的空指针改为指向前驱或后继的线索，等于把一棵二叉树转变成了一个双向链表，这样便于进行插入、删除结点和查找某个结点等操作。

线索二叉树及其线索链表的表示



标志域：

***ltag* = 0，*lchild*为左孩子指针**

***ltag* = 1，*lchild*为前驱线索**

***rtag* = 0，*rchild*为右孩子指针**

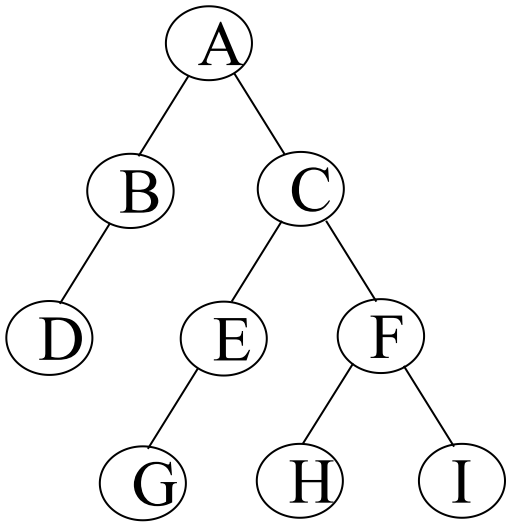
***rtag* = 1，*rchild*为后继线索**

中序结点序列：DBACE

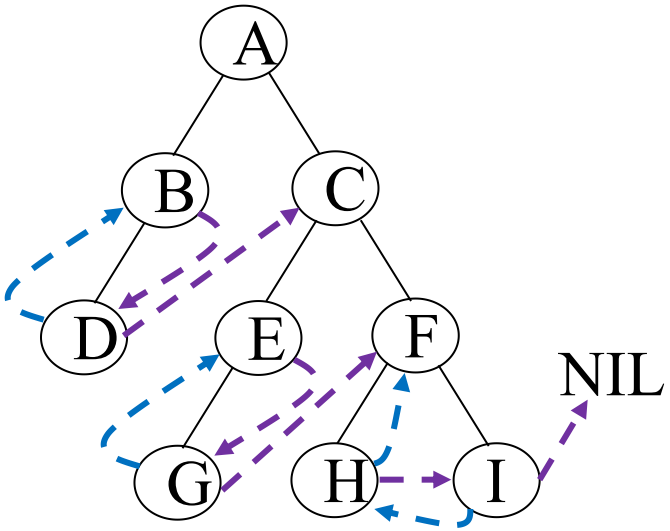
6.3 线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树

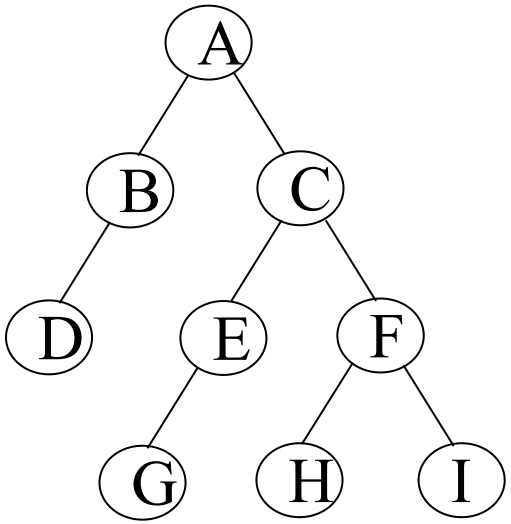


(b) 先序线索树的逻辑形式
结点序列：**ABDCEGFHI**

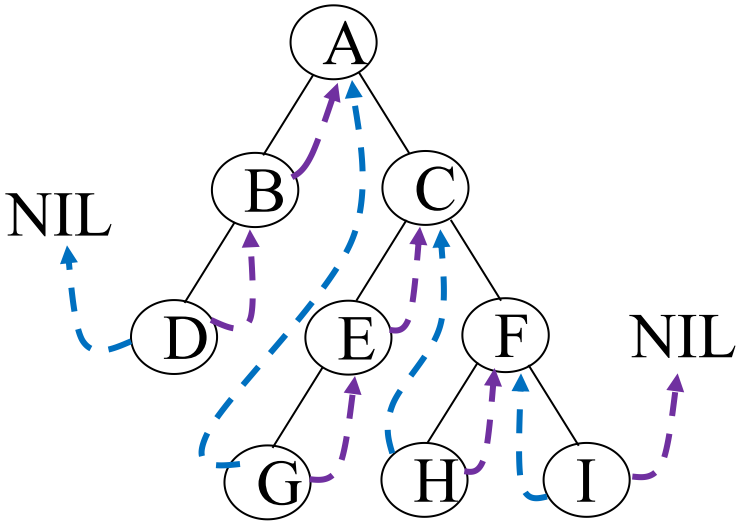
6.3 线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树

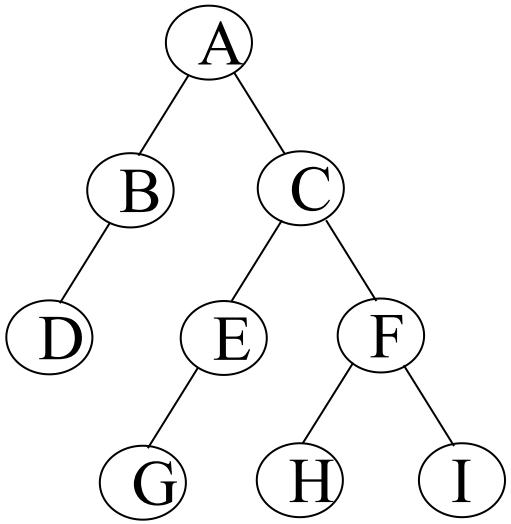


(c) 中序线索树的逻辑形式
结点序列：**DBAGECHFI**

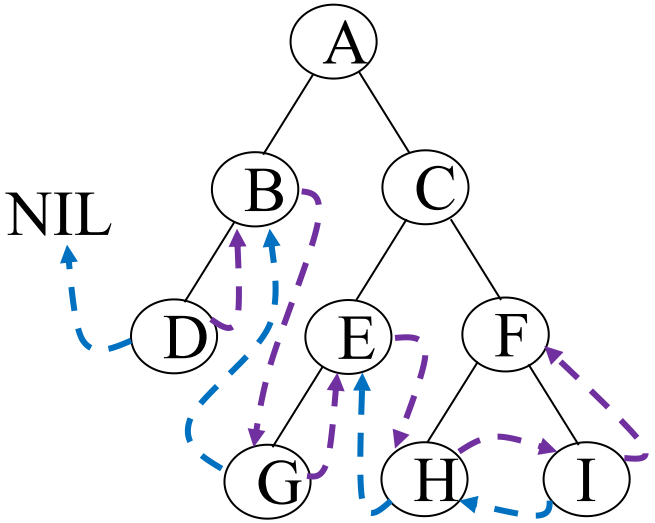
6.3 线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树



(d) 后序线索树的逻辑形式
结点序列：**DBGEHIFCA**

说明： 画线索二叉树时，实线表示指针，指向其左、右孩子；虚线表示线索，指向其直接前驱或直接后继。

6.3 线索二叉树

三. 线索二叉树的遍历

- 由于前驱和后继的信息只有在遍历该二叉树时才能得到，所以线索化的过程就是在遍历的过程中修改空指针使其指向直接前驱或直接后继的的过程。
  - 从遍历的第一个结点来看：
    先序序列中第一个结点必为根结点
    中、后序序列中第一个结点的左孩子定为空
  - 从遍历的最后一个结点来看：
    先、中序序列中最后一个结点的右孩子必为空
    后序序列中最后一个结点一定为根结点

步骤：

1)找遍历的第一个结点

2)不断地找遍历到的结点的后继结点，直到树中各结点都遍历到为止，结束。

作用：

对于遍历操作，线索树优于非线索树；遍历线索树不用设栈

6.3 线索二叉树

懂原理，最多会画即可，对于建立过程不需要掌握

三. 线索二叉树的遍历

- 中序遍历线索化
  - 线索化后继——处理前驱结点

a. 如果无前驱结点，则不必加线索

b. 如果前驱结点的右指针域非空，也不必加线索

c. 如果前驱结点的右指针域为空，则把当前结点的指针值赋给前驱结点的右指针域。

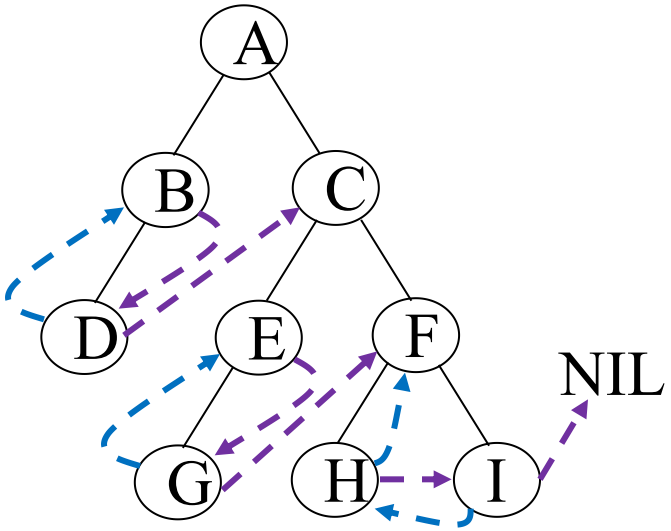
说明： 在线索树上进行遍历，只要先找到序列中的第一个结点，然后就可以依次找结点的直接后继结点直到后继为空为止。

6.3 线索二叉树

三. 线索二叉树的遍历

- 线索二叉树的先序遍历-非递归

```
void preorder_Thread(BiThrNode *T)
{  BiThrNode  *p=T ;
   while (p!=NULL)
    {  visit(p->data) ;
       if (p->Ltag==0)  p=p->Lchild ;
       else  p=p->Rchild
    }
}
```



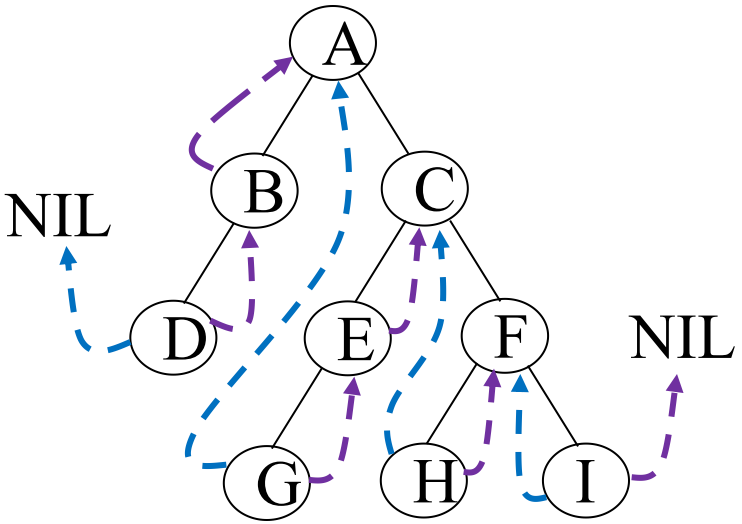
结点先序序列：
ABDCEGFHI

6.3 线索二叉树

三. 线索二叉树的遍历

- 线索二叉树的中序遍历-非递归

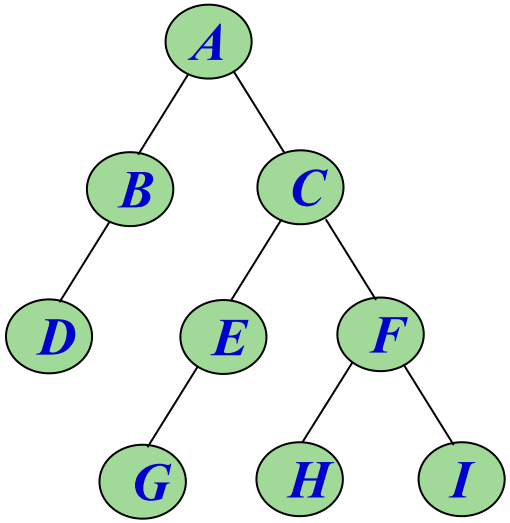
```
// 结点T是一个头结点
// T的左孩子指向根结点;
// T的右孩子指向中序遍历的尾结点
Status InOrder (BiThrTree T) {
  BiThrTree p;
  p = T->lchild;                    // p指向根结点
  while (p != T) {   // 空树或遍历结束时，p==T
        while (p->LTag==Link) p = p->lchild;
        cout<<p->data;
        while (p->RTag==Thread && p->rchild!=T) {
          p = p->rchild;
          cout<<p->data;
        }
    p = p->rchild;  // p进至其右子树根
  }
  return OK;
}
```



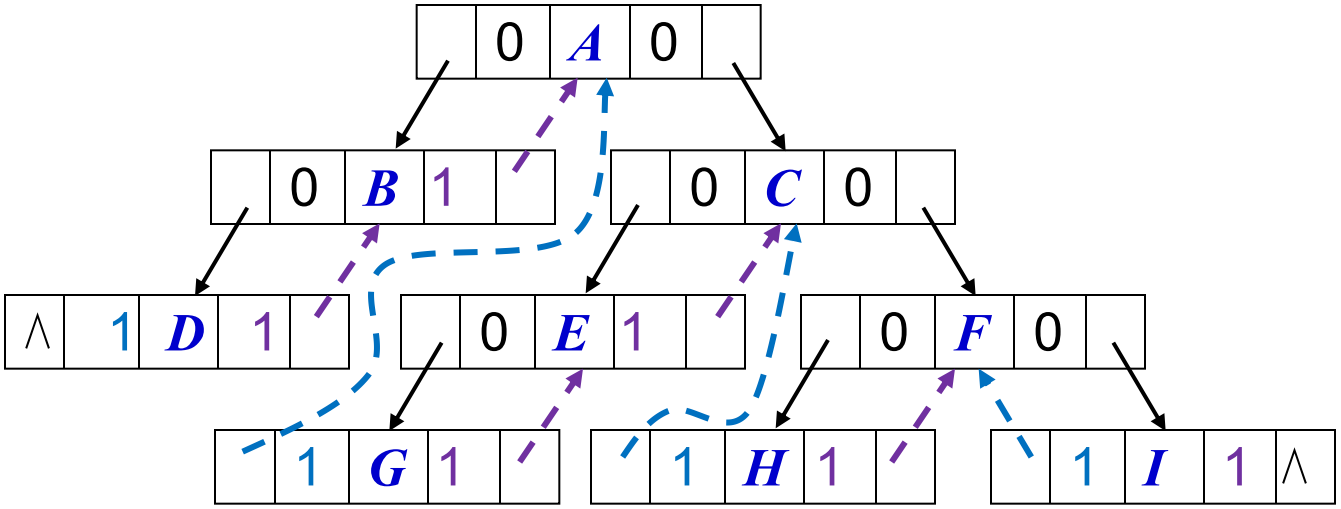
结点中序序列：
DBAGECHFI

练习：

1、对由图所示二叉树进行中序线索化。



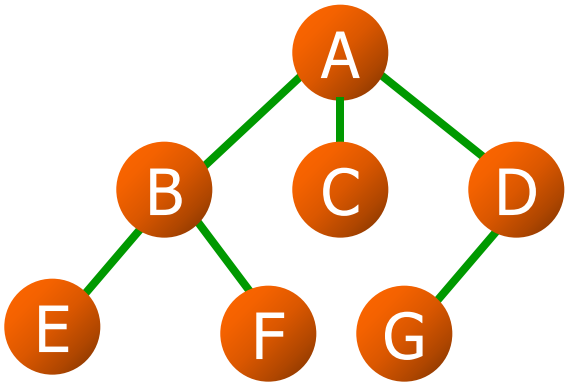
中序线索链表 结点序列：DBAGECHFI



6.4 树与森林

一. 树的存储结构

- 双亲表示法
  - 用一组连续的存储空间来存储树的结点，同时在每个结点中附加一个指示器(整数域)，用以指示双亲结点的位置(下标值)。



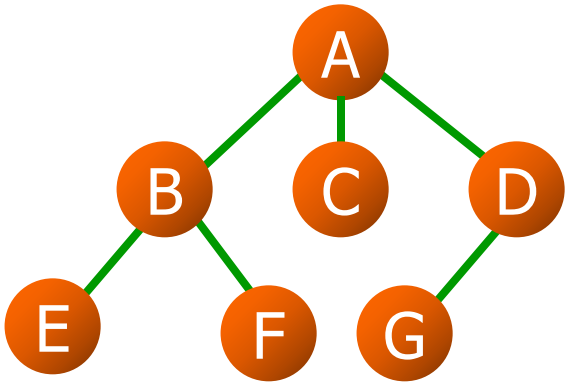
0	1	2	3	4	5	6
A	B	C	D	E	F	G
-1	0	0	0	1	1	3

这种存储结构利用了任一结点的双亲结点唯一的性质，可以方便地直接找到任一结点的双亲结点，但求结点的孩子结点时需要扫描整个数组。

6.4 树与森林

一. 树的存储结构

- 孩子表示法—多重链表
  - 可以采用多重链表，即每个结点有多个指针域
  - 最大缺点是空链域太多，[(d−1)n+1个]



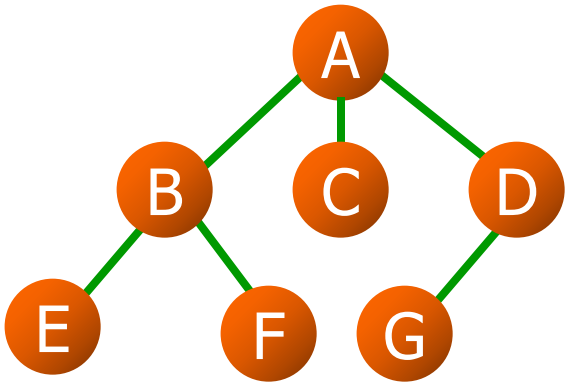
data	$child_1$	$child_2$	$child_3$	.....	$child_d$

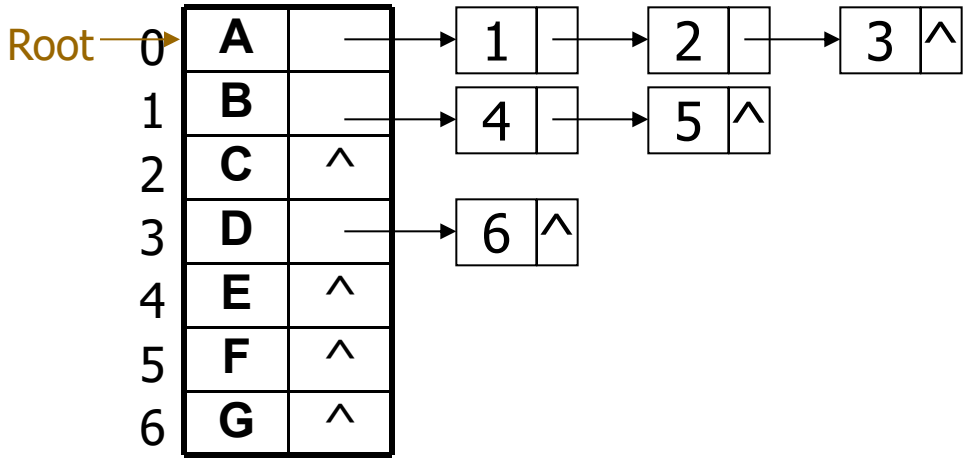
data	degree	$child_1$	$child_2$	.....	$child_d$

6.4 树与森林

一. 树的存储结构

- 孩子表示法—单链表
  - 将每个结点排列成一个线性表
  - 将每个结点的孩子排列起来，用单链表表示



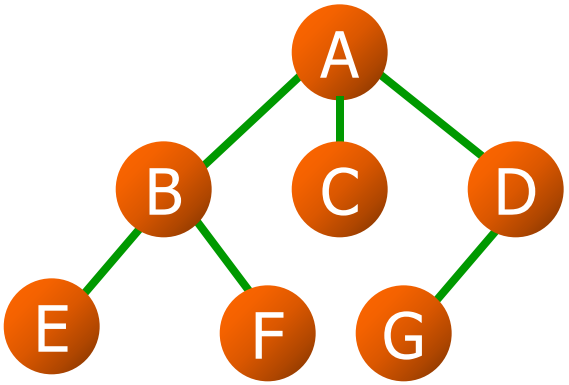


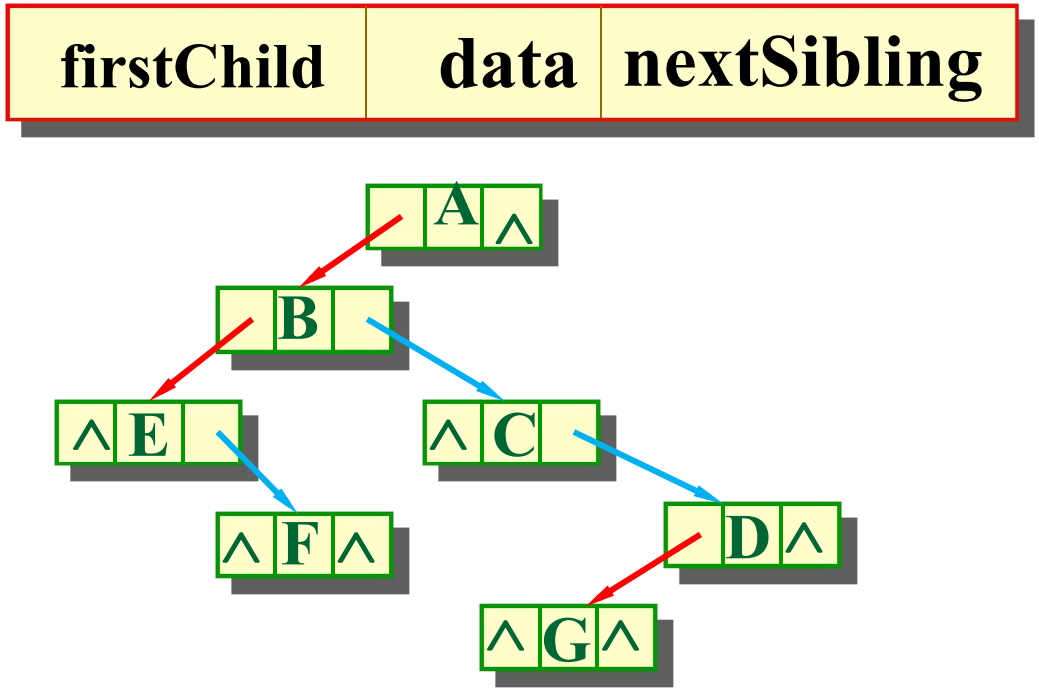
n个结点的树有n个(孩子)单链表(叶子结点的孩子链表为空)，而n个结点又组成一个线性表且以顺序存储结构表示。

6.4 树与森林

一. 树的存储结构

- 孩子兄弟表示法
  - 采用二叉链表
  - 左边指针指向第一个孩子，右边指针指向兄弟

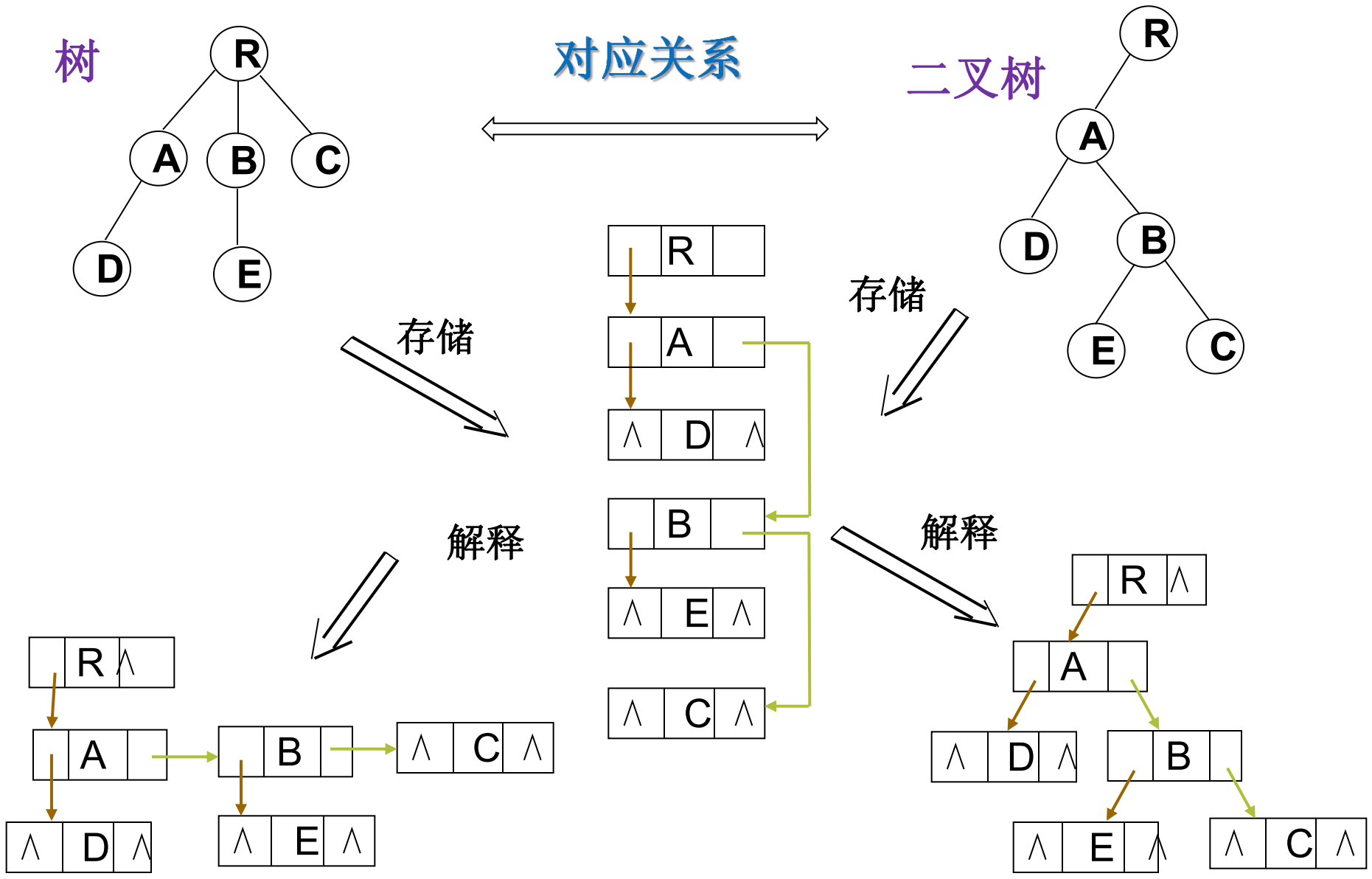




6.4 树与森林

二. 树、二叉树、森林的关系

- 树与二叉树的关系
  - 树与二叉树都可以采用二叉链表作存储结构
  - 任意给定一棵树，可以对应一个唯一的二叉树(根没有右子树)
- 树与二叉树的转换:
  - 以二叉链表作为存储结构，将其解释为树或二叉树，实现两者之间的转换。

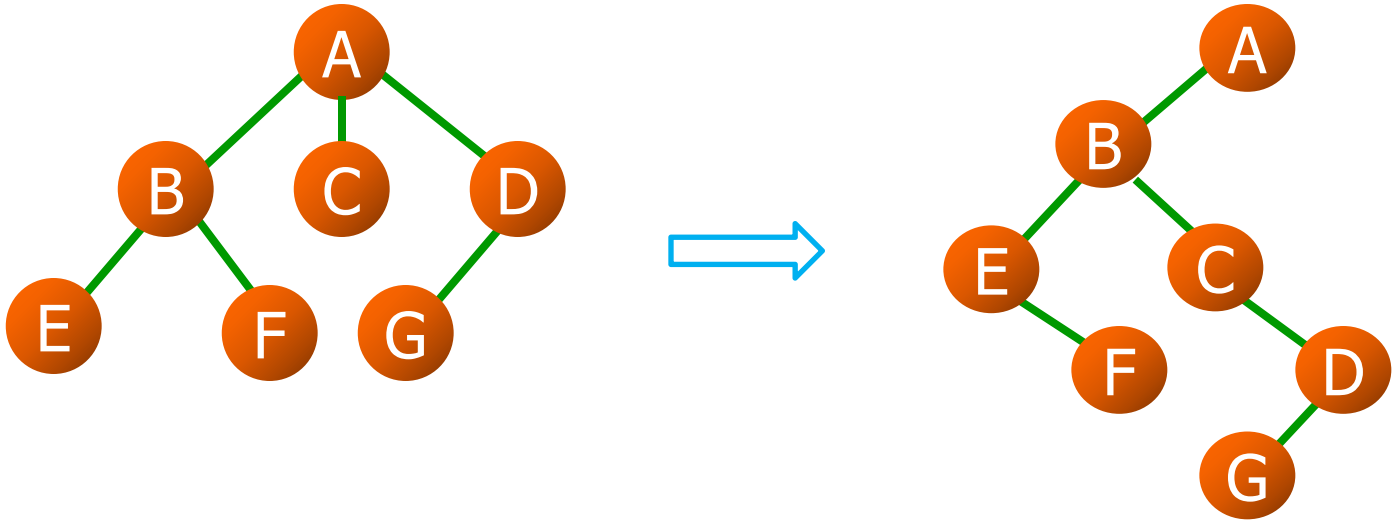


树与二叉树的对应关系

6.4 树与森林

二. 树、二叉树、森林的关系

- 树转二叉树
  - 左边指针指向第一个孩子，右边指针指向兄弟



6.4 树与森林

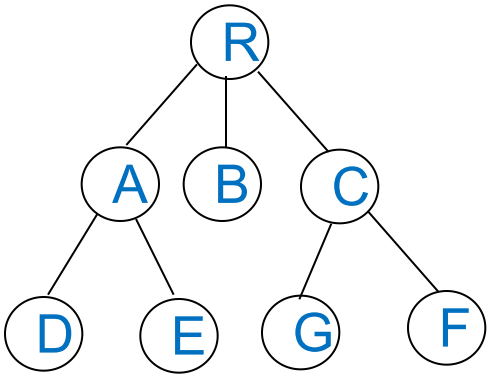
二. 树、二叉树、森林的关系

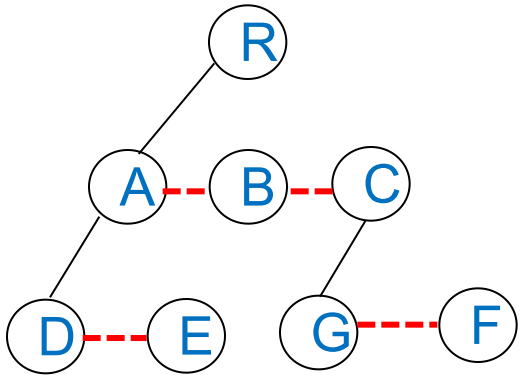
■ 树转二叉树

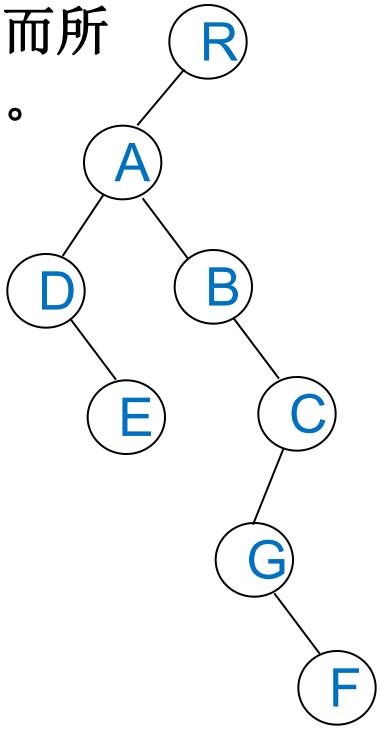
这样转换后的二叉树的特点是：

◆ 二叉树的根结点没有右子树，只有左子树；

◆ 左子树根结点仍然是原来树中相应结点的左孩子结点，而所有沿右链往下的右孩子结点均是原来树中该结点的兄弟结点。



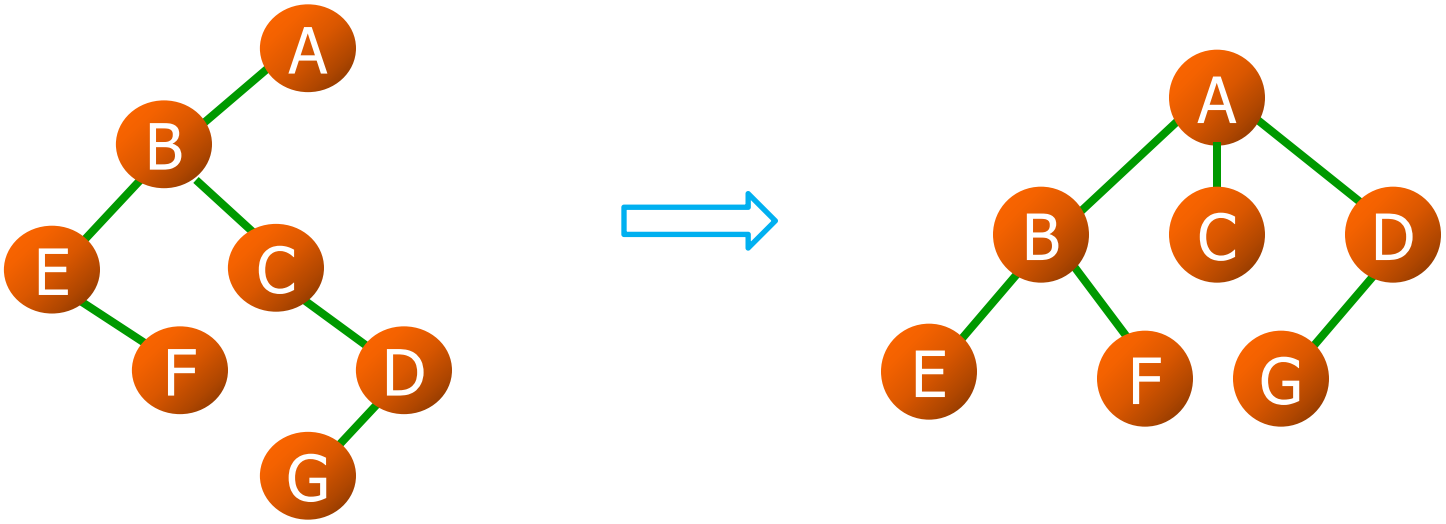




6.4 树与森林

二. 树、二叉树、森林的关系

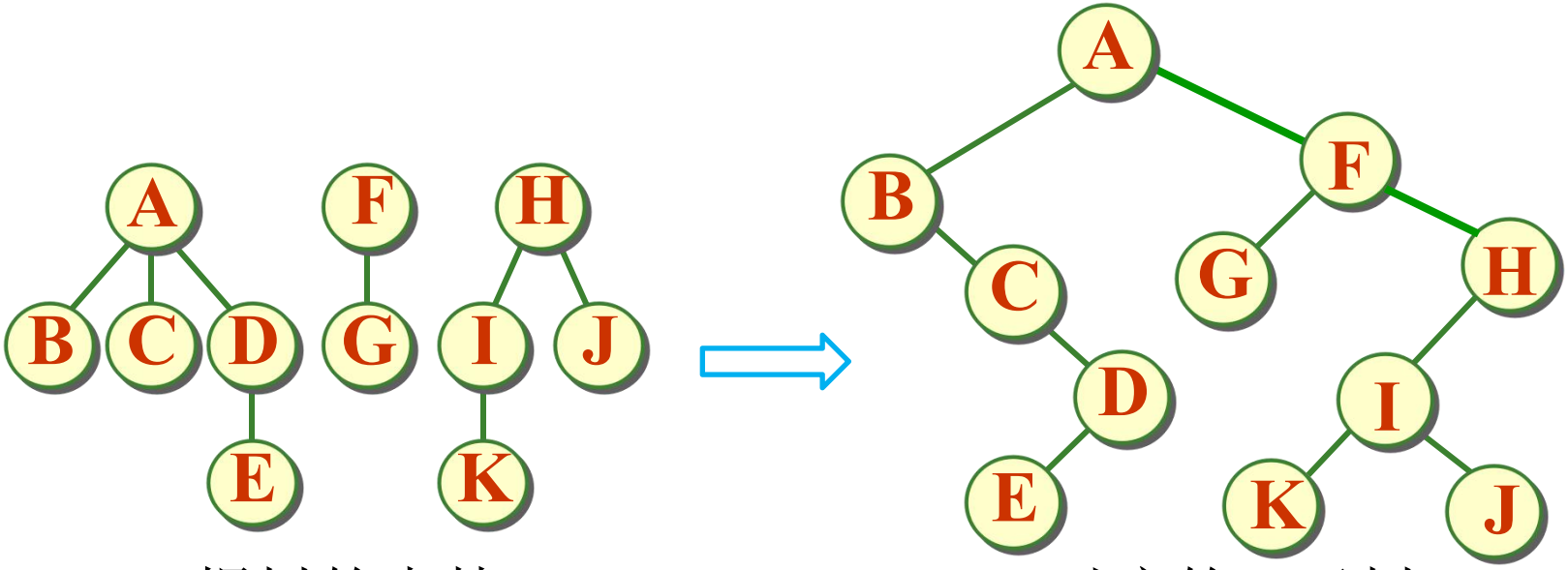
- 二叉树转树
  - 左指针做孩子，右指针做兄弟



6.4 树与森林

二. 树、二叉树、森林的关系

- 森林与二叉树的关系
  - 如果把森林中的第二棵树的根结点看作是第一棵树的根结点的兄弟，则可找到一个唯一的二叉树与之对应
  - 森林转二叉树，新树接入右孩子

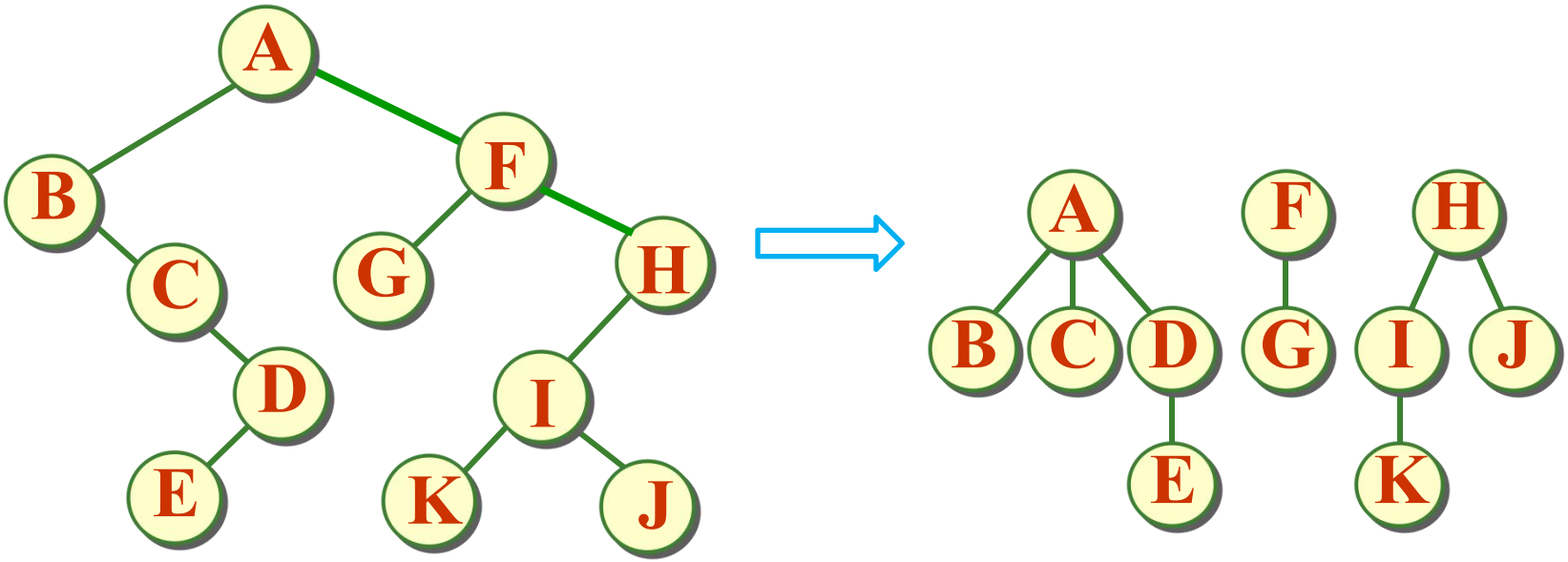


三棵树的森林　　对应的二叉树

6.4 树与森林

二. 树、二叉树、森林的关系

- 二叉树转森林
  - 根结点右孩子是一棵新树
  - 其他结点的右孩子是兄弟



6.4 树与森林

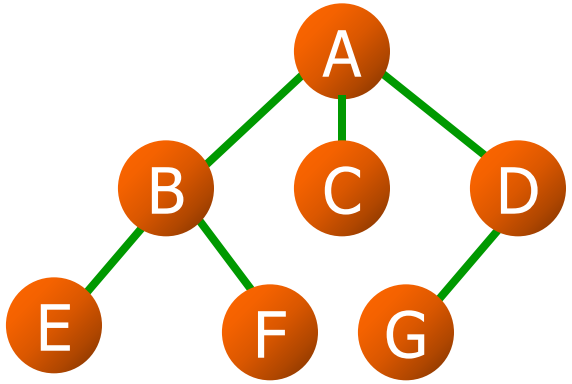
三. 树的遍历

- 对树的遍历主要有两种：

1. 先根（次序）遍历

当树非空时

- 访问根结点
- 依次先根遍历根的各棵子树

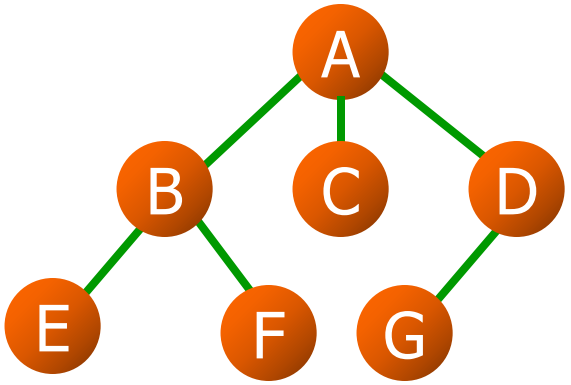


输出结果：ABEFCDG

2. 后根（次序）遍历

当树非空时

- 依次后根遍历根的各棵子树
- 访问根结点



输出结果：EFBCGDA

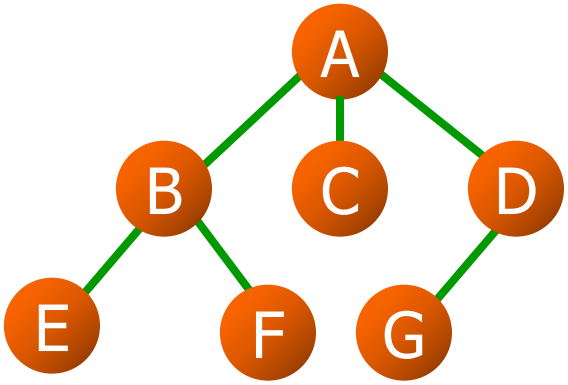
6.4 树与森林

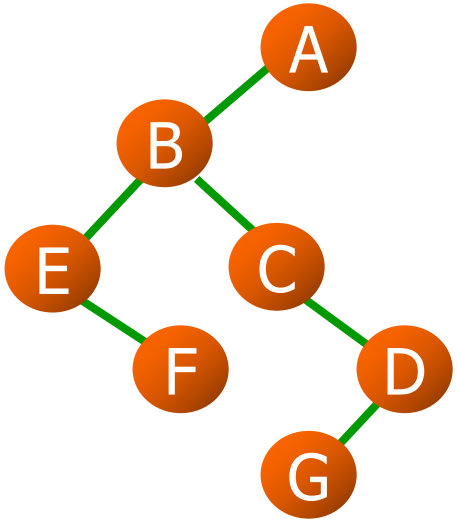
三. 树的遍历

- 与二叉树遍历的关系

（当采用孩子兄弟表示法表示树时）

- 树的先根遍历，与树对应的二叉树的先序遍历完全相同
- 树的后根遍历，与树对应的二叉树的中序遍历完全相同





先根遍历结果：ABEFCDG

后根遍历结果：EFBCGDA

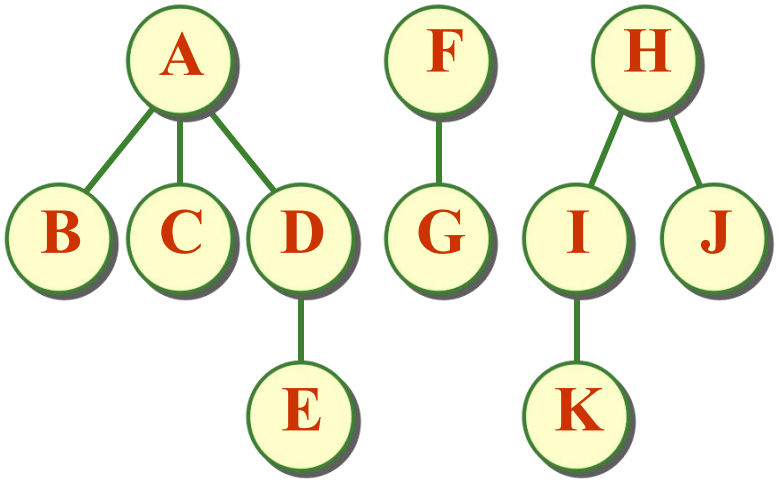
先根遍历结果：ABEFCDG

中序遍历结果：EFBCGDA

6.4 树与森林

四. 森林的遍历

- 对森林的遍历主要有两种：



1. 先序遍历

若森林不空，则

- 访问森林中第一棵树的根结点；
- 先序遍历森林中第一棵树的子树森林；
- 先序遍历森林中(除第一棵树之外)其余树构成的森林。

即：依次从左至右对森林中的每一棵树进行先根遍历。

先序遍历：

ABCDEFGHIKJ

6.4 树与森林

四. 森林的遍历

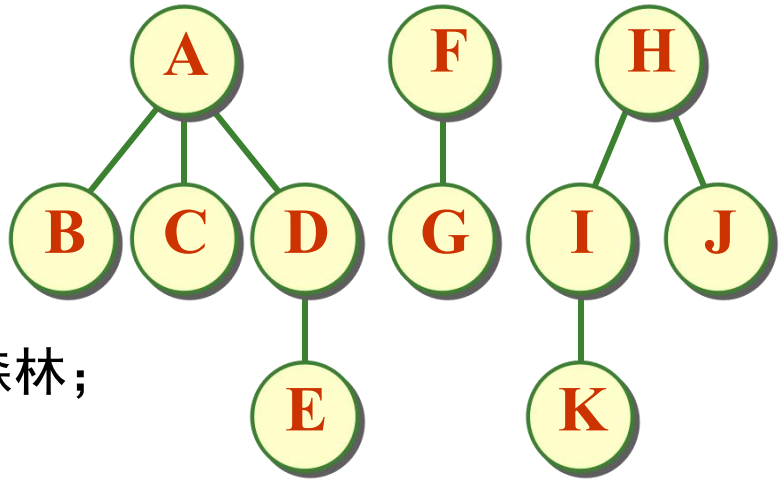
- 对森林的遍历主要有两种：

2. 中序遍历

若森林不空，则

- 中序遍历森林中第一棵树的子树森林；
- 访问森林中第一棵树的根结点；
- 中序遍历森林中(除第一棵树之外)其余树构成的森林。

即：依次从左至右对森林中的每一棵树进行后根遍历。



中序遍历：

BCEDAGFKIJH

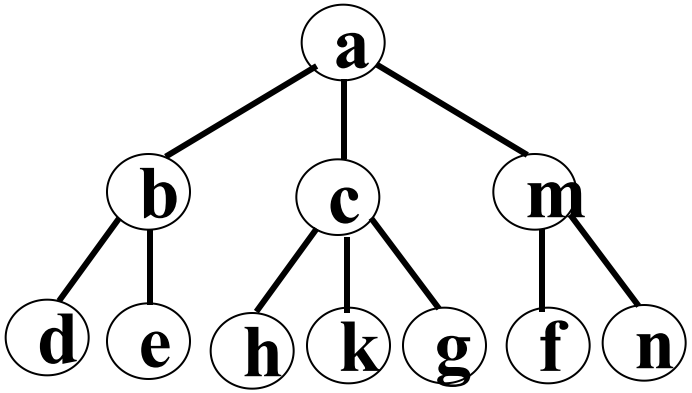
6.4 树与森林

遍历的对应关系

二叉树	树	森林
先序遍历	先根遍历	先序遍历
中序遍历	后根遍历	中序遍历

练习

1. 已知一棵树如下所示，完成以下要求
   - 请分别用双亲表示法、孩子表示法、孩子兄弟表示法给出该树的存储结构
   - 请将这棵树转换成二叉树
   - 请给出该树的先根遍历序列和后根遍历序列。



练习

2. 已知二叉树顺序存储是 A B C D E F 0 0 0 G 0 0 H，请画出它对应的树或森林

3. 已知二叉树顺序存储是 A B 0 C D 0 0 0 0 E，请画出它对应的树或森林

练习

4. 二叉树的中序和后序遍历结果分别为E F B C G H I D A 和 F E I H G D C B A 。

（1） 画出该树，求其先序遍历的结果；

（2）将其转换为树，画出转换后的树，并求其先根、后根遍历序列。

练习

- 已知森林如下所示，完成以下要求
  - 请将森林转换成二叉树
  - 请给出森林的先序遍历序列和中序遍历序列。