数据结构

树与二叉树

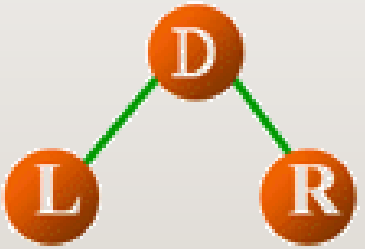
2021年10月

深圳大学电子与信息工程学院 周飞

2 第五章 树与二叉树

第三节 遍历二叉树

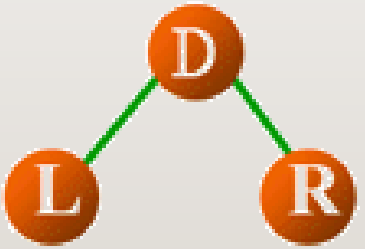
- 一、遍历二叉树
  - 树的遍历就是按照某种次序访问树中的结点，要求每个结点访问且仅访问一次。
  - 一个二叉树由根结点、左子树和右子树构成。
  - 记根结点为D，左子树表示为L，右子树表示为R。



3 第五章 树与二叉树

第三节 遍历二叉树

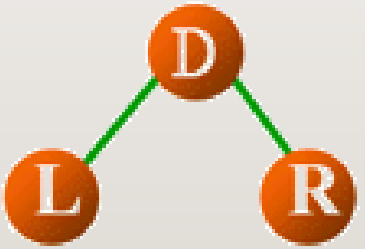
- 一、遍历二叉树
  - 规定：选左后右（访问左子树在访问右子树之前）
  - 根据访问根结点的次序，可以把遍历分成三种：
  - 1. D L R [先序遍历]
  - 2. L D R [中序遍历]
  - 3. L R D [后序遍历]



4 第五章 树与二叉树

第三节 遍历二叉树

- 二、先序遍历二叉树
  - 算法（D L R ）：
  - 1. 若树为空，则返回，否则
  - 2. 访问根结点（D）
  - 3. 访问左子树（L）：先序遍历左子树
  - 4. 访问右子树（R）：先序遍历右子树

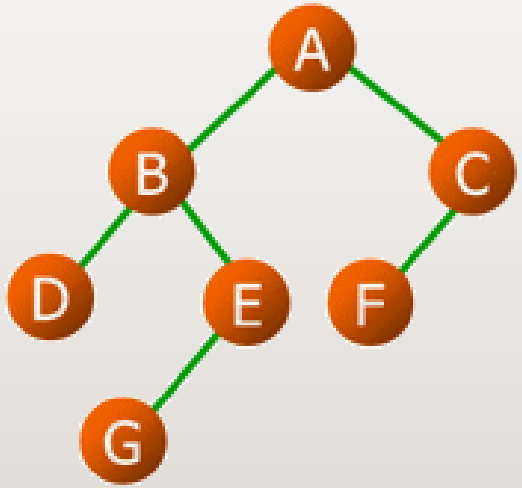


5 第五章 树与二叉树

第三节 遍历二叉树

- 二、先序遍历二叉树
  - 算法（举例）：
  - 1. 若树为空，则返回，否则
  - 2. 访问根结点（D）
  - 3. 访问左子树（L）：先序遍历左子树
  - 4. 访问右子树（R）：先序遍历右子树

A BDEG CF



6 第五章 树与二叉树

第三节 遍历二叉树

- 二、先序遍历二叉树
  - 算法（程序实现）：

指向根结点的指针

```
void PreOrderTraverse ( BinTree T ) {
    if (T) {
        cout << T->data;
        PreOrderTraverse ( T->lChild );
        PreOrderTraverse ( T->rChild );
    }
}
```

7 第五章 树与二叉树

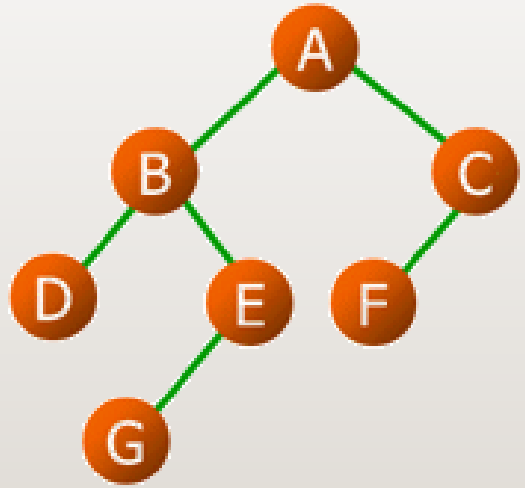
第三节 遍历二叉树

- 二、先序遍历二叉树
  - 特性：
  - 在先序遍历中，第一个输出的结点必为根结点。
  - 先序遍历的输出序列由

    根 + 左子树先序遍历序列 + 右子树先序遍历序列

    组成

A BDEG CF

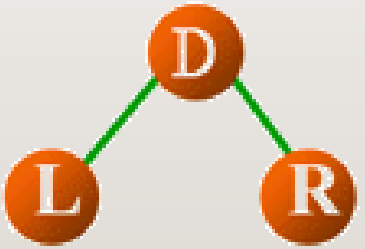


给定一个先序遍历的输出序列，能唯一确定二叉树树的结构吗？ 不能！

第五章 树与二叉树

第三节 遍历二叉树

- 三、中序遍历二叉树
  - 算法（L D R ）：
  - 1. 若树为空，则返回，否则
  - 2. 访问左子树（L）：中序遍历左子树
  - 3.访问根结点（D）
  - 4. 访问右子树（R）：中序遍历右子树

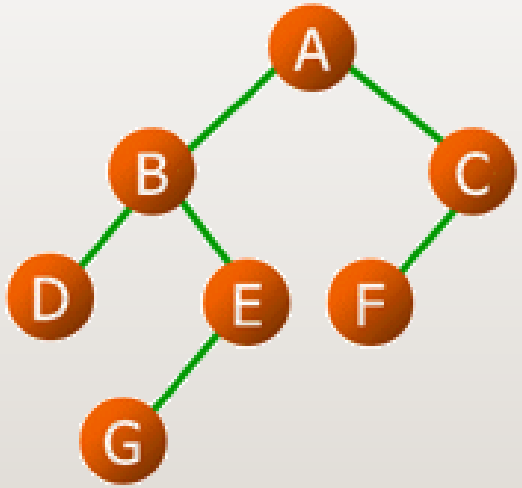


第五章 树与二叉树

第三节 遍历二叉树

- 三、中序遍历二叉树
  - 算法（举例）：
  - 1. 若树为空，则返回，否则
  - 2. 访问左子树（L）：中序遍历左子树
  - 3.访问根结点（D）
  - 4. 访问右子树（R）：中序遍历右子树

DBGE A FC



第五章 树与二叉树

第三节 遍历二叉树

- 三、中序遍历二叉树
  - 算法（程序实现）：

指向根结点的指针

```
void InOrderTraverse ( BinTree T ) {
    if (T) {
        InOrderTraverse ( T->lChild );
        cout << T->data;
        InOrderTraverse ( T->rChild );
    }
}
```

第五章 树与二叉树

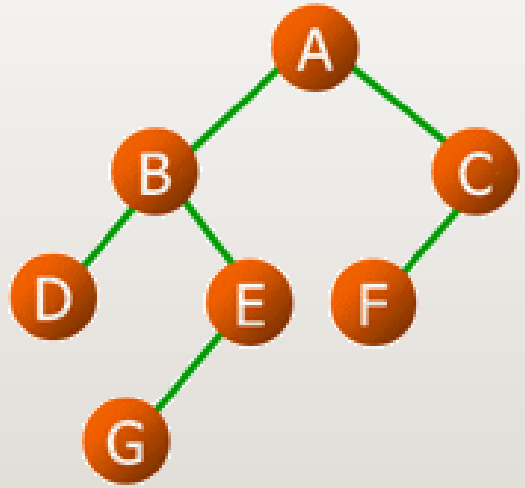
第三节 遍历二叉树

- 三、中序遍历二叉树
  - 特性：
  - 在中序遍历中，先于根结点输出的是左子树的结点，后于根结点输出的是右子树的结点。
  - 中序遍历的输出序列由

    左子树中序遍历序列 + 根 + 右子树中序遍历序列

    组成

DBGE A FC

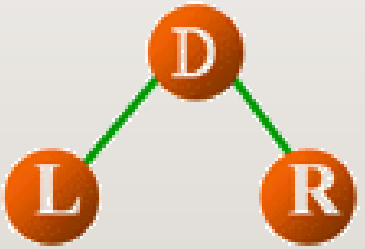


给定一个中序遍历的输出序列，能唯一确定二叉树树的结构吗？ 不能！

第五章 树与二叉树

第三节 遍历二叉树

- 四、后序遍历二叉树
  - 算法（ＬＲＤ）：
  - 1. 若树为空，则返回，否则
  - 2. 访问左子树（L）：后序遍历左子树
  - 3. 访问右子树（R）：后序遍历右子树
  - 4. 访问根结点（D）

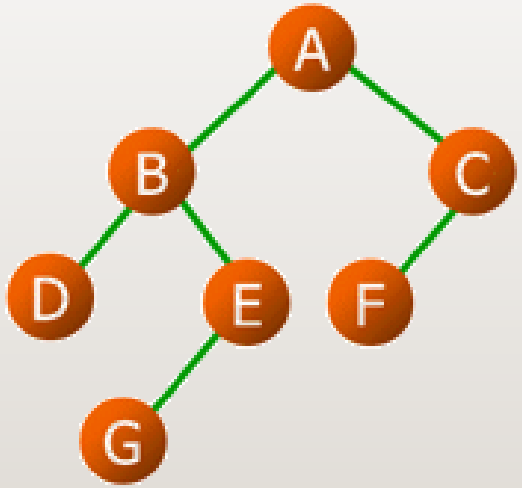


第五章 树与二叉树

第三节 遍历二叉树

- 四、后序遍历二叉树
  - 算法（举例）：
  - 1. 若树为空，则返回，否则
  - 2. 访问左子树（L）：后序遍历左子树
  - 3. 访问右子树（R）：后序遍历右子树
  - 4. 访问根结点（D）

DGEB FC A



第五章 树与二叉树

第三节 遍历二叉树

- 四、后序遍历二叉树
  - 算法（程序实现）：

```
void PostOrderTraverse ( BinTree T ) {
    if (T) {
        PostOrderTraverse ( T->lChild );
        PostOrderTraverse ( T->rChild );
        cout << T->data;
    }
}
```

第五章 树与二叉树

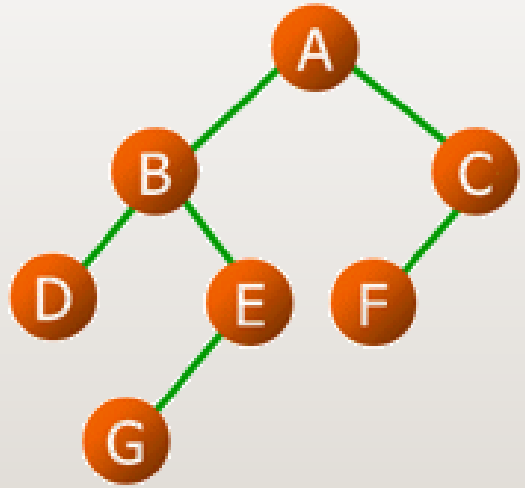
第三节 遍历二叉树

- 四、后序遍历二叉树
  - 特性：
  - 在后序遍历中，最后一个输出结点必为根结点。
  - 后序遍历的输出序列由

    左子树后序遍历序列 + 右子树后序遍历序列 + 根

    组成

DGEB FC A



给定一个后序遍历的输出序列，能唯一确定二叉树树的结构吗？ 不能！

第五章 树与二叉树

第三节 遍历二叉树

VS.

solo不能完成的任务

组团能不能完成？

答案是肯定的！

第五章 树与二叉树

第三节 遍历二叉树

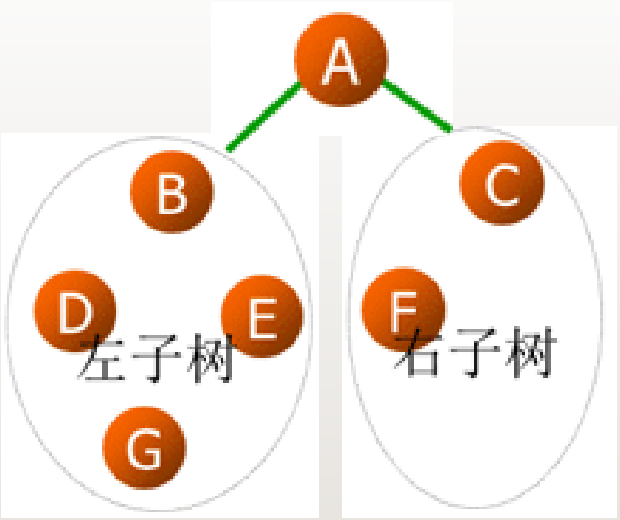
- 五、通过先、中序遍历序列求二叉树
  - 如果已知一棵二叉树的先序遍历和中序遍历序列，则可以唯一确定这棵二叉树。
  - 算法：
  - 1. 通过先序遍历确定根结点D：先序遍历的第一个输出就是根结点D。
  - 2. 通过根结点和中序遍历找到左子树L的结点和右子树R的结点。
  - 3. 对左右子树重复上述1-2步直到确定整个二叉树。

第五章 树与二叉树

第三节 遍历二叉树

- 五、通过先、中序遍历序列求二叉树

已知一棵二叉树的先序遍历序列为：ABDEGCF，中序遍历序列为：DBGEAFC，请画出这棵二叉树



- 根据先序遍历序列，可知根节点为A；
再根据中序遍历序列可知，左子树由DBGE组成
右子树由FC组成

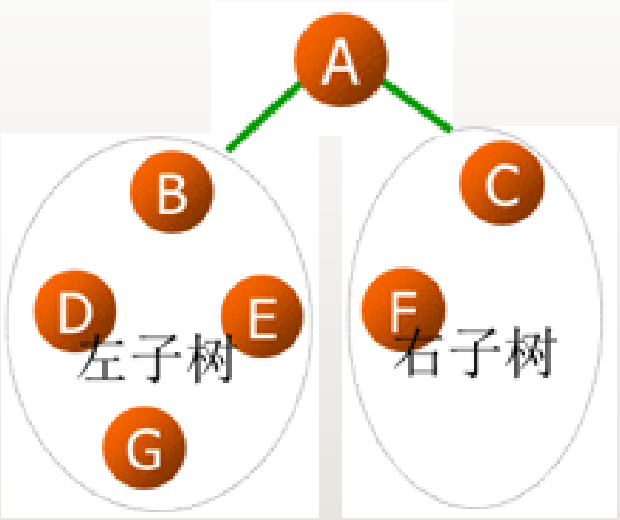
第五章 树与二叉树

第三节 遍历二叉树

- 五、通过先、中序遍历序列求二叉树

从整棵二叉树的先序遍历序列ABDEGCF可知，左子树的先序遍历序列为BDEG，右子树的先序遍历为CF

从整棵二叉树的中序遍历序列DBGEAFC右知，左子树的中序遍历为DBGE，右子树的中序遍历为FC



对左右子树重复上述的操作。

第五章 树与二叉树

第三节 遍历二叉树

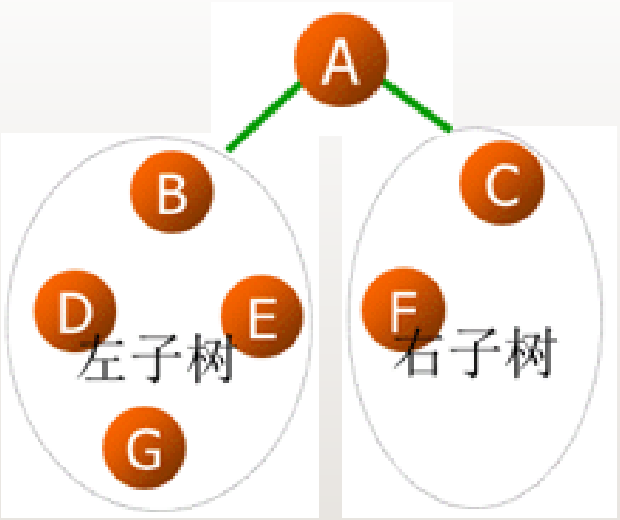
- 六、通过后、中序遍历序列求二叉树
  - 如果已知一棵二叉树的后序遍历和中序遍历序列，则可以唯一确定这棵二叉树。
  - 算法：
  - 1. 通过后序遍历确定根结点D：最后第一个结点就是根结点D。
  - 2. 通过根结点和中序遍历找到左子树L的结点和右子树R的结点。
  - 3. 对左右子树重复上述1-2步直到确定整个二叉树。

第五章 树与二叉树

第三节 遍历二叉树

- 六、通过后、中序遍历序列求二叉树

已知一棵二叉树的后序遍历序列为：DGEBFCA，中序遍历序列为：DBGEAFC，请画出这棵二叉树



- 根据后序遍历序列，可知根节点为A；

再根据中序遍历序列可知，左子树由DBGE组成
右子树由FC组成

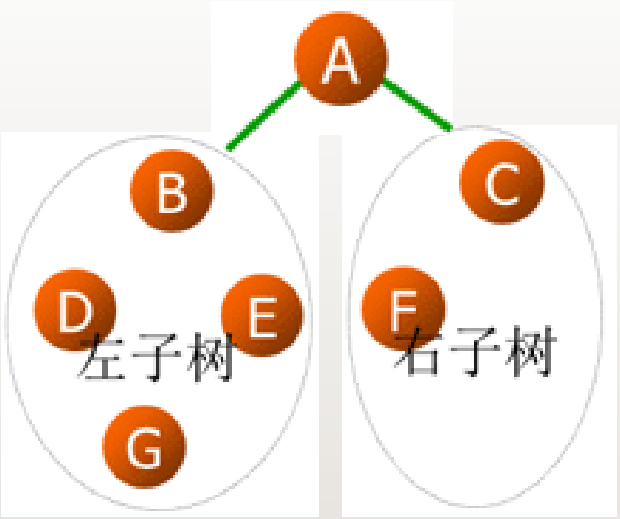
第五章 树与二叉树

第三节 遍历二叉树

- 六、通过后、中序遍历序列求二叉树

从整棵二叉树的后序遍历序列DGEBFCA可知，左子树的后序遍历序列为DGEB，右子树的后序遍历为FC

从整棵二叉树的中序遍历序列DBGEAFC右知，左子树的中序遍历为DBGE，右子树的中序遍历为FC



对左右子树重复上述的操作。

第五章 树与二叉树

第三节 遍历二叉树

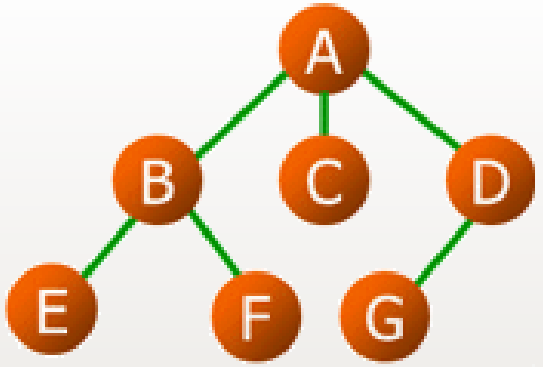
- 六、先、后序遍历序列是否可以求二叉树?

先序：ABC
后序：CBA

不行，不具备确定左右子树的能力。

第五章 树与二叉树

第四节 树与森林



- 一、树的存储结构
  - 1、双亲表示法
    - 采用一组连续的存储空间。
    - 在每个结点中附设一个指示器指示其双亲的位置。

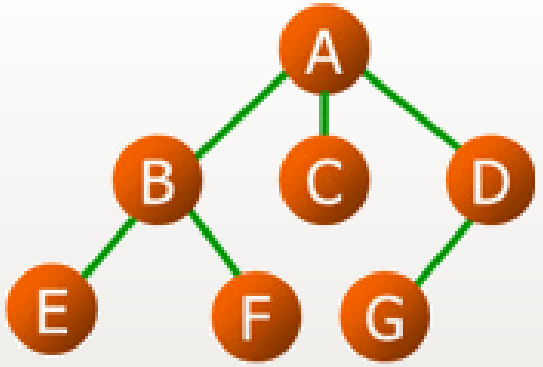
0	1	2	3	4	5	6	7
	A	B	C	D	E	F	G
	0	1	1	1	2	2	4

找双亲容易，找孩子难。

第五章 树与二叉树

第四节 树与森林

- 一、树的存储结构
  - 2、孩子表示法
    - 采用多重链表，即每个结点有多个指针域



data	$child_1$	$child_2$	$child_3$	·····	$child_d$

最大的缺点就是空指针（链）域太多

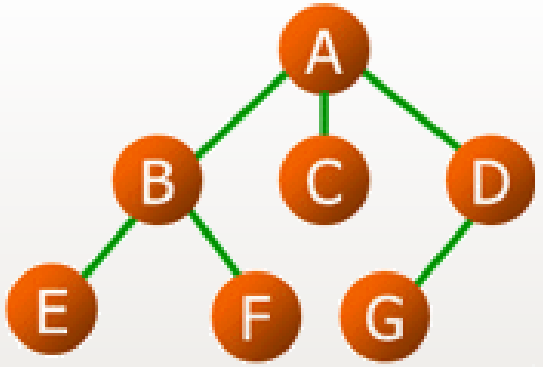
有些结点有很多孩子，有些结点的孩子很少：结构不均衡！

给定最大孩子数，是不是能找一个好的树的结构，使得空域变少呢？

第五章 树与二叉树

第四节 树与森林

- 一、树的存储结构
  - 2、孩子表示法



考察一个结点数是n的树，
假设其结点度的最大值（也称为树的度）为d

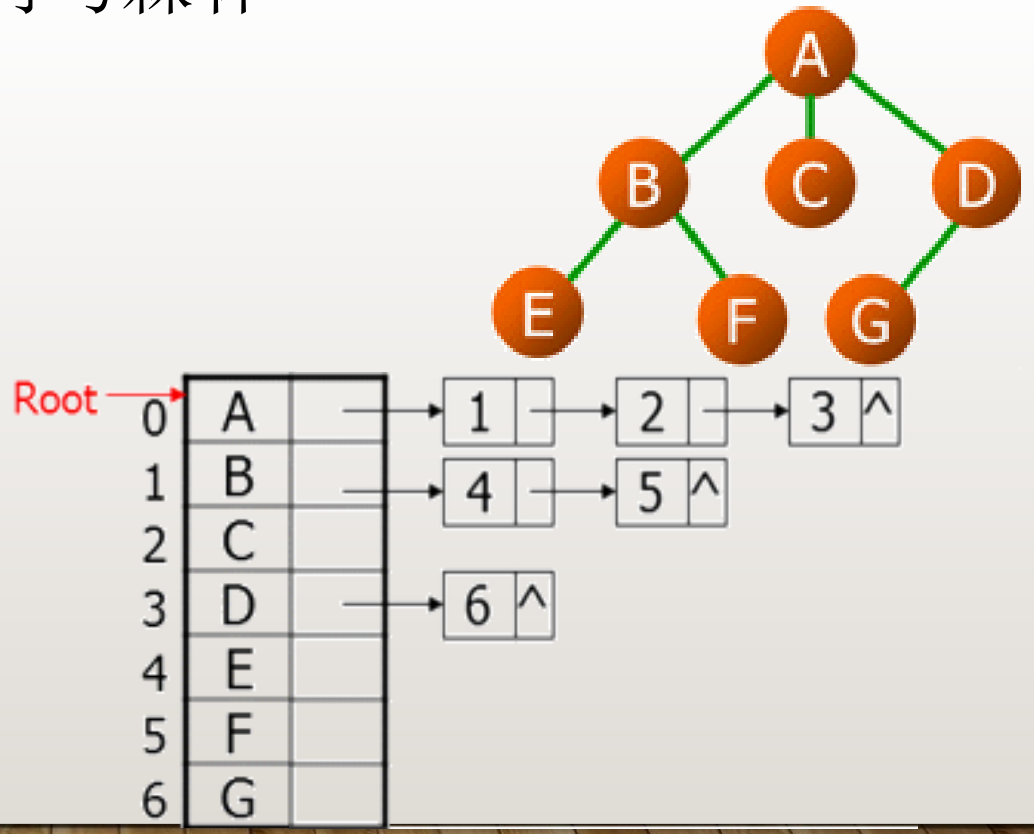
多链表表示法中，树中的空指针（链）域必有[(d-1)n+1]个

跟树中结点的结构没有关系，度越大，空域越多，浪费越多。

第五章 树与二叉树

第四节 树与森林

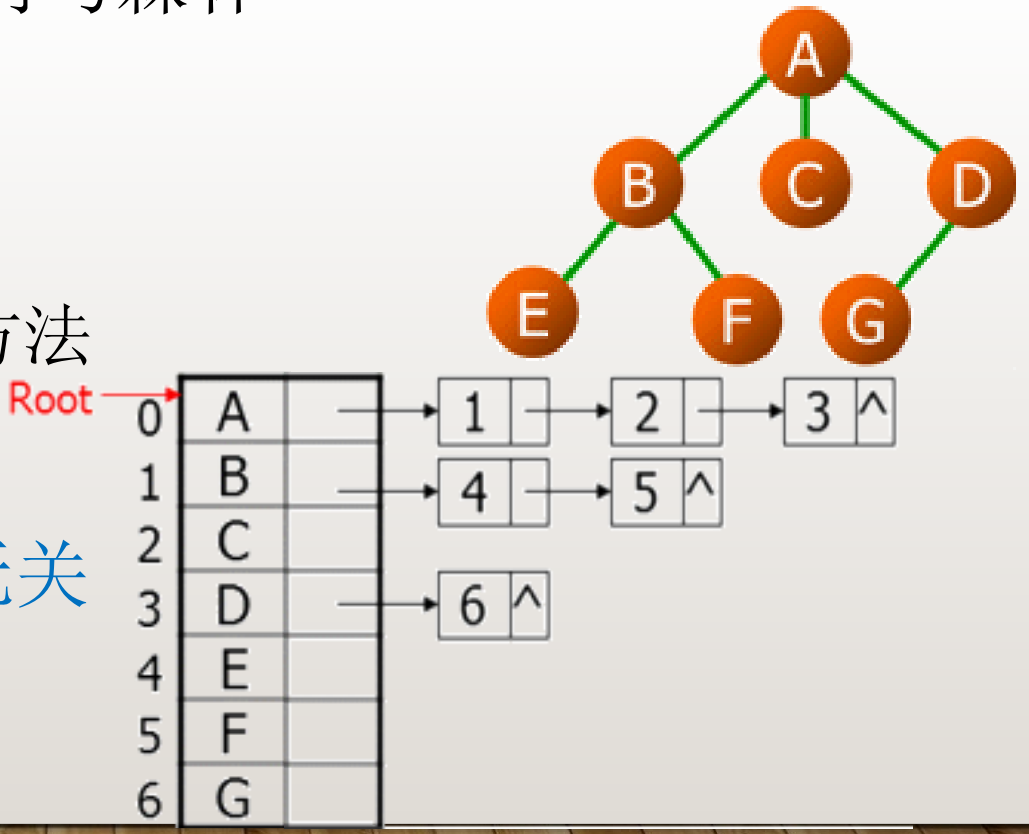
- 一、树的存储结构
  - 2、孩子表示法
    - 每个结点的孩子排列起来构成一个单链表
    - 将每个结点排列起来构成一个线性表



第五章 树与二叉树

第四节 树与森林

- 一、树的存储结构
  - 2、孩子表示法
    - 采用线性表加单链表的表示方法
    - 空指针域的个数是n个
    - 空指针域的个数跟树的度无关



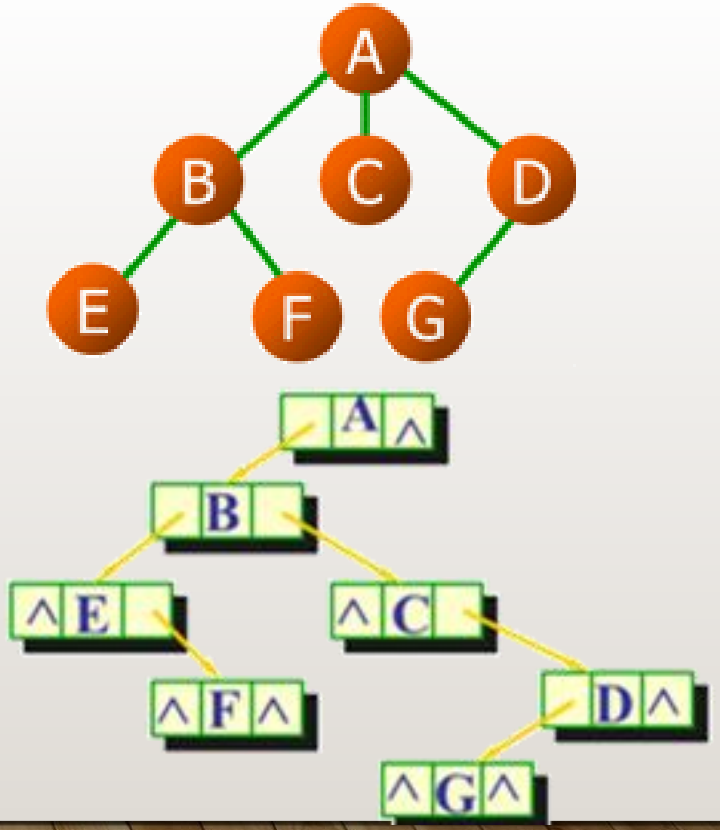
第五章 树与二叉树

第四节 树与森林

- 一、树的存储结构
  - 3、兄弟表示法
    - 采用二叉链表表示。

data	firstChild	nextSibling

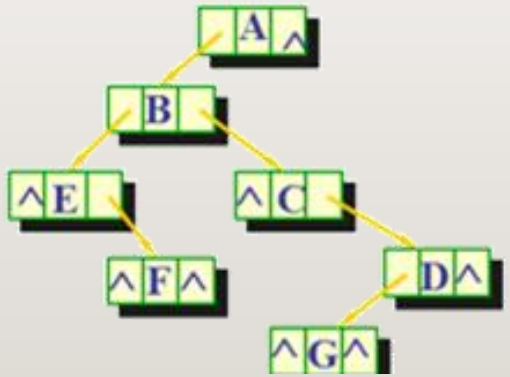
    - 左边指针指向自己的第一个孩子，右边指针指向自己的下一个兄弟。



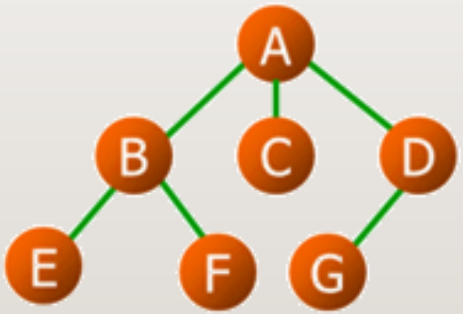
第五章 树与二叉树

第四节 树与森林

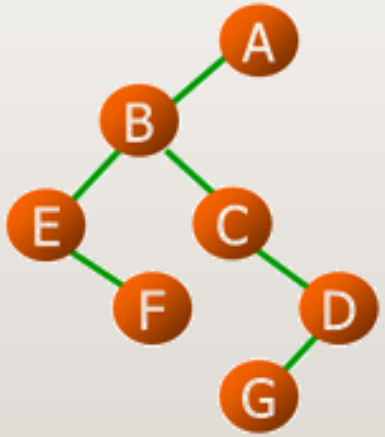
- 二、树与二叉树的对应关系
  - 树与二叉树都可以采用二叉链表的存储方式
  - 任意一个树都可以找到一个唯一的二叉树与之对应



二叉链表表示



树



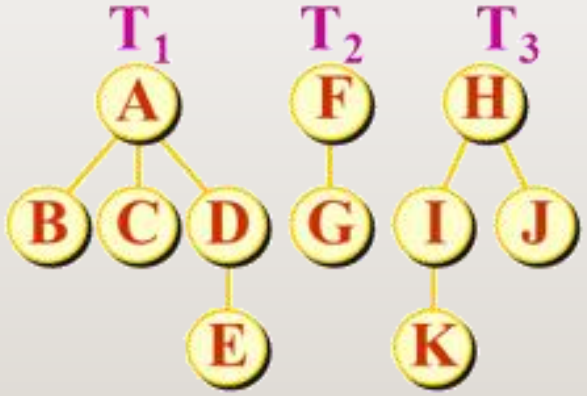
对应的二叉树

特性：
肯定没有右子树！

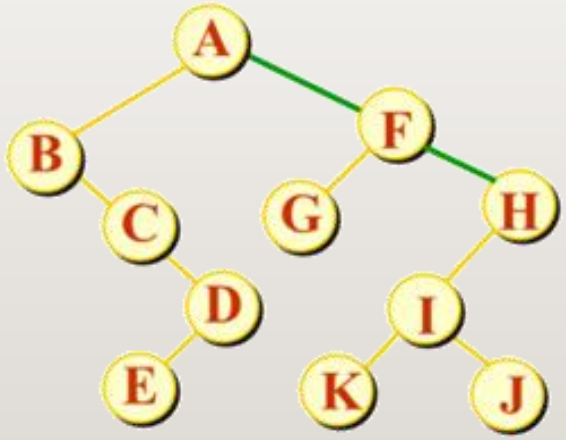
第五章 树与二叉树

第四节 树与森林

- 三、森林与二叉树的对应关系
  - 如果把森林中下一颗树的根结点看出上一个树根结点的兄弟，可以找到一个唯一的二叉树与该森林对应。



三棵树的森林

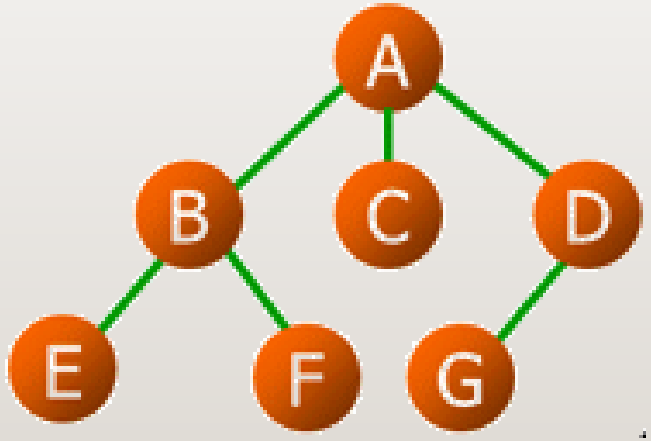


对应的二叉树

第五章 树与二叉树

第四节 树与森林

- 四、树的遍历
  - 1、先根（次序）遍历
  - 2、后根（次序）遍历



第五章 树与二叉树

第四节 树与森林

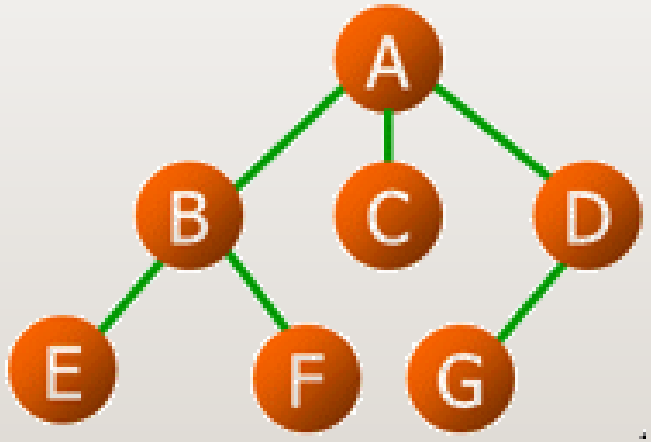
- 四、树的遍历
  - 1、先根（次序）遍历

    当树非空时，

    1）访问根结点；

    2）先根遍历根的各颗子树（从左至右）。

ABEFCDG



第五章 树与二叉树

第四节 树与森林

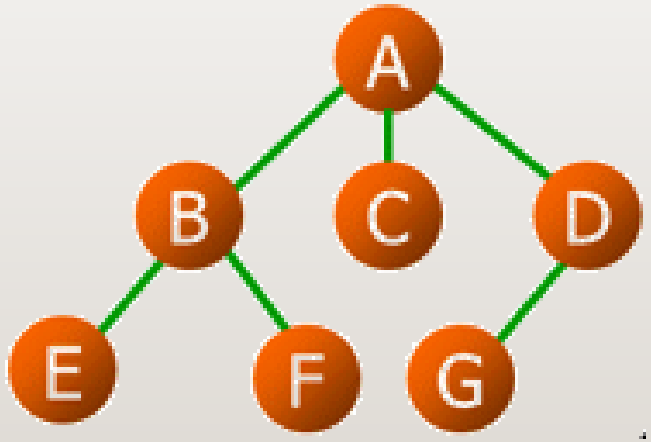
- 四、树的遍历
  - 2、后根（次序）遍历

    当树非空时，

    1）后根遍历根的各颗子树（从左至右）；

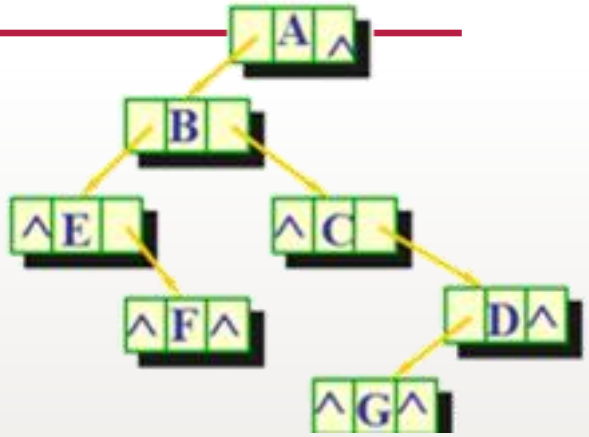
    2）访问根结点。

EFBCGDA



第五章 树与二叉树

第四节 树与森林



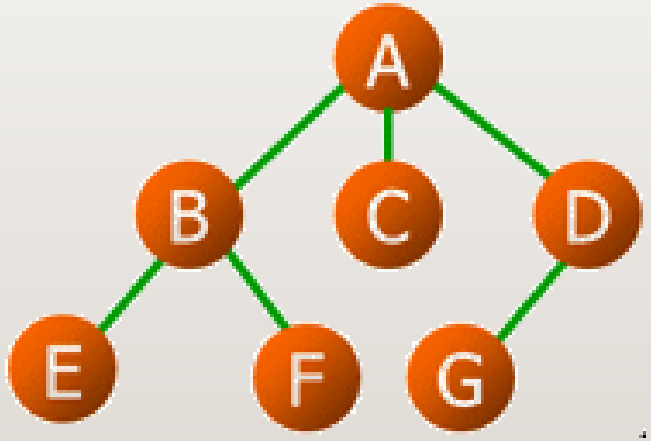
- 四、树的遍历
  - 3、与二叉树遍历的关系

    当使用兄弟表示法表示树时，

    树的先根遍历（ABEFCDG）

    等同于

    其对应二叉树的先根遍历（ABEFCDG）



第五章 树与二叉树

第四节 树与森林

- 四、树的遍历
  - 3、与二叉树遍历的关系

    当使用兄弟表示法表示树时，

    树的后根遍历（EFBCGDA）

    等同于

    其对应二叉树的中根遍历（ EFBCGDA ）