数据结构

深圳技术大学
大数据与互联网学院

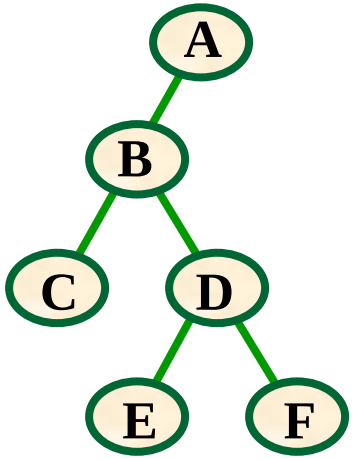
第六章 树和·二叉树

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
6.5 树与等价问题
6.6 赫夫曼树及其应用
6.7 回溯法与树的遍历
6.8 树的计数

6.3 遍历二叉树

二叉树的构建

- 通过先序遍历字符串建立二叉树链表，
  - 用特定字符表示空树
  - 例如ABC00DE0G00F000



6.3 遍历二叉树

二叉树的构建

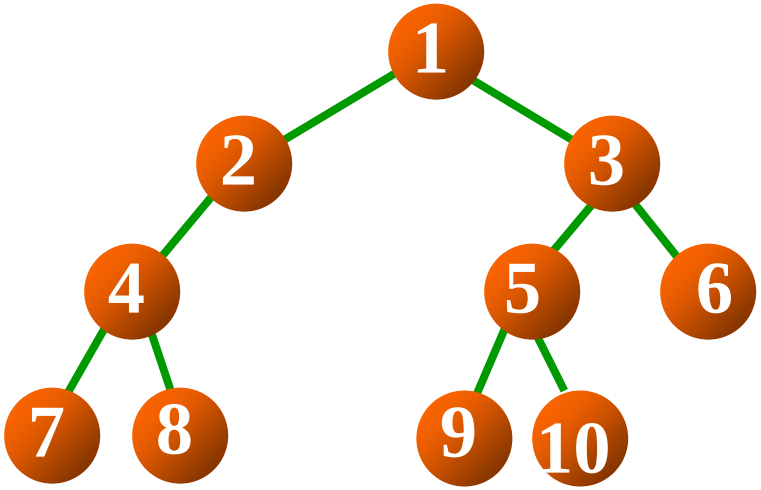
- 逐个输入字符构建链式存储结构

```
BiTree CreateBiTree(BiTNode* &T) {
  char ch;
  cin >> ch;
  if (ch=='0') T = NULL;
  else {
    T = new BiTNode;                  // 分配内存
    if (!T) return ERROR;             // 分配失败
    T->data = ch;                     // 生成根结点
    CreateBiTree(T->lchild);          // 构造左子树
    CreateBiTree(T->rchild);          // 构造右子树
  }
  return OK;
} // CreateBiTree
```

6.3 遍历二叉树

二叉树的构建

- 按完全二叉树的顺序表示，结合二叉树性质5构建

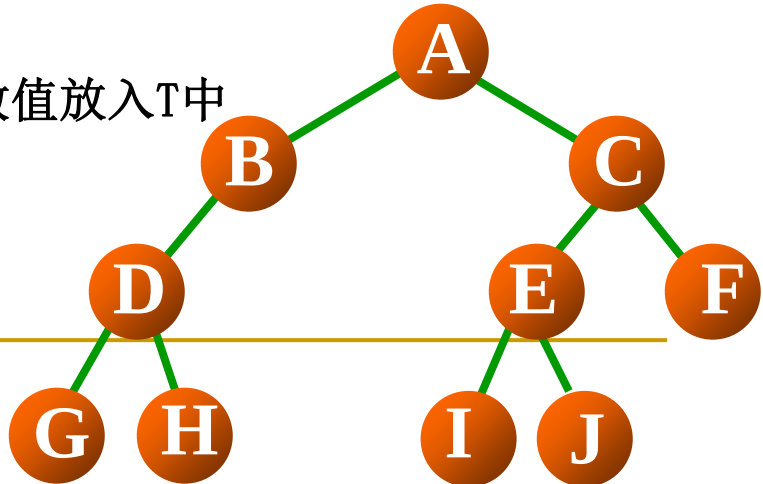


6.3 遍历二叉树

二叉树的构建

- 按完全二叉树的顺序表示，结合二叉树性质5构建
  - 根据二叉树性质5，因为数组从0编号，父节点是i，则左右孩子的位置是2i+1和2i+2
  - 采用递归实现，设定函数参数pos，表示当前结点在数组的位置
  - 函数作用：根据pos位置的数值创建结点t，返回结点t
  - 输入：一个数组的数据，实际就是一个字符串，其中字符0表示结点为空
  - 算法流程

1. 数组越界检查，检查pos如果超过数组长度，直接返回空
2. 获取pos位置的数值，如果为字符0，则结点t为空
3. 如果数值不为0，执行以下过程
   - 3.1 为结点T分配空间，使用new方法，并把数值放入T中
   - 3.2 在第2i+1位置递归创建t的左子树
   - 3.3 在第2i+2位置递归创建t的右子树
4. 返回结点t



6.3 遍历二叉树

二叉树的构建

- 根据两种遍历序列构建二叉树，根据遍历的特性已知：
  - 先序遍历中，第一个输出节点必为根节点
  - 中序遍历中，先于根节点D输出的节点为左子树的节点，后于根节点D输出的节点为右子树的节点
  - 后序遍历中，最后一个输出节点必为根节点
  - 先序+中序构建
  - 中序+后序构建

6.3 遍历二叉树

二叉树的构建

- 已知先中序遍历构建二叉树：
  - 在先序遍历序列中，第一个节点为根节点D
  - 在中序遍历序列中，根节点D左边的节点归为左子树，根节点D右边的节点归为右子树
  - 对每个子树反复使用1,2两步，直到确定二叉树

6.3 遍历二叉树

二叉树的构建

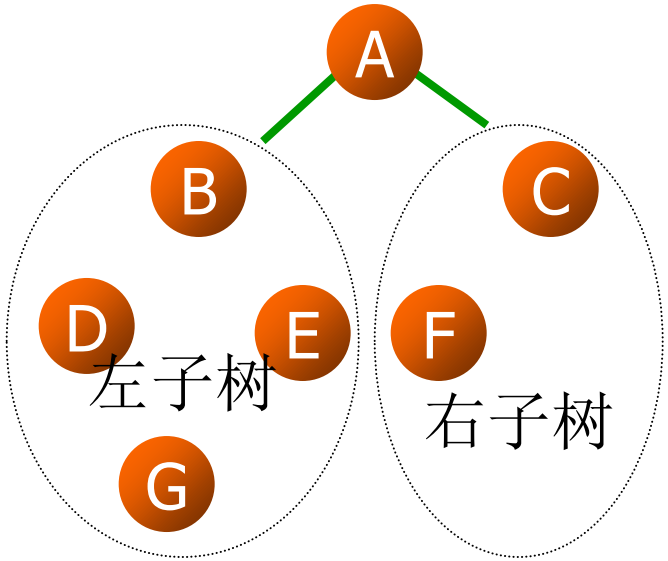
- 已知先中序遍历构建二叉树：
  - 先序遍历序列为：ABDEGCF
  - 中序遍历序列为：DBGEAFC

1、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

2、在左子树的DBGE中，根据先序找出根结点是B，然后根据中序遍历，又分为左孙子树为D，右孙子树为GE

3、在右子树的FC中，根据先序遍历找出根节点C，然后根据中序遍历，由分为左孙子树为F，右孙子树为空

以此类推



6.3 遍历二叉树

二叉树的构建

- 已知后中序遍历构建二叉树：
  - 在后序遍历序列中，最后一个节点为根节点D
  - 在中序遍历序列中，根节点D左边的节点归为左子树，根节点D右边的节点归为右子树
  - 对每个子树反复使用1,2两步，直到确定二叉树

6.3 遍历二叉树

二叉树的构建

- 举例

  后序遍历序列为：DGEBFCA

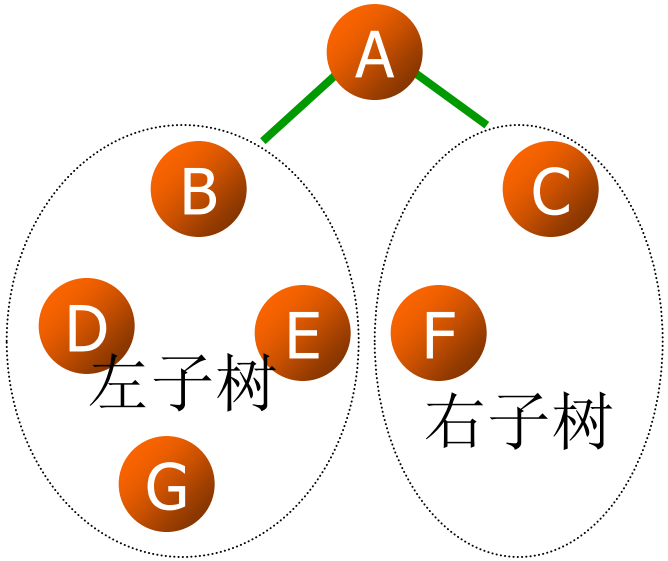
  中序遍历序列为：DBGEAFC

1、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

2、在左子树的DBGE中，根据后序找出根结点是B，然后根据中序遍历，又分为左孙子树为D，右孙子树为GE

3、在右子树的FC中，根据后序遍历找出根节点C，然后根据中序遍历，又分为左孙子树为F，右孙子树为空

以此类推



6.3遍历二叉树

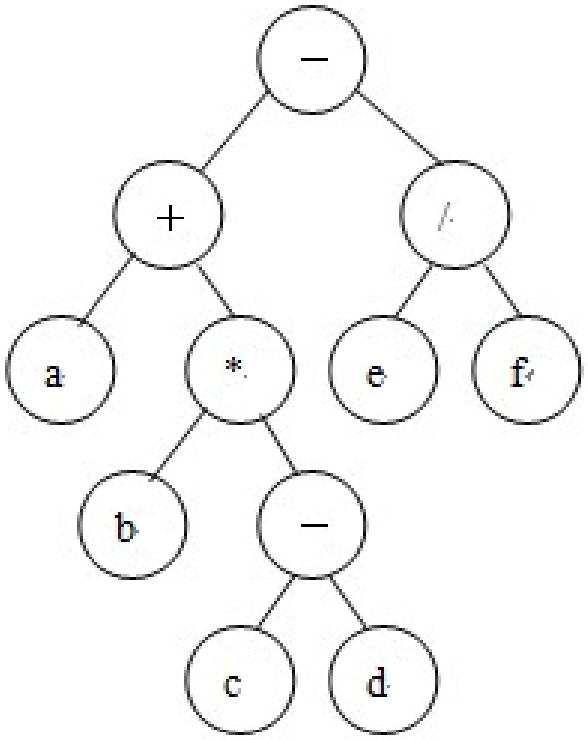
表达式与二叉树

- 后缀表达式：不包含括号，运算符放在两个运算对象的后面，所有的计算按运算符出现的顺序，严格从左向右进行，不再考虑运算符的优先规则
- 当我们对此二叉树进行先序、中序和后序遍历后，便可得到表达式的前缀、中缀和后缀形式：

  前缀：-+a*b-cd/ef

  中缀：a+b*c-d-e/f

  后缀：abcd-*+ef/-
- 在计算机中，往往用中缀形式是表示算术表达式的通常形式，只是没有括号。
- 往往使用后缀表达式易于求值

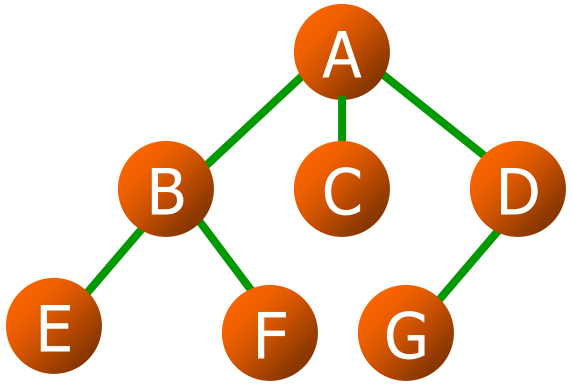


6.4 树与森林

一. 树的存储结构

- 双亲表示法
  - 采用一组连续的存储空间
  - 由于每个结点只有一个双亲，只需要一个指针

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

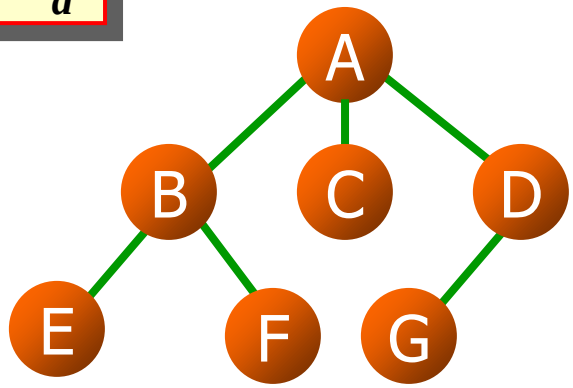


6.4 树与森林

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
  - 最大缺点是空链域太多，[(d-1)n+1个]

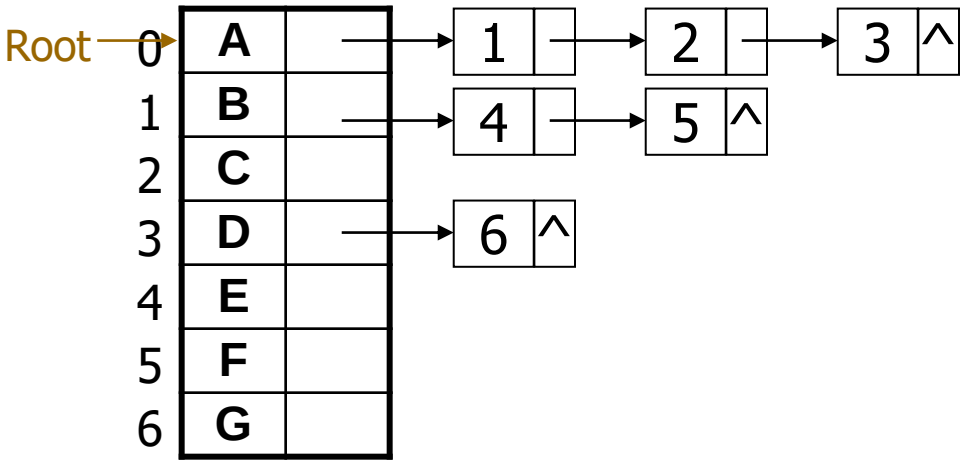
data	$child_1$	$child_2$	$child_3$	.....	$child_d$

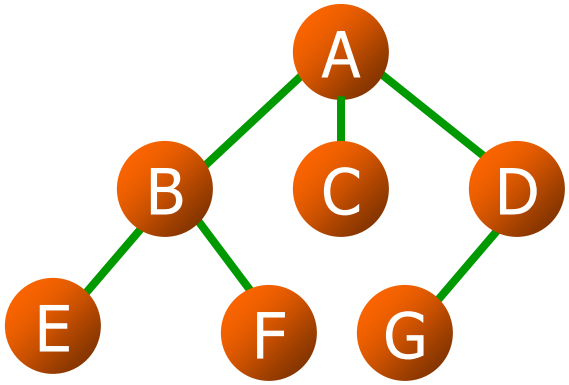


6.4 树与森林

一. 树的存储结构

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



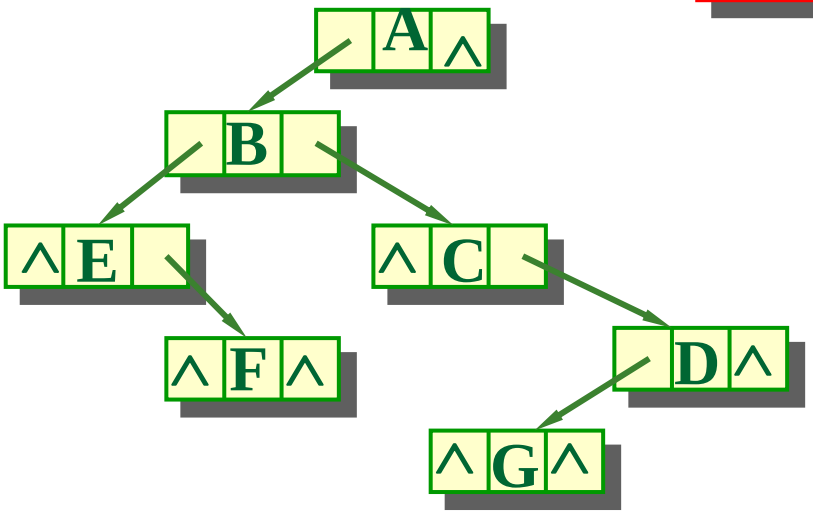


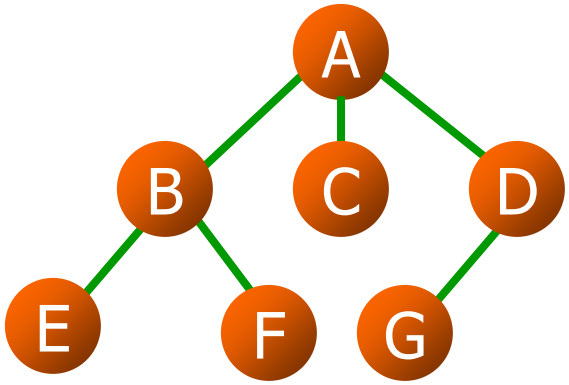
6.4 树与森林

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

data	firstChild	nextSibling





6.4 树与森林

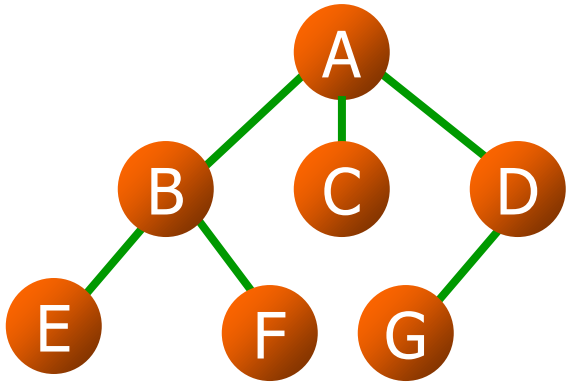
二. 树、二叉树、森林的关系

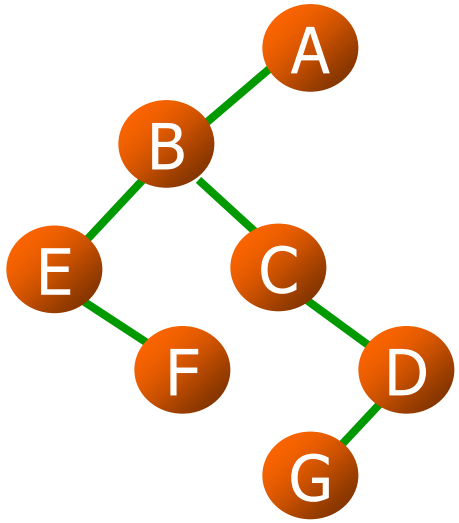
- 树与二叉树的关系
  - 树与二叉树都可以采用二叉链表作存储结构
  - 任意给定一棵树，可以找到一个唯一的二叉树(没有右子树

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

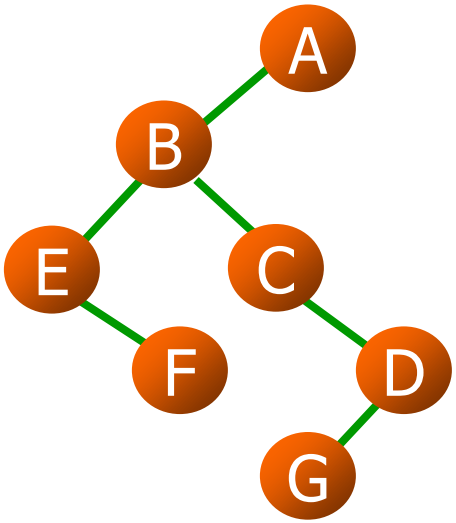


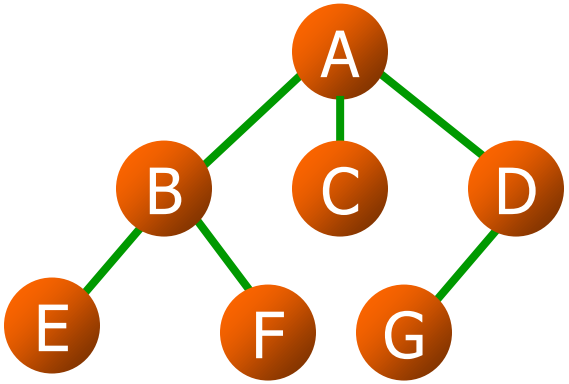


6.4 树与森林

二. 树、二叉树、森林的关系

- 二叉树转树
  - 左指针做孩子，右指针做兄弟

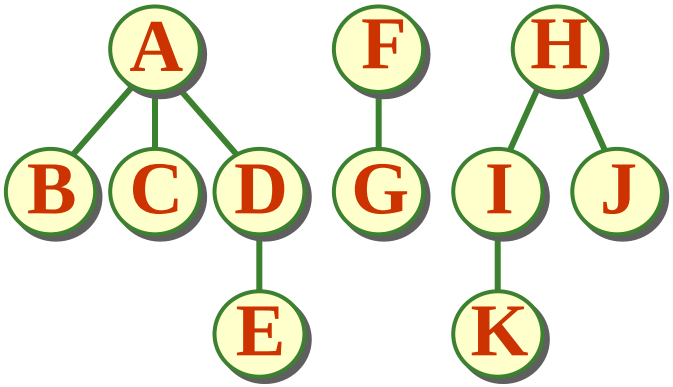




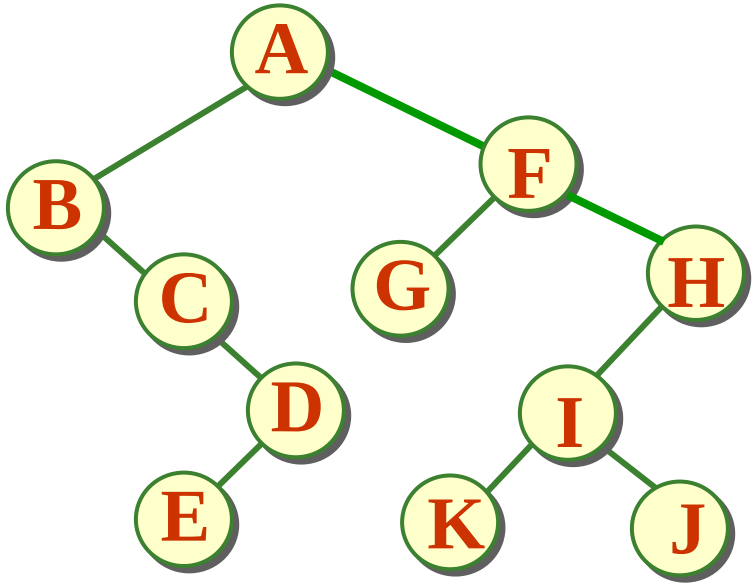
6.4 树与森林

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

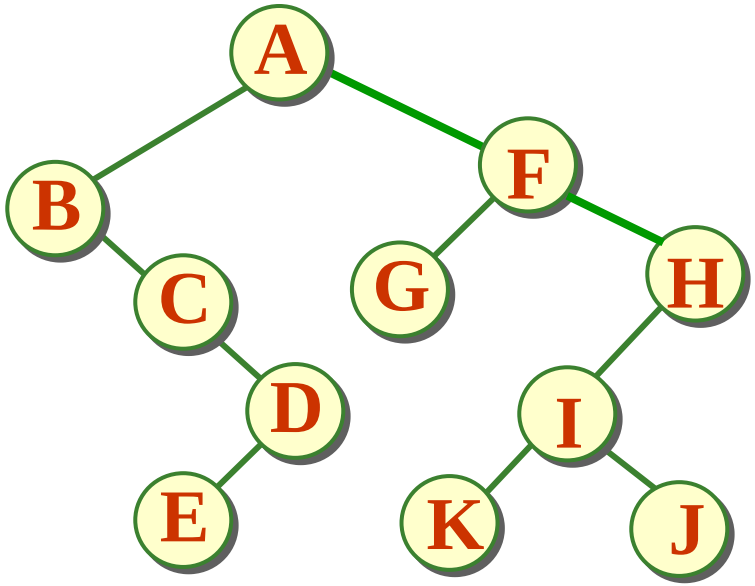


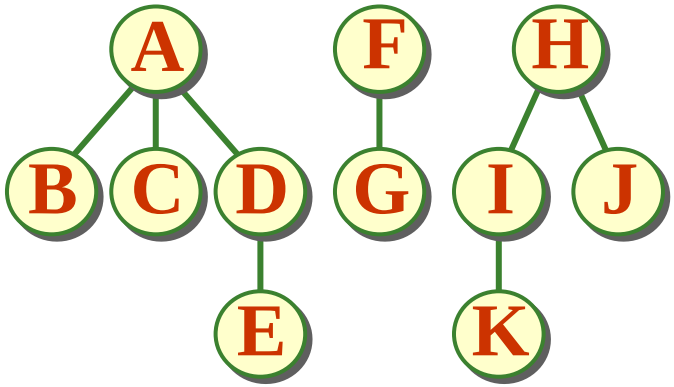
对应的二叉树

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：
  - 先根（次序）遍历
  - 后根（次序）遍历

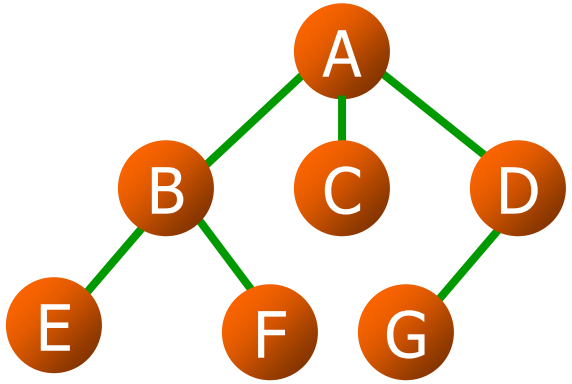
6.4 树与森林

三. 树的遍历

- 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

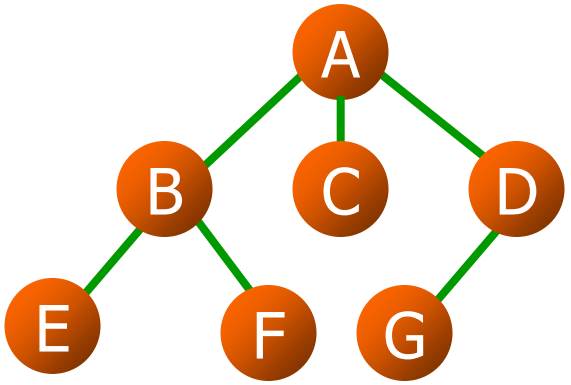


输出结果：ABEFCDG

- 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

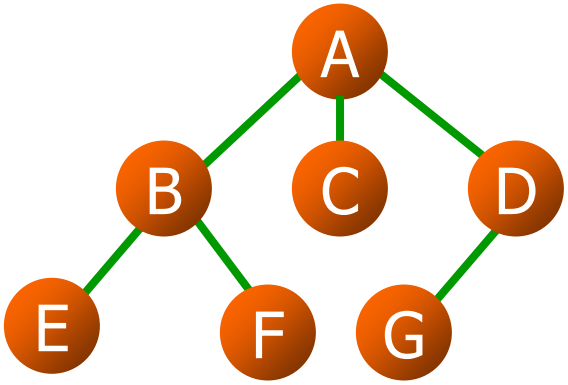
6.4 树与森林

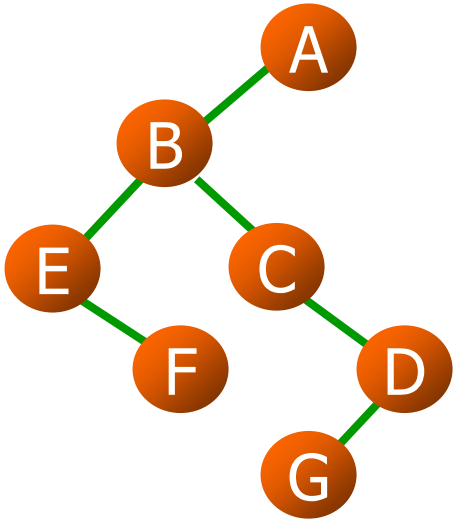
三. 树的遍历

- 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根遍历完全相同
- 树的后根遍历，与树对应的二叉树的中根遍历完全相同





先根遍历结果：ABEFCDG，后根遍历结果：EFBCGDA