树（Tree）

树是n（n ⩾ 0）个节点的有限集。当n = 0时，称为空树。在任意一棵非空树中应满足：

1）有且仅有一个特定的称为根的结点。

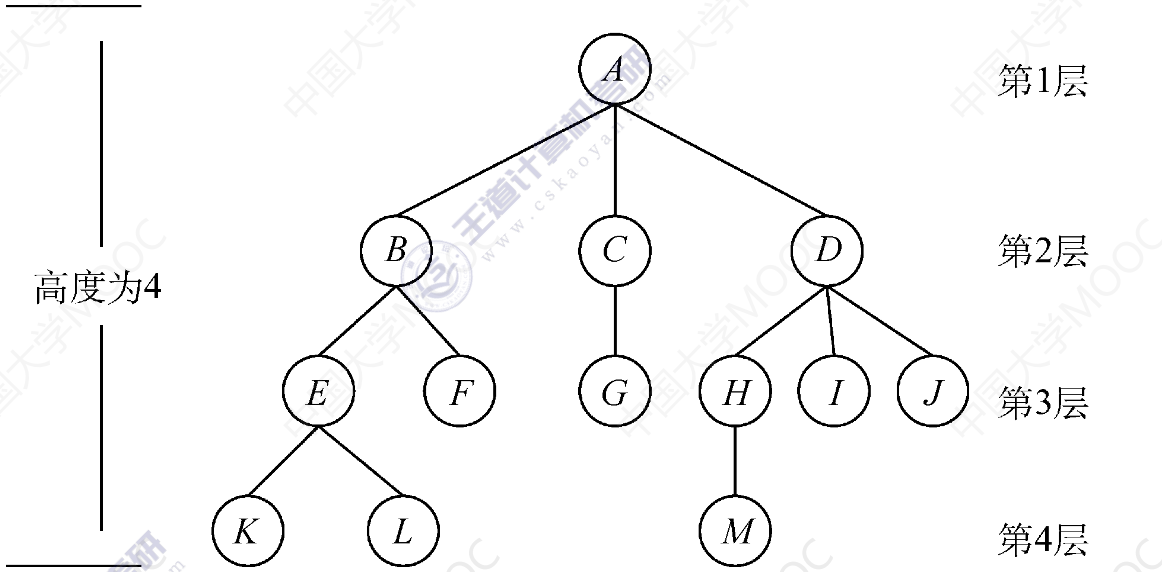
2）当n > 1时，其余节点可分为m（m > 0）个互不相交的有限集T1, T2,…, Tm，其中每个集合本身又是一棵树，并且称为根的子树。

树作为一种逻辑结构，同时也是一种分层结构，具有以下两个特点：

1）树的根结点没有前驱，除根结点外的所有结点有且只有一个前驱。

2）树中所有结点可以有零个或多个后继。

树的结构



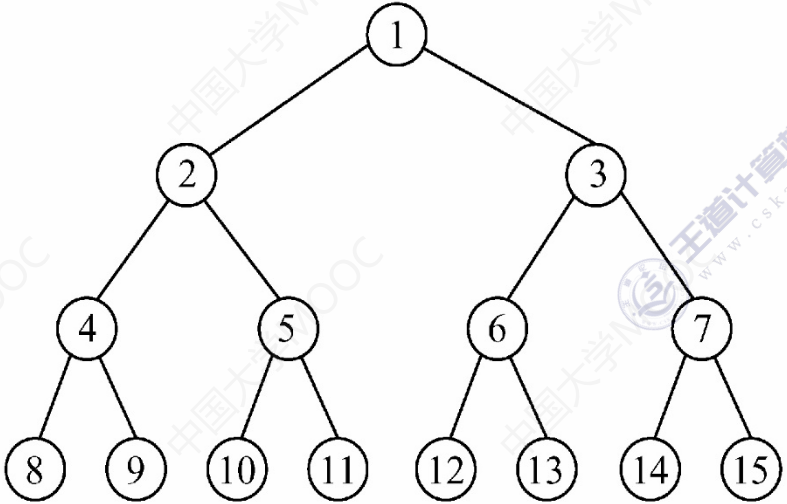
二叉树

二叉树是另一种树形结构，其特点是每个结点至多只有两棵子树（即二叉树中不存在度大于2的结点），并且二叉树的子树有左右之分，其次序不能任意颠倒。

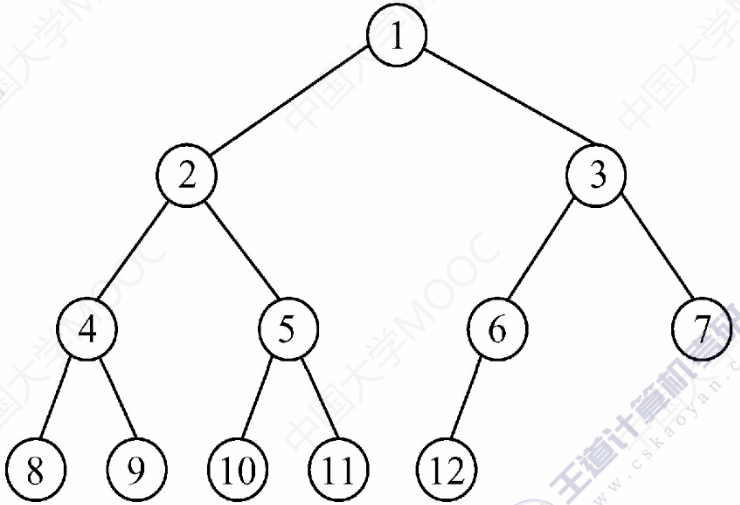
与树相似，二叉树也以递归的形式定义。二叉树是n（$n \geq 0$）个结点的有限集合：

① 或者为空二叉树，即$n = 0$。

② 或者由一个根结点和两个互不相交的被称为根的左子树和右子树组成。左子树和右子树又分别是一棵二叉树。

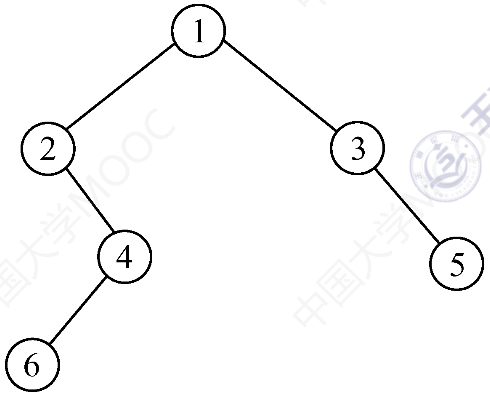


(a) 满二叉树



(b) 完全二叉树

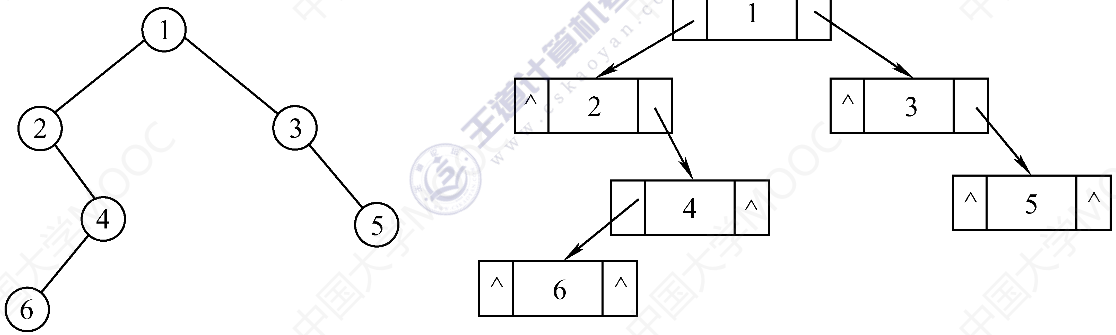
二叉树的顺序存储



1	2	3	0	4	0	5	0	0	6	0

顺序存储的实战我们到堆排时讲

二叉树的链式存储



树节点数据结构

```
typedef char BiElemType;
typedef struct BiTNode{
    BiElemType c;//c就是书籍上的data
    struct BiTNode *lchild;
    struct BiTNode *rchild;
}BiTNode,*BiTree;
```

先画图后
实战代码

依次是层次建树，先序遍历，中序遍历，后续遍历，层序遍历

线索二叉树

```
typedef char ElemType;
typedef struct ThreadNode{
        ElemType data;
        struct ThreadNode *lchild,*rchild;
        int ltag,rtag;//相对于二叉树多的
}ThreadNode,*ThreadTree;
```

线索二叉树

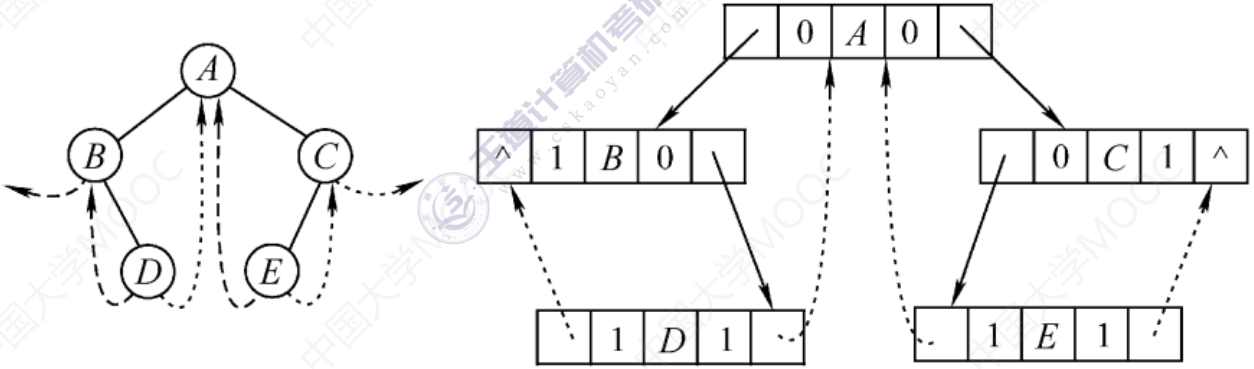
$$ltag\begin{cases}0, & lchild域指示结点的左孩子\\ 1, & lchild域指示结点的前驱\end{cases}$$

$$rtag\begin{cases}0, & rchild域指示结点的右孩子\\ 1, & rchild域指示结点的后继\end{cases}$$

线索二叉树

以中序线索二叉树的建立为例。附设指针pre指向刚刚访问过的结点，指针p指向正在访问的结点，即pre指向p的前驱。在中序遍历的过程中，检查p的左指针是否为空，若为空就将它指向pre；检查pre的右指针是否为空，若为空就将它指向p

线索二叉树



方法，先把树画出来，然后再线索化

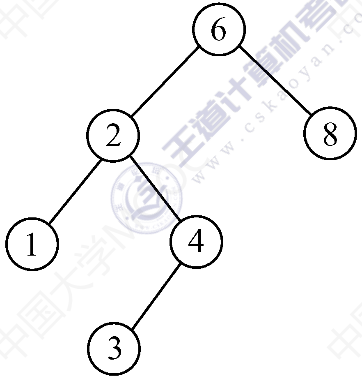
二叉排序树

二叉排序树

二叉排序树（也称二叉查找树）或者是一棵空树，或者是具有下列特性的二叉树：

1）若左子树非空，则左子树上所有结点的值均小于根结点的值。
2）若右子树非空，则右子树上所有结点的值均大于根结点的值。
3）左、右子树也分别是一棵二叉排序树。

二叉排序树



通过网站https://www.cs.usfca.edu/~galles/visualization/Algorithms.html来演示