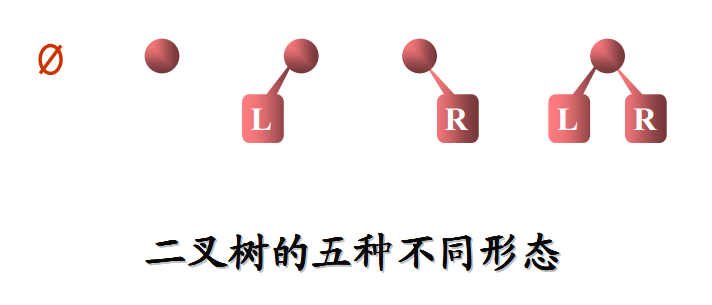
一.二叉树定义

二叉树是由n(n>=0)个结点组成的有限集合，该集合或者为空，或者由一个根结点加上两棵分别为

左子树和右子树的，互不相交的二叉树组成



二.二叉树的特殊类型

1.满二叉树

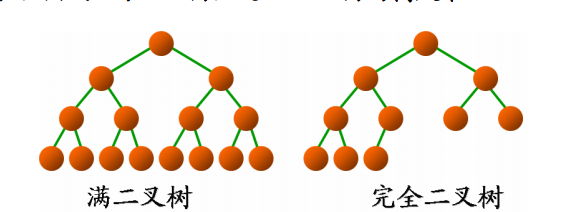
如果二叉树中所有分支结点的度数都为2，且叶子结点都在同一层次上，则称这类二叉树为

满二叉树

2.完全二叉树

如果一棵具有n个结点的高度为k的二叉树，它的每一个结点都与高度为k的满二叉树中编号

为1-n结点一一对应，则称这棵二叉树为完全二叉树（从上到下从左到右编号）



完全二叉树的性质

叶结点仅出现在最下面两层

最下层的叶结点一定出现在左边

倒数第二层的叶结点一定出现在右边

完全二叉树中度为1的结点只有左孩子

同样结点数的二叉树，完全二叉树的高度最小

三.二叉树的深层性质

1.在二叉树的第i层最多有2^(i-1)个结点

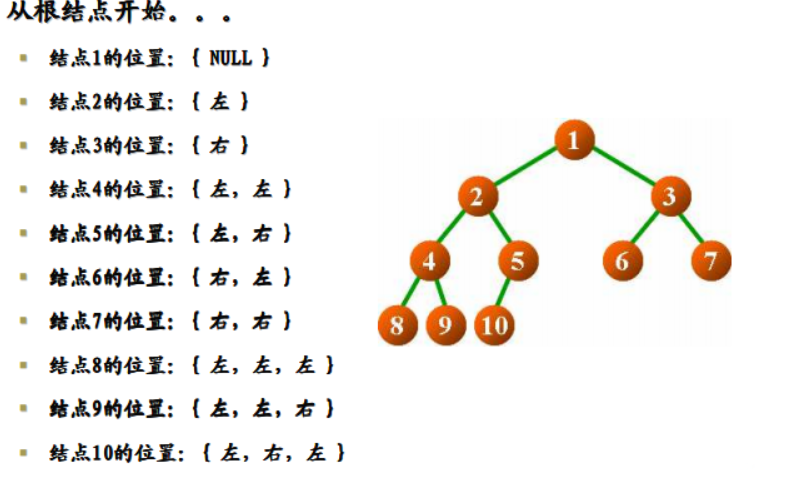
2.深度为k的二叉树最多有2^(k-1)个结点

3.任何一棵二叉树，如果其叶结点有n0个，度为2的非叶结点有n2个，则有n0 = n2+1

4.具有n个结点的完全二叉树的高度为

四.二叉树的实现

1指路法



通过bit位来指路，如果bit位为0，则左转，如果bit位为1，则右转

#define BT\_LEFT 0;

#define BT\_RIGHT 1；

typedef unsigned long long BTPos;//指定寻找路径

2.二叉树的存储结构

typedef struct \_tag\_BTreeNode BTreeNode;

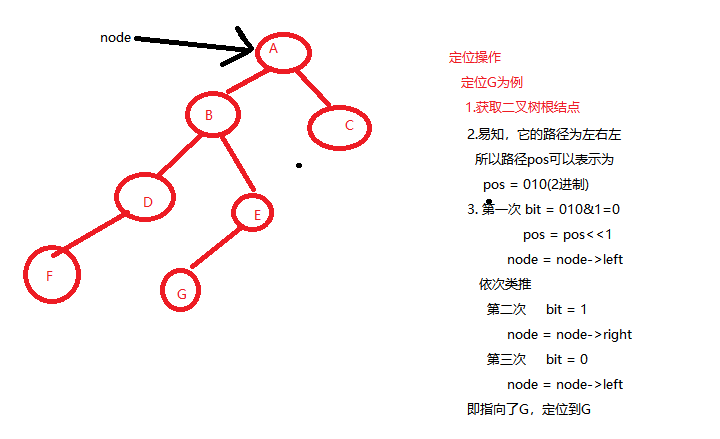
struct \_tag\_BTreeNode{

BTreeNode\* left;

BTreeNode\* right;

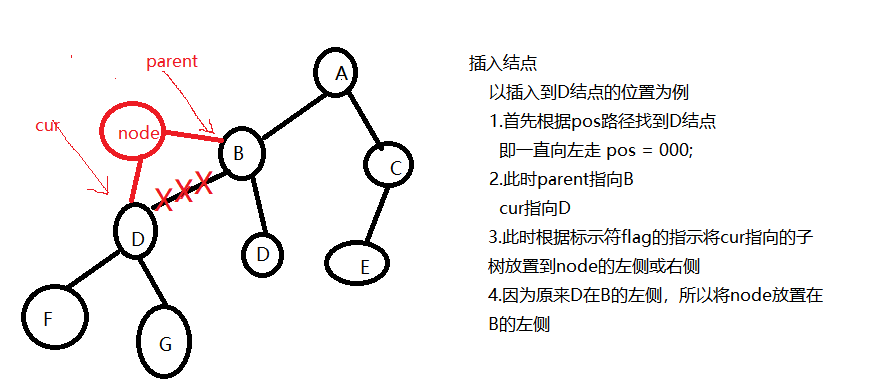
}

3.二叉树的定位操作



利用二进制中0和1分别表示left和right

4.插入操作



五.遍历二叉树

单链表的遍历是指从第一个结点开始(下标为0的结点)，按照某种次序访问每一个结点

二叉树的遍历是指从根结点开始，按照某种次序依次访问二叉树中所有结点

1.前序遍历（根左右）

若二叉树为空

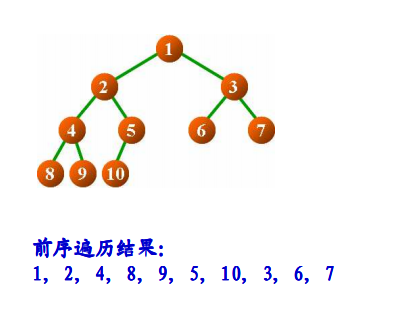
空操作返回

若二叉树不为空：

1.访问根结点中的数据

2.前序遍历左子树

3.前序遍历右子树



2.中序遍历

若二叉树为空

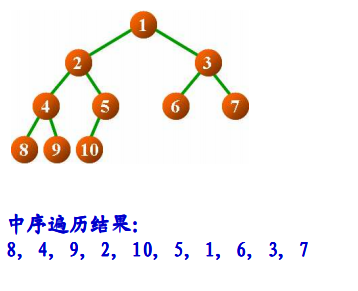
空操作返回

若二叉树不为空

1.中序遍历左子树

2.访问根结点中的数据

3.中序遍历右子树



3.后序遍历

若二叉树为空

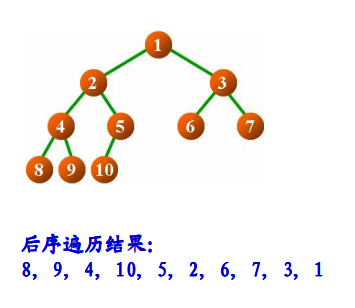
空操作返回

若二叉树不为空

1.后序遍历左子树

2.后序遍历右子树

3.访问根结点中的数据



4.层次遍历（借助队列来实现）

若二叉树为空

空操作返回

若二叉树不为空

访问根结点中的数据

访问第二层所有结点的数据

访问第三层所有结点的数据

………

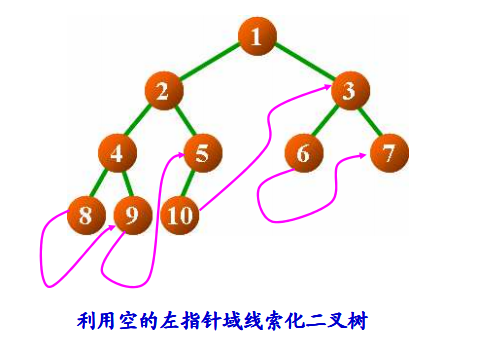
六 线索化二叉树

线索化二叉树指将二叉树中的结点进行逻辑意义上的"重排列",使其可以线性的方式访问每一个结点

二叉树线索化之后每个结点都有一个线性下标，通过线性下标可以快速访问结点

线索方法1；

利用结点中的空指针域，使其指向后继结点



算法思想

1.初始化位置指针

p = NULL;

2.前序遍历二叉树

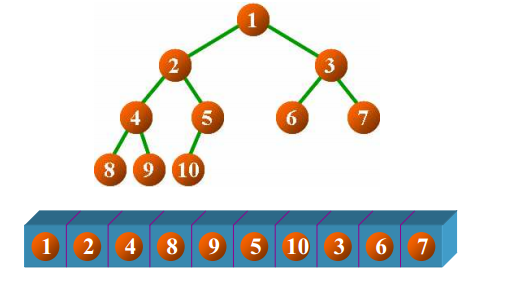
如p不为空，将p->left指向当前结点，并将p置为NULL

若当前结点的左子树为空时，将p指向当前结点

这种方式会破坏树的结构。

线索方法2：

利用线性表保存二叉树的遍历顺序



算法思想

初始化顺序表s1

前序遍历二叉树

遍历过程中将当前结点插入顺序表s1

这种方法不会破坏树的结构，不需要线索化时销毁链表即可

链表线索化方法可以很容易地以任何一种遍历顺序对二叉树进行线索化

霍夫曼树

1.给定n个数值{v1,v2,…,vn}

2.根据这个n个数值构造二叉树集合F

F={T1，T2，……,Tn}

3.在F中选取两棵根节点的值最小的树作为左右子树构造一棵新的二叉树

这棵二叉树的根节点中的值作为左右子树根节点中的值之和

4.在F中删除这两棵子树，并将构造的新的二叉树加入F中

5.重复3和4，直到F中只剩下一个树为止，这棵树就是霍夫曼树