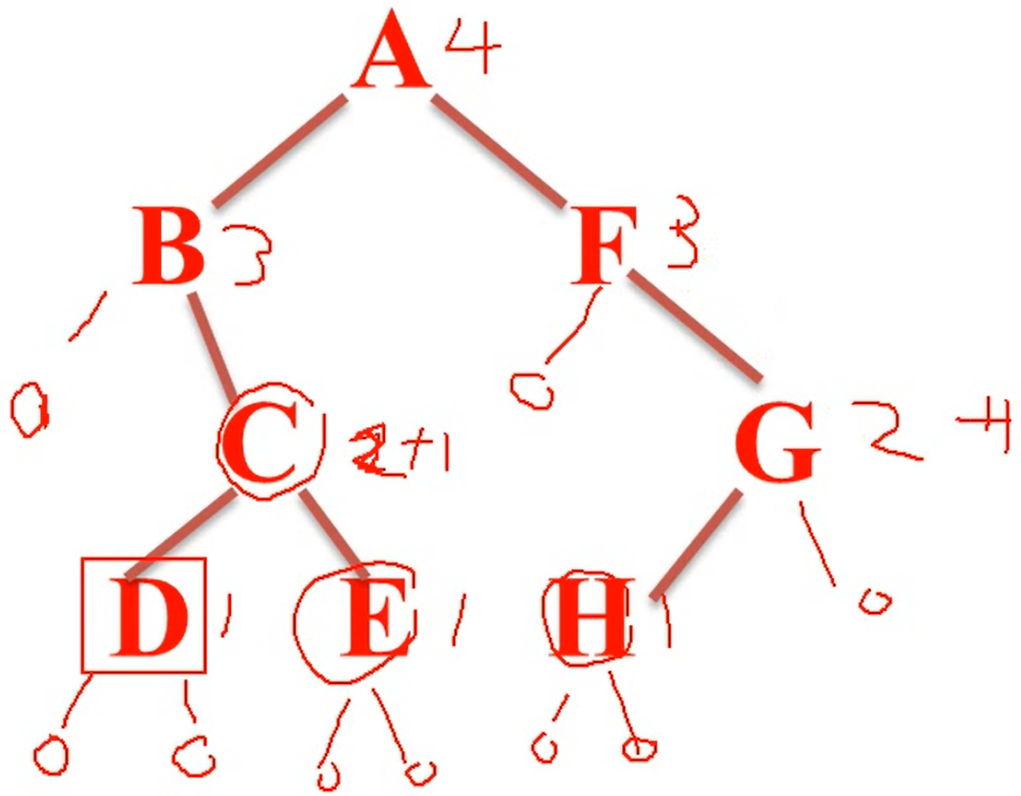
- **二叉树性质**
  - 性质 1: 在二叉树的第 i 层上至多有 2^{i-1} 个结点（i>0）
  - 性质 2: 深度为 k 的二叉树至多有 2^k-1 个结点（k>0）
  - 性质 3: 对于任何一棵二叉树，若度为 2 的结点数有 n2 个，则叶子数（n0）必定为 n2+1 （即 n0=n2+1）
  - 概念解释：
    - **满二叉树**

      一棵深度为 k 且有 2^k-1 个结点的二叉树。

~~~~~~~~~~~~~~将每一个节点看作子树

那么该子树的高度

high=l_h > r_h ? l_h+1 : r_h+1;


~~~~~~~~~~~~~~

```
int getTreeHeight(struct BinaryNode * root)
{
    if (root == NULL)
    {
        return 0;
    }

    //求出左子树的高度
    int lHeight = getTreeHeight(root->lChild);

    int rHeight = getTreeHeight(root->rChild);

    //取左子树 和 右子树中最大值 +1
    int height = lHeight > rHeight ? lHeight + 1 : rHeight + 1;


    return height;

}
```

二叉树的非递归遍历

需要用到栈，核心思想为：

首先给所有节点加一个标志位。默认为假

```
/*
    1、将根节点 压入栈中
    2、只要 栈size> 0  执行循环
        2.1 拿出栈顶元素
        2.2 如果栈顶元素的标志位 真    直接输出  执行下一次循环
        2.3 如果不是真 该flag的标志位真
        2.4 将  右子节点  和 左子节点  和 根 入栈
        2.5 执行下一次循环
*/
```

详细代码:

```c
void nonRecursion(struct BinaryNode * root)
{
    //初始栈
    seqStack myStack = init_SeqStack();

    push_SeqStack(myStack, root);


    while (size_SeqStack(myStack)>0)
    {
        //获取栈顶元素
        struct BinaryNode * topNode = top_SeqStack(myStack);

        //弹出栈顶
        pop_SeqStack(myStack);
```

```c
        //如果栈顶元素的标志位 真    直接输出  执行下一次循环
        if (topNode->flag == 1)
        {
            printf("%c ", topNode->ch);
            continue;
        }
```

```c
        //如果不是真 该flag的标志位真
        topNode->flag = 1;
        //将  右子节点  和 左子节点  和 根 入栈
        if (topNode->rChild != NULL)
        {
            push_SeqStack(myStack, topNode->rChild);
        }

        if (topNode->lChild != NULL)
        {
            push_SeqStack(myStack, topNode->lChild);
        }

        push_SeqStack(myStack,topNode);
    }

    //栈销毁掉
    destroy_SeqStack(myStack);
    myStack = NULL;
}
```