二叉树结构

非洲大草原上的树(很有特点)：

# 二叉树的定义

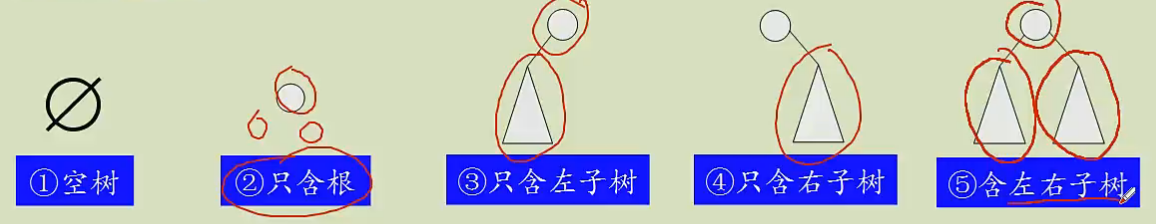
## 二叉树的递归定义

二叉树是有限的节点集合—递归定义:

这个集合**或者是空**

或者**由一个根节点和两颗互不相交的左子树和右子树的二叉树组成**。

## 五种基本形态



# 二叉树的性质

## 性质1：非空二叉树上叶节点数等于双分支节点数加1，即n0 = n2+1。

证明：

思路：根据两种求节点总数的方式推导出n0=n2+1；一是根据节点种类；二是根据节点总数=分支总数+根节点。

令二叉树的叶节点数为n0，单分支节点数为n1，双分支节点数为n2。

二叉树中总共就这三种节点，则根据节点种类计算总节点数为n=n0+n1+n2；

除根节点外，一个节点对应一个分支，则通过分支计算总节点数为n=n1+2\*n2+1；

则n0+n1+n2=n1+2\*n2+1,消项后有**n0 = n2+1**。

## 性质2：非空二叉树上第i层至多有2i-1个节点。

证明：根据树的性质2可以推出。(很好理解)

度为2的树中第i层(i>0)上至多有mi-1个节点。对于二叉树度m=2，则有2i-1。

## 性质3：高度为h的二叉树至多有2h-1个节点(h>0)。

证明：由树的性质3可推出：高度为h的m次树至多有(mh-1)/(m-1)个节点。

二叉树m=2，则为2h-1。

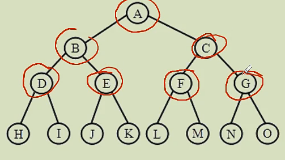
理解：就是一个等比数列求和：首项为1，公比为2的前h项和，根据等比数列求和公式得2h-1。()。

# 特殊的二叉树

## 满二叉树

### 定义1：在一颗二叉树中，如果所有分支节点都有左孩子节点和右孩子节点，并且叶节点都集中在二叉树的最下一层，这样的二叉树称为满二叉树。

### 定义2:一颗高为h，且有2h-1个节点的二叉树称为满二叉树。



### 特点

只有度为0和度为2的节点，没有度为1的节点。

叶子节点(度为0的节点)都在最下一层；

高度h确定，满二叉树就可以完全确定；节点总数确定，满二叉树也可以确定。

第i层的节点数目是固定的，为2i-1；高度为h的满二叉树的节点总数是固定的，为2h-1。

### 数量关系

节点总数：n = 2h-1；

高度：h = log2(n+1);

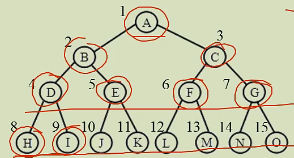
叶节点数：n0 = 2h-1；

度为1的节点数：n1=0；

度为2的节点数：n2 = n0-1=2h-1-1或者n2=n-n0-n1=2h-1-1。

### 节点编号

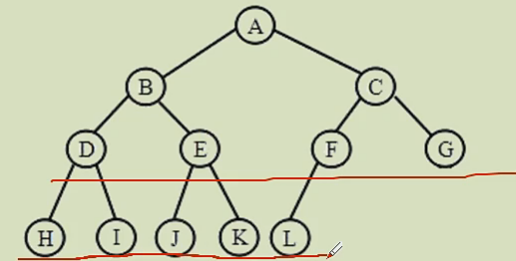
对满二叉树节点的连续编号:树根为1，按照层数从小到大、同一层从左到右的次序进行。



## 完全二叉树

### 定义：

若二叉树最多只有**最下面两层的节点的度数可以小于2**，并且最下面一层的叶节点都依次排列在该层最左边的位置上，这样的二叉树称为完全二叉树。



### 特点：

叶子节点只能出现在层次最大的两层上出现；

最大层次上的叶子节点都依次排列在该层的最左边的位置上；

有度为1的节点最多只能有1个(可以没有)，且该节点只有左孩子节点，没有右孩子节点；

**除最大层次之外，每层都达到最大节点数，为2i-1；最大层次小于等于2i-1**；

**去掉最后一层，就是一个满二叉树。**

### 数量关系

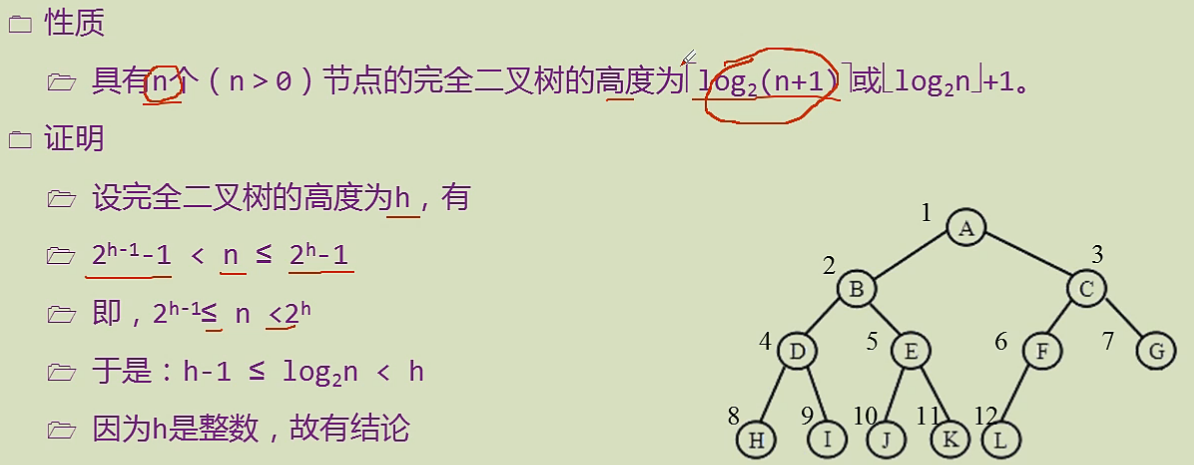
节点总数n一旦确定，完全二叉树的树形就确定了。

度为1的节点数目：n1=0或1；当n为偶数时，n1=1；当n为奇数时，n1=0。

性质：具有n个(n>0)节点的完全二叉树的高度为

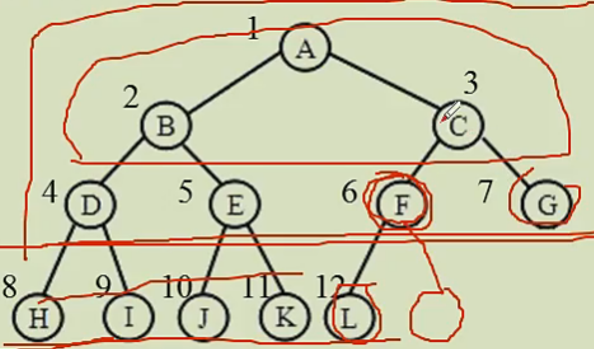


证明：



### 节点编号：

对满二叉树节点的连续编号:树根为1，按照层数从小到大、同一层次从左到右的次序进行。



按层编号后，一旦出现某节点(编号为k)为叶子节点或只有左孩子节点，则编号大于k的节点都是叶子节点。

### 完全二叉树与满二叉树的区别

完全二叉树的前n-1层就是一个满二叉树；

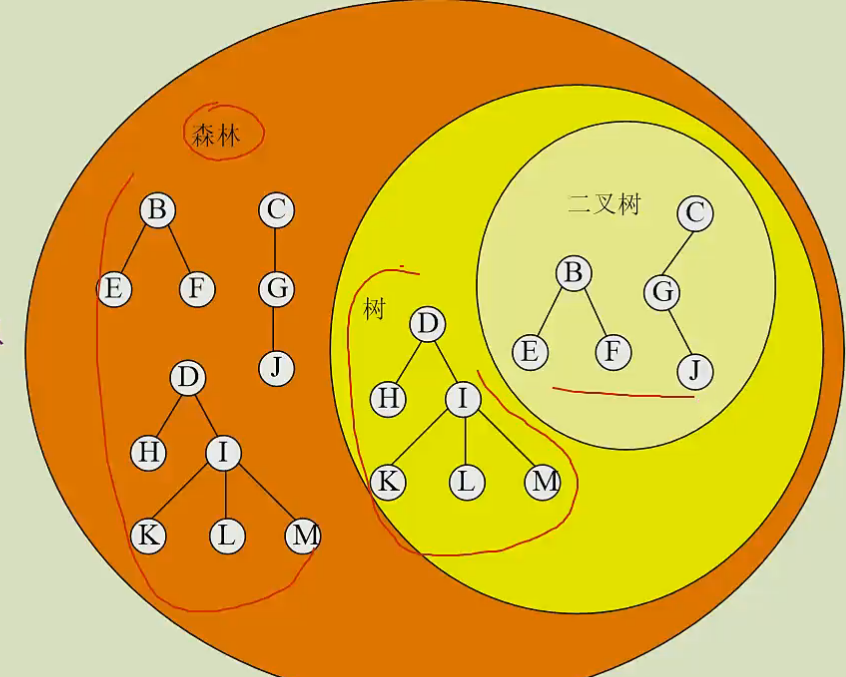
完全二叉树可以看做是满二叉树去掉最后一层的几个叶子节点；

满二叉树是一种特殊的完全二叉树。

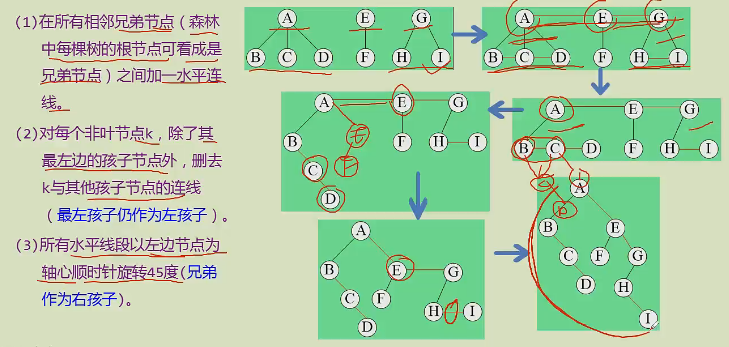
### 完全二叉树的编号性质



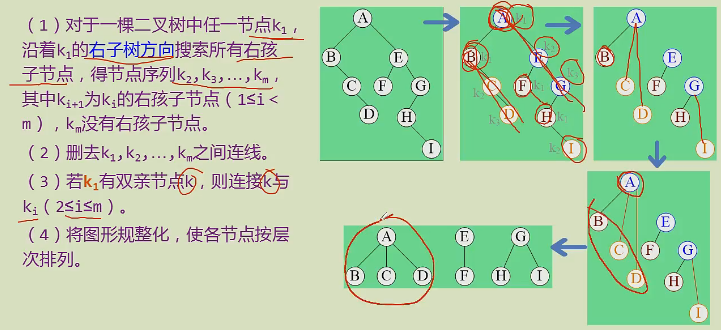
# 森林、树与二叉树之间的转换



## 将森林、树转换成二叉树：



## 将二叉树还原为森林、树：

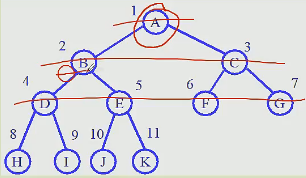


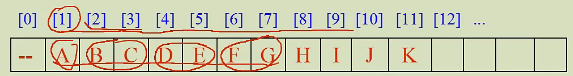
# 二叉树的存储结构

## 顺序存储结构

### 对于完全二叉树：

按照编号次序存储节点：对二叉树中的每个节点进行编号，其编号从小到大的顺序就是节点在连续存储单元中的先后次序。



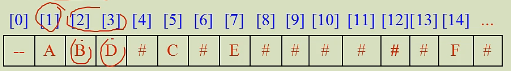


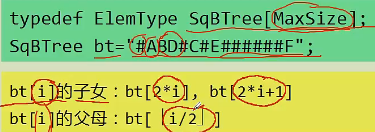
完全二叉树的编号(下标)的关系：

**编号为i的节点的左孩子节点的编号为2i，右孩子节点的编号为2i+1；**

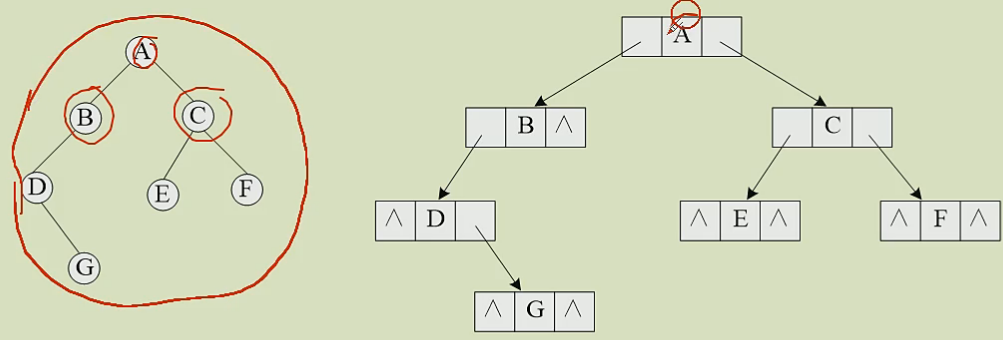
**除树根节点外，若一个节点的编号为i，则它的双亲节点的编号为**。

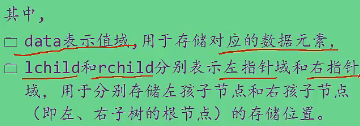
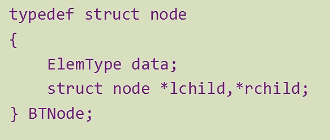
### 一般的二叉树：

先用空节点补全成为完全二叉树，再对节点进行编号，最后确定存储。

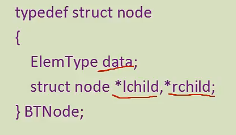
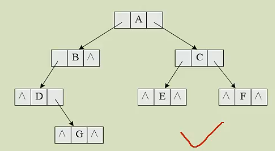


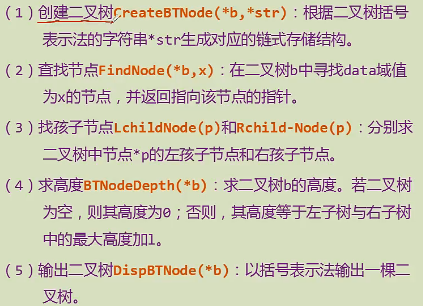
## 链式存储结构(重要)



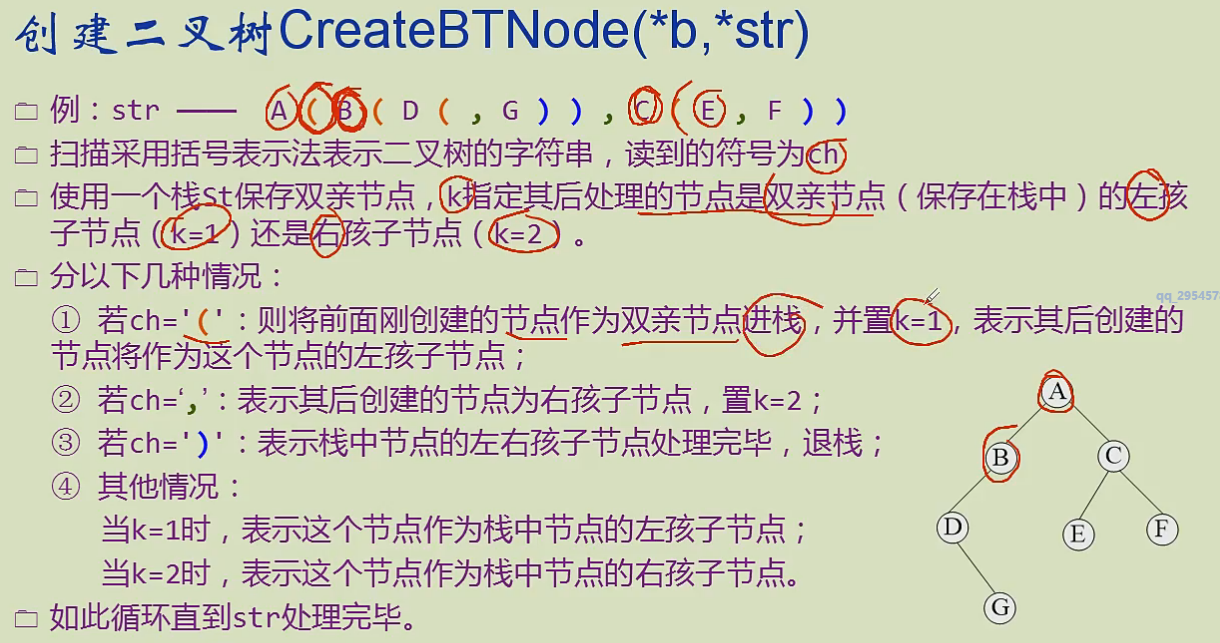


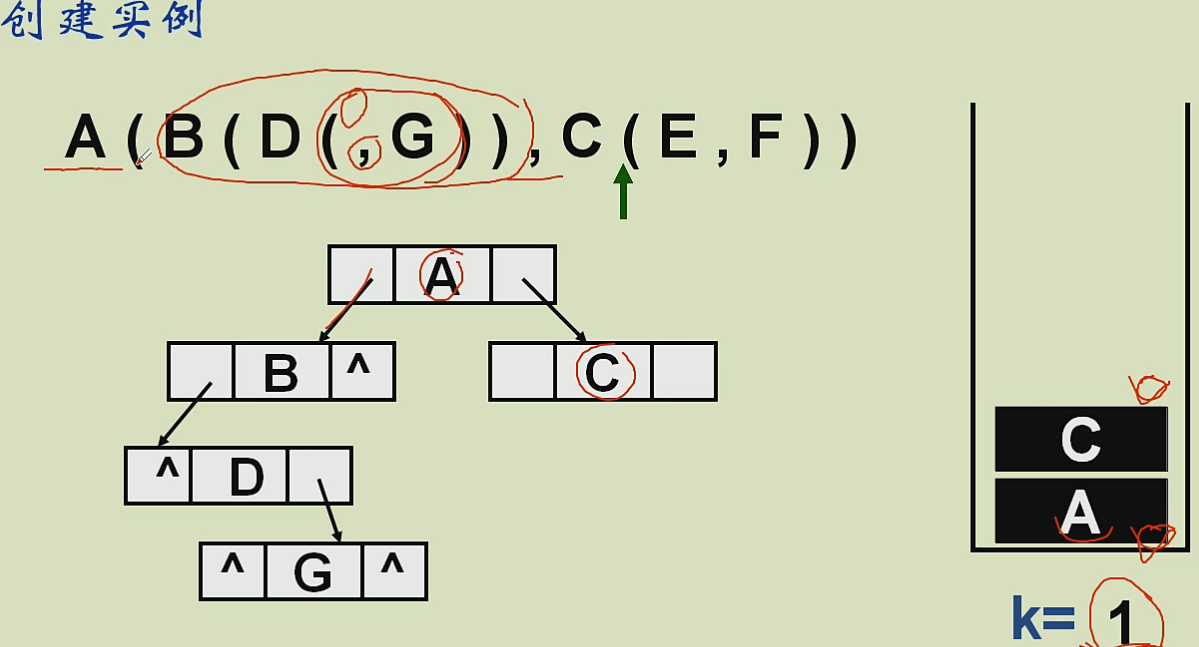
# 二叉树的基本运算(基于链式存储结构)

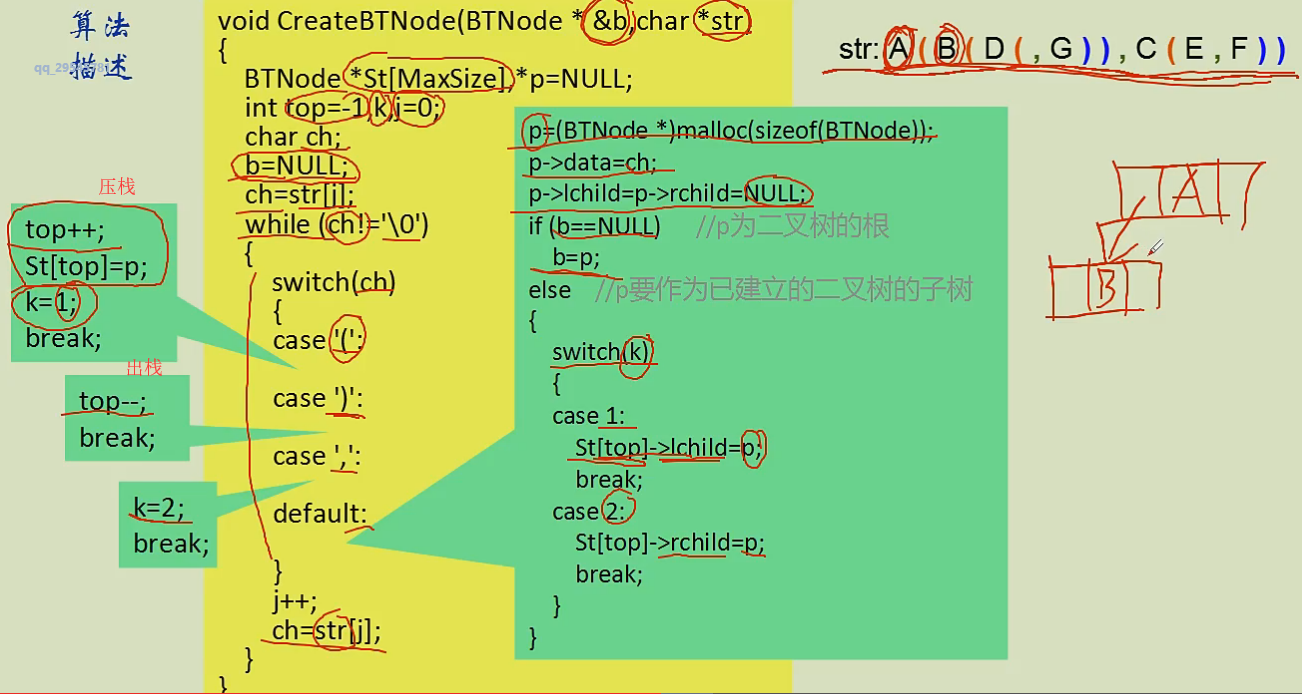




## 创建二叉树CreateBTNode







**栈的作用：存放双亲节点。为什么呢？举例说明,A有B和C两个左右孩子节点，但是在读取到B、C之间有很多其他节点，对于叶子节点来说，直接存储就可以，但是对于双亲节点来说，需要修改双亲节点的左右孩子节点的指针，因此必须保存相对应的双亲节点，恰好是先进后出，所以利用栈存储双亲节点的指针最合适。**

**压栈：当发现左括号的时候，需要压栈；**

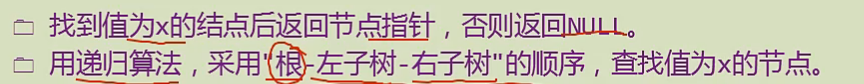
出栈：当发现右括号的时候，需要出栈。出栈的时候，只需要将指针减一就行，不需要获取具体的出栈元素。

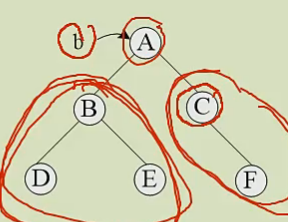
当整个树创建完毕后，栈中应该是空的。

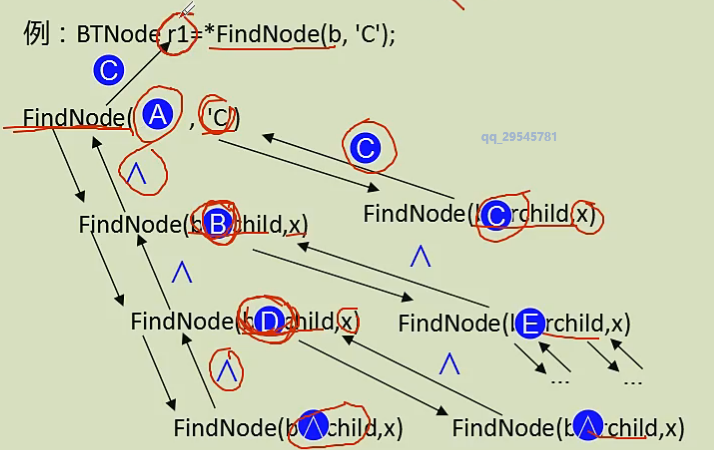
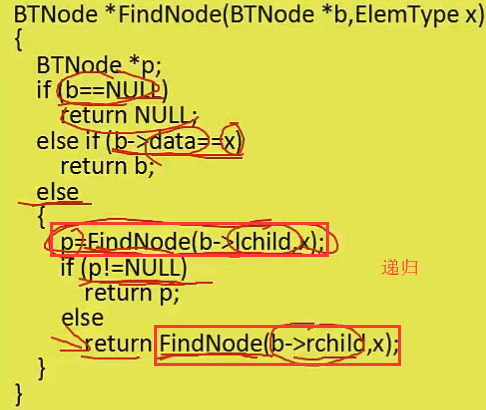
分配一个节点需要的存储空间。

## 查找节点FindNode(\*b,x)

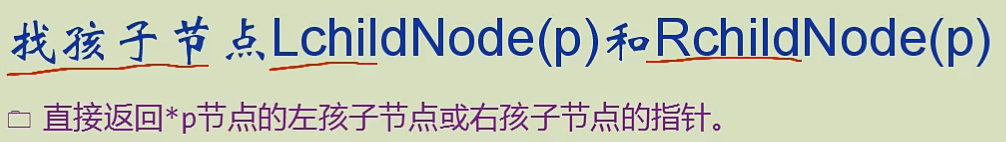




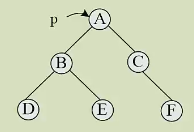
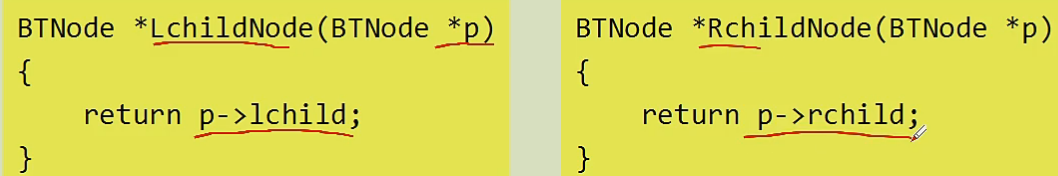




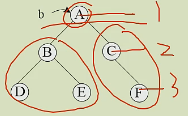
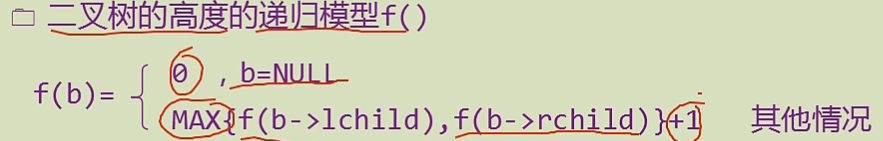
## 找孩子节点

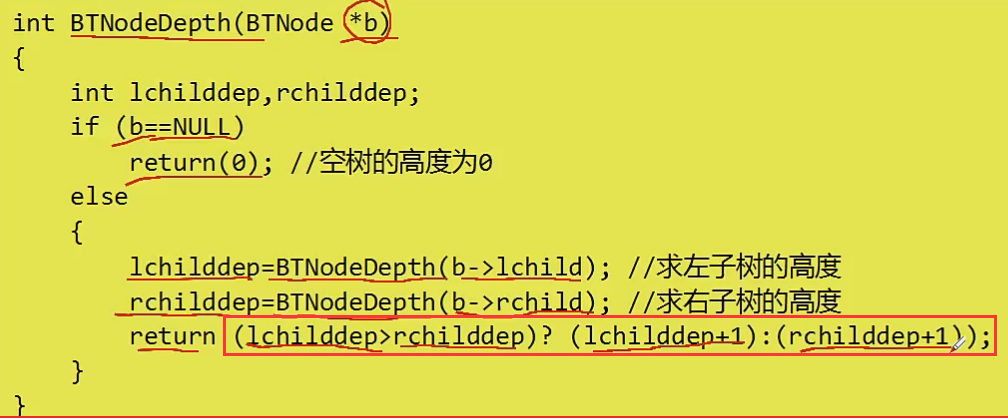


若无某个孩子节点，直接返回NULL。



## 求高度BTNodeDepth(\*b)





## 输出二叉树DispBTNode(\*b)

仍然是采用**递归的思想**进行处理：

