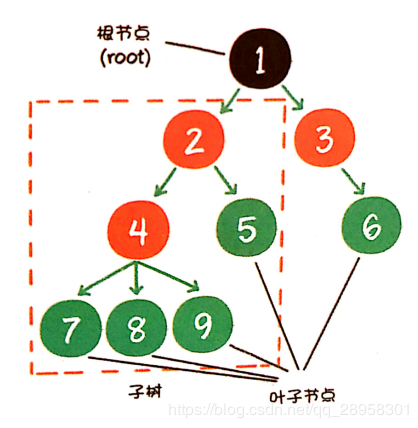
## 树

**n0=n2+1**

树是**n个节点的有限集。**当n=0时，称为空树。在任意一个非空树中，有如下特点：

* 有且仅有一个特定的称为根的节点；
* 当n>1时，其余节点可分为m个互不相交的有限集，每一个集合本身又是一个树，并称为根的子树。
* 树的最大层级数，称为树的高度或深度。

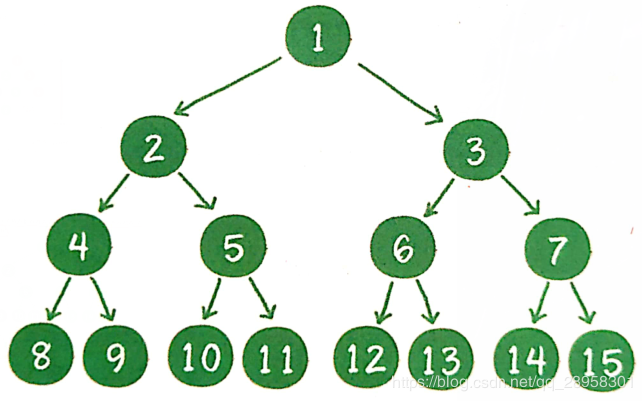


## 2. 二叉树

树的每个节点最多有2个孩子节点。

### 2.1 满二叉树

一个二叉树的所有非叶子节点都存在左右孩子，并**且所有叶子节点都在同一层级上。**



### 2.2 完全二叉树

对一个有n个节点的二叉树，按层级顺序编号，则所有节点的编号为1到n。要求这个树所有节点和同样深度的满二叉树的编号为从1到n的节点位置相同。**完全二叉树只需保证最后一个节点之前的节点都齐全即可。**

## 3. 二叉树的应用

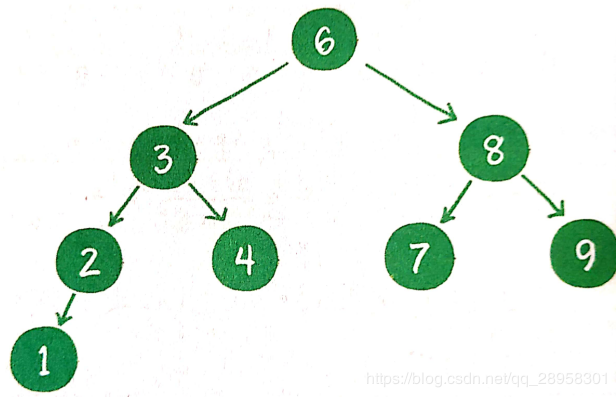
### 3.1 查找

二叉查找树在二叉树的基础上增加了以下几个条件：

如果左子树不为空，则左子树上所有节点的值均小于根节点的值；

如果右子树不为空，则右子树上所有节点的值均大于根节点的值；

左、右子树也都是二叉查找树。



对于一个节点分布相对均衡的二叉查找树来说，如果节点总数是n，那么搜索节点的时间复杂度都是O(logn)，和树的深度是一样的。

### 3.2 维持相对顺序（插入）

二叉查找树的特性保证了二叉树的有序性，因此还有另外一个名字：二叉排序树。

插入的过程中，可能会出现需要二叉树进行自平衡，例如下图的情况：

二叉树的自平衡的方式有很多种，如红黑树、AVL树、树堆等。

## 4. 二叉树的遍历

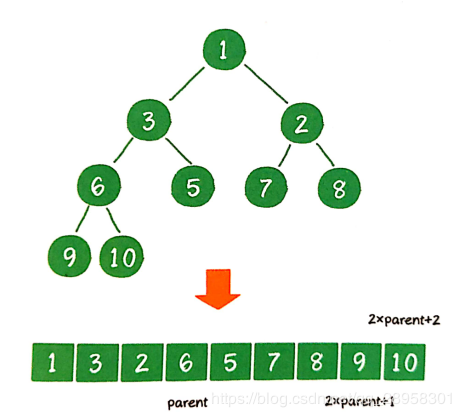
<https://blog.csdn.net/coder__666/article/details/80349039>

## 5. 二叉堆

二叉堆本质上是一种完全二叉树，分为2个类型：

* 最大堆：任何一个父节点的值，都大于或等于它左、右孩子节点的值；
* 最小堆：任何一个父节点的值，都小于或等于它左、右孩子节点的值。

二叉堆虽然是一个完全二叉树，**但它的存储方式并不是链式存储，而是顺序存储，**如



**假设父节点的下标是parent，那么它的左孩子的下标就是2 \* parent + 1，右孩子的下标就是2 \* parent + 2。**

二叉堆的3种操作（假设是最小堆）：

### 插入节点

插入节点是通过“上浮”操作完成的：当二叉堆插入节点时，插入位置是完全二叉树的最后一个位置，将该节点与它的父节点进行比较，如果该节点小于它的父节点，那么该与它的父节点交换位置，直到比较到堆顶位置。

### 删除节点

删除节点是通过“下沉”操作完成的：将要删除的节点看作是堆顶，只看该节点及它下面的部分。因为堆顶元素要进行删除，将最后一个节点元素替换堆顶元素，将替换后的元素与它的左、右子树进行比较，如果左、右孩子节点中最小的一个比该节点小，那么该节点“下沉”，直到叶子节点。

### 构建二叉堆

构建二叉堆就是把一个无序的完全二叉树调整为二叉堆，本质就是让所有非叶子节点依次“下沉”。

## 6.面试知识点

可唯一确定一棵二叉树：

给定一棵二叉树的后序和中序遍历序列。给定一棵二叉树的先序和中序遍历序列。

度：分叉

叶子结点总数=度数\*该度数对应的结点数+1