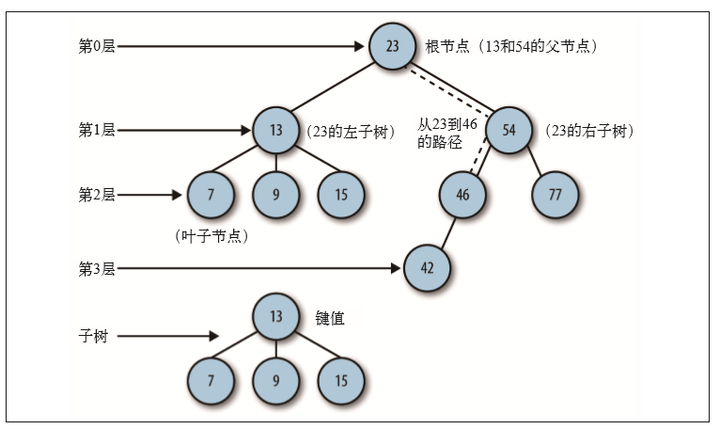
**什么是树？**

树由一组以边连接的节点组成，一棵树最上面的节点称为根节点，如果一个节点下面连接多个节点，那么该节点称为父节点，它下面的节点称为子 节点。一个节点可以有 0 个、1 个或多个子节点。没有任何子节点的节点称为叶子节点。



树可以分为几个层次，根节点是第 0 层，它的子节点是第 1 层，子节点的子节点是第 2 层，以此类推。树中任何一层的节点可以都可以看做是子树的根，该子树包含根节点的子节 点，子节点的子节点等。我们定义树的深度就是树的高度，我们可以通过数边的个数获取树的高度。最后，每个节点都有一个与之相关的值，该值有时被称为键（key）。

**什么是二叉树(Binary Tree)？**

二叉树，每个节点最多只能有两个节点，左子节点和右子节点。二叉树并不要求每个节点都有两个子节点，有的节点只有左子节点，有的节点只有右子节点.

**什么是满二叉树？**

叶子节点都在最底层，除了叶子节点，每个节点都有左右两个节点。

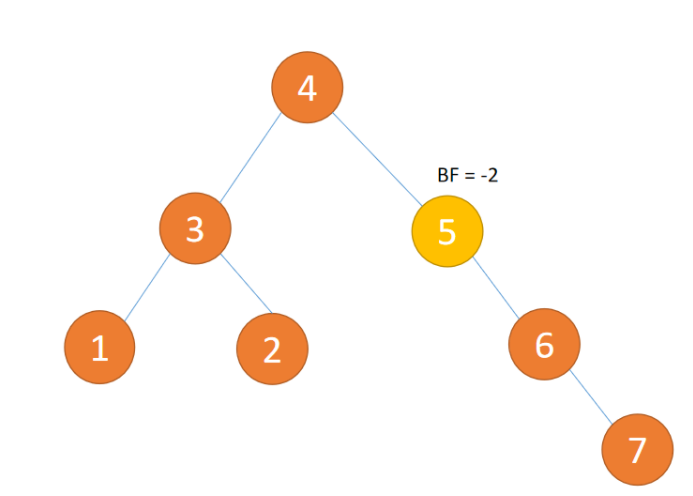
**什么是二分搜索树，二叉查找树(Binary Search Tree)？**

二分搜索树是基于二叉树实现的，二叉树存储元素的时候，一个节点的值永远大于他左子树所有节点的值，同理一个节点的值永远小于右子树所有节点的值。

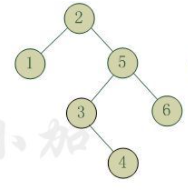
**什么是平衡二叉树？**

如果在二叉树中，任一节点的左右子树的高度差不超过一，则该二叉树为平衡二叉树。

例如：



上图不是平衡二叉树。因为对于节点5，其左子树没有只有子树根节点5深度为0。右子树以节点5作为子树根节点，其深度为2，所以高度差为2大于1.



上图不是平衡二叉树。因为对于节点2，其左子树的深度为1。.右子树以节点5作为子树根节点，其深度为3，所以高度差为2大于1.

**树的遍历方法：**

树主要有两种遍历风格。分为深度优先算法(Depth first search, DFS)以及广度优先算法(breath first search, BFS)。

**深度优先算法**尽可能的访问深度的节点然后进行回溯。

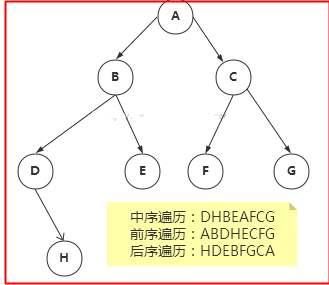
深度优先遍历风格主要包括以下三种实现方式：树的前序遍历(Pre-order Traversal)，中序遍历(In-Order Traversal)，后序遍历(Post-order Traversal)。其中中序遍历的比较流行的深度优先遍历。

前序遍历，中序遍历，后序遍历是以当前访问节点的角度来说。所以

前序遍历为current node->左子树->右子树, 对于刚开始时候当前节点为根节点时为root node->左子树->右子树

中序遍历为左子树->current node-> 右子树 .对于刚开始时候当前节点为根节点时为左子树->root node-> 右子树. 另外特殊的是， 对二分查找树进行中序遍历，将形成一个从小到大排序的序列。

后序遍历为左子树-> 右子树->current node. 对于刚开始时候当前节点为根节点时为左子树-> 右子树->root node.

例如：

**广度优先算法**将访问同一层的节点以后再访问下一深度的节点。

广度优先遍历风格实现方式为水平顺序遍历(level order traversal)

例如：