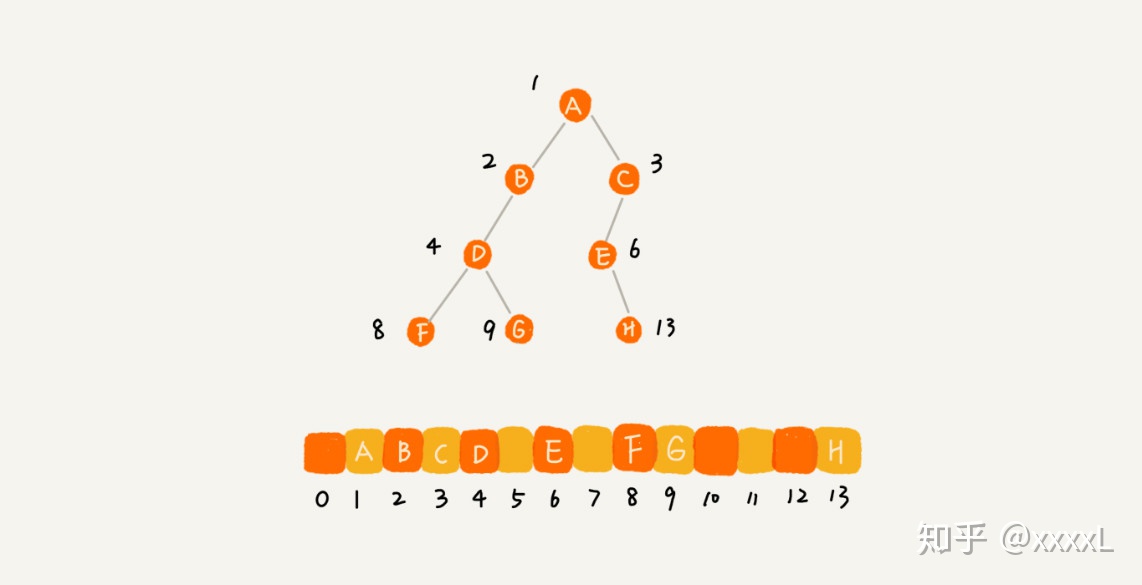
# 二叉树学习总结

相对数组/链表/散列表来说，树是一种非线性（型？）数据结构，特别的当每个节点分叉不多于两个时成为二叉树。相邻的两个节点上面叫父节点，下面叫子节点。

**二叉树的两种存储方法**

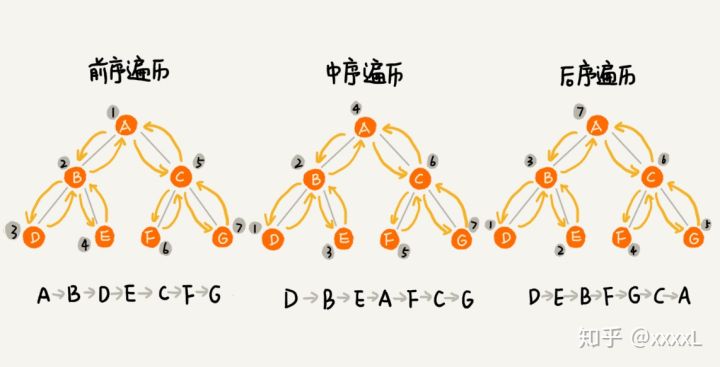
A.链式存储法（常用）：每个节点保存left，right两个指针指向子节点

B.顺序存储法：按层数从高到低，每一层从左到右存到数组中，如果有空则跳过



**二叉树的遍历**

* 前序遍历是指，对于树中的任意节点来说，先打印这个节点，然后再打印它的左子树，最后打印它的右子树。
* 中序遍历是指，对于树中的任意节点来说，先打印它的左子树，然后再打印它本身，最后打印它的右子树。
* 后序遍历是指，对于树中的任意节点来说，先打印它的左子树，然后再打印它的右子树，最后打印这个节点本身。



每个节点最多会被访问两次，所以遍历操作的时间复杂度，跟节点的个数 n 成正比，也就是说二叉树遍历的时间复杂度是 O(n)

**二叉查找树（Binary Search Tree）**

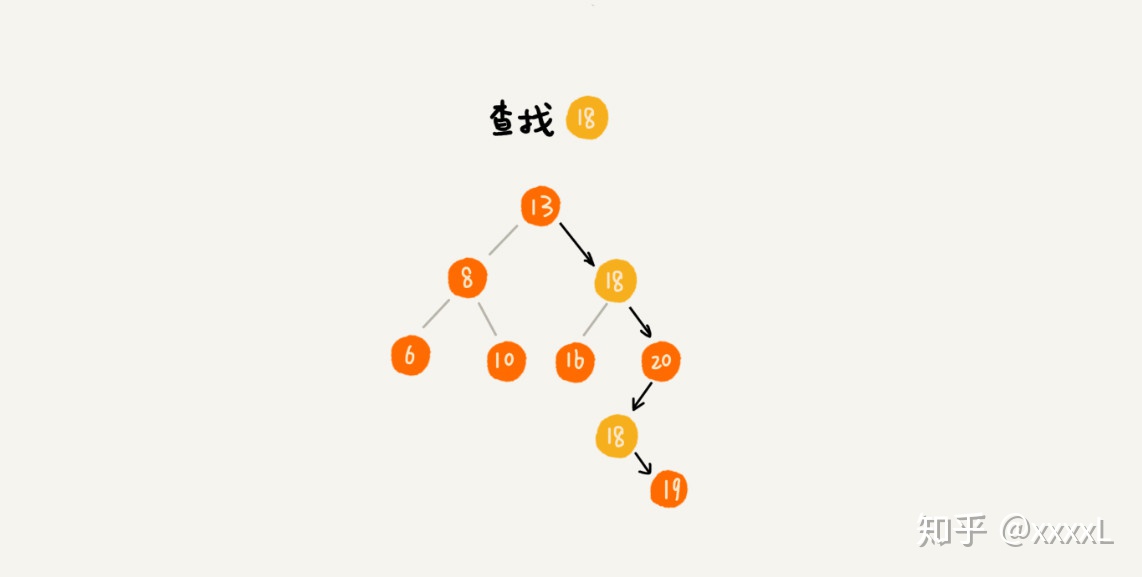
在树中的任意一个节点，其左子树中的每个节点的值，都要小于这个节点的值，而右子树节点的值都大于这个节点的值。

除了，增删查，二叉查找树中还可以支持快速地查找最大节点和最小节点、前驱节点和后继节点

**支持重复数据的二叉查找树**

第一种方法比较容易。二叉查找树中每一个节点不仅会存储一个数据，因此我们通过链表和支持动态扩容的数组等数据结构，把值相同的数据都存储在同一个节点上。

第二种方法比较不好理解，不过更加优雅。每个节点仍然只存储一个数据。在查找插入位置的过程中，如果碰到一个节点的值，与要插入数据的值相同，我们就将这个要插入的数据放到这个节点的右子树，也就是说，把这个新插入的数据当作大于这个节点的值来处理。



增删查需要追溯到叶子节点

**确定二叉树高度的两种思路**

第一种是深度优先思想的递归，分别求左右子树的高度。当前节点的高度就是左右子树中较大的那个+1；

第二种可以采用层次遍历的方式，每一层记录都记录下当前队列的长度，这个是队尾，每一层队头从0开始。然后每遍历一个元素，队头下标+1。直到队头下标等于队尾下标。这个时候表示当前层遍历完成。每一层刚开始遍历的时候，树的高度+1。最后队列为空，就能得到树的高度。