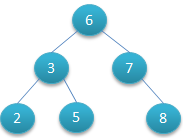
B-Tree是平衡多路查找树，B代表的是Balance，不是Binary。B+Tree是从最早的平衡二叉树演化来的。（需要了解一些前提知识：二叉查找树Binary Search Tree、平衡二叉树AVL树或者Balanced Binary Tree）。

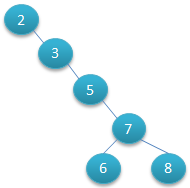
# Binary Search Tree

二叉查找树的性质是：左子树的键值小于根的键值，右子树的键值大于根的键值。



对该二叉树的节点进行查找发现深度为1的节点的查找次数为1，深度为2的查找次数为2，深度为n的节点的查找次数为n，因此其平均查找次数为 (1+2+2+3+3+3) / 6 = 2.3次。

但是二叉查找树可以任意构造，同样是2,3,5,6,7,8这6个数字也可以构造成下面的形式：

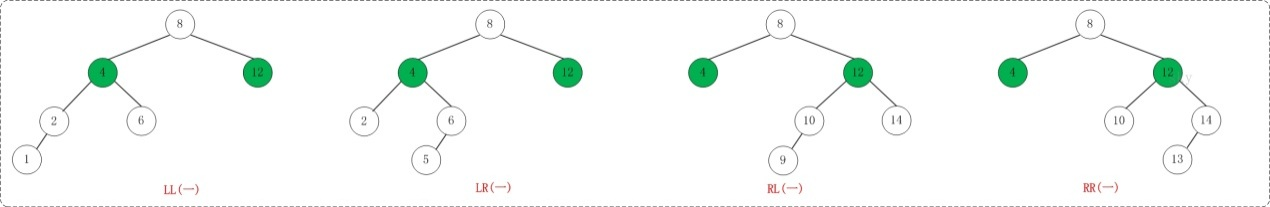


但是这棵二叉树的查询效率就低了。因此若想二叉树的查询效率尽可能高，需要这棵二叉树是平衡的，从而引出新的定义——平衡二叉树，或称AVL树。

# AVL Tree

平衡二叉树是在符合二叉查找树的条件下，还满足任何节点的两个子树的高度最大差为1。

在AVL树中插入或者删除节点可能会导致AVL树失去平衡，需要通过一次旋转让AVL树恢复平衡。分为LL，RR，LR和RL。



LL（根节点左孩子的左孩子出了问题）：解决办法分为三步

1. 根的左孩子作为根节点（4），即左孩子**右移**；
2. 左孩子的右孩子（6）作为根的左孩子。
3. 原来的根节点（8）**右移**作为新的根节点（4）的右孩子；

# 参考

https://www.cnblogs.com/vianzhang/p/7922426.html