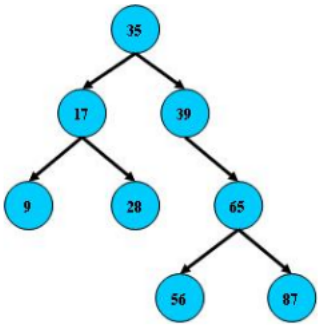
BST

即二叉搜索树，又叫二叉排序树。

1. 所有的非叶子节点至多拥有两个儿子。

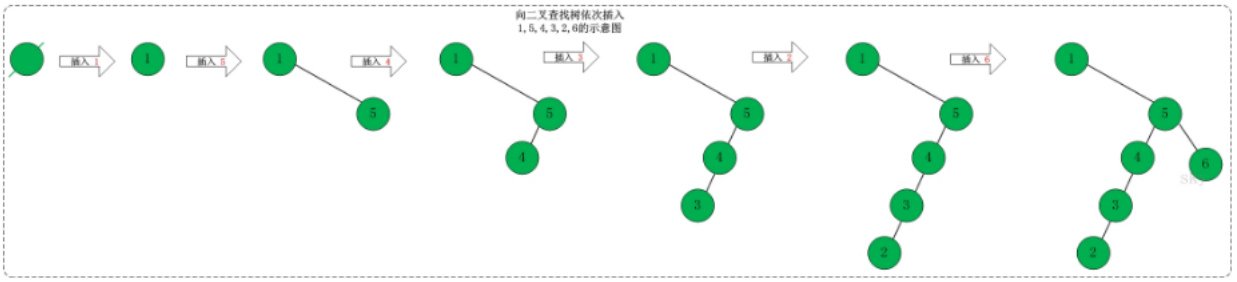
2. 所有节点存储一个关键字。

3. 非叶子节点的左指针指向小于其关键字的子树，右指针指向大于其关键字的子树、



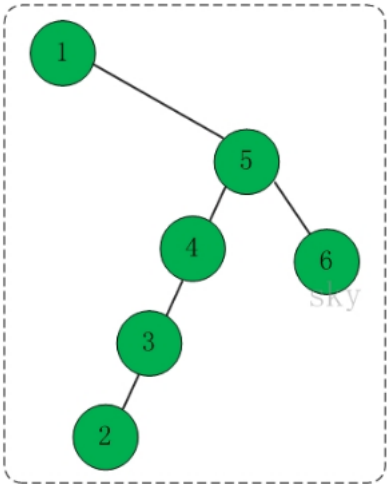
B树的搜索，从根节点开始，如果搜索的关键字与节点的关键字想的相等，那么就命中；否则，如果查询关键字比节点关键字要小，就进入左儿子；如果比节点关键字大，就进入右儿子了；如果左儿子和右儿子的指针为空，则报告找不到相应的关键字。如果B树的所有非叶子节点的左右子树的节点数目均保持差不多（平衡），那么B树的搜索性能逼近二分查找；但它比连续内存空间的二分查找的优点是，改变B树结构（插入与删除节点）不需要移动大段的内存数据，甚至通常是常数开销。

(02) 向二叉查找树中依次插入1,5,4,3,2,6 。如下图所示：



(03) 遍历和查找

插入1,5,4,3,2,6之后，得到的二叉查找树如下：



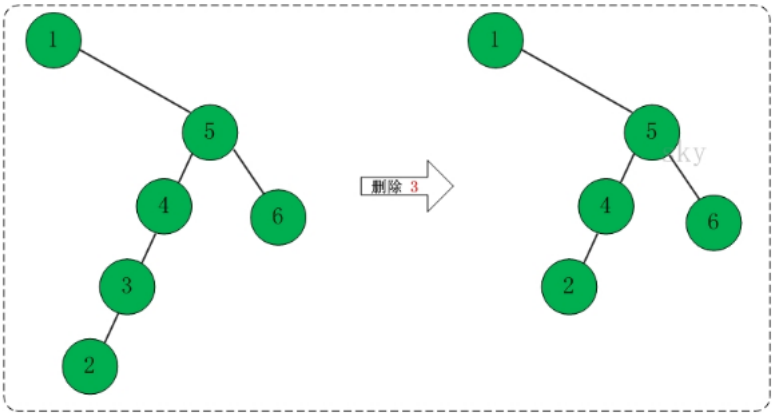
前序遍历结果: **1 5 4 3 2 6**

中序遍历结果: **1 2 3 4 5 6**

后序遍历结果: **2 3 4 6 5 1**

最小值是1，而最大值是6。

(04) 删除节点4。如下图所示：



(05) 重新遍历该二叉查找树。
中序遍历结果：**1 2 4 5 6**

红黑树

```
/**
 * @author SHENHONG
 * 红黑树的数据结构
 */
public class Red_Black_Tree_Node {

    private Red_Black_Tree_Node left;
    private Red_Black_Tree_Node right;
    private int data;
    private int color;    //0表示红色，1表示黑色

}
```

2. 红黑树的性质：

一般的，红黑树，满足以下性质，即只有满足以下全部性质的树，我们才称之为红黑树。

1. 每个节点要么是红的，要么是黑的。
2. 根节点是黑的。
3. 红色节点不能连续（也即是，红色节点的孩子和父亲都不是红色）
4. 对于每个节点，从该节点至叶子节点包含相同个数的黑色节点。

3. 红黑树相比于BST和AVL树有什么优点？

红黑树是牺牲了严格的高度平衡的优越条件为代价，它只要求部分地达到了