1190201215-冯开来

AVL树的删除操作

(一)删除操作

1.删除的结点左右子树都为空：

(1)可能是根节点，那么我们直接删掉他

(2)可能是叶子结点，在删除时我们认为它被删除，然后以它为基点回溯调整树，最后删掉它。

2.删除的结点左子树为空，右子树不为空：

由于AVL树删除之前是平衡的，左子树为空，右子树不能超过1个结点。那么我们以右子树唯一一个的结点为后继，把他的值替换给待删除结点，我们认为后继节点被删除了，以后继节点为基点回溯。

3.删除的结点 右子树为空，左子树不为空：

由于AVL树删除之前是平衡的，右子树为空，左子树上不能超过1个结点。那么我们以左子树唯一一个结点为前驱，把他的值替换给待删除结点，我们认为前驱结点被删除了，以前驱结点为基点回溯。

4.删除的结点 左右子树都不为空：

删除结点的左右子树都不为空，因为只能找一个结点替换值，那么我们以找后继为先，左右子树都不为空就一定可以找到后继节点，但是，因为左右子树都不为空，后继结点可能会有右子树(因为后继结点一定不会有左子树)，当然也可能不会有右子树。那么我们先把后继节点值赋值给待删结点，此时 我们需要把后继结点删除，焦点被引向了后继结点。

待删除的对象 变成了后继结点，后继节点一定没有左子树，那么就转换到了上面的情况2，待删除的结点左子树为空，右子树不为空的情况。

(二)删除后的调整

如果左边失衡，也就是失衡节点(unbalance)平衡因子值为2，根据失衡结点的左孩子(unbLChild)的平衡因子的值不同就会产生以下三种情况：

1.unbLChild=1左左失衡，与插入时一样，我们调整unbalance和unbLChild的平衡因子等于0，以unbalance为基点右旋转即可。

2.unbLChild=-1左右失衡,与插入一样 ，根据unbLChild的右孩子的平衡因子值又分为同样的三种情况。

3.unbLChild=0左失衡unbLChild的平衡因子值为0，根据平衡因子为0的视角来看，一种情况是unbLChild是叶子结点。另一种情况是，unbLChild有左右子树，左右子树平衡：

第一种情况：unbLChild是叶子结点。unbLChild是叶子结点，那么失衡节点左子树只有一个结点，不会发生左失衡，所以不存在unbLChild是叶子结点的情况。

第二种情况：unbLChild左右子树平衡，我们需要右旋转，但是右旋转过后，树2比树3高1，所以根结点平衡因子要先被设定为 1 ，unbLChild变成了根结点，那么 树1 比 树2+1少1，所以unbLChild平衡因子要被设置为-1。这样旋转之后平衡了，平衡因子也正确了。

如果右边失衡，也就是失衡节点(unbalance)平衡因子值为-2，根据失衡结点的左孩子(unbRChild)的平衡因子的值不同就会产生以下三种情况：

1.unbRChild=-1右右失衡，和插入时一样的操作，右右失衡一次左旋就OK，平衡因子和插入也是一样的调整。

2.unbRChild = 1右左失衡，和插入时一养的操作，右左失衡先右旋再左旋，平衡因子和插入也是一样的调整。

3.unbRChild = 0右失衡unbRChild的平衡因子值为0。unbRChild的左右子树平衡，unbalance右失衡，左旋。由于左旋后树1变成右孩子，树1-树3 = 1，所以平衡因子要改为-1，树1变为右孩子，那么，右子树为树2，120比树2高1，所以根平衡因子应改为1。旋转后就平衡了。