1 实现原理

此报告叙述了 AVL 树删除节点操作的实现方法，我们在执行删除操作时不仅需要保持二叉搜索树的性质，并且还要按 AVL 树的方式保持平衡。

1.1 删除策略

我们采用递归实现删除操作，有以下三种情况：

1. 待删除节点为叶节点：直接删除
2. 待删除节点有一个子节点：用子节点替换当前节点
3. 待删除节点有两个子节点：用右子树中的最小值替换当前节点，并删除最小值节点

1.2 平衡维护

删除节点后，我们自底向上更新节点高度以保持 AVL 树的平衡。主要步骤如下：

1. 更新节点高度：$height = max(leftHeight, rightHeight) + 1$
2. 计算平衡因子：$balance = leftHeight - rightHeight$
3. 根据平衡因子进行旋转操作：
   - 左左情况（balance $>$ 1）：右旋
   - 左右情况（balance $>$ 1）：先左旋后右旋
   - 右右情况（balance $<$ -1）：左旋
   - 右左情况（balance $<$ -1）：先右旋后左旋

2 关键代码实现

删除操作的核心在于平衡的维护。每次删除节点后，我们都需要更新受影响路径上的节点高度，并检查是否需要重新平衡。重平衡过程的实现步骤如下：

- 调用 updateHeight 更新节点高度
- 计算平衡因子
- 根据平衡因子的值确定是否需要旋转以及使用哪种旋转方式
- 旋转完成后重新更新受影响节点的高度

为了提高效率，本实现在递归返回过程中进行平衡维护，这样可以保证每个节点最多只需要调整一次。同时我们按照要求使用移动而非复制来处理节点值的替换，减少了不必要的对象拷贝。