1 AVL 树的删除操作实现

在实现 AVL 树的删除操作时，我采用了以下策略：

1.1 基本删除流程

删除操作分为三种情况处理：

1. 被删除节点是叶子节点或只有一个子节点：直接删除并重新连接
2. 被删除节点有两个子节点：
   - 使用 detachMin 函数找到右子树中的最小节点
   - 将该最小节点替换到被删除节点的位置
   - 保持原有的左右子树连接关系

1.2 平衡维护

删除节点后，需要自底向上维护 AVL 树的平衡性：

1. 更新节点高度：计算左右子树的最大高度 +1
2. 计算平衡因子：左子树高度减右子树高度
3. 根据平衡因子进行旋转操作：
   - 左左情况 (balance $>$ 1 且左子树左偏)：右旋
   - 右右情况 (balance $<$ -1 且右子树右偏)：左旋
   - 左右情况 (balance $>$ 1 且左子树右偏)：先左旋后右旋
   - 右左情况 (balance $<$ -1 且右子树左偏)：先右旋后左旋

1.3 实现优化

为了提高效率，我做了以下优化：

1. 使用 detachMin 而不是 findMin，避免多余的节点复制
2. 在旋转操作中直接更新高度，减少重复计算
3. 仅在必要时进行平衡操作，避免不必要的旋转

通过这种实现方式，确保了删除操作后树仍然保持 AVL 平衡特性，即任意节点的左右子树高度差不超过 1。同时，通过优化的实现方式，保证了操作的效率。