摘要

本次编程作业主要重写了 `BST.h` 中的 `remove()` 函数，实现了按 AVL 树的方式删除。

1 `remove()` 函数的设计思路

1. 整体思路：AVL 树和 BST 只有一点不同，即 AVL 树需要两侧子树的平衡。因此在实现 `remove()` 的时候，新增一个 `balance()` 函数，用于平衡子树。具体地，用之前 BST 的 `remove()` 方式找到要删除的节点，删掉；然后依次回溯父节点，检查节点是否平衡，不平衡进行 `balance()`，知道回溯到 root 节点停止。由于需要回溯父节点，这次作业不会继续使用上次 `remove()` 的方式，改用递归。

2. `balance()` 的具体实现：如果是左子树的左节点引起失衡，右旋一次；右子树的右孩子引起失衡，左旋一次；左子树的右节点引起失衡，左旋一次，右旋一次；右子树的左节点引起失衡，右旋一次，左旋一次。其中左旋和右旋分别通过 `singleLeftRotate` 和 `singleRightRotate` 实现。