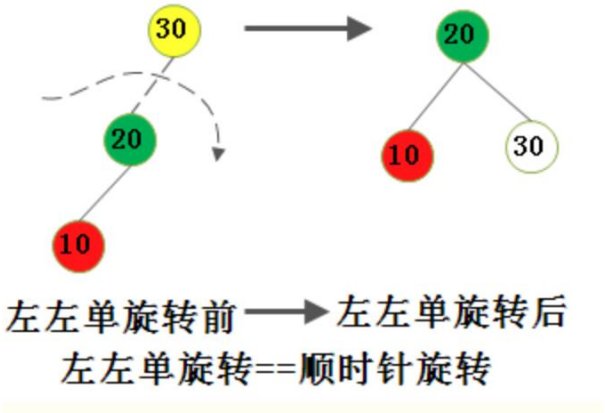
AVL Tree

简介

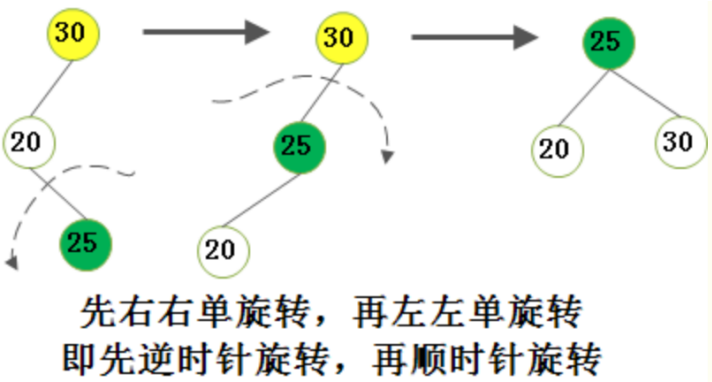
AVL 树是较早发明的自平衡二叉搜索树。在 AVL 树中，任何节点的两个子树的高度最大差别为 1。

为了实现 AVL 树的高度差，AVL 树会进行旋转。根据旋转的方向我们可以分为两大种类型：单旋（向左旋、向右旋）以及双旋（先左旋后右旋、先右旋后左旋）。

单旋示意图：



双旋示意图：



向 AVL 树插入节点后，自底向上向根节点折回，将所有不平衡的节点进行

旋转来实现平衡。因为折回到根节点的路途上最多有 1.5 * log n 个节点，而每次 AVL 旋转都耗费恒定的时间，插入处理在整体上耗费 O(log n) 时间。

因为插入和删除的时候都可能会导致 AVL 树不平衡，因此在插入和删除时都需要考虑 AVL 旋转。

实验部分

本次实验需要对 AVL 树的插入部分进行实现，并且比较一下 AVL 树的插入性能与之前实现的跳表的插入性能。

具体需要实现的内容如下：

1. 实现 AVL 树的插入算法。插入的数据的格式应与跳表插入的数据格式保持一致。
2. 通过计时函数计算 AVL 树进行插入所需要的时间。
3. 随机生成一组插入序列，插入元素的个数分别为 50，100，500，1000，2000。计算 AVL 树插入不同个数的元素所需要的时间，画出折线图。
4. 针对插入元素的个数为 2000 的 AVL 树和跳表，对比下列三种情况它们的插入所需时间的差异。

   1）顺序插入（后一个插入的 key 恒大于前一个值）

   2）随机插入（随机生成插入序列）

   3）逆序插入（后一个插入的 key 恒小于前一个值）
5. 根据 3、4 得到的数据，与理论值进行比较并进行分析，并谈谈你觉得在什么情况下使用跳表、在什么情况下使用 AVL 树更加合理。将 3、4、5 的内容做成一份文档。

提交内容

请将你的实验结果和分析写成报告提交，与代码和输入样例一起做成压缩包上传。命名使用“学号+姓名+hw4”，如“520123456789+张三+hw4.zip”。

注意事项

1. 实验部分的 4 中针对 AVL 树和跳表的插入进行比较时，使用的输入内容（输入样例）需要相同。
2. 本次作业会在 2022.03.30 23:59PM 截止。
3. 有任何问题请与顾翼成助教沟通。