红黑树与 AVL 的对比

实验介绍

红黑树是二叉搜索树的一种变体。普通的二叉搜索树会在多次插入后产生“不平衡”的情况，如顺序插入 1 到 10，则普通的二叉搜索树会退化为链表，导致查找效率退化为 O(n)。为此平衡二叉树选择不断通过旋转调整树节点，使得单次查找在最坏情况下也可保证为 O(logn)，从而提升了查找效率。红黑树也是基于类似的思想，红黑树通过红黑节点配合来达到 2-3-4 树的效果（2-3-4 树是 4 阶的 B 树），如下图所示。理解了转换后，红黑树的增删改查操作也就可以对应到相应的 2-3-4 树模型。

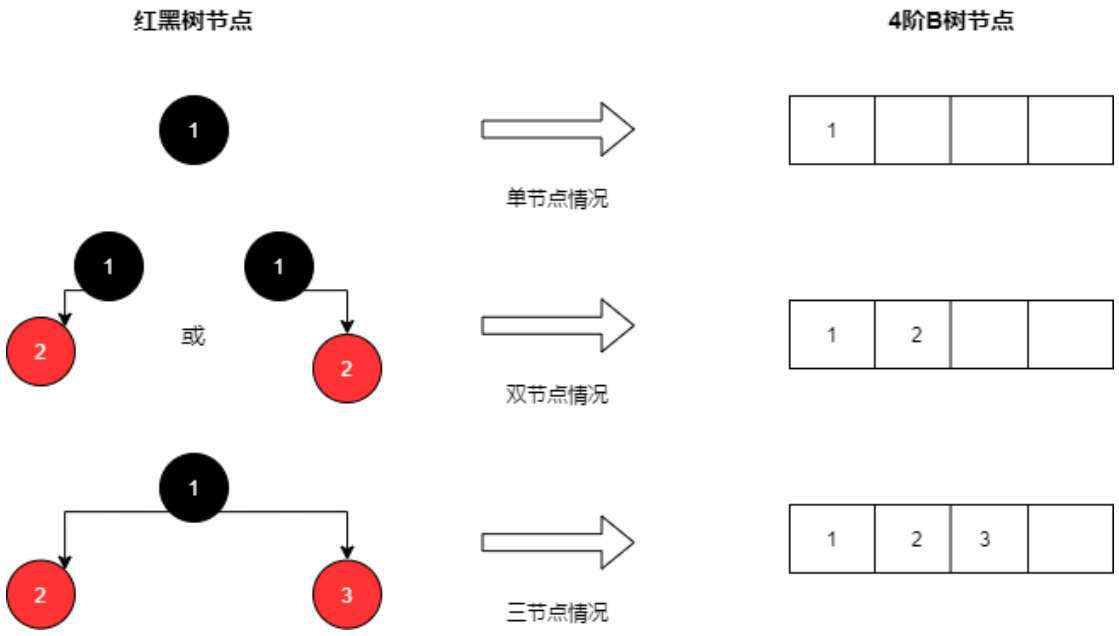

图 1 红黑树与 2-3-4 树的节点转换

在理论上，AVL 与红黑树的单次插入操作和查找操作的复杂度都是 O(logn)，实际上二者的性能也基本相似。**由于红黑树对于“平衡”的要求没有 AVL 严格，所以红黑树的插入为了达到“平衡”的旋转次数要更少一些，同时红黑树的高度也会**

因为不严格的“平衡”而更高，因此查找操作的性能要更低一些。

实验要求

1. 实现一个红黑树，至少有插入和查找操作
2. 设计至少 5 组不同大小的输入集，并分别使用顺序和乱序两种输入方式（建议保存为文件而不是运行时再生成）
3. 测试在红黑树和 AVL 树下，每组输入集使用顺序和乱序两种方式插入时的耗时和旋转操作次数，并将结果制作为表格
4. 设计至少 5 组不同大小的查找集，并分别使用顺序和乱序两种输入方式
5. 测试在红黑树和 AVL 树下，每组输入集使用顺序和乱序两种方式查找时的耗时，并将结果制作为表格
6. 分析所测红黑树与 AVL 树的插入和查询操作开销是否符合理论（见实验说明，至少要对比在顺序和乱序输入场景下的红黑树和 AVL 的插入操作的耗时和旋转次数的对比以及插入操作的耗时对比），并说明自己对两种数据结构差异的理解

请将你的实验文档与代码和数据压缩后一并上传。