华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析	年级：21级	上机实践成绩：
指导教师：金澈清	**姓名**：包亦晟	
上机实践名称：红黑树	**学号**：10215501451	**上机实践日期**：2023.4.27
上机实践编号：No.9	**组号**：1-451	

一、目的

1．熟悉算法设计的基本思想

2．掌握构建红黑树的方法

二、内容与设计思想

1. 编写随机整数生成算法，生成S到T范围内的N个随机整数并输出；
2. 编写红黑树构建算法，中序遍历各节点，输出颜色和值；
3. 随机生成1e2、1e3、1e4、1e5、1e6个**不同的**数，使用红黑树构建算法，并画图描述不同数据量下的运行时间差异；
4. （思考题选做）对比红黑树和普通搜索二叉树在不同情况下插入和查找的性能差异。

三、使用环境

推荐使用C/C++集成编译环境。

四、实验过程

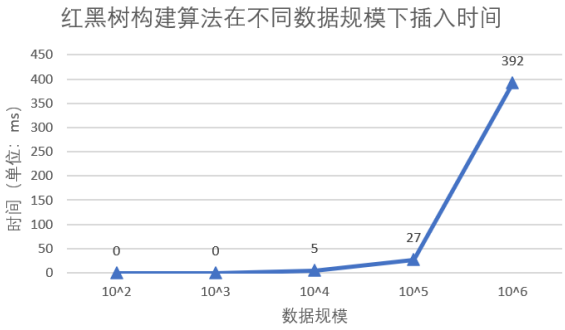
生成不同的随机数

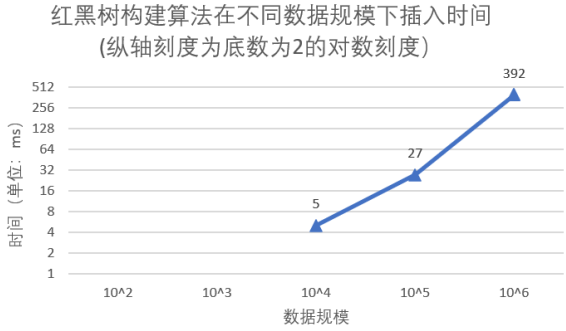
本次实验中生成数据的要求和以往不同，要求生成**不同**的数。按照以往的方式，虽然数据重复可能性很小很小，但依然是有可能的。所以要考虑不同的数据生成方式：

```
vector<int> a;
for (int i = s; i <= t; i++) {
        a.push_back(i);
}
random_shuffle(a.begin(), a.end());
```

因为本次还加了数据范围的要求，所以我就将[s,t]之间的数全部放入容器当中。然后再通过random_shuffle对该容器内数据进行打乱，最后取前N个数就满足要求了。

折线图





五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

折线图分析

当红黑树有n个节点，插入一个新节点的时间复杂度为$O(\lg n)$，所以经过估算，构建一棵有n个节点的红黑树的时间复杂度差不多为$O(n \lg n)$。从图表中可以看出，在对数坐标下，红黑树构建算法随数据规模的增大基本上呈现的是线性增长，与我们的分析基本吻合。

红黑树和普通搜索二叉树的性能差异

插入性能差异

普通搜索二叉树的插入操作的时间复杂度为 $O(\lg n)$，但这只是平均情况下的。当插入的元素有序时，普通的搜索二叉树就会退化成链表，每次插入操作的时间复杂度为$O(n)$。而且，与快速排序中的情况不同的是，快速排序可以通过随机选取主元规避最坏情况的发生，而普通搜索二叉树通常都要根据输入直接插入，**无法规避最坏情况发生**。因此，在插入元素有序的情况下，普通搜索二叉树的性能是比较差的。

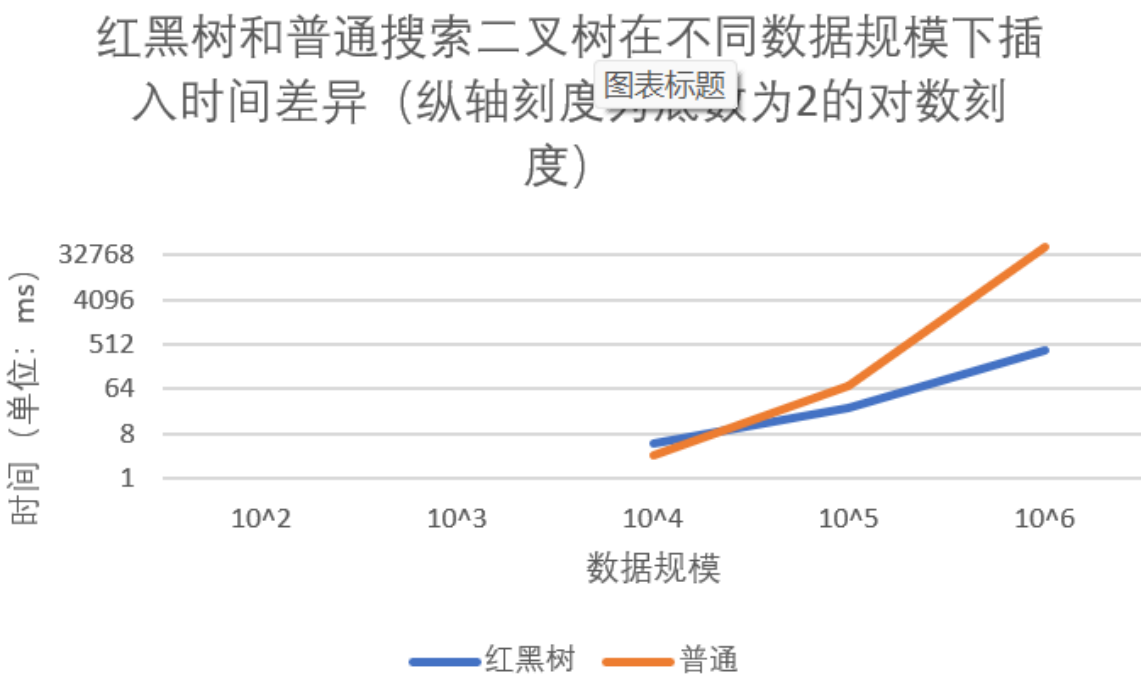
相比之下，红黑树平衡的特性可以保证树的高度始终为$O(\lg n)$。所以无论是面对怎样的数据集，红黑树的插入操作的时间复杂度皆为 $O(\lg n)$。即使红黑树相较于普通搜索二叉树需要额外旋转和变色，效率会低一点，但是时间复杂度始终为 $O(\lg n)$。不会像普通搜索二叉树那样，直接从$O(\lg n)$跌落到$O(n)$。

查找性能差异

查找的分析和插入的分析是类似的。在数据集有序的情况下，普通搜索二叉树查找的时间复杂度会退化到$O(n)$ 。相比之下，红黑树的查找操作的时间复杂度始终为 $O(\lg n)$，因为红黑树因为树的高度始终为 $O(\lg n)$。

综上所述，普通搜索二叉树在节点数较少且元素无序时可能表现较好，但在节点数较多且元素有序时表现非常差。而红黑树始终保持平衡，因此在任何情况下都能够提供稳定的性能表现。

我做了两者在插入性能上的实验，结果如下：



我们发现当数据规模达到10^6的时候，普通的速度要远远慢于红黑树。我目前的猜想是数据集中的数据部分有序导致的，但是因为数据集过大我也无法验证。但我觉得多半是因为这个原因。

那么问题来了，既然红黑树相较于普通搜索二叉树优势在于其平衡性，那么红黑树相较于AVL树的区别又在哪里呢?

红黑树与AVL树的性能差异

首先在平衡度方面。虽然红黑树和AVL树都是平衡二叉搜索树，但是红黑树的平衡度是略逊一筹的。红黑树只是"近似"平衡的二叉搜索树，它的高度差是要大于AVL树的。也就是说，在最坏情况下查找复杂度红黑树是要逊于AVL树的。当然，之间的差距不是很大。

但是插入和删除的时候就是另外一回事情了。对AVL树进行插入和删除操作的时候，是很容易导致树结构的不平衡的，而红黑树自身的平衡度要求较低。所以，单在调整树结构的频率上面来说，红黑树是小于AVL树的。

插入引起不平衡时，AVL树和红黑树都需要至多2次旋转；删除引起不平衡时，AVL至多需要$\lg n$次，而红黑树至多需要3次。

虽然红黑树要多出一个变色操作，但是由于操作很简单，实际运行中变色操作是比较快的。

然后在空间效率层面，红黑树只需要为每个节点额外保存颜色即可，我们可以通过一个bit位来进行表示。而AVL树却需要额外存储高度，多出了许多空间损耗。

所以，红黑树的总体统计性能是优于AVL树的。

感想

我不得不感叹，红黑树真的是个重量级数据结构。抛去实现代码不谈，光光是记住那几种变化规则就已经快要了我老命了。更别说还需要具体代码实现了。也难怪之前在网上一直看到对于面试中手写红黑树的吐槽，这真的是很难凭空写出来。

但是我觉得吧，学习数据结构和算法的关键是不在于完完整整地背出数据结构和算法的代码实现，毕竟我们是人，不是机器。我们真正要学会的是，**要在什么时候以及如何将不同的数据结构运用到自己的代码里**。这点才是关键。记忆方面，我们只需要记住数据结构和算法的基本思路即可，细枝末节的东西到时候查一下就足够了。