二叉搜索树的实现

LUIS LUZERN YUVEN *
数学科学学院，信息与计算科学专业
学号 ： 3190300985

November 20, 2020

1 问题

实现二叉搜索树的构造，析构函数，min，max，successor，predecessor，tree_search，insert，inorder_treewalk 功能，讨论二叉搜索树的排序算法的稳定性与效率，且与随即快速排序算法相比。

min	：返回二叉搜索树中的最小值
max	：返回二叉搜索树中的最大值
successor	：返回排序之后的下一个元素
predecessor	：返回排序之后的上一个元素
tree_search	：返回所搜索的数的地址
insert	：插入一个数到二叉搜索树
Inorder_treewalk	：把二叉搜索树中的元素从小到大排序

2 实验方案

测试运行环境为虚拟机 Virtual Box 下的 Ubuntu 16.04 。由于测试的是相对时间效率，因此对具体的机器性能不敏感，这里不再列出再多配置细节。

2.1 项目文件说明

程序采用 2011 标准的 C++ 编写，项目名称为 QuickSort，项目文件结构如下：

```
BinarySearchTree
    |---generate
    |---stat
    |---BST
```

其中，

generate 能自动产生指定长度提供排序测试的从 1 到 100 的整数，提高出现重复数字的可能性，并采 用 C++11 提供 random 库确保随机性，其中随 机数种子采用对一个具体程序过程的实时统计得到，而时间计算则调用了 C++11 提供的。chrono 库，精 度到纳秒 （10^{-9} 秒）。

*email : luzernyl@gmail.com

stat 用于对批量产生时间数据的统计，目前只实现了多次重复测试的平均时间。

BST 提供计算一个具体排序运行时间的程序和脚本以及实现上述的功能，其中 BinarySearchTree.h 用于实现所有的功能， main.cpp 用于 实际时间计 算和 功能的测试，同样通过 chrono 库完成； bash 脚本 batch_test 共有 4 个 用户参数，依次分别代表 ：

$1 测试总组数
$2 测试的起始数组长度
$3 每组测试的增量
$4 每组测试的重复次数

专门用于测试排序的相对时间效率。而脚本 test4matlab 功能与参数设置 和 batch_test 一致，区别在于其输出为一个 Matlab 脚本文件的格式；

3 稳定性和效率

二叉搜索树的排序算法， inorder_treewalk, 是一个稳定的排序算法。因为在 insert 函数里，如果遇到相等的数，则这个数变成 rightchild。

```
if (y == NULL)
    root = p;
else if (p->data < y->data)
    y->lc = p;
else
    y->rc = p;
```

而在 inorder_treewalk 函数里，先用递归在一个 Node 的 leftchild (lc)来找适当的函数，若找到，直接 print 这个 Node 的值，才再用递归在 rightchild (rc)找下一个元素。

```
void BinaryTree::inorder_treewalk(Node* _x) const
{
    if (_x != NULL)
    {
        inorder_treewalk(_x->lc);
        std::cout << _x->data << "\t";
        inorder_treewalk(_x->rc);
    }
};
```

显然，若有两个相同的数，第一次被输入的数一定排在第二次被输入的数的前面。故，inorder_treewalk 是一个稳定的算法

4 效率

实验结果见图 1。从 t_n 的增长趋势可以看出，随着 n logn 增大，这一现象与结论：

$$t_n = \theta(n \log n)$$

一致。

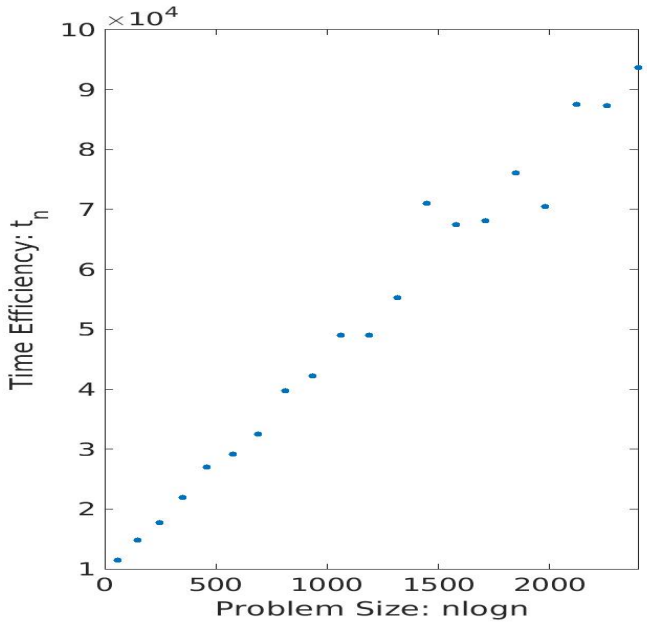
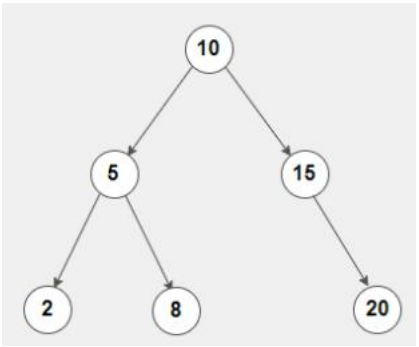
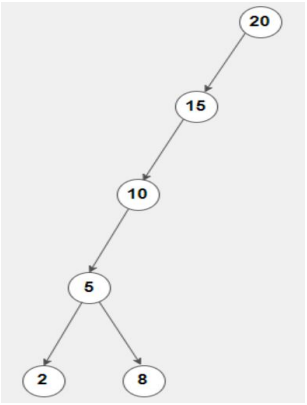


图 1 ： 快速排序对于随机分布整数数组的时间效率，这里每一个点代表一组重复了 100 次后取平均的实验结果。共 20 组，输入数组的规模从长度 20 起，等量递增到 400. 每一次重复实验均做了新抽取。

在一个二叉搜索树，排序的快速是取决于该树的平衡。若一个二叉搜索树是平衡或者接近平衡的，则排序快速比较快。反之，若一个二叉搜索树是不平衡的，则排序的快速会慢地很多。即，二叉搜索树排序的快速取决于该树的高度，树越高，排序快速越慢。



平衡二叉搜索树



不平衡二叉搜索树

二叉搜索树的最佳情形 ： 完全平衡树 (evenly balanced tree), $t_n = \theta(n \log n)$

二叉搜索树的最坏情形 ： 完全退化树 (fully degenerate tree), $t_n = \theta(n^2)$

二叉搜索树的一般情形 ： $t_n = \theta(n \log n)$

与随机快速排序的比较 ：

二叉搜索树的排序与随机快速排序有相同的时间效率， $\theta(n \log n)$

在随机快速排序，排序的快速是取决于随即划分的结果。若划分的一边的元素个数比另一边大的很多，则排序快速越慢

随机快速排序的最佳情形 ：随机划分之后的两边有相同的元素个数， $t_n = \theta(n \log n)$

随机快速排序分最坏情形 ：随机划分之后，一边只含有主元，另一边含有其余的元素，

$t_n = \theta(n^2)$

随机快速排序的一般情形 ：$t_n = \theta(n \log n)$

5 结论

二叉搜索树的排序算法是稳定的，而且效率为：

$$t_n = \theta(n \log n)$$

二叉搜索树的排序与随机快速排序有相同的性质，因为快速排序中的主元 (pivot) 也可以看为二叉搜索树中的根 (root)。若随机划分的结果不平衡，则对应的二叉搜索树也不平衡。