Report

Report of BST & its Sort Function

一.模板类的实现

私有成员变量

1.BinaryNode:二叉树结点结构体，包含该结点的值以及指向左子树和右子树（如果有的话）的根的指针.
2.root:根结点.

成员函数

1.insert:进行一个新元素的插入，由于是二叉搜索树，故不会随便找个叶子当父亲，而是从根开始根据元素大小递归插入.
2.remove:删除树上相应大小的结点，同样放在private里引址调用并利用递归思想.
3.findMin/findMax:返回树上最小/大元素的结点的地址.
4.contains:判断树上有无所给大小元素的结点.
5.makeEmpty:清空树上结点并释放旧空间内存，root指向nullptr.
6.printTree:前序遍历打印树.
7.clone:深复制.
8.isEmpty:判断是否为空树.
9.基本的构造函数、operator=重载、析构函数.

二.构造两个sort函数

一.BST_sort

构造方法

新建一个空二叉搜索树，将_data中元素依次插入树中，clear掉_data然后依次findMin树中的元素push_back到_data中并remove掉树中这个被push的元素，最终实现_data的排序.

二.Randomized_BST_sort

构造方法

基本同BST_sort，但是在sort前会先对_data进行乱序操作，这里我利用c++11的一个新特性产生一个巧妙的随机乱序方法:先利用random_device（本身是均匀分布整数随机数生成器）生成一个种子值rd，然后利用random库中的**std::mt19937**（一个相较于rand()函数速度快且周期长的高效随机数算法）与rd产生一个种子生成器(Standard mersenne_twister_engine seeded with rd())generator,然后利用shuffle函数与generator对_data内容进行随机排列.该函数会附带打印洗牌后的_data.

三.对两个sort函数的测试

具体见main.cpp，运行时会产生一个模拟操作系统，每次分别可以输入0/1/2/3/4来退出系统/清空_data/push新元素进_data（要push的元素数n由测试者自己输入，会有提示）/利用二叉搜索树库和两个sort函数对_data进行排序/打印_data.

四.sort函数复杂度分析

显然sort函数利用到二叉树，其建立导致空间复杂度为O(n).

1.查找:

普通的顺序查找时间复杂度是O(n^2)，而这里设计的sort函数的时间复杂度与二叉搜索树的平衡性有关，如果平衡性好的话时间复杂度可能仅仅是O(logn);相反平衡性差的话这时二叉树就跟边上附着几根枯枝散叶的长链表一般，查找时间复杂度为O(n),这时还浪费了空间，十分不划算.

2.排序:

普通的冒泡排序时间复杂度为O(n^2),而这里设计的sort函数的时间复杂度也与树的平衡性有关，平衡性好的话findMin和findMax以及remove函数就会高效工作，最终时间复杂度为O(nlogn);相反平衡性差的话，时间复杂度为O(n^2)（time~n(n-1)/2）.

五.结论

我们需要二叉搜索树的进化版AVL树的进化版红黑树！