算法基础LAB2实验报告

PB19071535徐昊天

一.实验内容

1. 将Hash和Heap组合起来，维护一个成绩列表，其中每个元素由名字和成绩组成。
2. 维护一个红黑树。
3. 给定一个序列$a_1,\ldots,a_n$，设计一个算法来找到最长严格递增子序列$s_1,\ldots,s_m$的长度m。

二.实验目的

1. 利用OJ编程，强化编程能力，培养编程细节，提升编程素养。
2. 熟悉书本和课堂上涉及的算法，进一步巩固算法知识。
3. 设计数据结构，强化处理问题的能力，并用特殊的数据结构提升处理问题的效率。

三.实验思路

I.成绩单

将hash和heap结合维护成绩列表。分别设计hash和heap结点数据结构，其中hash结点储存下标对应的学生在堆结点数组中的下标位置，heap结点储存学生的成绩和名字。解决该问题需设计以下算法：

- 哈希函数

  由于学生姓名限定为由大写字母构成的最多五位的字符串，故只需考虑固定范围内的字符串。通过对每一位字符转化为整型数并逐渐提升位数，最终可获得对应字符串的唯一的整型数，将该整型数作为哈希表的下标，表中该位置即可储存对应学生在堆中的位置。

- 插入节点

参照算法导论中提供的算法，将堆大小加一，并在堆的最后一位插入一个成绩为负数的结点，并将该结点的成绩提升为待插入结点的成绩。在结点移动位置的同时维护哈希表，即可实现插入操作。该操作时间复杂度为$O(lgn)$。

- 删除结点

  参照算法导论中提供的算法，将待删除结点的成绩提升到大于100的数，并删除堆中最大结点，堆大小减一。在结点移动位置的同时维护哈希表，即可实现删除操作。该操作时间复杂度为$O(lgn)$。

- 查找结点

  根据待查找的学生名字直接在哈希表中搜索对应堆结点即可查找到学生成绩。该操作时间复杂度为$O(1)$。

- 输出最高成绩的学生

  在堆中自顶向下查找所有成绩与堆的首位结点成绩相同的结点并将名字储存，执行基数排序后输出。该操作时间复杂度为$O(k)$。

主函数中根据输入的大写字母标识合理调用以上算法即可实现成绩单的维护。

II.红黑树

根据算法导论上对红黑树的定义，分别定义红黑树结点和红黑树的数据结构。并基于以上数据结构设计以下算法：

- 红黑树初始化

  初始红黑树只有一个$T->nil$结点，并设置该结点为黑色。

- 旋转

  参考算法导论，若对红黑树执行增删操作，需通过旋转调整指针结构，从而保持红黑树性质。

- 插入

  参考算法导论，在红黑树中插入新结点，需调用插入函数，并在插入函数中调用**RB_INSERT_FIXUP**函数来保持红黑性质。

- 删除

参考算法导论，从红黑树中删除结点需基于**RB_DELETE**过程。并且需特别设计一个供**RB_DELETE**过程调用的子过程**RB_TRANSPLANT**，并调用一个辅助过程**RB_DELETE_FIXUP**通过改变颜色和执行旋转来恢复红黑性质。

- 查找第k大元素

  参考算法导论并做出对应修改，利用红黑树结点结构体中的**size**元素，通过递归过程可得到红黑树中的第k大元素。

- 查找[a,∞]中的最小元素

  通过从根结点向下迭代查找，即可查询红黑树中大于a的最小元素。

- 查找[l,r]中的元素个数

  可先设计一个函数返回红黑树中小于等于k的元素个数，然后调用这个函数分别返回小于等于r和l-1的元素个数并相减，即为所求结果。

主函数中即可根据输入的字符标识对应的操作调用以上函数实现红黑树的维护。

III.最长递增子序列

通过遍历序列并合理调用二分查找设计算法查找最长递增子序列。

维护两个数组b,c。其中c数组中$c[i]$储存数组$a[1..i]$中以$a[i]$为最后一个元素的最长递增子序列长度，b数组中$b[i]$储存a数组中长度为i的最长递增子序列的最后一个元素的最小值。

length用于记录每次迭代结束后a[0..i]的最长递增子序列长度，迭代结束后即为a[0..n-1]的最长递增子序列长度。

迭代过程中，可分为三种情况分别处理：

1. $a[i] < b[1]$

   即查找到了a[0..i]中的最小元素，故将b[1]更新为a[i]并将c[i]更新为1。

2. $a[i] > b[length]$

   查找到了比已知最长递增子序列最后一个元素更大的元素，故应当更新最长递增子序列长度，并对b,c数组做出适当调整。

3. 其他

   利用二分查找法找到满足$b[j-1] < a[i] <= b[j]$的j，并对b,c数组做出适当调整。

最后在主函数中调用以上算法即可得到所求最长递增子序列长度。

由于**FIND_LIS**算法需执行n次循环，每次循环最多执行一次二分查找，故时间复杂度为$O(nlogn)$。

四.实验收获与感想

1. 利用OJ编程，养成了更好的编程习惯，促进自身追求更优美的代码风格，提升了编程素养。
2. 提升了编程能力，提前适应上机编程的难度，为将来刷算法题或机试打下了基础。
3. 增强了对代码细节的把控能力，对程序的追求不止于结果正确，而是追求更低的时间空间消耗、数据的完整性和精确性等。
4. 增强了对经典算法的理解，通过复现算法更加深入了解了算法的原理和过程。
5. 通过设计数据结构并处理相应问题，对经典数据结构的理解和运用有了较大的进步。