Algorithm Homework3

PB18111704 Zhu Enzuo

2020 年 10 月 24 日

0.1 Prob1

排序算法的稳定性分析:
插入排序: 稳定
归并排序: 稳定
堆排序: 不稳定
快速排序: 可稳定可不稳定
计数排序: 不稳定
稳定化方法: 将原位置信息作为排序的指标之一。在两个元素相等时比较原位置大小, 小的在前大的在后。时间复杂度: 不变。空间复杂度: O (N)

0.2 Prob2

序列如下:
$S =< 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 >$

0.3 Prob3

证明:
记 N 为插入数据的量, l 是决策树的叶节点树, h 是决策树高度。
我们考虑以节点插入这个过程中每一个时刻树的中序遍历和相应操作所对应的决策树。其中如果我们当前在某一个节点进行比较, 则我们认为这个节点的中序遍历为 $< leftchild, key, inserting\_key, rightchild >$; 对于尚未插入的节点我们将其放在决策树的中序遍历之后。
则我们可以发现叶子结点是一个维护好的 BST。而且这个决策树和基于比较的排序的决策树等价。

因此我们有 $N! \leq l \leq 2^h$，所以 $h = \Omega(NlogN)$

0.4 Prob4

先考虑求 Tree-Successor 的算法：

1、如果节点存在右子，则找到右子树的最左节点

2、如果该节点无右子，则找到第一个左子树包含这个节点的父亲。

对树的深度 h 进行归纳。

当树的深度 h 为 1 时，显然 $k \leq 2^(h-1)$，故总复杂度为 $O(h+k)$。

当树的深度 h 有 $\forall k \leq h' \leq h-1, f(h',k) = O(h+k)$，我们可以得出

对于该树的任意一个查询都可以分治为在左子树上的连续查询和在右子树上的连续查询。

不妨设我们求后继的次数 $k = l + m + r$，其中 l 为后继在左子树伤的次数，m 为后继是根节点的次数，r 为后继在右子树上的次数，那么根据归纳假设有 $f(h) = f(h-1,l) + O(h+m) + f(h-1,r)$。

故 $f(h) = O(h-1+l) + O(h+m) + O(h-1+r) = O(h+l+m+r) = O(h+k)$