算法基础 HW3

PB18111697 王章瀚

2020 年 10 月 27 日

1

下面的排序算法中哪些是稳定的：插入排序、归并排序、堆排序、快速排序和计数排序？给出一个能使任何排序算法都稳定的方法。你所给出的方法带来的额外时间和空间开销是多少？

- 稳定的排序算法有: 插入排序, 归并排序, 计数排序[1](但其实真的需要所有的都可以稳定的...)
- 使任何排序算法都稳定的方法:
  只需要将需要排序的元素对象封装为一个数据结构, 这个数据结构包含这个元素以及一个元素原本的次序。这样，在做排序的时候，优先按照元素排序，若相同，就按照这个次序域的值排序（要求按原本次序），即可完成稳定的排序。
  下面是一个通用的示例程序，以方便助教理解我的意思：

```
template <class T>
class Element {
public:
    T elem;
    int order;

    Element(T elem, int order): elem(elem), order(order) {};
    bool operator< (const Element &e) {
        if (this->elem < e.elem) return true;
        else if(this->elem == e.elem) return (this->order < e.order);
        else return false;
    }
    // 其它运算符也做类似重载
}
```

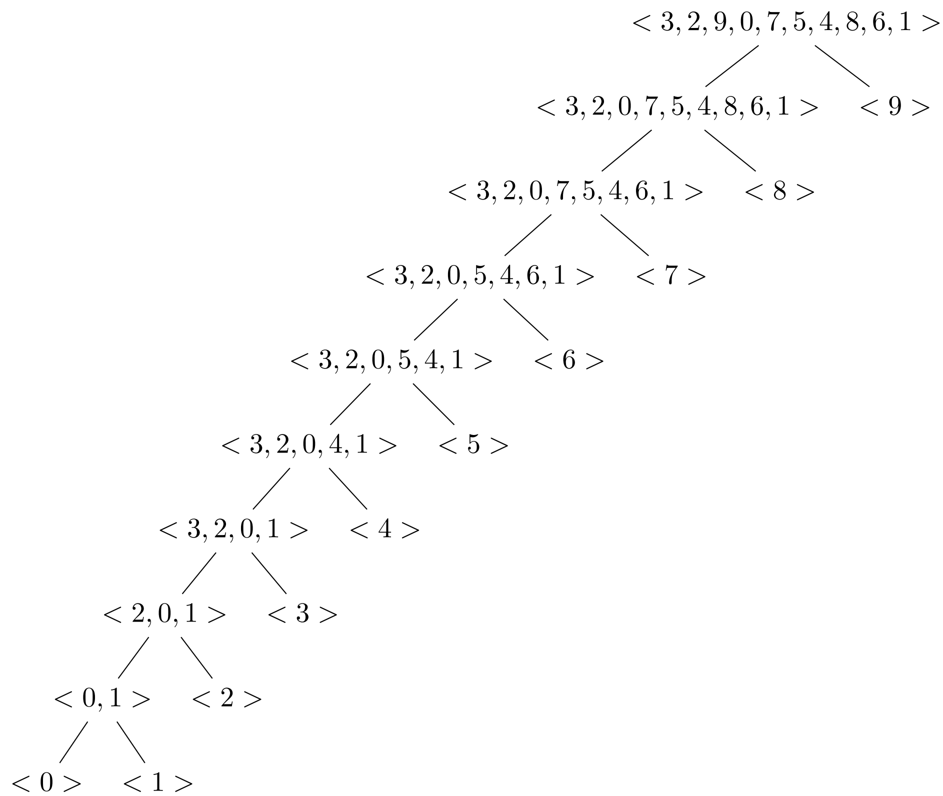
- 这种方法带来额外时间开销只是需要多比较一个order, 而空间开销则是$\Theta(n)$的(为n个元素各加了1个order域)

[1] 参考网站(助教就当没看见哈☺):https://www.cnblogs.com/Leophen/p/11397731.html

2

假设用 Random-Selecect 去选择数组 $A = <3, 2, 9, 0, 7, 5, 4, 8, 6, 1>$的最小元素, 给出能够导致 Random-Selecect 最坏情况的一个划分序列.

可以每次都选择最大元素, 那么就会有这样的最坏划分序列:



3

因为在基于比较的排序模型中, 完成n个元素的排序, 其最坏情况下需要$\Omega(n \lg n)$时间. 试证明: 任何基于比较的算法从n个元素的任意序列中构造一棵二叉搜索树, 其最坏情况下需要$\Omega(n \lg n)$的时间.

反证假设: 如果构造一棵二叉搜索树, 最坏情况下不需要这么多($\Omega(n \lg n)$)的时间,
那么, 我们可以直接构造一棵二叉搜索树, 然后经过一次中序遍历($\Theta(n)$)就能得到排序好的数组,
这时, 总的时间必然是少于$\Omega(n \lg n)$的,
这与"在基于比较的排序模型中, 完成n个元素的排序, 其最坏情况下需要$\Omega(n \lg n)$时间"相矛盾.
因此命题成立. $\square$

4

证明: 在一棵高度为h的二叉搜索树上, 无论从哪个结点开始, k次连续的TREE-SUCCESSOR 调用所需时间为$O(k+h)$.

无论从哪个结点开始, 从访问第一个到访问第k个, 其间至多有$2h$个结点是访问了但无效的(不是要找的后继的结点), 这$2h$个结点来自

- 刚开始需要从比较靠近叶子的结点不断向上直到不再是右结点, 才能得到后继, 这里最多h个
- 快结束了需要从靠近根处往下, 寻找右子树的最小元, 这里最多需要h个

此外, 剩下的都是对有用节点的访问, 这里每个结点最多被访问到3次(当有左右子节点的时候), 因此这里最多是$3k$次的访问.
综上, 总共最多访问$2h+3k<3(h+k)$次结点, 所以可以认为题述情况所需时间为$O(k+h)$