HW3

肖桐 PB18000037

2020 年 10 月 23 日

解 1. 稳定排序算法：插入排序、归并排序.
非稳定排序算法：堆排序、快速排序、计数排序.
改进算法：可以将每个数据都包装为一个结构体, 其中不仅包含原来的数据 $value$, 还应该包含每个数据在原来数组中的下标 $index$.
这样只需要改变在排序时的判断条件：$(value_1 < value_2) \mid\mid ((value_1 == value_2)$ && $(index_1 < index_2))$ 才能进行交换.
额外时间开销为每次比较都要额外比较两个表达式, 额外空间开销为原来的一倍.

解 2. 划分时每次取当前数组中最大的元素可以使得发生最坏情况. 因此划分序列为 $9, 8, 7, 6, 5, 4, 3, 2, 1, 0$.

解 3. 先考虑构造二叉树的最佳情况. 即是每次向二叉树中插入节点时二叉树深度都为 $\lfloor \lg n \rfloor + 1$. 则时间复杂度为：

$$\begin{aligned}
\sum_{k=1}^{n} (\lfloor \lg k \rfloor + 1) &\le \sum_{k=1}^{n} (\lg k + 2) \\
&= 2n + \sum_{k=1}^{n} \lg k = 2n + \lg \prod_{k=1}^{n} k \\
&\le 2n + \lg n^n = 2n + n \lg n \\
&= \Omega(n \lg n)
\end{aligned}$$

又因为最坏情况下的时间复杂度必然比最佳情况的要大. 因此最坏情况下时间复杂度也是 $\Omega(n \lg n)$.

解 4. 记初始节点为 x, x 经 k 次 TREE_SUCCESSOR 操作后到达的节点, z 为 x, y 高度最高的公共祖先节点.
则因为 TREE_SUCCESSOR 操作实际上是一部分的树的遍历操作, 因此每条树边不会被访问两次以上, 每个节点不会被访问 3 次以上.
同时, 路径 $x \to z$ 和 $y \to z$ 上值不在 x, y 之间的节点被最多访问一次. 又因为每次 TREE_SUCCESSOR 操作访问节点个数的上界为 h.
因此总的访问节点个数的上界为 $3k + 2h = O(k + h)$. 因此时间复杂度也为 $O(k + h)$.

$$(N, C_{\text{in}}, H, W)$$

$$(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$$

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$