算法作业 5

孟妍廷 2015202009

2017 年 10 月 22 日

8.3-2

解：插入排序是稳定的，因为它是从某一位置起依次向前比较，不会改变两个相同元素的相对位置。归并排序也是稳定的，因为它递归的过程中每次将当前数组分为两半，直到分成单个元素，最后进行比较和合并，不改变两个相同元素的相对位置。堆排序是不稳定的，因为它不能保证分别位于完全二叉树的左右子树的两个相同元素的相对位置不被改变。快速排序也是不稳定的，因为 partition 过程中若有元素与主元相等，他们的相对位置可能会被改变。
使任何排序算法都稳定的算法 (假设被排序的数组为 A, 长度为 n):
加入一个长度为 n 的数组 B[n], 记录原数组中每个元素最初的位置, 即 B[i]=i. 在排序过程中,
在每次出现 swap(A[i],A[j]) 时同时执行 swap(B[i],B[j]). 由于排序后相同的元素是连在一起的,
在所有排序后加上:

$$for\ i = 1\ to\ n$$

$$while(A[i] == A[i+1])$$

$$count++;$$

根据$B[i]$到$B[i+count]$的大小对$A[i]$到$A[i+count]$进行排序

由于在排序中，相同的元素应该是常数个，否则排序失去意义，没有增加额外的时间开销，
额外的空间开销为 O(n).

8.2
解: 对于题目中的关键字的值为 0 或 1 的理解: 待排序的数组的元素值为 0 或 1

a. 要求算法满足时间复杂度为 O(n) 且是稳定的, 则计数算法满足要求
因为计数算法是稳定的, 且取值只有 0 和 1, 所以 k 为 2, 远远小于 n, 故时间复杂度为 O(n)

b. 要求算法时间复杂度为 O(n) 且是原址排序，只需要固定的额外存储空间
修改快速排序中的 partition 函数如下:

$$partition(A, p, r)$$

$$i\ =\ p-1$$

$$for\ j\ =\ p\ to\ r$$

$$if\ A[j]\ ==\ 0$$

$$i\ =\ i+1$$

$$swap(A[i], A[j])$$

满足要求, 排序后数组左边为为 0 的元素, 右边为为 1 的元素

c. 要求算法是稳定的且是原址排序，只需要固定的额外的存储空间
插入排序满足要求, 只是时间复杂度不为 O(n).
d. 算法 (a), 即计数排序可以使基数排序的时间代价是 O(bn)
由于计数排序是稳定的, 且 k=2, 故由引理 8.3 可证。(题目中的如何处理是什么意思?)
而算法 (b) 不是稳定的, 算法 (c) 的时间复杂度不是 O(n).

e. 依题意修改计数排序算法如下:
$counting\_sort(A, k)$
$let\ C[0....k]\ be\ a\ new\ array$

$for\ i\ =\ 0\ to\ k$
  $C[i]\ =\ 0$
$for\ j\ =\ 1\ to\ A.length$
  $C[A[j]]\ =\ C[A[j]]+1$
$for\ i\ =\ 1\ to\ k$
  $C[i]\ =\ C[i]+C[i-1]$
$i\ =\ 1$
$while\ i\ <\ C.length$
  $for\ j\ =\ C[i-1]\ to\ C[i]-1$
    $A[j]=i$
  $i++$
由于该算法相当于统计出每个元素应该在的位置, 把原数组覆盖掉
所以该算法可以在 O(n+k) 时间内实现对 n 条记录的原址排序, 但是该算法不稳定

9.3-1
证明:
假设每组 x 个元素 (x 为奇数), 则除了 n 不能整除 x 时产生的元素少于 x 个的组和包含 x 的组之外，至少有一半的组中有 $\lceil \frac{x}{2} \rceil$ 个元素大于 x, 则大于 x 的元素个数至少为:

$$\lceil \frac{x}{2} \rceil (\lceil \frac{1}{2} \lceil \frac{n}{x} \rceil \rceil - 2) \geq \frac{n}{4} - x$$

所以只有当所有小于 x 的数都被遍历时达到最坏情况, 最多有 $\frac{3}{4}n + x$ 个
则在最坏情况下的递归式为:

$$\begin{aligned} T(n) &= T(\lceil \frac{n}{x} \rceil) + T(\frac{3}{4}n + x) + O(n) \\ &\leq \frac{cn}{x} + c + \frac{3}{4}cn + cx + an \\ &= (\frac{1}{x} + \frac{3}{4})cn + cx + c + an \\ &= cn - \frac{x-4}{4x}cn + cx + c + an \quad (*) \end{aligned}$$

可知若要上式满足 $(*) < O(n)$, 需满足 $x - 4 > 0$ 即 $x > 4$, 否则时间复杂度 >O(n)
因此每组 7 个元素时仍然是线性时间，每组 3 个元素时 SELECT 的运行时间不是线性时间。