第二周作业-solution

LECTURER: 杨启哲 LAST MODIFIED: 2023 年 12 月 23 日

1. (教材习题 4.17) 对于下列给出的 8 个数的数列，用图来说明算法 *RadixSort* 的运算过程：
   - 16543，25895，18674，98256，91428，73234，16597，73195

解答. 我们这里只给出算法每一轮结束过后，当前的数组排列情况：

1. 16543，18674，73234，25895，73195，98256，16597，91428.
2. 91428，73234，16543，98256，18674，25895，73195，16597.
3. 73195，73234，98256，91428，16543，16597，18674，25895.
4. 91428，73195，73234，25895，16543，16597，98256，18674.
5. 16543，16597，18674，25895，73195，73234，91428，98256.

□

2. (教材习题 4.18) 当输入由下列区间中的 n 个正整数组成时，以 n 为大小说明算法 *RadixSort* 的时间复杂性。
   - $[1,\ldots,n]$.
   - $[1,\ldots,2^n]$.

解答. 只需要计算至多需要多少位表示数即可。

1. 当数据范围在 $1\sim n$ 时，其用来表示的数位至多为 $\lceil\log n\rceil$，从而算法的时间复杂性为 $\Theta(n\log n)$.
2. 当数据范围在 $1\sim 2^n$ 时，其用来表示的数位至多为 n，从而算法的时间复杂性为 $\Theta(n^2)$.

□

3. (教材习题 5.4) 设 $A[1,\ldots,n]$ 是一个由 n 个整数组成的数组，x 是一个整数，请给出一个分治算法，求 x 在数组 A 中的频度，即 x 在 A 中出现的次数。你算法的时间复杂性是什么？

解答. 将 A 分解成两个数组，$A_1=A[1,\ldots\lfloor\frac{n}{2}\rfloor]$ 和 $A_2=A[\lfloor\frac{n}{2}\rfloor+1,\ldots,n]$，然后递归地求解 A_1 和 A_2 中 x 的频度，最后将两者相加即可。算法的时间复杂性为 $T(n)=2T(\frac{n}{2})+O(1)$，根据主定理可知，其时间复杂性为 $O(n)$. 算法伪代码如下所示：

算法: 计算数组频度

输入: 数组 $A[1,\ldots,n]$ 和整数 x
输出: x 在 A 中出现的次数
1: return $Count(A,1,n,x)$

过程: $Count(A, low, high, x)$
if $low = high$ then
    return $A[low] == x$
end if
$mid \leftarrow \lfloor (low + high)/2 \rfloor$
return $Count(A, low, mid, x) + Count(A, mid + 1, high, x)$

□

4. (计算逆序对) 数组中的一个逆序对 (i, j) 指的满足 $i < j$, $a[i] > a[j]$ 的一个二元组。现在给定一个数组，请设计一个 $O(n \log n)$ 时间的算法来计算其中逆序对的个数。

解答. 这道题的核心是在于发现逆序对的计算可以通过归并排序来完成。事实上，当数组被划分成 $A_1 = A[1, \ldots \lfloor \frac{n}{2} \rfloor]$ 和 $A_2 = A[\lfloor \frac{n}{2} \rfloor + 1, \ldots, n]$ 两个数组时，其逆序对个数满足：

$$A\text{中逆序对个数} = A_1\text{中逆序对个数} + A_2\text{中逆序对个数} + A_1\text{和}A_2\text{之间的逆序对个数}$$

当 A_1, A_2 各自的逆序对计算好时，对其进行排列不会影响两者之间的逆序对个数，因此其中 A_1 和 A_2 之间的逆序对个数事实上可以在将两个有序数组合并成一个有序数组的过程中计算出来，如下例子：

$$A = [3, 5, 4, 10], \ B = [2, 1, 6, 7]$$

可以看到 A 和 B 各自有一个逆序对。当计算其之间的逆序对时，我们可以先将其变成有序的数组，此时合并上述两个数组时，在比较 $A[i]$ 和 $B[j]$ 时，如果 $A[i] > B[j]$，则 $A[i]$ 和 $B[j]$ 之间的逆序对个数为 $4 - i + 1$，因为 $A[i]$ 后面的元素都大于 $B[j]$。基于此，我们可以设计如下的算法：

算法: 计算数组逆序对个数

输入: 数组 $A[1, \ldots, n]$
输出: 逆序对个数
return $Count(A, 1, n)$
过程: $Count(A, low, high)$
if $low = high$ then
    return 0
end if
$mid \leftarrow \lfloor (low + high)/2 \rfloor$
return $Count(A, low, mid) + Count(A, mid + 1, high) + MergeCount(A, low, mid, high)$
过程: $MergeCount(A, p, q, r)$
$s \leftarrow p,\ t \leftarrow q + 1,\ k \leftarrow p,\ count \leftarrow 0$
while $s \leq q$ and $t \leq r$ do
    if $A[s] \leq A[t]$ then
        $B[k] \leftarrow A[s]$
        $s \leftarrow s + 1$
    else
        $B[k] \leftarrow A[t]$
        $t \leftarrow t + 1$
        $count \leftarrow count + q - s + 1$
    end if

    $k \leftarrow k + 1$
end while
if $s = q + 1$ then $B[k, \ldots r] \leftarrow A[t, \ldots, r]$
else $B[k, \ldots r] \leftarrow A[s, \ldots, q]$
end if
$A[p, \ldots, r] \leftarrow B[p, \ldots, r]$
return *count*

$\square$