算法设计与分析作业二

作者：吴润泽　　　**学号：**181860109

Email：181860109@smail.nju.edu.cn

2020 年 3 月 6 日

目录

PART I

problem 6.8

算法分析 假定 n 总是 k 的倍数，且 n 和 k 都是 2 的幂。

利用快排的思想，将数组从中间划分为两段 $A[0\cdots n/2]$, $A[n/2+1\cdots n]$，且左段元素小于右段元素。

对于子序列继续递归划分，得到 $A[0\cdots n/2^m], A[n/2^m+1\cdots n/2^{m-1}]\cdots A[n-n/2^m+1\cdots n]$，当 $2^m = k \rightarrow m = \log k$ 时，划分完成。

因此寻找中位数划分的函数时间复杂度应为 $O(n)$，划分函数的递归方程为 $W(n) = 2W(n/2) + O(n)$，划分左右子段 $\log k$ 次，方能使得总的时间复杂度达到 $O(n\log k)$。

具体算法实现请见算法 k-sorted：算法 1

算法时间复杂度 对于 findk_pos，每次递归代价为 $O(n)$，每次子问题缩小为原来一半的规模，且子问题只有一个，可列出递归方程为 $T(n) = T(n/2) + O(n)$，由主定理可以得出 $T(n) = O(n)$。

对于 k_sorted，每次递归代价为 $O(n)$，每次子问题缩小为原来一半，而需要划分左右两序列，子问题为两个，可列出递归方程为 $W(n) = 2W(n/2) + O(n)$，注意结束条件为递归调用了 $\log k$ 层，每层代价均为 $O(n)$，因此时间复杂度为 $O(n\log k)$ 满足题目要求。

算法 1 k-sorted 算法

输入: 待划分序列 $A[1\cdots n]$，划分段数 k

输出: 划分后的的序列 A

function FINDK_POS$(A, k_pos, begin, end)$ \\返回该段数组第 k 小

/* 利用快排思想，选定一个 key，将大于 key 的元素放在其右边，小于 key 放于左边。

判断 key 插入的位置是否为 k，如果是则函数返回，如果插入位置大于 k 说明第 k 小位于左子序列对左边递归寻找，否则对右子序列递归寻找。 */

  $split \leftarrow begin, key \leftarrow A[begin]$

  for $i \leftarrow begin + 1$ *to* end **do**

    $A[i] \le key$? $swap(A[++split], A[i])$

  end for

  $split > k\_pos$? **return** $findk\_pos(A, k\_pos, begin, split - 1)$

  $split < k\_pos$? **return** $findk\_pos(A, k\_pos, split + 1, end)$

  return $split$

end function

function K_SORTED$(A, begin, end, k, count = 1)$

/* count 记录当前的段数, 每次调用 findk_pos，A 被分为 [begin,mid] 和 [mid+1,end] 两段，段数变为原来两倍，且左段元素小于右段，调用层数达到 logk 层算法结束，否则继续划分左右子序列 */

  $mid \leftarrow (end - begin)/2 + begin, count \leftarrow count * 2$

  $findk\_pos(A, mid, begin, end)$

  if $count == k$ **then**

划分 k 段，算法结束

    return A

  end if

  $k\_sorted(A, begin, mid, k, count)$

  $k\_sorted(A, mid + 1, end, k, count)$

end function

problem 6.9

算法分析 同样利用快速排序的思想，令螺钉为 A，螺母为 B：

1. 在 A 数组中拿一个，根据 A 和螺母的大小关系，可以分成三部分，B1：比螺钉小的，B2：比螺钉大的，B3：完全匹配的。

2. 用 B3，同样可以把 A 分为三部分，A1：比螺母小的，A2：比螺母大的，A3：完全匹配的。

3. B1 与 A1 匹配，B2 与 A2 匹配，分别执行上述算法，直至全部匹配。

具体算法实现请见算法 match：算法 2

算法时间复杂度 对于每次递归代价：

1. 首先寻找分割点，遍历了一次数组，时间复杂度为 $O(n)$。

2. 之后根据分割点遍历 A,B 两数组将其分为两部分，时间复杂度为 $O(n)$。

3. 最后将分割后两序列进行递归操作继续划分。因此每次递归操作总的代价为 $O(n)$。

因此可推得算法的递推方程为 $T(n) = 2T(n/2) + O(n)$。根据主定理可得时间复杂度为 $O(n\log n)$ 满足题目要求。

算法 2 match 算法

输入: 螺钉数组 A，螺母数组 B

输出: 螺钉螺母对应下标一一匹配后的数组

function MATCH(A, B, l, r)
/*找到分割点，mark 记录 B 等于 A 首元素的下标，
count 记录 B 中小于 A 的个数*/
  $count \leftarrow 0, mark \leftarrow 0$
  for $i \leftarrow l\ to\ r$ **do**
    $A[l] == B[i]\ ?\ mark = i$
    $A[l] > B[i]\ ?\ count+ = 1$
  end for
/*为 B 和 A 的左半部分分配 count 个元素*/
  $swap(A[l], A[l + count]),\ swap(B[mark], B[l + count])$
  $mark \leftarrow mark + count, i \leftarrow l,\ j \leftarrow r$
  while $i < mark$ **and** $j > mark$ **do**\\将 a 分成两部分
    while $i < mark$ **and** $a[i] < b[mark]$ **do**
      $i \leftarrow i + 1$
    end while
    while $j > mark$ **and** $a[j] < b[mark]$ **do**
      $j \leftarrow j - 1$
    end while
    $swap(a[i++], a[j--])$
  end while
  $i \leftarrow l,\ j \leftarrow r$
  while $i < mark$ **and** $j > mark$ **do**\\将 b 分成两部分
    while $i < mark$ **and** $b[i] < a[mark]$ **do**
      $i \leftarrow i + 1$
    end while
    while $j > mark$ **and** $b[j] < a[mark]$ **do**
      $j \leftarrow j - 1$
    end while
    $swap(b[i++], b[j--])$
  end while
  $l < mark\ ?\ match(A, B, l, mark - 1)$
  $r < mark\ ?\ match(A, B, mark + 1, r)$
end function

problem 6.10

(1)

算法分析 利用归并排序的思想，在归并排序中合并左右数组 A，B

1. 由归并排序定义可知，A、B 两数组已经有序。

2. 在合并过程中，就需要计算 a[i]，b[j] 分别来自左右两部分的逆序对数，同时遍历两个数组，对于遍历的两个数进行比较大小，如果 $a[i] < b[0]$ 显然没有逆序。

4. 如果 $a[i] > b[j]$，那么 $a[i+1 \cdots n] > b[j]$ 均成立，即逆序对数有 $n-i+1$ 个因此遍历时没遇到 $a[i] > b[j]$，逆序对数加 $n-i+1$ 即可。

具体算法实现请见算法 MergeSort：算法 3

算法时间复杂度 与归并排序算法相同，在合并函数中仅添加一条赋值语句，复杂度仍为 $O(n \log n)$ 符合题目要求。

(2)

算法分析 同样利用归并排序的思想，在归并排序中合并左右数组 A，B

1. 同样若 $a[i] < b[j]$ 必不存在广义逆序,而对于 $a[i] > b[j] \& a[i] > C \cdot b[j]$，那么 $a[i+1 \cdots n] > C \cdot b[j]$ 均成立，与算法 3 相似。

具体算法实现 在算法 3 合并操作 (算法第 24 行) 中添加判断条件：
若满足$L[i] > C \cdot R[j-1]$ 则 $sum \leftarrow sum + n1 - i + 1$ 即可。

算法时间复杂度 与算法 3 相同，复杂度仍为 $O(n \log n)$。

算法 3 MergeSort 算法

输入: 无序序列 A, l, r

输出: 总逆序对数和 sum

$sum \leftarrow 0$
function MERGESORT(A, l, r)
  $l == r$? **return**
  $mid \leftarrow \frac{l+r}{2}$
  $MergeSort(A, l, mid),\ MergeSort(A, mid + 1, r)$
  $Merge(A, l, mid, r)$
end function
function MERGE(A, l, mid, r)
  $n1 \leftarrow mid - l + 1,\ n2 \leftarrow r - mid$
`Let L[1..(n1+1)] and R[1..(n2+1)] be new arrays`
  for $i \leftarrow\ to\ n1$ **do**
    $L[i] \leftarrow A[l - i + 1]$
  end for
  for $i \leftarrow\ to\ n2$ **do**
    $R[i] \leftarrow A[mid + i]$
  end for
  $L[n1 + 1] \leftarrow \infty,\ R[n2 + 1] \leftarrow \infty$
  $i \leftarrow 1,\ j \leftarrow 1$
  for $k \leftarrow l\ to\ r$ **do**
    if $L[i] < R[j]$ **then**
      $A[k] \leftarrow L[i++]$
    else
      $A[k] \leftarrow R[j++]$
      $sum \leftarrow sum + n1 - i + 1$\\仅添加这句，更新 sum 逆序对和
    end if
  end for
end function

problem 6.13

(1)

①证明 易知当 R 中元素个数为 13 倍数相同元素均为 k 个，则 R 中存在 13 个常见元素。假设可以有 14 个常见元素：则这 14 个常见元素个数总和 $sum >= 14 \cdot \lceil \frac{n}{13} \rceil > n$，与总元素个数为 n 矛盾。
因此 R 中存在最多 13 个不同的常见元素，证毕。

②证明 x 为 $R[1..n]$ 中常见元素，则设 x 在 $R[1..\lfloor \frac{n}{2} \rfloor]$ 中有 k_1 个，在 $R[\lfloor \frac{n}{2} \rfloor + 1..n]$ 中有 k_2 个，则 $k_1 + k_2 >= \lceil \frac{n}{13} \rceil$，假设 x 在两个数组中均不为常见元素，则 $k_1 < \lceil \frac{n}{26} \rceil$，$k_2 < \lceil \frac{n}{26} \rceil$，即 $k_1 + k_2 < 2 \cdot \lceil \frac{n}{26} \rceil < \lceil \frac{n}{13} \rceil$，产生矛盾，因此 x 至少是两个数组中一个数组的常见元素。

③算法设计

(2)

(3)

存在

算法 5 $Majority$ 算法

Function $Majority\ (A[1 \cdots n])$

$res \leftarrow A[1],\ count \leftarrow 1$

for $i \leftarrow 2\ to\ n\ do$

  if $res == A[i]\ do\ count \leftarrow count + 1$

  else if $count == 0\ do\ res \leftarrow A[i], count \leftarrow 1$

  else $count \leftarrow count - 1$

$count \leftarrow 1$

for $i \leftarrow 1\ to\ n\ do$

  if $res == A[i]\ do\ count \leftarrow count + 1$

return $count \geq \lceil \frac{n}{2} \rceil\ ?\ res\ :\ 0$

算法 4 Majority 算法

function MAJORITY($A[1...n]$)
  $res \leftarrow A[1],\ count \leftarrow 1$
  for $i \leftarrow 2$ *to* n **do**
    if $res == A[i]$ **then**
      $count \leftarrow count + 1$
    else if $count == 1$ **then**
      $res = A[i]$
    else
      $count \leftarrow count - 1$
    end if
  end for
  $count \leftarrow 1$
  for $i \leftarrow$ *to* n **do**
    $A[i] == res\ ?\ count{+}{+}$
  end for
  return $count \geq \lceil \frac{n}{2} \rceil\ ?\ res\ :\ 0$
end function

(4)

problem 6.15

PART II

problem 7.1

problem 7.2

problem 7.3

problem 7.4

problem 7.6