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 $2^m = k \to m = \log k$ 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 $O(n\log k)$。

具体算法实现请见下页 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**
        $A[i] \leq key$? $swap(A[++split], A[i])$
    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。

具体算法实现请见下页 算法 match：算法 2

算法时间复杂度 对于每次递归代价：

首先寻找分割点，遍历了一次数组，时间复杂度为 $O(n)$。

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 $O(n)$。

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

```
1: function MATCH(A, B, l, r)
2: /*找到分割点，mark 记录 B 等于 A 首元素的下标，
3: count 记录 B 中小于 A 的个数*/
4:     count ← 0, mark ← 0
5:     for i ← l to r do
6:         A[l] == B[i] ? mark = i
7:         A[l] > B[i] ? count+ = 1
8:     end for
9: /*为 B 和 A 的左半部分分配 count 个元素*/
10:    swap(A[l], A[l + count]) swap(B[mark], B[l + count])
11:    mark ← mark + count, i ← l, j ← r
12:    while i < mark and j > mark do\\将 a 分成两部分
13:        while i < mark and a[i] < b[mark] do
14:            i ← i + 1
15:        end while
16:        while j > mark and a[j] < b[mark] do
17:            j ← j − 1
18:        end while
19:        swap(a[i + +], a[j − −])
20:    end while
21:    i ← l, j ← r
22:    while i < mark and j > mark do\\将 b 分成两部分
23:        while i < mark and b[i] < a[mark] do
24:            i ← i + 1
25:        end while
26:        while j > mark and b[j] < a[mark] do
27:            j ← j − 1
28:        end while
29:        swap(b[i + +], b[j − −])
30:    end while
31:    l < mark ? match(A, B, l, mark − 1)
32:    r < mark ? match(A, B, mark + 1, r)
33: end function
```

problem 6.10

problem 6.13

problem 6.15

PART II

problem 7.1

problem 7.2

problem 7.3

problem 7.4

problem 7.6