算法作业

丁元杰 17231164

2019 年 9 月 20 日

1

1.1

解 可知：

$$T(n) = \left\lceil \frac{n}{2} \right\rceil$$

由归纳法，$T(1) = 1$，$T(2) = 1$，满足边界条件。
如果

$$T(n-2) = \left\lceil \frac{n-2}{2} \right\rceil = \left\lceil \frac{n}{2} \right\rceil - 1$$

，那么

$$T(n) = \left\lceil \frac{n}{2} \right\rceil - 1 + 1 = \left\lceil \frac{n}{2} \right\rceil$$

综上所述，

$$T(n) = \left\lceil \frac{n}{2} \right\rceil = O(n)$$

1.2

解 可知：

$$T(n) = \log_2 n + 1$$

由归纳法，

$$T(1) = 1$$

边界满足条件，如果

$$T(n/2) = \log_2(n/2) + 1 = \log_2 n - \log_2 2 + 1 = \log_2 n$$

那么

$$T(n) = T(n/2) + 1 = \log_2 n + 1$$

由此得到：

$$T(n) = \log_2 n + 1 = O(\log n)$$

1.3

解 由主定理：

$$T(n) = O(n^{\log_2 2} \lg n) = O(n \log n)$$

1.4

解 由主定理：

$$T(n) = O(n^{\log_2 2}) = O(n)$$

1.5

解 由主定理：

$$T(n) = O(n^{\log_2 4}) = O(n^2)$$

1.6

解 由主定理：

$$T(n) = O(n^2)$$

1.7

解 由观察可以发现，设$n = 2^m$，则

$$\begin{aligned}
T(n) &= T(2^m) \\
&= \sum_{i=0}^{m} \log 2^i \\
&= \sum_{i=0}^{m} i \log 2 \\
&= \log 2 \sum_{i=0}^{m} i \\
&= \log 2 \cdot O(m^2) = O(m^2) \\
&= O(\log n^2)
\end{aligned}$$

2

算法思路

先考虑另一个问题KTH-ELEMENT(A, k)，表示求解数组A中第k小的元素，并用这个元素对原序列进行划分。设A排序后的数组为A'，那么$A[1..n]$中的第k小元素就被定义为$A'[k]$。

对于 KTH-ELEMENT 问题，需要使用到快速排序中的函数 PARTITION。在 KTH-ELEMENT$(A[1..n], k)$问题中，随机从$A[1..n]$中选取一个元素$A[p]$作为 pivot，设经过PARTITION之后的数组为A'，pivot 在A'中的下标为p'（即$A[p]$是A中的第p'小），那么

1. 如果$p' \geq k$，原问题化为子问题KTH-ELEMENT$(A'[1..p'], k)$
2. 如果$p' < k$，原问题化为子问题KTH-ELEMENT$(A'[p'+1..n], k - p')$

设原问题为ROUGHLY-SORT(A, k)，表示将A数组化为k重有序数组。那么可以通过不断地对原数组进行 KTH-ELEMENT$(A, n/2)$，直到长度小于n/k。或者形式地讲，解决 ROUGHLY-SORT$(A[1..n], k)$ 问题，

1. 如果$k=1$，那么$A[1..n]$即是所求
2. 否则先对序列进行 KTH-ELEMENT($A[1..n], n/2$)，然后分别进行 ROUGHLY-SORT($A[1..n/2], k/2$)，以及 ROUGHLY-SORT($A[n/2+1..n], k/2$)

伪代码

算法 1 将序列排成k重有序

输入: A数组，k有序数组的重数

输出: 排序后的数组A

function KthElement(A, k)
    randomly choose a number $p \in [1, n]$
    call Partition(A, p), and set the new location of $A[p]$ to p'
    if $p' \geq k$ **then**
        KthElement($A[1..p'], k$)
    else
        KthElement($A[p'+1..n], k-p'$)
    end if
end function
function RoughlySort($A[1..n], k$)
    if $k = 1$ **then**
        return
    else
        call KthElement($A[1..n], n/2$)
        call RoughlySort($A[1..n/2], k/2$)
        call RoughlySort($A[n/2+1..1], k/2$)
    end if
end function

复杂度分析

首先分析KTH-ELEMENT($A[1..n], k$)的复杂度$f(n)$。声明$f(n)$的期望复杂度为：

$$f(n) = O(n)$$

现用归纳法证明当n足够大时，$f(n) < 6n$。显然地有

$$f(1) = 1 < 6$$

假设对$\forall i < n$，都有

$$f(i) < 6i$$

设选取到用于 PARTITION 的 pivot 元素是$A[p]$，那么$f(n)$由这两部分组成：

1. n的时间用于 PARTITION 操作，此时间与p的选取无关

2. 以$\frac{1}{n}$的概率选到某个元素p作 pivot 之后，需要花费 $f(p)$或$f(n-p)$的时间进行后面的 KTH-ELEMENT 操作。更加明确地讲，是$\max\{p, n-p+1\}$

所以可以看出

$$\begin{aligned} f(n) &= n + \frac{1}{n}\sum_{i=1}^{n} f\left(\max\{n-i+1, i\}\right) \\ &= n + \frac{2}{n}\sum_{i=1}^{n/2} f\left(\frac{n}{2}+i-1\right) \\ &< n + \frac{2}{n}\times 6 \times \sum_{i=1}^{n/2}\frac{n}{2}+i-1 \\ &= n + \frac{12}{n}\times\left(\frac{n^2}{4}-\frac{n}{2}+\frac{n^2+2}{8}\right) \\ &= n + 6\times\frac{3n-4+\frac{2}{n}}{4} \\ &< \frac{11}{2}n - 6 + \frac{3}{n} \\ &= 5.5n + o(n) \\ &< 6n = O(n) \end{aligned}$$

所以可以得出结论

$$f(n) = O(n)$$

其次分析ROUGHLY-SORT的时间复杂度$T(n,k)$，可以发现递归式：

$$T(n,k) = \begin{cases} 1 & , k = 1 \\ 2T(n/2, k/2) + f(n) & , \text{else} \end{cases}$$

其中$f(n) = O(n)$。所以，可以得出，$T(n,k) = O(n\log k)$。

3

算法思路

考虑$A[1..n]$序列的差分序列$B[1..n-1]$，满足条件$B[i] = A[i+1] - A[i]$。
如果$A[i^*]$是A序列中的极大值，则必有$B[i^*-1] > 0$且$B[i^*] < 0$。
此时可以考虑在B序列上进行二分查找，设初始$l_0 = 1, r_0 = n-1$，可知$B[l_0] < 0$，而$B[r_0] > 0$。
接下来进行迭代，在第k次迭代中，维护循环不变性$l_k < r_k$，以及$B[l_k]B[r_k] < 0$：

1. 如果$l_k + 1 = r_k$，那么算法结束，$A[r_k]$就是所求的局部最大值。
2. 记$m_k = \left\lfloor \frac{l_k + r_k}{2} \right\rfloor$，显然有$l_k < m_k < r_k$，且以下两个条件之一成立：
   (a) 如果$B[l_k]B[m_k] < 0$，则$l_{k+1} = l_k$，$r_{k+1} = m_k$
   (b) 如果$B[r_k]B[m_k] < 0$，则$l_{k+1} = m_k$，$r_{k+1} = r_k$

算法 2 求局部最大值

输入: A数组，n数组长度

输出: 局部最大值在A中的下标

function LOCALMAXIMUM(A, n)
    $l \leftarrow 1,\ r \leftarrow n - 1$
    while $l < r$ **do**
        if $l + 1 = r$ **then**
            return r
        else
            $m \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor$
            $b_l \leftarrow A[l+1] - A[l]$
            $b_m \leftarrow A[m+1] - A[m]$
            if $b_l * b_m < 0$ **then**
                $r \leftarrow m$
            else
                $l \leftarrow m$
            end if
        end if
    end while
end function

伪代码

复杂度分析

设算法时间复杂度为$T(n)$，其中n为A数组长度。那么很明显的有：

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(n/2) + 1 & , \text{else} \end{cases}$$

显然，$T(n) = O(\log n)$

4