算法设计与分析 (2019年秋季学期)

第一次作业参考答案

1 对下面每一对表达式 (A, B)，请判断 A 和 B 之间的关系是 O, Ω 还是 Θ。注意他们之间可能满足多种关系。(每小题 5分，共 25 分)

1. $A = n^3 - 100n, B = n^2$;
2. $A = \log n, B = \log_{1.1} n$;
3. $A = 2^{2n}, B = 2^{3n}$;
4. $A = 2^{\log n}, B = n$;
5. $A = \log\log n, B = 10^{100}$.

解：

1. $A = \Omega(B)$;
2. $A = O(B), A = \Omega(B), A = \Theta(B)$;
3. $A = O(B)$;
4. $A = O(B), A = \Omega(B), A = \Theta(B)$;
5. $A = \Omega(B)$.

2 请给出 $T(n)$ 尽可能紧凑的渐进上界并予以说明，可以假定 n 是 2 的幂次。(每小题 4 分，共 28 分)

1.

$$T(1) = T(2) = 1$$
$$T(n) = T(n-2) + 1 \quad if \quad n > 2$$

2.

$$T(1) = 1$$
$$T(n) = T(n/2) + 1 \quad if \quad n > 1$$

3.

$$T(1) = 1$$
$$T(n) = T(n/2) + n \quad if \quad n > 1$$

4.

$$T(1) = 1$$
$$T(n) = 2T(n/2) + 1 \quad if \quad n > 1$$

5.

$$T(1) = 1$$
$$T(n) = 4T(n/2) + 1 \quad if \quad n > 1$$

6.

$$T(1) = 1$$
$$T(n) = 3T(n/2) + n^2 \quad if \quad n > 1$$

7.

$$T(1) = 1$$
$$T(n) = T(n/2) + \log n \quad if \quad n > 1$$

解：

1. $T(n) = O(n)$
2. $T(n) = O(\log n)$
3. $T(n) = O(n)$
4. $T(n) = O(n)$
5. $T(n) = O(n^2)$
6. $T(n) = O(n^2)$
7. $T(n) = O(\log^2 n)$

   原式展开为：

   $$\log n + \log(n/2) + \log(n/4) + \cdots + \log 2 + 1 = 1 + (1 + 2 + \cdots + \log n) \leq \sum_{i=1}^{\log n} i = O(\log^2 n)$$

3 k 路归并问题 (22 分)

现有 k 个有序数组 (从小到大排序)，每个数组中包含 n 个元素。您的任务是将它们合并成 1 个包含 kn 个元素的有序数组。首先来回忆一下课上讲的归并排序算法，它提供了一种合并有序数组的算法 $Merge$。如果我们有两个有序数组大小分别为 x 和 y，$Merge$ 算法可以用 $O(x+y)$ 的时间来合并这两个数组。

1. 如果我们应用 $Merge$ 算法先合并第一个和第二个数组，然后由前后两个数组合并后的数组与第三个合并，再与第四个合并，直到合并完 k 个数组。请分析这种合并策略的时间复杂度 (请用关于 k 和 n 的函数表示)。
2. 针对本题的任务，请给出一个更高效的算法，并分析它的时间复杂度。(提示：此题若取得满分，所设计算法的时间复杂度应为 $O(nk\log k)$。)

解：

1. 题目中给出的 $Merge$ 算法时间复杂度是线性的，根据题目中的策略对数组进行和并，每次和并的复杂度分别为 $n+n, 2n+n, \ldots, (k-1)n+n$。总的复杂度为：

$$\left(n\sum_{i=1}^{k-1} i\right) + (k-1)n = n\frac{k(k-1)}{2} + (k-1)n = n\frac{k^2-k}{2} + k - 1 = O(nk^2)$$

2. 一种更高效的做法是把 k 个有序数组平均分为两份递归进行合并得到两个数组，然后再合并这两个数组。这种方法的复杂度递归式为 $T(k) = 2T(k/2) + O(nk), T(1) = O(n)$，解出时间复杂度为 $O(nk\log k)$。算法实现可以参考 **Algorithm 1**。

Algorithm 1 $k\_Merge(A, l, r)$

Input:
k 个包含 n 个元素的有序数组, $A[1..k][1..n]$;
递归区间左端点, l;
递归区间右端点, r;
Output:
归并后的包含 $(r-l+1)n$ 个元素的有序数组;
if $l = r$ **then**
  return $A[l][1..n]$;
end if
$m \leftarrow \lfloor \frac{l+r}{2} \rfloor$;
return $Merge(k\_Merge(A, l, m), k\_Merge(A, m+1, r))$;

4 局部最小值问题 (25 分)

给定一个由 $n(n \geq 3)$ 个互不相同的整数组成的数组 $A[1..n]$，其满足 $A[1] > A[2]$ 并且 $A[n-1] < A[n]$。我们定义数组的**局部最小值**为比它的两个相邻元素 (如果存在) 都小的整数。换言之，$A[x]$ 是局部最小值当且仅当它满足 $A[x] < A[x-1]$ 并且 $A[x] < A[x+1] (1 < x < n)$。例如，下图所示数组中包含两个局部最小值，分别为 3 和 1。

9	3	7	2	1	4	5

求局部最小值显然有一个 $O(n)$ 的做法，仅需要扫描一遍整个数组就可以找到所有的局部最小值。请你给出一个算法可以在 $O(\log n)$ 的时间复杂度内找出一个数组的局部最小值。如果局部最小值有多个，仅需要找出任意一个局部最小值即可。(提示：我们给出的限制条件保证数组至少有一个局部最小值。)

解：

如果 n 等于 3，做法是显然的。考虑 n 大于 3 的情况, 此时令 $m = \lfloor \frac{n}{2} \rfloor$，来检查 $A[m-1], A[m], A[m+1]$ 之间的大小关系：

1. 如果 $A[m-1] > A[m]$ 并且 $A[m] < A[m+1]$，那么 A[m] 就是局部最小值，算法结束。
2. 如果 $A[m-1] < A[m] < A[m+1]$，那么 $A[1..m]$ 中一定存在局部最小值，我们递归处理数组 $A[1..m]$ 即可。
3. 如果 $A[m-1] > A[m] > A[m+1]$，那么 $A[m..n]$ 中一定存在局部最小值，我们递归处理数组 $A[m..n]$ 即可。
4. 如果 $A[m-1] > A[m]$ 并且 $A[m] < A[m+1]$，那么左右两个区间中都存在局部最小值，我们任选一个区间递归处理即可。

任何一种情况每次递归都缩减了一半的规模，因此可以得出递归式 $T(n) \leq T(n/2) + O(1)$，解出算法的时间复杂度为 $T(n) = O(\log n)$