计算机学院《算法设计与分析》
(2021 年秋季学期)

第一次作业参考答案

1 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 4 的整数次幂。(每小题 3 分，共 21 分)

1.

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

2.

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

3.

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

4.

$$T(1) = 1$$
$$T(n) = 8T(n/4) + 2n \quad if \quad n > 1$$

5.

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

6.

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

7.

$$T(1) = 1$$

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

答案：

1. $T(n) = O(n^2)$
2. $T(n) = O(n^2)$
3. $T(n) = O(2^n)$
4. $T(n) = O(n^{\frac{3}{2}})$
5. $T(n) = O(n \log n)$
6. $T(n) = O(n^{\log_2 3})$
7. $T(n) = O(n)$

2 k 路归并问题 (19 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。你的任务是将他们合并成 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的大小分别为 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）。（9 分）
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 $O(nk\log k)$）。（10 分）

答案：

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合并，每次合并的复杂度分别为 $n+n, 2n+n, \ldots, (k-1)n+n$。

   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。算法实现请参考 **Algorithm** 1。

   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 $O(nk\log k)$。

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 $Merge(k\_Merge(A, l, m), k\_Merge(A, m+1, r))$

3 填数字问题 (20 分)

给定一个长度为 n 的数组 $A[1..n]$，初始时数组中所有元素的值均为 0，现对其进行 n 次操作。第 i 次操作可分为两个步骤:

1. 先选出 A 数组长度最长且连续为 0 的区间，如果有多个这样的区间，则选择最左端的区间，记本次选定的闭区间为 $[l, r]$;
2. 对于闭区间 $[l, r]$，将 $A\left[\lfloor \frac{l+r}{2} \rfloor\right]$ 赋值为 i，其中 $\lfloor x \rfloor$ 表示对数 x 做向下取整。

例如 $n = 6$ 的情形，初始时数组为 $A = [0, 0, 0, 0, 0, 0]$。
第一次操作为选择区间 $[1, 6]$，赋值后为 $A = [0, 0, 1, 0, 0, 0]$;
第二次操作为选择区间 $[4, 6]$，赋值后为 $A = [0, 0, 1, 0, 2, 0]$;
第三次操作为选择区间 $[1, 2]$，赋值后为 $A = [3, 0, 1, 0, 2, 0]$;
第四次操作为选择区间 $[2, 2]$，赋值后为 $A = [3, 4, 1, 0, 2, 0]$;
第五次操作为选择区间 $[4, 4]$，赋值后为 $A = [3, 4, 1, 5, 2, 0]$;
第六次操作为选择区间 $[6, 6]$，赋值后为 $A = [3, 4, 1, 5, 2, 6]$，为所求。
请设计一个高效的算法求出 n 次操作后的数组，并分析其时间复杂度。

答案:
本题可以利用分治预先得到所有操作选择的区间，再根据规则排序依次操作赋值。

考虑若某一次操作的区间是 $[l, r]$，那么 $[l, mid)$ 和 $(mid, r]$ 是两个待操作的区间 (其中 $mid = \lfloor \frac{l+r}{2} \rfloor$)。则可以考虑如下分治算法生成所有需要操作的区间:

Algorithm 2 $SpiltAll(L, R)$

Input:
当前操作区间的左右断点 L, R;

Output:
当前区间最终分裂成的操作区间集合

if $L > R$ **then**
  return $\varnothing$
end if
$mid = \lfloor \frac{l+r}{2} \rfloor$
return $[L, R] \cup SpiltAll(L, mid-1) \cup SpiltAll(mid+1, r)$

之后，按照区间长度为第一关键字，区间左端点为第二关键字依次操作每个区间，得到 n 次操作后的数组。故总算法框架如 **Algorithm**3所示。

Algorithm 3 $GenArray(A, n)$

Input:

数组 A 及其长度 n

Output:

数组 A

```
LIST ← SpiltAll(1, n)
将 LIST 按照区间长度为第一关键字、区间左端点为第二关键字排序
for [l_i, r_i] ∈ LIST do
    A[⌊(l_i + r_i)/2⌋] ← i {有序枚举 LIST 里面的所有区间，记第 i 个被枚举区间为 [l_i, r_i]}
end for
return A
```

用分治得到所有可能被选择的区间需要 $T(n) = 2T(n/2) + O(1) = O(n)$ 的时间，后续排序选择区间需要 $O(n \log n)$ 的时间，故总的时间复杂度为 $O(n \log n)$。

4 奇数因子和问题 (20 分)

定义 $god(i)$ 为整数 i 的最大奇数因子，例如 $god(3) = 3$，$god(14) = 7$。

请你设计一个高效算法，计算一个整数区间 $[A, B]$ 内所有数的最大奇数因子和，即 $\sum_{i \in [A,B]} god(i)$，并分析该算法的时间复杂度。

例如，区间 $[3, 9]$ 的计算结果为 $3 + 1 + 5 + 3 + 7 + 1 + 9 = 29$。

答案：

考虑区间 $[a, b]$，将区间内的奇偶数分开处理。

首先考虑区间中的奇数，奇数 k 的最大奇数因子 $god(k)$ 为其自身 k，因此可以对区间内所有奇数求和。

然后考虑区间内的偶数 k，观察发现 $god(k) = god(k/2)$，因此处理区间 $[a, b]$ 中的偶数，等于处理区间 $[\lceil \frac{a}{2} \rceil, \lfloor \frac{b}{2} \rfloor]$ 中的整数。

详细算法如 **Algorithm 4**所示。

Algorithm 4 $sumgod(l, r)$

Input:

区间左端点, l

区间右端点, r

Output:

区间计算结果 $\sum_{i \in [A,B]} god(i)$

```
if l > r then
    return 0
end if
odd_l ← l − l%2 + 1
odd_r ← r + r%2 − 1
sum_odd ← (odd_l + odd_r) × (odd_r − odd_l + 1)/2
return sum_odd + sumgod(⌈l/2⌉, ⌊r/2⌋)
```

该算法将原问题转化为一个可在 $O(1)$ 时间内计算的式子与一个规模为 $n/2$ 的子问题，故可列出递归式 $T(n) = T(n/2) + O(1)$，解得 $T(n) = O(\log n)$。

5 区间计数问题 (20 分)

给定一个整数数组 $C = [c_1, c_2, \cdots, c_n]$，询问有多少个区间的区间和小于等于常数 X，即询问有多少对 l, r 满足 $l \le r$ 且 $\sum_{i=l}^{r} c_i \le X$。

例如，给定常数 $X = 3$，数组 $C = [1, 2, 3]$，$l = 1, r = 1$，$l = 2, r = 2$，$l = 3, r = 3$，和 $l = 1, r = 2$ 均满足条件，因此答案为 4。

请设计一个高效的算法来解决此问题，并分析该算法的时间复杂度。

答案:

本题可以借鉴求解数组中逆序数对个数的方法。在求解逆序数对个数的问题中，我们是要统计数组 C 中：$i<j, C[i]>C[j]$ 的数对 (i,j) 的个数，也即 $i<j, C[i]-C[j]>0$ 的数对 (i,j) 的个数。

在本题中，我们是要求解满足：$\sum_{k=l}^{r} C[k] \leq X$ 的 (l,r) 的个数。对于 $\sum_{k=l}^{r} C[k]$ 我们可以使用前缀和数组进行变换，记前缀和数组 $preSum[k]=\sum_{i=1}^{k} C[k]$。

对于区间 $[l,r]$ 的和有 $\sum_{k=l}^{r} C[k]=preSum[r]-preSum[l-1]$。所以本题的求解目标可以转变为 $l \leq r, preSum[r]-PreSum[l-1] \leq X$ 的 (l,r) 个数。过程与求解逆序数对个数相似。

具体实现参考 **Algorithm** 5。

Algorithm 5 $SortAndCount(L, X)$

Input:

前缀和数组 $preSum$; 区间和的上界 X;

Output:

满足 $l \leq r, preSum[r]-preSum[l-1] \leq X$ 的 (l,r) 个数;

将 $preSum$ 划分为两个子数组 A, B

$(r_a, A) \leftarrow SortAndCount(A, X)$

$(r_b, B) \leftarrow SortAndCount(B, X)$

$(r, L) \leftarrow MergeAndCount(A, B, X)$

return $r+r_a+r_b, L$

Algorithm 6 $MergeAndCount(A, B, X)$

Input:

前缀和数组 A, B; 区间和的上界 X;

Output:

满足 $B[r]-A[l] \leq X$ 的 (l,r) 个数;

$ans, r \leftarrow 0, 1$

$L \leftarrow \emptyset$

for $l \leftarrow 1$ **to** $A.length$ **do**

    while $r \leq B.length$ 并且 $B[r]-A[l] \leq X$ **do**

        $r \leftarrow r+1$

    end while

    $ans \leftarrow ans+r-1$

end for

$L \leftarrow Merge(A, B)$

return ans, L

每个问题划分为了两个子问题来解，两个子问题的合并过程所需时间复杂度为 $O(n)$，据此可以写出递归式 $T(n)=2T(n/2)+O(n)$。解得总的时间复杂度为 $O(n\log n)$。