算法设计与分析 Week 3

第三次作业-solution

Lecturer: 杨启哲 Last modified: 2025 年 10 月 20 日

1. 考虑算法 $SlowMinmax$，它是将算法 $Minmax$ 的检验条件 **if** $high - low = 1$ 修改为 **if** $high = low$，并对此算法做一些相应改变而得出的。这样，在算法 $SlowMinmax$ 中，当输入数组的大小为 1 时，递归停止。计算由此算法找出数组 $A[1,...,n]$ 中的最大值和最小值所需要的比较次数，这里 n 是 2 的幂。并解释为什么此算法的比较次数大于算法 $Minmax$ 的比较次数。

   Hint: 在这种情形下，初始条件是 $C(1) = 0$

   解答. 注意到在题设条件下，算法 $SlowMinmax$ 的递归式为

$$C(n) = \begin{cases} 0, & n = 1 \\ 2C(\frac{n}{2}) + 2, & n > 1 \end{cases}$$

   则考察 $n = 2^k$ 我们有:

$$\begin{aligned} C(n) &= 2C(\frac{n}{2}) + 2 \\ &= 2(2C(\frac{n}{4}) + 2) + 2 \\ &\vdots \\ &= 2^k C(\frac{n}{2^k}) + 2(1 + 2 + \cdots + 2^{k-1}) \\ &= 2^k C(1) + 2(1 + 2 + \cdots + 2^k) \\ &= 2(2^k - 1) \\ &= 2n - 2 \end{aligned}$$

   其比较次数大于书上 MINMAX 的算法,这是因为在这个算法中,递归深度更深了一层,MINMAX 递归到 $n = 2$ 就停止了，而 $SlowMinmax$ 则递归到 $n = 1$ 才停止。 □

2. 求解下列递推式，并针对每个递推式给出一个 Θ 界限:

   (1) $T(n) = 3T(\frac{n}{3}) + O(n)$

   (2) $T(n) = 9T(\frac{n}{2}) + 2n^3$

   (3) $T(n) = 3T(\frac{n}{3}) + 3$

   解答. 由主定理可得

   (1) $T(n) = \Theta(n \log n)$.

   (2) $T(n) = \Theta(n^{\log_2 9})$.

(3) $T(n)=\Theta(n)$.

特别的，后两个递推式我们可以算出严格的解：

(1) 设 $n=2^k$，则

$$\begin{aligned}
T(n) &= 9T(\frac{n}{2})+2n^3 \\
&= 9(9T(\frac{n}{4})+2(\frac{n}{2})^3)+2n^3 \\
&\vdots \\
&= 9^kT(1)+2n^3(1+\frac{9}{8}+\cdots+(\frac{9}{8})^{k-1}) \\
&= 9^kT(1)+2n^3\frac{(\frac{9}{8})^k-1}{\frac{9}{8}-1} \\
&= n^{\log_2 9}T(1)+16n^3((\frac{9}{8})^{\log_2 n}-1) \\
&= n^{\log_2 9}T(1)+16n^3(n^{\log_2 9-3}-1) \\
&= n^{\log_2 9}(T(1)+16)-16n^3=\Theta(n^{\log_2 9})
\end{aligned}$$

(2) 设 $n=3^k$，则

$$\begin{aligned}
T(n) &= 3T(\frac{n}{3})+3 \\
&= 3(3T(\frac{n}{9})+3)+3 \\
&\vdots \\
&= 3^kT(1)+3(1+3+\cdots+3^{k-1}) \\
&= 3^kT(1)+3\frac{3^k-1}{3-1} \\
&= (T(1)+\frac{3}{2})n-\frac{3}{2}=\Theta(n)
\end{aligned}$$

□

3. 给定 n 个元素互不相同的整数数组 $A[1,\ldots,n]$ 和整数 $k(1\leqslant k\leqslant n)$，现在希望返回该数组中前 k 小的元素，一个很自然的算法是先对数组进行排序，然后返回前 k 个元素。该算法的时间复杂度为 $O(n\log n)$，现在请设计一个时间复杂度为 $O(n)$ 的算法来解决该问题。注意到，在这里 k 不是常数，因此 $O(kn)$ 的算法并不符合要求。

解答. 注意到 k 不是常数，因此不能通过返回 k 次最小的元素来实现。

$O(n)$ 算法的关键在于我们只需要**返回前 k 小的元素，而不需要对它们进行排序**。因此，我们可以使用类似划分的思想来实现该算法，具体来说：

(1) 随机选定一个主元 x, 并将数组划分成两部分 $A_1=\{a\in A\mid a<x\}$ 和 $A_2=\{a\in A\mid a>x\}$。

(2) 若 $|A_1|=k-1$，则返回 $A_1\cup\{x\}$。

(3) 若 $|A_1|>k-1$，则在 A_1 中递归查找前 k 小的元素。

(4) 若 $|A_1| < k-1$，则在 A_2 中递归查找前 $k-|A_1|-1$ 小的元素，并将结果与 $A_1 \cup \{x\}$ 合并返回。

注意到上述算法的递推式为：

$$T(n) = T(\max\{|A_1|, |A_2|\}) + cn$$

由于每次划分后，期望地 $\max\{|A_1|, |A_2|\} \leqslant \frac{3}{4}n$，因此我们有 $T(n) = T(\frac{3}{4}n) + cn$，从而由主定理可知 $T(n) = \Theta(n)$。 □

Remark 0.1

为了叙述简单，我们给了个随机版本的划分算法，事实上如果不希望引入随机，那么可以使用上课时提到的 SELECT 算法进行主元的选取再进行划分。

4. 对于某个整数 $g \geqslant 3$，用 g 来表示算法 $Select$ 中每组的规模，导出用 g 表示的算法的运行时间。当 $g = 3, 7, 9, 11$ 时，哪个选择可以保证算法在最坏情况下执行的次数依旧是 $\Theta(n)$?

解答. 我们用 $2k+1$ 来表示 g 这个奇数，则算法的递推式为：

$$T(n) = T(\frac{1}{2k+1}n) + T(\frac{3k+1}{4k+2}n) + cn$$

注意到 $g = 3$ 时 $k = 1$，从而 $\frac{1}{2k+1} + \frac{3k+1}{4k+2} = 1$，而 $g > 5$ 时有 $k > 1$，则 $\frac{1}{2k+1} + \frac{3k+1}{4k+2} < 1$，并且该值随 g 增加而递减，从而递归展开可知：

- 当 $g = 3$ 时 $T(n) = \Theta(n \log n)$。
- 当 $g = 7, 9, 11$ 时 $T(n) = \Theta(n)$。

□

Remark 0.2

由上述分析可知，$g = 5$ 时算法有最快的运行速度。

一开始的答案版本混淆了 k 与 g，会有一些理解的困难，在新版已经完成了改正。

5. $3-SAT$ 问题的定义如下：其实例是一个合取范式，其中每个子句都包含三个变量的析取。例如，$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_5) \wedge (x_3 \vee \neg x_5 \vee x_6)$，其中 $x_1, x_2, x_3, x_4, x_5, x_6$ 是布尔变量。$3-SAT$ 问题是判断是否存在一种赋值方式，使得合取范式为真。

在课程的后半段，我们会知道该问题是一个 NP 完全问题。现在，考虑一个其受限的版本，假设将其变量从 $1, \ldots, n$ 编号，公式中的每个子句包含变量的编号差异在 ± 10 以内。请给出一个线性时间的算法，判断这样的公式是否是可满足的。

解答. 注意到变量具有局部性的性质，我们只需要保留 10 个变量的所有可能取值，从而得到一个线性时间的算法。我们先介绍一些记号，令输入的实例为 $C_1 \wedge C_2 \wedge \cdots \wedge C_m$，变量记为 $x_1, \ldots, x_n$，定义如下集合：

- $V_1 = \{C_i \mid \exists j \in \{1, 2, \ldots, 10\}\ x_j \in C_i \text{ or } \neg x_j \in C_i\}$
- $V_k = \{C_i \mid \exists j \in \{10k-9, 10k-8, \ldots, 10k\}\ x_j \in C_i \text{ or } \neg x_j \in C_i\} \setminus V_{k-1}$

注意到对每个集合 V_k 其满足出现的变量至多为 $x_{10k-19}, \ldots, x_{10k+10}$，从而可以构造算法如下:

(1) 枚举变量 $x_1, \ldots, x_{20}$ 的所有可能赋值，并记录下所有使得 V_1 中成真的对 $x_1 1, \ldots, x_{20}$ 的赋值，令该集合为 A。

(2) 若 A 为空，则返回 $False$。

(3) 对于 $j = 2, 3, \ldots k$，重复如下步骤:

   (i) 枚举集合 A 当前的每种赋值情况，对每个赋值情况，枚举 $x_{10j-9}, \ldots, x_{10j+10}$ 的所有可能赋值。

   (ii) 记录下所有使得 V_j 中成真的对 $x_{10j-9}, \ldots, x_{10j}$ 的赋值，枚举结束后将 A 更新成该集合。

(4) 若 A 不为空，则返回 $True$。

注意到 20 个变量的所有取值情况为 $2^{20} = 1048576$ 是常数，集合 A 的大小至多为 $2^1 0 = 1024$ 也是常数，从而上述算法的时间复杂性为 $O(m)$，即是线性时间的。 $\square$