Technische Universität München
Institut für Informatik
Theoretical Computer Science

WS 2017/18

Fundamental Algorithms 4 - Solution Examples

Exercise 1

1. Try the Recursion Tree Method for the following recurrence:
$$T(n) = T\left(\tfrac{1}{3}n\right) + T\left(\tfrac{2}{3}n\right) + \theta(n),$$
assuming that all occurring divisions are without remainder and $T(1) = 0$.
2. Show that the height of the recursion tree is in $O(\log(n))$.
3. What could be a flaw using the recursion tree method for such unbalanced trees?
4. Show that $T(n) \in \theta(n \log(n))$, anyway, by using the substitution method.

Solution:

1. Let c be the constant in the $\theta(n)$ term. We then obtain the recursion tree

$$\begin{tikzcd}
 & & & cn \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & c\cdot\frac{1}{3}n \arrow[dl, dash] \arrow[dr, dash] & & & & c\cdot\frac{2}{3}n \arrow[dl, dash] \arrow[dr, dash] & \\
c\cdot\frac{1}{9}n & & c\cdot\frac{2}{9}n & & c\cdot\frac{2}{9}n & & c\cdot\frac{4}{9}n \\
\iddots\ \ddots & & \iddots\ \ddots & & \iddots\ \ddots & & \iddots\ \ddots
\end{tikzcd}$$

On each level, we obtain a total of cn operations.

2. The longest path in the recursion tree is the rightmost path with problem size $n \to \frac{2}{3}cn \to (\frac{2}{3})^2 cn \to \cdots \to 1$ until we stop at problem size 1. The height h of the tree can be determined via the equation $\min_h (\frac{2}{3})^h cn \leq 1$, leading to $h = \lceil \log_{\frac{3}{2}} cn \rceil$ (since $\log_a b = -\log_{\frac{1}{a}} b$).

We could expect the total cost to be $O(cn \log_{\frac{3}{2}} n) = O(n \log n)$.

3. Problem: If the tree was a complete binary tree, we would have
$$2^h = 2^{\log_{\frac{3}{2}} n} = 2^{\frac{\log_2 n}{\log_2 \frac{3}{2}}} = n^{\frac{1}{\log_2 \frac{3}{2}}} = n^{\frac{\log_{\frac{3}{2}} 2}{\log_{\frac{3}{2}} \frac{3}{2}}} = n^{\log_{\frac{3}{2}} 2}$$
leaves. Thus, the number of terms would be $\Theta(n^{\log_{\frac{3}{2}} 2})$ on the last level. As $\log_{\frac{3}{2}} 2 \approx 1.7095 > 1$, this especially means that the number of terms would be $\omega(n)$. Hence, the simple approach of assuming constant effort c for $T(1)$ on the final level does no longer work: In that case, the costs would sum up to $\Theta(n^d)$ on the last level – and not cn!

Hence, we'd have to explicitly consider that the tree starts to thin out much earlier (starting at level $1 + \log_3 n$), and we would have to examine the exact cost on all subsequent levels, which is more tedious than our tree diagram suggests.

4. We simplify and assume that the total cost is $T(n) = an\log n \in \theta(n \log n)$ and use the substitution method to verify this:

$$\begin{aligned}
T(n) &= T(\tfrac{1}{3}n) + T(\tfrac{2}{3}n) + cn \\
&= a(\tfrac{1}{3}n)\log(\tfrac{1}{3}n) + a(\tfrac{2}{3}n)\log(\tfrac{2}{3}n) + cn \\
&= \tfrac{1}{3}an(\log\tfrac{1}{3} + \log n) + \tfrac{2}{3}an(\log\tfrac{2}{3} + \log(n)) + cn \\
&= an\log n - a\left(\tfrac{1}{3}n\log 3 + \tfrac{2}{3}n\log\tfrac{3}{2}\right) + cn \\
&= an\log n - an\left(\tfrac{1}{3}\log 3 + \tfrac{2}{3}\log 3 - \tfrac{2}{3}\log 2\right) + cn \\
&= an\log n - an\left(\log 3 - \tfrac{2}{3}\right) + cn.
\end{aligned}$$

By choosing a such that

$$an\left(\log 3 - \tfrac{2}{3}\right) = cn,$$

i.e.

$$a = \frac{c}{\log 3 - \frac{2}{3}},$$

we obtain

$$T(n) = an\log n$$

Exercise 2

For the so-called BFPRT Algorithm, an algorithm to determine the *median* element of an array, we obtain the following (slightly simplified) recurrence equation for its running time $T(n)$ (depending on the number n of elements):

$$T(n) = s(n,k) + T\left(\tfrac{1}{k}n\right) + T\left(\tfrac{l}{2k}n\right).$$

k and l are parameters (k usually small, for example $k = 3$ or $k = 5$) where $k = 2l + 1$. For the function s, we can assume $s(n,k) \in \Theta(n\log k)$.

1. Show that $T(n) \in O(n)$.
2. Does it make sense to use growing values for k (and l, respectively)?

Solution:
We try to prove the claim by inserting the assumed solution $T(n) \le cn$ into the recurrence equation:

$$\begin{aligned}
cn &\ge s(n,k) + c\frac{n}{k} + c\frac{l}{2k}n \\
cn\left(1 - \frac{1}{k} - \frac{l}{2k}\right) &\ge s(n,k)
\end{aligned}$$

As $s(n,k) \in \Theta(n\log k)$, there is a constant C_s such that $s(n,k) \ge C_s n\log k$ for large n. Therefore,

$$cn\left(1 - \frac{1}{k} - \frac{l}{2k}\right) \ge s(n,k) \ge C_s n\log k,$$

and thus

$$c \ge \frac{C_s\log k}{1 - \frac{1}{k} - \frac{l}{2k}} \in O(\log k)$$

Hence, we can choose a suitable, large enough c that is independent of n, and thus prove $T(n) \in O(n)$. But, c has to slightly grow with k, as $c \in O(\log k)$, consequently k should be of limited size.