Technische Universität München
Institut für Informatik
Scientific Computing in Computer Science

WS 2011/12
Worksheet
22.11.2011

Fundamental Algorithms 3

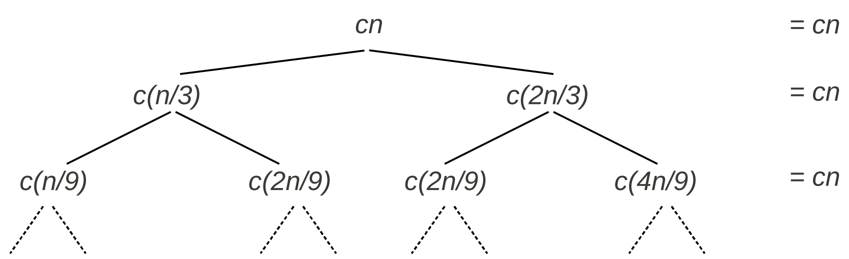
Exercise 1

Try the Recursion Tree Method (compare lecture) for the following recurrence:

$$T(n) = T(n/3) + T(2n/3) + O(n)$$

Show that the height of the recursion tree is in $O(\log(n))$.

- We assume that all occurring n are multiples of 3. Further, let c be the constant in the $O(n)$ term. We then obtain the recursion tree



  On each level, we obviously obtain cn operations, independent of the level.

- The longest path in the recursion tree is the rightmost path with problem size $n \to 2/3n \to (2/3)^2 n \to \cdots \to 1$ until we stop at problem size 1. The height h of the tree can be determined via the equation $(2/3)^h n = 1$, leading to $h = \log_{3/2} n$.

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

What could be a flaw using the recursion tree method for such unbalanced trees?
Show that $T(n) \in O(n \log(n))$, anyway, by using the substitution method.

- Problem: If the tree was a complete binary tree, we would have $2^{\log_{3/2} n} = n^{\log_{3/2} 2}$ leaves. Assuming constant effort c for T(1), on the last level the costs would sum up to $\Theta(cn^{\log_{3/2} 2})$. Thus, on that level, the cost would be $\omega(n \log n)$ – and not cn! Of course, the tree thins out starting at level $1 + \log_3 n$, thus we would have to count the exact cost on the subsequent levels.
- We simplify and assume that the total cost are $O(n \log n)$ and use the substitution method to verify this:

  Assuming that $T(n) \leq an \log n$ for a suitable constant a, we obtain

$$\begin{aligned}
T(n) &\leq T(n/3) + T(2n/3) + cn \\
&\leq a(n/3)\log(n/3) + a(2n/3)\log(2n/3) + cn \\
&= a3n/3 \log n - a\left((n/3)\log 3 + (2n/3)\log(3/2)\right) + cn \\
&= an \log n - a\left((n/3)\log 3 + (2n/3)\log 3 - (2n/3)\log 2\right) + cn \\
&= an \log n - an\left(\log 3 - 2/3 \log 2\right) + cn \\
&\leq an \log n
\end{aligned}$$

  for $d \geq c/(\log 3 - 2/3 \log 2)$.

Exercise 2

Consider a partitioning algorithm that, in the worst case, will partition an array of m elements into two partitions of size $\lfloor \epsilon m \rfloor$ and $\lceil (1-\epsilon)m \rceil$, where ϵ is fixed, and $0 < \epsilon < 1$. Show that a quicksort algorithm based on this partitioning has a worst-case complexity of $O(n \log n)$.

Solution:

Again, we will only count comparisons between array elements.

Using that the partitioning step will require at most n comparisons, we get the following recurrence for the necessary number $C(n)$ of comparisons:

$$\begin{aligned}
C(1) &= 0 \\
C(n) &= C(\epsilon n) + C((1-\epsilon)n) + n
\end{aligned}$$

We guess $C(n) := an \log_2 n + b$ as the solution, and try to find constants a and b such that the recurrence is satisfied:

case $n = 1$**:**

$$C(1) = a \cdot 1 \cdot \log_2 1 + b = 0 \quad \Leftrightarrow b = 0,$$

hence, $C(n) = an \log_2 n$.

case $n > 1$**:** We insert our guess into the recurrence:

$$
\begin{aligned}
a n \log_2 n = C(n) &= C(\epsilon n) + C((1-\epsilon)n) + n \\
\Leftrightarrow \quad a n \log_2 n &= a\epsilon n \log_2(\epsilon n) + a(1-\epsilon)n \log_2((1-\epsilon)n) + n \\
\Leftrightarrow \quad a n \log_2 n &= a\epsilon n\,(\log_2 \epsilon + \log_2 n) + a(1-\epsilon)n\,(\log_2(1-\epsilon) + \log_2 n) + n \\
\Leftrightarrow \quad a n \log_2 n &= a\epsilon n \log_2 \epsilon + a\epsilon n \log_2 n + \\
&\quad\; a(1-\epsilon)n \log_2(1-\epsilon) + a(1-\epsilon)n \log_2 n + n \\
\Leftrightarrow \quad a n \log_2 n &= a\epsilon n \log_2 \epsilon + a\epsilon n \log_2 n + \\
&\quad\; a n \log_2(1-\epsilon) - a\epsilon n \log_2(1-\epsilon) + a n \log_2 n - a\epsilon n \log_2 n + n \\
\Leftrightarrow \quad 0 &= a\epsilon n \log_2 \epsilon + a n \log_2(1-\epsilon) - a\epsilon n \log_2(1-\epsilon) + n \\
\Leftrightarrow \quad 0 &= a n\,(\epsilon \log_2 \epsilon + (1-\epsilon)\log_2(1-\epsilon)) + n \\
\Leftrightarrow \quad a &= \frac{-1}{\epsilon \log_2 \epsilon + (1-\epsilon)\log_2(1-\epsilon)}
\end{aligned}
$$

Thus, the recurrence is satisfied if

$$C(n) = \frac{-n \log_2 n}{\epsilon \log_2 \epsilon + (1-\epsilon)\log_2(1-\epsilon)}$$

Note that the constant a will be very large for values of ϵ that are close to either 0 or 1. Thus, even very bad partitions will not destroy the $O(n \log n)$ complexity, provided that the respective partition sizes are bounded by ϵn and $(1-\epsilon)n$. However, bad partitions will still lead to slow algorithms due to the large constant factor involved.