Institut für Informatik
der Universität München
Prof. M. Hofmann
Dr. Andreas Abel
Dipl. Inf. Roland Axelsson

SS 2009

20. Mai 2009

Musterlösung
Algorithmen und Datenstrukturen

Blatt 1

Aufgabe H-1:

Beweisen oder widerlegen Sie die folgenden Behauptungen:

a) $f(n) + g(n) = \Theta(max(f(n), g(n)))$.

- Obere Schranke: Es gilt $f + g \leq 2max(f, g)$

- Untere Schranke: Es gilt $f + g \geq max(f, g)$ [da $f, g \geq 0$]

b) Wenn $f(n) = O(g(n))$ ist, so auch $2^{f(n)} = O(2^{g(n)})$.

Es gibt c, N, so dass für alle $n \geq N$, $f(n) \leq c \cdot g(n)$. Daraus erhalten wir $2^{f(n)} \leq (2^{g(n)})^c$. Falls $c > 1$, gilt nicht dass $2^{f(n)} \leq c' \cdot 2^{g(n)}$ für ein fixes c'.

Gegenbeispiel: $2n = O(n)$, aber $2^{2n} = 4^n = \omega(2^n)$.

c) $f(n) = \Theta(f(\frac{n}{3}))$.

Abwegig, gilt zwar für rationale Funktionen oder Logarithmus, aber

$2^n = \omega(2^{\frac{n}{3}}) = \omega((2^{\frac{1}{3}})^n)$.

d) $(f(n) + g(n))^2 = O(f(n)^2) + O(g(n)^2)$.

$(f(n) + g(n))^2 = f(n)^2 + 2f(n)g(n) + g(n)^2$, also gilt die Behauptung, falls $f(n)g(n) = O(f(n)^2) + O(g(n)^2)$. Sei n_0 so, dass $f(n), g(n) > 0$ für alle $\geq n_0$.

Fall 1: $f(n) \geq g(n)$, dann ist $f(n)g(n) \leq f(n)^2 \leq f(n)^2 + g(n)^2$

Fall 2: $f(n) < g(n)$, dual.

Aufgabe H-2:

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

$a = 9, b = 3, f(n) = n, \log_b a = 2$

Da $f(n) = O(n^{2-\epsilon})$ für $\epsilon = 1$, ist Fall 1 des Master-Theorems anwendbar und somit $T(n) = \Theta(n^2)$.

b) $T(n) = 3T(\frac{3n}{4}) + (n^2 + n)^2$

$a = 3, b = \frac{4}{3}, f(n) = n^4 + 2n^3 + n^2, 3 < \log_b a < 4$

Es ist also $f(n) = \Omega(n^{\log_b a + \epsilon})$ für ein $\epsilon > 0$ und desweiteren gilt:

$a \cdot f(\frac{n}{b}) = 3 \cdot f(\frac{3}{4}n) = \frac{243}{256}n^4 + O(n^3) <= \frac{243}{256}f(n)$ für genügend große n. Anwendung von Fall 3 des Master-Theorems: $T(n) = \Theta(f(n)) = \Theta(n^4)$.

c) $T(n) = 2T(\frac{n}{4}) + \sqrt{n \log n}$

$a = 2, b = 4, f(n) = n^{\frac{1}{2}} \cdot (\log n)^{\frac{1}{2}}, \log_b a = \frac{1}{2}$

Es gilt $f(n) = \omega(n^{\frac{1}{2}})$ aber $f(n) \neq \Omega(n^{\frac{1}{2}+\epsilon})$ für alle $\epsilon > 0$. Deshalb ist das Master-Theorem nicht anwendbar!

d) $T(n) = 32T(\frac{n}{4}) + n^2\sqrt{n}$

$a = 32, b = 4, f(n) = n^{\frac{5}{2}}, \log_b a = \frac{5}{2}$

Da $f(n) = \Theta(n^{\frac{5}{2}})$, kann Fall 2 des Master-Theorems angewendet werden und somit ist $T(n) = \Theta(n^{\frac{5}{2}} \log n)$.

Aufgabe H-3:

MERGE(A, p, q, r)	Zeit	wie oft?
$i \leftarrow p; j \leftarrow q+1; k \leftarrow 1$	c_1	1
while $i \leq q$ **and** $j \leq r$ **do**	c_2	K
if $A[i] \leq A[j]$	c_3	K
then $B[k] \leftarrow A[i]; i \leftarrow i+1$	c_4	$I - p$
else $B[k] \leftarrow A[j]; j \leftarrow j+1$	c_5	$J - (q+1)$
$k \leftarrow k+1$	c_6	K
if $i \leq q$ **then** $A[r-q+i \ldots r] \leftarrow A[i \ldots q]$	$c_7 + (q - I + 1)c_8$	1
$A[p \ldots p+k-2] \leftarrow B[1 \ldots k-1]$	$c_9 + (K-1)c_{10}$	1

Dabei sei K der Wert der Variablen k nach Ablauf des Programms. Ebenso sei dies mit I für i und J für j. Man erhält so:

$T(n) = c_1 + (c_2 + c_3 + c_6)K + c_4(I - p) + c_5(J - (q+1)) + c_7 + (q - I + 1)c_8 + c_9 + (K-1)c_{10}$.

Dies ist $O(n)$, weil

$$\begin{aligned} K &\leq q - p + 1 + r - (q+1) + 1 + 1 = r - p + 2 = n + 1 \\ I - p &\leq q + 1 - p \leq n \\ J - (q+1) &\leq r + 1 - (q+1) \leq n \\ q - J + 1 &\leq q - p + 1 \leq n \end{aligned}$$

Aufgabe H-4:

ContainsSum(A,r):	$n = \text{length}[A]$
MergeSort(A)	$\Theta(n \log n)$
$l \leftarrow 1, u \leftarrow \text{length}[A]$	c_1
while $l \leq u$ do	$c_2 \cdot n$
$\text{sum} \leftarrow A[l] + A[u]$	$c_3 \cdot n$
if $\text{sum} = r$ then return true	$c_4 \cdot n$
if $\text{sum} < r$ then $l = l + 1$	$c_5 \cdot n$
if $\text{sum} > r$ then $u = u - 1$	$c_6 \cdot n$
return false	c_7

Komplexität insgesamt: $\Theta(n \log n) + O(n) = \Theta(n \log n)$

Korrektheit formal

Es gelten folgende Invarianten zu Beginn der Schleife:

a) Für $i < l$ ist $A[i] + A[j] \neq r$ für alle j und alle $i \neq l$

b) Für $i > u$ ist $A[i] + A[j] \neq r$ für alle j und alle $i \neq l$

Falls $l > u$, ist damit $A[i] + A[j] \neq r$ für alle j und alle $i \neq l$. Deswegen ist die Antwort `false` korrekt.

Die Invariante gilt zu Beginn der Schleife, da die Quantifikation leer ist.

Erhalt der Invariante:

Falls sum kleiner r ist, wird l auf $l' = l + 1$ erhöht. u bleibt, also ist b) trivialerweise erfüllt. Wir beweisen:

Für $i < l + 1$ ist $A[i] + A[j] \neq r$ für alle j.

Es genügt, diese Aussage für $i = l$ zu zeigen, da sie für $i < l$ nach a) schon erfüllt ist.

Es ist $A[l] + A[u] < r$, da $\text{sum} < r$, also ist wegen der Sortierung automatisch $A[l] + A[j] < r$ für alle $j \leq u$. Für $j > u$ gilt aber nach b) schon $A[l] + A[j] \neq r$. Also gilt a) auch für $l + 1$.

Der Fall $\text{sum} > r$ verläuft analog.