Algorithmen und Datenstrukturen
WS 2018
von Luxburg/Lohaus

Präsenzübung 1

Besprechung: 22.10.18 – 27.10.18

Aufgabe 1: Logarithmus (mündlich, keine Punkte)

Der Logarithmus einer Zahl $y \in \mathbb{R}_{>0}$ zur Basis $b \in \mathbb{R}_{>0} \backslash \{1\}$ ist die Zahl x, mit der man b potenzieren muss, um y zu erhalten:

$$\log_b(y) = x \iff b^x = y.$$

(a) Zeigen Sie, dass $b^{\log_b(a)} = a$.

(b) Zeigen Sie:

(i) $\log_b(x \cdot y) = \log_b(x) + \log_b(y)$ (ii) $\log_b(x^r) = r \log_b x$.

(c) Vereinfachen Sie

$$-\frac{1}{3} \log(x^2 y^{-2} z) + \frac{1}{3} \log(x^{-1} y z).$$

(d) Beweisen Sie das Gesetz zur Basisumrechnung ($a > 0$ und $a \neq 1$):

$$\log_a x = \frac{\log_b x}{\log_b a}.$$

(e) Berechnen Sie

(i) $\log_2 \frac{32}{10} + 2 \log_2 \sqrt{10}$ (ii) $2^{\log_4(n)}$.

Aufgabe 2: Landau-Notation (mündlich, keine Punkte)

Bestimmen Sie für alle folgenden Beispiele, welcher der drei folgenden Fälle vorliegt: $f \in o(g)$, oder $f \in \omega(g)$, oder $f \in \Theta(g)$.

	$f(n)$	$g(n)$		$f(n)$	$g(n)$
(a)	$n/2$	$4n + 250$	(e)	$n^{1.01}$	$n \log(n)^5$
(b)	$10n^2 + 8n + 100$	n^3	(f)	2^n	2^{n+1}
(c)	$10 \cdot \log(n)$	$\log(n^2)$	(g)	$n!$	2^n
(d)	n	$n \log(n)$	(h)	$(\log_2 n)^{\log_2 n}$	$2^{(\log_2 n)^2}$

Aufgabe 3: Induktion (mündlich, keine Punkte)

Folgender rekursiver Algorithmus wird bei Eingabe $n \geq 1$ insgesamt $C(n)$ Mal aufgerufen (d.h. Zeile 1 ausgeführt).

```
1: function StupidAlg(n)
2:     if n = 1 then
3:         return 1
4:     else
5:         return StupidAlg(n − 1) · StupidAlg(n − 1)
6:     end if
7: end function
```

(a) Was berechnet der Algorithmus?

Web page: http://www.tml.cs.uni-tuebingen.de/teaching/2018_algorithmen/
Login: “machine”, Password: “learning”

(b) Ermitteln Sie $C(n)$. Beweisen Sie durch vollständige Induktion.

Aufgabe 4: Laufzeitanalyse (mündlich, keine Punkte)

Die folgende Funktion Fun erhält als Eingabe ein Array $X = [x_1, x_2, \ldots, x_n]$ der Länge n. Fun ruft sich rekursiv mit Teilarrays auf, wobei z.B. $X[5 \ldots 42]$ das Teilarray der Indizes $5 \ldots 42$ bezeichnet.

```
function Fun(X)
    ℓ ← length(X)
    if ℓ ≤ 1 then
        Print("Hallo")
        return
    end if
    p ← ⌈ℓ/3⌉
    Fun(X[1 ... p])
    Fun(X[(ℓ − p + 1) ... ℓ])
    x ← 0
    while x < ℓ do
        x ← x + √ℓ
    end while
end function
```

a) Bestimmen Sie eine Rekurrenzgleichung für die Laufzeit von Fun in Abhängigkeit von n (in der Form, wie wir sie für das Master-Theorem benötigen).

b) Nutzen Sie das Master-Theorem, um die resultierende asymptotische Laufzeit zu ermitteln.

c) Nehmen Sie an, Fun wird mit einem Array der Länge n aufgerufen, wobei n eine 3er-Potenz ist. Wie oft wird "Hallo" ausgegeben?