Algorithmische Mathematik I

Wintersemester 2009/2010
Prof. Dr. Mario Bebendorf
Dr. Jan Hamaekers

universitätbonn

Übungsblatt 5.

Abgabe am **Mittwoch, 25.11.2009 (vor der Vorlesung)**.

Aufgabe 1. (Zwei–Term–Rekursion)

Für $n = 0, 1, 2, \ldots$ seien die Integrale

$$y_n := \int_0^1 \frac{x^n}{x+a}\, dx$$

für exakt gegebenes $a > 0$ gegeben.

a) Man leite die Rekursion

$$y_0 = \ln(1+a) - \ln a,$$
$$y_n = \frac{1}{n} - a\, y_{n-1}, \quad n > 0,$$

her.

b) Man interpretiere y_n als Funktion des Startwerts y_0. Wie lautet dann die Konditionszahl dieser Funktion?

c) Welche Konditionszahl hat die Formel $y_0(a)$?

(10 Punkte)

Aufgabe 2. (Christoffel–Darboux)

Sei $\{p_n\}$ eine Folge reeller Orthogonalpolynome und

$$p_n(x) = (A_n x + B_n)p_{n-1}(x) - C_n p_{n-2}(x), \quad n > 0,$$

ihre dreigliedrige Rekursionsformel mit $C_n = A_n/A_{n-1}$, $A_0 = A_1$, $p_{-1} \equiv 0$. Man beweise die Formel von Christoffel–Darboux für $n \geq 0$:

$$\sum_{k=0}^{n} p_k(x)p_k(y) = \frac{1}{A_{n+1}} \frac{p_{n+1}(x)p_n(y) - p_n(x)p_{n+1}(y)}{x-y} \qquad \text{für } x \neq y.$$

(10 Punkte)

Aufgabe 3. (Asymptotische Laufzeitnotationen)

Zur Geschwindigkeits- und Speicherverbrauchsanalyse bei Algorithmen verwendet man die Notationen

$$O(g(n)) := \{f\colon \mathbb{N} \to \mathbb{N} \,|\, \exists c > 0 \text{ und } \exists n_0 \in \mathbb{N} \text{ mit } f(n) \leq c \cdot g(n)\ \forall n \geq n_0\},$$
$$o(g(n)) := \{f\colon \mathbb{N} \to \mathbb{N} \,|\, \forall c > 0\ \exists n_0 \in \mathbb{N} \text{ mit } f(n) \leq c \cdot g(n)\ \forall n \geq n_0\},$$
$$\Omega(g(n)) := \{f\colon \mathbb{N} \to \mathbb{N} \,|\, \exists c > 0 \text{ und } \exists n_0 \in \mathbb{N} \text{ mit } f(n) \geq c \cdot g(n)\ \forall n \geq n_0\},$$
$$\Theta(g(n)) := O(g(n)) \cap \Omega(g(n)).$$

Wenn bspw. die Anzahl der Rechenschritte eines Programms durch $f(n)$ in Abhängigkeit der Eingabegröße n gegeben ist, bedeuten obige Definitonen

$f(n) = O(g(n))$**:** auf *jedem* Rechner benötigt man *maximal* irgendeine Konstante mal g Rechenschritte (spätestens ab einem gewissen n).

$f(n) = o(g(n))$**:** Die Laufzeit von f hat *echt kleinere* Größenordnung als g (man könnte schreiben $O(f(n)) < O(g(n)) \Leftrightarrow f(n) = o(g(n))$).

$f(n) = \Omega(g(n))$**:** man braucht *mindestens* $c \cdot g(n)$ Rechenoperationen (untere Schranke).

$f(n) = \Theta(g(n))$**:** wegen $f(n) = O(g(n))$ braucht f maximal $\bar{c} \cdot g$ Schritte und wegen $f(n) = \Omega(g(n))$ braucht f minimal $\underline{c} \cdot g$ Schritte.

a) Sind die folgenden Aussagen wahr oder falsch? Führen Sie einen Beweis.

1. $n^6 = O(2^n)$
2. $\log(n+1) = O(\log n)$
3. $42n^3 + 13n^2 + 2n + 500 = O(n^3)$
4. $42n^3 + 13n^2 + 2n + 500 = o(n^2)$
5. $g(n) = o(f(n))$ impliziert $f(n) + g(n) = \Theta(f(n))$
6. $\frac{n^{15}}{3^n} = O(1)$
7. $f(n) = \Theta(g(n))$ impliziert $g(n) = \Theta(f(n))$

b) Sortieren Sie die folgenden Funktionen bezüglich ihrer Größenordnung, d.h. geben Sie eine Reihenfolge $g_1, \ldots, g_k$ an mit $g_i = \Omega(g_{i+1})$. Geben Sie dabei an, wenn mehrere g_i dieselbe Größenordnung haben (d.h. wenn $g_i(n) = \Theta(g_j(n))$).

$$2^{\log(n)}, \quad (\sqrt{2})^{\log(n)}, \quad n^2, \quad (\log(n))!, \quad \left(\frac{3}{2}\right)^n, \quad n^3,$$

$$\log^2(n), \quad \log(n!), \quad 2^{(2^n)}, \quad n^{\frac{1}{\log(n)}}, \quad 2^{\log(n)}, \quad 2^{\sqrt{2\log(n)}},$$
$$10^4, \quad e^n, \quad 2^n, \quad n\log(n), \quad 2^{2^{n+1}}, \quad n!$$

(10 Punkte)

Aufgabe 4. (Suchperformance)

Sei $T(n)$ die maximale Anzahl von Vergleichen bei der sequentiellen Suche nach einem Wert a in einem unsortierten Array $A_1, \ldots, A_n$ der Größe n. Das Array enthalte kein Element doppelt.

a) Zeigen Sie, dass für das *Mittel* gilt: $\overline{T}(n) = \Theta(n)$. Dabei sei jede Permutation des Arrays gleich wahrscheinlich.

b) Falls das Eingabearray der Größe nach sortiert wäre, könnte man auch *binäre Suche* durchführen. Binäre Suche nach a in einem *sortierten* Array mit n Elementen prüft ($a \leq A_{n/2}$?) und wird dann ggf. rekursiv auf $A_1, \ldots, A_{n/2}$ aufgerufen. Andernfalls wird es rekursiv für $A_{n/2+1}, \ldots, A_n$ aufgerufen. Falls $n = 1$, wird ($a = A_1$?) geprüft und ggf. terminiert. Zeigen Sie, daß binäre Suche $O(\log n)$ viele Vergleiche benötigt.

c) Nun soll nach mehreren Werten $a_1, \ldots, a_k$ mit $k < n$ in einem unsortierten Array gesucht werden. Es kann dabei durchaus sein, daß k von n abhängt, z.B. $k = k(n) = \frac{n}{2}$. Geben Sie das $k(n)$ an, ab dem eine Sortierung des Arrays gefolgt von k binären Suchen asymptotisch schneller ist als k lineare Suchen auf dem unsortieren Array[1].

(10 Punkte)

[1] Gehen Sie dabei davon aus, daß Sortieren $\Theta(n \log n)$ viele Vergleiche benötigt, binäre Suche $\Theta(\log n)$ viele Vergleiche benötigt und lineare Suche $\Theta(n)$ viele Vergleiche benötigt.