Algorithmische Mathematik I

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

universitätbonn

Übungsblatt 4.

Abgabe am **Mittwoch, 18.11.2009 (vor der Vorlesung)**.

Aufgabe 1. (Drei–Term–Rekursion)

Sei zu festem $j \in \mathbb{N}_0$ jeweils die Drei–Term–Rekursion

$$g_{j,k} = a_k\, g_{j,k-1} + b_k\, g_{j,k-2} + \delta_{j,k}, \quad k = j, j+1, j+2, \ldots$$

mit den Startwerten

$$g_{j,j-2} = g_{j,j-1} = 0$$

gegeben. Dabei seien $a_k, b_k \in \mathbb{R}$ fest und

$$\delta_{j,k} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}.$$

Zeigen Sie: Für die inhomogene Drei–Term–Rekursion

$$p_k = a_k\, p_{k-1} + b_k\, p_{k-2} + c_k, \quad k = 1, 2, 3, \ldots$$

mit den Startwerten

$$p_0 = c_0,\ p_{-1} = 0$$

gilt die Darstellung

$$p_k = \sum_{j=0}^{k} c_j\, g_{j,k}, \quad k = 0, 1, 2, \ldots .$$

(10 Punkte)

Aufgabe 2. (Orthonormalpolynome und Drei–Term–Rekursion)

Es sei für Funktionen $f, g\colon [a,b] \to \mathbb{R}$ das Integral

$$\langle f, g\rangle := \int_a^b \omega(x) f(x) g(x) dx$$

mit Gewichtsfunktion $\omega : (a,b) \to \mathbb{R}^+$, und $a < b$ gegeben.[1] Weiterhin sei

$$\Pi_k = \left\{ c_k x^k + c_{k-1} x^{k-1} + \ldots + c_1 x + c_0 \right\}$$

der Raum der Polynome vom Grad kleiner oder gleich k mit Koeffizienten $c_j \in \mathbb{R}$.
Die Elemente einer Folge von Polynomen $\{P_k\}_k \subset \Pi_k$ vom exakten[2] Grad k heißen Orthonormalpolynome über $[a,b]$ bezüglich der Gewichtsfunktion ω, falls gilt

$$\langle P_i, P_j\rangle = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

[1] $\langle f, g\rangle$ ist ein Skalarprodukt auf $C[a,b]$. Für alle $f, g, h \in C[a,b]$, $s \in \mathbb{R}$ gilt insbesondere: $\langle f, g\rangle = \langle g, f\rangle$, $\langle f+g, h\rangle = \langle f, h\rangle + \langle g, h\rangle$, $\langle sf, g\rangle = s\langle f, g\rangle$, $\langle f, f\rangle \geq 0$ und $\langle f, f\rangle = 0 \Leftrightarrow f = 0$.
[2] Exakt Grad k heißt für $p \in \Pi_k$, daß $c_k \neq 0$.

Dann kann man auch für jedes $p \in \Pi_k$ eindeutige Koeffizienten $\gamma_j \in \mathbb{R}$ finden, so daß $p = \sum_{j=0}^{k} \gamma_j P_j$.
Zeigen Sie:
Die Folge von Orthonormalpolynomen $\{P_k\} \subset \Pi_k$ erfüllt die Drei–Term–Rekursion

$$\alpha_{n+1}\, P_{n+1}(x) = (x - \beta_n)\, P_n(x) - \alpha_n\, P_{n-1}(x), \quad n = 1, 2, \ldots$$
$$\alpha_n = \langle x\, P_{n-1}, P_n\rangle, \quad \beta_n = \langle x\, P_n, P_n\rangle$$

mit den Startwerten $P_{-1} = 0$, $P_0 = \frac{1}{\sqrt{\int_a^b \omega(x)\, dx}}$.
Hinweis: Sie können verwenden, dass jedes $p \in \Pi_{n+1}$ mit eindeutigen Koeffizienten $\gamma_j \in \mathbb{R}$ geschrieben werden kann als $p = \sum_{j=0}^{n} \gamma_j P_j + \gamma_{n+1} \cdot x P_n$.

(10 Punkte)

Aufgabe 3. (Matrixmultiplikation)

a) Seien $n \in \mathbb{N}$ und $A, B \in \mathbb{R}^{n\times n}$ Matrizen

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \ldots & b_{nn} \end{pmatrix}$$

Das Matrixprodukt $C = AB$, $(C \in \mathbb{R}^{n\times n})$ kann mittels der gewöhnlichen Summenformel berechnet werden:

$$C = \begin{pmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \ldots & c_{nn} \end{pmatrix}, \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}. \tag{1}$$

Bestimmen Sie, wie viele elementare Additionen und Multiplikationen reeller Zahlen zur Berechnung des Produkts $C = AB$ mit der gewöhnlichen Summenformel (1) notwendig sind. Drücken Sie die benötigte Zeit für die Matrixmultiplikation mit Hilfe der „groß O Notation“ aus der Vorlesung aus.

b) Sei nun n eine Zweierpotenz, also $n = 2^m$ mit $m \in \mathbb{N}$. Dann kann das Matrixprodukt $C = AB$ von Matrizen $A, B \in \mathbb{R}^{n\times n}$ mit Hilfe kleinerer Matrizen geschrieben werden: Zerlege A und B in jeweils vier $\frac{n}{2} \times \frac{n}{2}$-Matrizen

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

Analog zur gewöhnlichen Summenformel (1) gilt dann:

$$C = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}.$$

Alternativ schlug Strassen[4] folgendes Verfahren vor: Berechne die sieben Hilfsprodukte

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}), & P_5 &= (A_{11} + A_{12})B_{22}, \\ P_2 &= (A_{21} + A_{22})B_{11}, & P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}), \\ P_3 &= A_{11}(B_{12} - B_{22}), & P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}), \\ P_4 &= A_{22}(B_{21} - B_{11}). \end{aligned}$$

Dann gilt

$$C = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{pmatrix}.$$

[4] V. Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik, 13:354–356 (1969).

Damit reduziert sich die Zahl der Multiplikationen von $\frac{n}{2} \times \frac{n}{2}$-Matrizen von acht auf sieben gegenüber dem naiven Verfahren, während die Zahl der Additionen gestiegen ist. Dieses Verfahren wird nun rekursiv zur Berechnung der sieben kleineren Produkte angewandt. Zeigen Sie, dass die benötige Zeit $T(n)$ für die Multiplikation zweier $n \times n$-Matrizen der Rekurrenzrelation

$$T(n) = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2, \qquad T(1) = 1 \tag{2}$$

genügt. Entsprechend Satz 4.12. (Master-Theorem) aus der Vorlesung folgt damit, dass die Matrixmultiplikation nach Strassen die Komplexität $O(n^{\log_2 7}) \approx O(n^{2.807})$ besitzt. Verifizieren Sie dies, indem Sie per Induktion zeigen, dass

$$T(2^m) = 7^m + \frac{9}{2}\sum_{i=0}^{m-1} 7^i 2^{2m-2i}$$

und folgern Sie daraus die Komplexität $O(n^{\log_2 7}) \approx O(n^{2.807})$.

c) Zeigen Sie, dass $O(n^2)$ eine untere Schranke für die Komplexität der Matrixmultiplikation ist.

(10 Punkte)

Programmieraufgabe 2. (Merge–Sort)

Implementieren Sie den Merge–Sort–Algorithmus aus der Vorlesung in `C` oder `C++`. Als Eingabe sollen Arrays des Types `signed int` akzeptiert werden. Folgende Anforderungen soll Ihr Programm erfüllen:

1. Vor jedem Divide–Schritten soll das Array ausgegeben werden, um Ergebnisse leichter zu verifizieren.
2. Das Programm soll beliebige Arrays von $n = 2^k$ Zahlen, $k \in \mathbb{N}$, sortieren können. Beliebige $n \in \mathbb{N}$ ist praxisrelevanter, aber nicht gefordert.
3. Die Laufzeitanforderungen dürfen nicht größer als $O(n \log n)$ sein. Dasselbe gilt auch für den Speicherverbrauch.

Wenden Sie Ihre Implementierung auf die Zahlenfolge
503 087 512 061 908 170 897 275 653 426 154 509 612 677 765 703
an. (10 Punkte)

Hinweise:

- Um beliebig große Arrays zu speichern, benötigt man dynamische Speicherverwaltung. Auf der Vorlesungswebseite unter

  `http://wissrech.ins.uni-bonn.de/teaching/algmath/ws09_i`

  finden Sie erläuterte Beispiele, wie dies in `C` mit `calloc()` und `free()` bzw. in `C++` mit `vector` durchgeführt werden kann.
- Es ist möglich, Merge–Sort mit lediglich 2 Arrays mit jeweils Größe n durchzuführen. Tatsächlich geht es sogar mit noch weniger. Freiwillige können sich gerne daran versuchen.

Achtung: Die Punkte für die Programmieraufgabe werden gesondert gezählt!
Die Programmieraufgaben werden im CIP-Pool (Wegelerstraße 6, Raum E02) abgegeben. Bezüglich des konkreten Termines der Abgabe der Programmieraufgabe hängen in der Woche vom 23.11.-27.11.2009 im CIP-Pool Listen aus. Siehe dazu auch die Vorlesungshomepage `http://wissrech.ins.uni-bonn.de/teaching/algmath/ws09_i`.

Abgabe innerhalb der Woche 30.11.–4.12.2009 im CIP-Pool