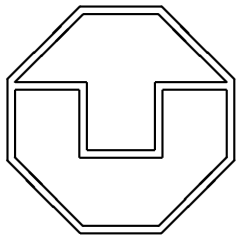
Technische Universität Dresden
Institut für Wissenschaftliches Rechnen

Wolfgang V. Walter



Programmieren 2

Aufgabenblatt 12 — Abgabetermin: 6.7.2018

Aufgabe 21: Schnelle(?) Matrixmultiplikation nach Strassen [12]

Es ist die übliche Matrixmultiplikation mit der sogenannten schnellen Matrixmultiplikation nach Strassen zu vergleichen. Dabei werden im folgenden nur noch quadratische $n \times n$-Matrizen betrachtet, deren Dimension eine Zweierpotenz ist ($n = 2^k$).

Nach *Volker Strassen (1969)* wird die Multiplikationsaufgabe

$$C = AB$$

durch Unterteilung der Matrizen in vier $n/2 \times n/2$-Matrizen geschrieben als

$$\left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right) = \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) \cdot \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)$$

und dann nach folgendem Schema vorgegangen:

$$\begin{aligned}
M1 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \\
M2 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M3 &= (A_{11} - A_{21}) \cdot (B_{11} + B_{12}) \\
M4 &= (A_{11} + A_{12}) \cdot B_{22} \\
M5 &= A_{11} \cdot (B_{12} - B_{22}) \\
M6 &= A_{22} \cdot (B_{21} - B_{11}) \\
M7 &= (A_{21} + A_{22}) \cdot B_{11} \\
C_{11} &= M1 + M2 - M4 + M6 \\
C_{12} &= M4 + M5 \\
C_{21} &= M6 + M7 \\
C_{22} &= M2 - M3 + M5 - M7
\end{aligned}$$

wobei auch die Matrizen $M1, \ldots, M7$ die Dimension $n/2$ haben. Strassen definiert nun diesen Prozess **rekursiv**, d.h. die 7 Matrixmultiplikationen zur Berechnung von $M1, \ldots, M7$ werden wiederum durch Viertelung der Matrizen durchgeführt.

Die Laufzeitkomplexität dieses Verfahrens ist $\mathcal{O}(n^{\log_2 7}) = \mathcal{O}(n^{2.807355})$, also im Prinzip geringer als die Komplexität der üblichen Matrixmultiplikation, die $\mathcal{O}(n^3)$ ist.

Implementieren Sie in einem Fortran 95–Modul die Methoden `strassen_matmul` und `simple_matmul`, welche die Matrixmultiplikation nach Strassen bzw. die herkömmliche Matrixmultiplikation (mit 3 ineinander geschachtelten Schleifen) realisieren.

Schreiben Sie ein Fortran 95–Hauptprogramm, welches eine ganze Zahl k einliest, daraus die Matrixgröße $n = 2^k$ berechnet und sodann zwei $n \times n$-Matrizen `A` und `B` anlegt und mit Daten füllt. Wahlweise sollen die Daten aus einer Datei gelesen werden oder Hilbert-Matrizen (mit $(a_{ij}) := 1/(i + j - 1)$) generiert werden können. Berechnen Sie sodann das Produkt `C` $=$ `A` $\cdot$ `B` dieser beiden Matrizen

- einmal mit der intrinsischen Matrixmultiplikation `MATMUL`,
- einmal mit der Matrixmultiplikation nach Strassen und
- einmal nach der herkömmlichen Methode

und prüfen Sie, ob alle drei Verfahren (bis auf Rundungsfehler) dieselben Ergebnisse liefern.

Messen Sie die Zeiten, die für die Multiplikationen gebraucht werden. Nutzen Sie dazu die Funktion `dtime` des `g95`-Compilers, welche entweder die Zeit seit ihrem letzten Aufruf oder, falls dieser noch nicht aufgetreten ist, die bisherige Laufzeit des Programms als reelle Zahl zurückgibt.

Was lässt sich über die „schnelle“ Matrixmultiplikation nach Strassen aussagen? Testen Sie die Methoden auch für große Matrizen (mit Dimensionen von mindestens 256) und verwenden Sie verschiedene Optimierungsstufen (Compileroption `-O`n).

Zusatzfragen: Der implementierte Algorithmus ist deterministisch. Wieso treten aber bei mehrmaliger Ausführung des Programms stets andere Laufzeiten auf?

Recherchieren Sie das Thema „effiziente Matrixmultiplikation“. Welche algorithmischen Fortschritte konnten auf diesem Gebiet in den letzten 50 Jahren erzielt werden? Welche asymptotischen Komplexitäten konnten inzwischen erreicht werden?

Aufgabe 22: Das Acht-(oder n)-Damen-Problem (Backtracking) 10

Beim Schachspiel können sich Damen (beliebig weit) horizontal, vertikal und diagonal bewegen und auf diese Weise andere Figuren bedrohen bzw. schlagen. Beim Acht-Damen-Problem sind acht Damen so auf einem Schachbrett zu verteilen, dass keine Dame eine andere schlagen kann. Allgemeiner sind n Damen auf einem quadratischen $(n \times n)$ Spielfeld so zu platzieren, dass keine Dame eine andere bedroht.

Schreiben Sie ein Fortran 95–Programm, welches nach der „Trial and Error“-Methode versucht, in einer Schleife für einzulesende $n \in \mathbb{N}$ jeweils alle Lösungen des n-Damen-Problems zu finden und geeignet auszugeben (entweder graphisch oder pseudo-graphisch mittels geeigneter Zeichen oder Buchstaben, die in n Zeilen und n Spalten angeordnet sind). Zählen Sie hierbei die Lösungen im Programm mit. Die Schleife ist abzubrechen, sobald ein $n < 1$ eingelesen wird.

Bei der Suche nach Lösungen wird jeweils eine Dame in die erste Zeile der nächsten freien Spalte gesetzt und überprüft, ob diese Dame eine andere schlagen kann. Wenn nicht wird die nächste Dame in die nächste Spalte gesetzt und wieder geprüft. Sollte die zuletzt gesetzte Dame jedoch geschlagen werden können, dann wird sie eine Zeile tiefer gesetzt. Falls sie schon in der untersten Zeile steht, wird sie entfernt und die Dame in der Spalte davor eine Zeile tiefer gesetzt (sogenanntes **Backtracking**).

Wenn die n-te Dame gesetzt ist und keine andere schlagen kann, ist eine Lösung gefunden. Wenn die erste Dame in der untersten Zeile steht und weitergesetzt werden müsste, dann sind alle Kombinationen überprüft und somit alle Lösungen gefunden.

Diese Art der Lösungssuche kann am besten *rekursiv* programmiert werden. Schreiben Sie eine rekursive Subroutine, die jeweils eine Dame in die nächste freie Spalte setzt und überprüft, ob sie sicher ist. Ist dies der Fall, so ruft sich die Subroutine selbst rekursiv auf, ansonsten versucht sie, eine andere Position innerhalb derselben Spalte mit einer Dame zu besetzen.

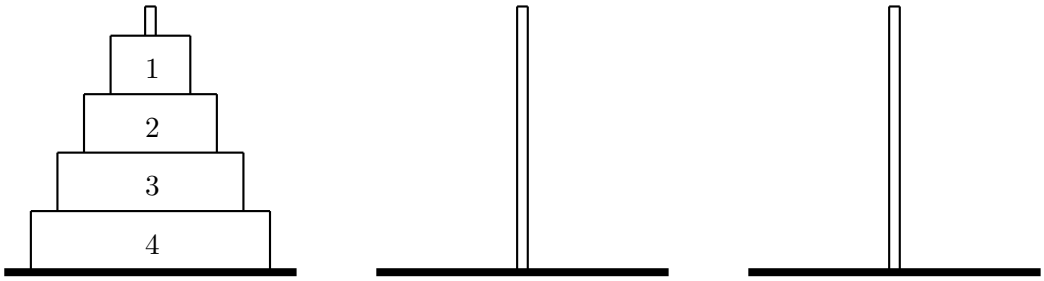
Zur Überprüfung, ob die zuletzt gesetzte Dame geschlagen werden kann, ist eine Funktion zu schreiben.

Optional können Sie versuchen, festzustellen, wieviele *fundamental verschiedene* Lösungen es gibt, wenn symmetrische Lösungen nicht mitgezählt werden.

Aufgabe 23: Türme von Hanoi, Stacks, Rekursion

16

Vor gut 100 Jahren wurde das orientalische Spiel „Türme von Hanoi“, dessen Anfangszustand hier mit 4 Scheiben schematisch dargestellt ist, als Umlegepuzzle nach Europa gebracht.



Das Ziel des Spiels besteht darin, die Scheiben vom linken Turm unter Benutzung des mittleren Turms auf den rechten Turm umzulegen. Dabei sind folgende Regeln zu beachten:

1. In jedem Zug wird jeweils die oberste Scheibe von einem der drei Türme auf einen anderen gelegt.
2. Es darf nie eine Scheibe mit größerem Durchmesser über einer Scheibe mit kleinerem Durchmesser liegen. (Hierbei wird vorausgesetzt, dass diese Regel bereits im Anfangszustand erfüllt ist und dass die Durchmesser aller Scheiben paarweise verschieden sind.)

Das Spiel wird von der Legende begleitet, in einem Tempel in Benares in Indien seien seit vielen Jahrhunderten Mönche auf Geheiß des Gottes Brahma damit beschäftigt, diese Umlegeprozedur, die sie mit einem nur wenige Zentimeter hohen Turm aus 64 dünnen Goldscheiben begonnen haben, auszuführen. Sobald die Mönche diese Aufgabe vollendet haben, so die Prophezeiung, werde der Tempel zu Staub zerfallen und die Welt mit einem Donnerschlag untergehen.

Vielleicht bleibt jedoch noch Zeit, die folgende Aufgabe zu bearbeiten ...

Schreiben Sie ein Fortran 95-Programm, das für eine einzulesende Anzahl n von Scheiben den Anfangszustand herstellt und das Spiel mit einer minimalen Anzahl von Zügen spielt. Dabei ist der neue Zustand nach jedem Zug graphisch darzustellen. Die Anzahl der ausgeführten Züge ist mitzuzählen, um festzustellen, ob sie wirklich der minimalen Anzahl von $2^n - 1$ Umlegungen entspricht. Die graphische Darstellung kann sehr einfach gestaltet werden, d.h. die Scheiben können als farbige Rechtecke (optional mit einer Nummer versehen) und die Türme ohne Basis und Mittelzapfen als an einer bestimmten Position mittig übereinandergestapelte Rechtecke gezeichnet werden. Um die Veränderungen in der Graphik optisch erfassbar zu machen, sollte nach jeder Veränderung in der Graphik eine Verzögerungsprozedur aufgerufen werden.

Es sind folgende Teilaufgaben zu bearbeiten:

6

a) Implementieren Sie die drei Türme als Kellerspeicher (Stapel, Stacks) mit Hilfe eines Fortran 95-Moduls `STACKMOD`. Die in der Vorlesung vorgestellte Stackimplementierung kann hierfür als Vorlage dienen.

   Definieren Sie dazu innerhalb des Moduls erstens den strukturierten Datentyp `DISC` mit den ganzzahligen Komponenten `NUM` für die Nummer, `COLOR` für den Farbcode und `DIAM` für den Durchmesser einer Scheibe, zweitens den strukturierten Datentyp `STACKELEM` mit einer Komponente `DATA` vom Typ `DISC` und einer Zeigerkomponente `SUCC` vom Typ `STACKELEM`, sowie drittens den strukturierten Datentyp `STACK` mit einer Zeigerkomponente `TOP` vom Typ `STACKELEM`, welche auf das oberste Element des Stapels zeigt, und einer ganzzahligen Komponente `TOP_POS` für die Anzahl der Scheiben im Turm (woraus die vertikale Position seiner obersten Scheibe bestimmt werden kann).

   Im Modul `STACKMOD` sollen des weiteren die folgenden Unterprogramme definiert werden:

- die Subroutine `INIT(S)`, in der der Stack `S` initialisiert wird, wobei alle Typkomponenten in einen konsistenten, definierten Zustand versetzt werden sollen, der erkennen lässt, dass der Stack leer ist,
- die logische Funktion `EMPTY(S)`, die testet, ob der Stack `S` leer ist,
- die Subroutinen `PUSH(S,ELEM)` und `POP(S,ELEM)`, die ein Element `ELEM` vom Typ `DISC` auf den Stack `S` legen bzw. von ihm herunternehmen. Es ist darauf zu achten, dass neben einer Modifikation der verzeigerten Stack-Struktur immer auch die Komponente `TOP_POS` aktualisiert wird. Es empfiehlt sich, das Zeichnen bzw. Löschen (Überzeichnen mit der Hintergrundfarbe) der Scheiben innerhalb dieser Routinen zu implementieren. Deshalb sollen alle notwendigen Konstanten und Variablen für Koordinaten, Farbcodes, Scheibendurchmesser, etc. im Modul vereinbart werden.

4

b) Im Hauptprogramm, welches den Modul `STACKMOD` benutzt, ist ein Feld mit drei Elementen vom Typ `STACK` zu vereinbaren. Zunächst ist die Anzahl n der zu verwendenden Scheiben einzulesen und der linke Turm (mit dem kleinsten Index) mit diesen n Scheiben regelkonform zu besetzen.

   Lösen Sie das Problem nun mit Hilfe einer **rekursiven** Subroutine `HANOI(I,J,K,m)`, die das regelkonforme Umlegen der oberen `m` Scheiben vom Turm mit der Nummer `I` auf den Turm mit der Nummer `K` über den als Zwischenlager dienenden Turm mit der Nummer `J` realisiert.

6

c) Implementieren Sie des weiteren einen **nichtrekursiven** Algorithmus, der das gestellte Problem mittels der korrekten Abfolge von Bewegungen einzelner Scheiben direkt (iterativ) löst.