Optimale Klammerung

Das Problem der optimalen Klammerung tritt auf, wenn man das Produkt von k Matrizen $M_1 \cdot \ldots \cdot M_k$ berechnen will, wobei M_i eine $(n_{i-1} \times n_i)$-Matrix ist $(i = 1, \ldots, k)$.

Wir sollen die optimale Ausführungsreihenfolge (= Klammerung) ermitteln, um möglichst wenige sogenannte *Skalarmultiplikationen* zu benötigen.

Wieso haben wir hier überhaupt eine Freiheit? Weil das Assoziativgesetz gilt!

Wieviele Skalarmultiplikation benötigt man für eine einzelne Matrixmultiplikation?
M sei $(\ell \times m)$-, N sei $(m \times n)$-Matrix, dann $\ell \cdot m \cdot n$.

Wir betrachten $M_1 \cdot M_2 \cdot M_3$, also $k = 3$.
Die Dimensionen seien (10×2), (2×8),(8×3).

Für $(M_1 \cdot M_2) \cdot M_3$ werden 400 skalare Multiplikationen benötigt.
Für $M_1 \cdot (M_2 \cdot M_3)$ werden 108 skalare Multiplikationen benötigt.

Plan für einen Algorithmus

Die grobe Struktur ist, wie bei fast allen Algorithmen dieses Typs, eine Tabelle mit k Zeilen und k Spalten. Allerdings wird jedes „Indexpaar“ $\{i, j\}$ nur einmal gebraucht, d.h. es ergibt sich eine Dreiecksstruktur:

i \ j	1	2	3	4	.	.	.	k
1	0							
2		0						
3			0					
4				0				
.					.		$T_{i,j}$	
.						.		
.							.	
k								0

$T_{i,j}$ soll für $i < j$ die Kosten für die Teilmultiplikation

$$M_i \cdot \ldots \cdot M_j$$

enthalten.

Diese Kosten berechnen wir mit der Formel:

$$T_{i,j} = \min_{i \le m < j} \left(T_{i,m} + T_{m+1,j} + n_{i-1} \cdot n_m \cdot n_j \right)$$

Pseudo-Code

```
FOR i := 1 TO k DO                                   Initialisierung
        T[i,i] := 0;
        FOR j := i + 1 TO k DO T[i,j] := ∞ ENDFOR
ENDFOR;
FOR d := 1 TO k − 1 DO                          d ist Differenz
        FOR i := 1 TO k − d DO                        zwischen i und j
                j := i + d;
                FOR m := i TO j − 1 DO        m ist der „Trennpunkt“
                        t := T[i,m]+T[m + 1,j] + n[i − 1] ∗ n[m] ∗ n[j];
                        IF t < T[i,j] THEN T[i,j] := t; B[i,j] := m ENDIF
                ENDFOR                              Trennpunkte in B
        ENDFOR                                            protokollieren
ENDFOR
```

Beispiel

Wir wollen die folgende Matrixmultiplikation ausführen:

$$M_1 \cdot M_2 \cdot M_3 \cdot M_4$$

M_1 ist eine (3×11)-, M_2 eine (11×5)-, M_3 eine (5×7)-, M_4 eine (7×2)-Matrix.

Bitte selbst durchführen...

Als Ergebnis sollten Sie die folgenden Matrizen erhalten:

$$T: \begin{matrix} 0 & 165 & 270 & 246 \\ & 0 & 385 & 180 \\ & & 0 & 70 \\ & & & 0 \end{matrix} \qquad B: \begin{matrix} - & 1 & 2 & 1 \\ & - & 2 & 2 \\ & & - & 3 \\ & & & - \end{matrix}$$

Also ist die optimale Klammerung: $M_1 \cdot (M_2 \cdot (M_3 \cdot M_4))$

Backtracking

Oft kommt es dazu, dass der Lösungsraum für ein Problem aus n Komponenten, die jeweils einen von endlich vielen Werten annehmen können, besteht. Dann gibt es exponentiell viele mögliche Lösungen, die schlimmstenfalls alle durchprobiert werden müssen.

Mit der Backtracking-Methode versucht man, in manchen Fällen schneller zu sein als das Brute-Force-Durchprobieren.

Als Beispiel betrachten wir eine logische Formel mit n Variablen. Gesucht ist eine erfüllende Belegung.

Ist die Formel in 3-KNF gegeben, können wir versuchen, immer wieder ein Tripel von Variablen zu finden, das in einer Klausel vorkommt und noch nicht belegt ist. Gibt es dieses, können wir alle noch möglichen Belegungen dafür ausprobieren.

SAT mit Backtracking

Aber wenn es kein solches Variablen-Tripel mehr gibt, ist die verbliebene Teilformel leicht lösbar (2-KNFSAT...).

Nun zum Fall, dass wir ein solches Tripel gefunden haben. Wir setzen die 3 Variablen auf alle möglichen Werte – das sind a priori 8 Möglichkeiten, von denen aber mindestens eine auf Grund der verwendeten Klausel ausgeschlossen werden kann.

Das führt zur Rechenzeit $T(n) = 7 \cdot T(n-3)$ für die rekursiven Aufrufe, also zu einer Rekursionstiefe von $n/3$ und folglich einer Gesamtlaufzeit des Algorithmus in $O(7^{n/3})$, d.h.

$$O(1.913^n)$$

und damit in $o(2^n)$.

Wieso 1.913 ? Weil dieser Wert die dritte Wurzel aus 7 (aufgerundet) ist!