# *Errata, Corrigenda et Addenda*
# Grundlegende Algorithmen

*Volker Heun*

Vieweg-Verlag, 2. Auflage, ISBN 3-528-13140-3

**Stand: 25. August 2009**

**Seite 9, Zeile −7** *(Korrektur):*

*Ersetze:* Da symmetrische Matrizen unter Matrizenmultiplikation abgeschlossen sind (d.h. das Produkt zweier symmetrischer Matrizen ist wieder symmetrisch), ...

*durch:* Im Allgemeinen ist das Produkt zweier symmetrischer Matrizen nicht symmetrisch. Es gilt jedoch, dass Produkte von Matrizen, die jeweils Potenzen einer gegebenen symmetrischen Matrix sind, wiederum symmetrisch sein müssen. Somit ...

*[Dank an G. West; 2004-01-14]*

**Seite 117, Zeile −2** *(Korrektur):*

*Ersetze:* **Aufgabe 3.2** Zeigen Sie, dass zur Bestimmung des Medians von 5 Elementen 7 Vergleiche hinreichend und notwendig sind.

*durch:* **Aufgabe 3.2** Zeigen Sie, dass zur Bestimmung des Medians von 5 Elementen 6 Vergleiche hinreichend und notwendig sind.

*[Dank an S. Gerke; 2004-11-09]*

**Seite 194, Zeile −21** *(Ergänzung):*

In der Prozedur `decrease_key` wurde beim Erniedrigen eines Schlüssels in der Wurzelliste vergessen, den Zeiger auf das minimale Element zu aktualisieren. Der Pesudo-Code ist wie folgt zu ergänzen:

DECREASE_KEY ()

```
...
if (parent[v] ≠ NULL) cut(v);
elsif (key[v] < key[min_root]) min_root = v;
...
```

*[Dank an T. Preclik; 2008-10-28]*

**Seite 195, Zeile +12** *(Ergänzung):*

In der Prozedur `delete_min` wurde im Fall, dass die Wurzel mit dem minimalen Element keine Kinder hat, vergessen, diese Wurzel selbst aus der Wurzelliste zu entfernen. Der Pesudo-Code ist wie folgt zu ergänzen:

DELETE_MIN ()

```
...
else
{
    new_root = right[min_root];
    left[right[min_root]] = left[min_root];
```

```
        right[left[min_root]] = right[min_root];
    }
    ...
```

*[Dank an T. Preclik; 2008-10-28]*

**Seite 195, Zeile −8** *(Ergänzung):*
In der Prozedur `cleanup` wurde beim Wiederherstellen der Wurzelliste vergessen, bei den dort eingehängten Wurzeln die Markierungen zu entfernen. Der Pesudo-Code ist wie folgt zu ergänzen:

```
CLEAN_UP ()
    ...
    if (root[i] ≠ NULL)
    {
        marked[root[i]] = FALSE;
        if (min_root == NULL)
            ...
    }
    ...
```

*[Dank an F. Prilmeier; 2003-12-09]*

**Seite 281, Zeile +3** *(Korrektur):*
*Ersetze:* **if** $(n == 1)$**return**$(a_0)$
*durch:* **if** $(n < 1)$**return**$(a_0)$
Da der Eingabevektor $2n$ Einträge besitzt, muss die Abbruchbedingung wie oben lauten. Je nachdem, ob die Division durch 2 im rekursiven Aufruf als Division auf reellen Zahlen oder als ganzzahlige Division interpretiert wird, ist dann $n = 0.5$ oder $n = 0$.
*[Dank an S. Szott; 2005-11-07]*

**Seite 283, Zeile +4/ + 5** *(Korrektur):*
*Ersetze:* $(a_0, \ldots, a_{2n-1}) = \mathrm{FFT}((a_0, \ldots, a_{n-1}, 0, \ldots, 0), 2n, \omega)$;
*durch:* $(a_0, \ldots, a_{2n-1}) = \mathrm{FFT}((a_0, \ldots, a_{n-1}, 0, \ldots, 0), n, \omega)$;
Der Eingabevektor besitzt für das Argument $n$ genau $2n$ Einträge.
*[Dank an S. Szott; 2005-11-07]*

**Seite 285, Zeile −3** *(Korrektur):*
*Ersetze:* $T(n) \leq T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + 6n.$
*durch:* $T(n) \leq T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + 8.5n.$

In der Rekursionsgleichung wurden leider die Kosten für die folgenden Addition vergessen:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2\lceil \frac{n}{2} \rceil} + \left[a^{(1)} \cdot b^{(0)} + a^{(0)} \cdot b^{(1)}\right] \cdot 2^{\lceil \frac{n}{2} \rceil} + a^{(0)} \cdot b^{(0)}.$$

Die Addition des ersten und letzten Summanden kosten nichts, da die beiden Bit-Vektoren nur geeignet zusammengesetzt werden müssen. Die eigentliche Addition des mittleren Summanden kostet $2n$ Bit-Operationen. Aufgrund eines Übertrags können weiter $n/2$ Bit-Operationen anfallen. Damit verändern sich natürlich auch die weiteren Rechnungen wie folgt angegeben.

Wir wollen nun die folgende Rekursionsgleichung lösen:

$$T(n) = 3T(\lceil n/2 \rceil + 1) + \frac{17n}{2}.$$

Man kann wie im Abschnitt 2.2.2 zeigen, dass die Lösung dieser Rekursionsgleichung auch eine Abschätzung für die Lösung unserer ursprünglichen Rekursionsgleichung liefert. Zuerst halten wir die folgende nützliche Beziehung fest:

$$\left\lceil \frac{\lceil \frac{n}{2^i} \rceil + 2}{2} \right\rceil + 1 = \left\lceil \frac{n}{2^{i+1}} \right\rceil + 2.$$

Diese Beziehung benötigen wir gerade, wenn wir die Rekursionsgleichung durch Iteration lösen wollen.

$$\begin{aligned}
T(n) &= 3 \cdot T\left(\left\lceil \frac{n}{2} \right\rceil + 1\right) + \frac{17n}{2} \\
&\leq 3^2 \cdot T\left(\left\lceil \frac{n}{2^2} \right\rceil + 2\right) + 3 \cdot 6\left(\left\lceil \frac{n}{2} \right\rceil + 2\right) + \frac{17n}{2} \\
&= 3^3 \cdot T\left(\left\lceil \frac{n}{2^3} \right\rceil + 2\right) + 3^2 \cdot 6\left(\left\lceil \frac{n}{2^2} \right\rceil + 2\right) + 3 \cdot 6\left(\left\lceil \frac{n}{2} \right\rceil + 2\right) + \frac{17n}{2}
\end{aligned}$$

Nach $k$ Iterationen erhalten wir:

$$\leq 3^k \cdot T\left(\left\lceil \frac{n}{2^k} \right\rceil + 2\right) + \sum_{i=0}^{k-1} 3^i \cdot \frac{17}{2}\left(\left\lceil \frac{n}{2^i} \right\rceil + 2\right)$$

Für $k = \lfloor \log(n) \rfloor$ gilt dann $\lceil \frac{n}{2^{\lfloor \log(n) \rfloor}} \rceil \leq 2$ und somit:

$$\begin{aligned}
&\leq 3^{\log(n)} \cdot T\Bigg(\underbrace{\left\lceil \frac{n}{2^{\lfloor \log(n) \rfloor}} \right\rceil + 2}_{\leq 4}\Bigg) + \frac{17}{2} \sum_{i=0}^{\lfloor \log(n) \rfloor - 1} 3^i \left(\left\lceil \frac{n}{2^i} \right\rceil + 2\right) \\
&\leq 2^{\log(3) \cdot \log(n)} \cdot T(4) + \frac{17}{2} \sum_{i=0}^{\lfloor \log(n) \rfloor - 1} 3^i \left(\frac{n}{2^i} + 3\right)
\end{aligned}$$

Da aufgrund der Schulmethode $T(4) \leq 2 \cdot 4^2 - 3 \cdot 4 = 20$ gilt, erhalten wir:

$$\begin{aligned}
&\leq 20n^{\log(3)} + \frac{17}{2} n \sum_{i=0}^{\lfloor \log(n) \rfloor - 1} \frac{3^i}{2^i} + \frac{51}{2} \sum_{i=0}^{\lfloor \log(n) \rfloor - 1} 3^i \\
&\leq 20n^{\log(3)} + \frac{17}{2} n \frac{\frac{3}{2}^{\log(n)} - 1}{\frac{3}{2} - 1} + \frac{51}{2} \frac{3^{\log(n)} - 1}{3 - 1} \\
&\leq 20n^{\log(3)} + 17n 2^{\log(3/2) \log(n)} + \frac{51}{4} \cdot 2^{\log(n) \log(3)} \\
&\leq 20n^{\log(3)} + 17n^{\log(3)} + \frac{51}{4} n^{\log(3)} \\
&\leq \frac{199}{4} \cdot n^{\log(3)}
\end{aligned}$$

Damit haben wir also das folgende Theorem bewiesen.

**Theorem 7.45** *Zwei $n$-stellige Binärzahlen können mit höchstens $\frac{199}{4} \cdot n^{\log(3)}$ Bit-Operationen multipliziert werden.*

Allerdings ist diese Methode erst für Binärzahlen der Länge 2312 effizienter als die Schulmethode.

### 7.6.4 Verbesserung des Algorithmus von Karatsuba und Ofman

Es sieht auf den ersten Blick so aus, als wäre der Karatsuba-Ofman-Algorithmus nicht praktikabel. Mit etwas Grips erkennt man, dass es für kleine Binärzahlen günstiger ist, auf die Schulmethode umzusteigen. Für eine genauere Analyse brechen wir die Rekursion bei Binärzahlen der Länge $m+2$ ab.

$$\begin{aligned} T(n) &\leq 3^k \cdot T\left(\left\lceil \frac{n}{2^k} \right\rceil + 2\right) + \sum_{i=0}^{k-1} 3^i \cdot \frac{17}{2}\left(\left\lceil \frac{n}{2^i} \right\rceil + 2\right) \\ \text{Wähle } k = \lceil \log(\tfrac{n}{m}) \rceil \text{ und setze } \varepsilon := \lceil \log(\tfrac{n}{m}) \rceil - \log(\tfrac{n}{m}), \text{ also } k &= \log(\tfrac{n}{m}) + \varepsilon \\ &\leq 3^{\log(\frac{n}{m})+\varepsilon} \cdot T\left(\left\lceil \frac{n}{2^{\log(\frac{n}{m})+\varepsilon}} \right\rceil + 2\right) + \sum_{i=0}^{\lceil \log(\frac{n}{m}) \rceil - 1} 3^i \cdot \frac{17}{2}\left(\left\lceil \frac{n}{2^i} \right\rceil + 2\right) \\ &\leq 3^{\log(n)} 3^{-\log(m)} 3^{\varepsilon} \cdot T\left(\frac{m}{2^\varepsilon} + 3\right) + \sum_{i=0}^{\lceil \log(\frac{n}{m}) \rceil - 1} 3^i \cdot \frac{17}{2}\left(\frac{n}{2^i} + 3\right) \\ &\leq 3^{\varepsilon} n^{\log(3)} 3^{-\log(m)} \cdot T\left(\frac{m}{2^\varepsilon} + 3\right) + \frac{17n}{2} \sum_{i=0}^{\lceil \log(\frac{n}{m}) \rceil - 1} \frac{3^i}{2^i} + \frac{51}{2} \sum_{i=0}^{\lceil \log(\frac{n}{m}) \rceil - 1} 3^i \\ &\leq 3^{\varepsilon} n^{\log(3)} 3^{-\log(m)} \cdot T\left(\frac{m}{2^\varepsilon} + 3\right) + \frac{17n}{2} \frac{\frac{3}{2}^{\log(\frac{n}{m})+\varepsilon} - 1}{\frac{3}{2} - 1} + \frac{51}{2} \frac{3^{\log(\frac{n}{m})+\varepsilon} - 1}{3 - 1} \end{aligned}$$

Da für die Schulmethode $T(n) \leq 2n^2 - 3n$ gilt:

$$\begin{aligned} &\leq n^{\log(3)} \left( \frac{\frac{2m^2}{2^{2\varepsilon}} + 12\frac{m}{2^\varepsilon} + 18 - \frac{3m}{2^\varepsilon} - 9}{3^{\log(m)-\varepsilon}} + \frac{\frac{17m}{2^\varepsilon}}{3^{\log(m)-\varepsilon}} + \frac{\frac{51}{4}}{3^{\log(m)-\varepsilon}} \right) \\ &\leq n^{\log(3)} \left[ \frac{2m^2/2^{2\varepsilon} + 26m/2^\varepsilon + \frac{87}{4}}{3^{\log(m)-\varepsilon}} \right]. \end{aligned}$$

Wir überlegen uns nun, für welches $m$ die „Konstante“ hinter dem Term $n^{\log(3)}$ am kleinsten wird. Dies kann man in etwa aus Bild 7.12 ablesen. Hier ist diese „Konstante“ in Abhängigkeit von $m \in [10:50]$ und $\varepsilon \in [0,1)$ dargestellt. Für den genauen Umschaltpunkt betrachten wir das Bild 7.13. Hier ist der Wert der „Konstanten“ für $m \in [28:30]$ in Abhängigkeit von $\varepsilon \in [0,1)$ aufgetragen. Daraus erkennen wir, dass beim Umschalten bei $m = 29$, d.h. bei Binärzahlen der Länge 31, die Konstante auf unter 12 sinkt.

**Theorem 7.46** *Zwei $n$-stellige Binärzahlen können mit maximal $12 \cdot n^{\log(3)}$ Bit-Operationen multipliziert werden.*

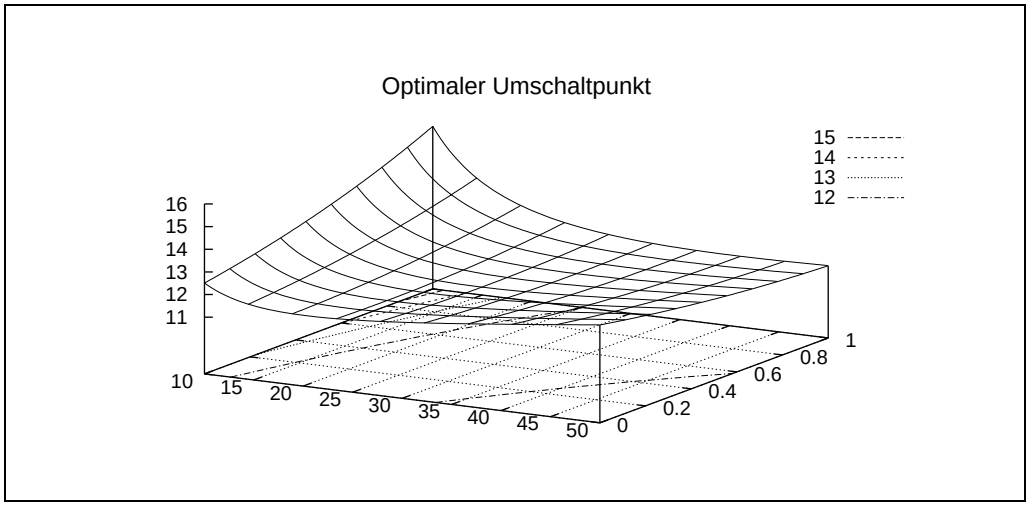


Bild 7.12: Wahl des Umschaltens bei der Multiplikation von Binärzahlen

Damit ist der modifizierte Karatsuba-Ofman-Algorithmus schon für Binärzahlen ab etwa 80 Stellen effizienter als die Schulmethode, wie man aus Bild 7.14 ablesen kann. Man sollte hier anmerken, dass bei Verschlüsselungsalgorithmen mittlerweile mit Zahlen von 500 Binärstellen und mehr gerechnet wird. Beschränkt man sich (wie in der Praxis üblich) bei der Implementierung auf Binärzahlen, deren Längen Zweierpotenzen sind, so lässt sich dass Ergebnis noch verbessern. Bei der vorigen Abschätzung ist dann $\varepsilon = 0$ und die störenden Gaußklammern entfallen. Man kann leicht nachrechnen, dass man dann mit maximal

$$n^{\log(3)} \left[ \frac{2m^2 + 22m + \frac{21}{2}}{3^{\log(m)}} \right]$$

Operationen auskommt, wobei $m$ nun ebenfalls eine Zweierpotenz sein muss. Steigt man bei $m = 16$ auf die Schulmethode um, so erhält man das folgende Ergebnis.

**Theorem 7.47** *Zwei $n$-stellige Binärzahlen können mit maximal $10{,}8 \cdot n^{\log(3)}$ Bit-Operationen multipliziert werden, wenn $n$ eine Zweierpotenz ist.*

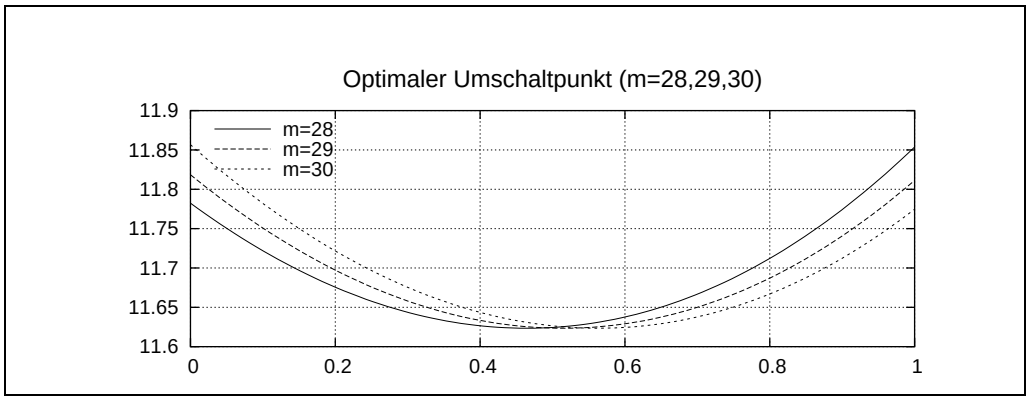


Bild 7.13: Umschaltpunkt bei der Multiplikation von Binärzahlen bei $m \approx 29$

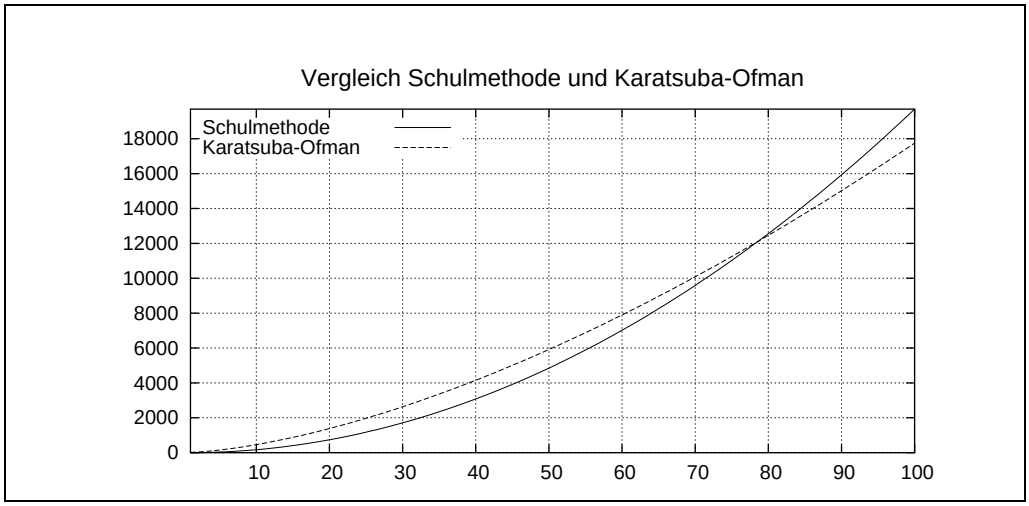


Bild 7.14: Vergleich Schulmethode gegen modifizierten Karatsuba-Ofman

Dieses Verfahren schlägt die Schulmethode bereits für Binärzahlen in der 64-Bit-Darstellung.
*[Dank an P. Überholz; 2005-09-22]*