Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

Teile & Herrsche

Teile & Herrsche (Divide & Conquer)

- Teile Eingabe in mehrere Teile auf
- Löse das Problem rekursiv auf den Teilen
- Füge die Teillösungen zu einer Gesamtlösung zusammen

Teile & Herrsche

Weiteres Beispiel

- Problem: Finde Element in sortiertem Feld
- Eingabe: Sortiertes Feld A, gesuchtes Element $b \in A[1,\ldots,n]$
- Ausgabe: Index i mit $A[i] = b$

BinäreSuche(A,b,p,r)
1. **if** p=r **then return** p
2. **else**
3. $q \leftarrow \lfloor (p+r)/2 \rfloor$
4. **if** $b \leq A[q]$ **then return** BinäreSuche(A,b,p,q)
5. **else return** BinäreSuche(A,b,q+1,r)

Teile & Herrsche

Satz 9

- Algorithmus BinäreSuche(A,b,p,r) findet den Index einer Zahl b in einem sortierten Feld A[p..r], sofern b in A[p..r] vorhanden ist.

Beweis

- Wir zeigen die Korrektheit per Induktion über n=r-p. Ist n<0, so ist nichts zu zeigen. Wir nehmen an, dass b in A[p..r] ist, da es sonst nichts zu zeigen gibt.
- (I.A.) Für n=0, d.h. p=r, gibt der Algorithmus p zurück. Dies ist der korrekte (weil einzige) Index.
- (I.V.) Für alle r,p mit m=r-p und 0≤m≤n findet BinäreSuche(A,b,p,r) den Index einer Zahl b in einem sortierten Feld A[p..r].

Teile & Herrsche

Satz 9

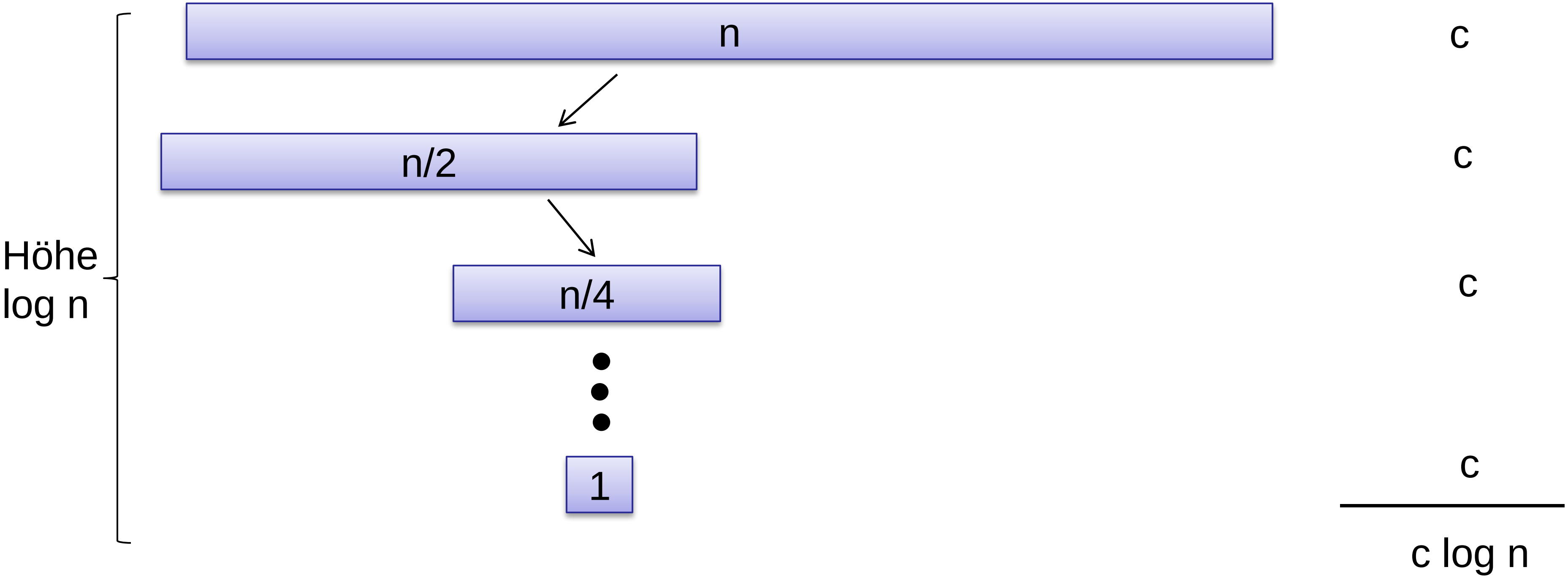
- Algorithmus BinäreSuche(A,b,p,r) findet den Index einer Zahl b in einem sortierten Feld A[p..r], sofern b in A[p..r] vorhanden ist.

Beweis

- (I.V.) Für alle r,p mit m=r-p und $0 \leq m \leq n$ findet BinäreSuche(A,b,p,r) den Index einer Zahl b in einem sortierten Feld A[p..r].
- (I.S.) Wir betrachten den Aufruf von BinäreSuche für beliebige p, r mit n+1 = r-p. Da n+1>0 folgt p<r und der Algorithmus führt den **else**-Fall aus. Dort wird q auf $\lfloor (p+r)/2 \rfloor$ gesetzt. Es gilt $q \geq p$ und q<r. Ist $b \leq A[q]$, so wird BinäreSuche rekursiv für A[p..q] aufgerufen. Da A[p..r] sortiert ist, liegt b in A[p..q]. Damit folgt aus (I.V.), dass der Index von b gefunden wird. Ist b>A[q], so wird BinäreSuche rekursiv für A[q+1..r] aufgerufen. Da A[p..r] sortiert ist, liegt b in A[q+1..r]. Damit folgt aus (I.V.), dass der Index von b gefunden wird.

Teile & Herrsche

Auflösen von $T(n) \leq T(n/2) + c$ (Intuition; wir ignorieren Runden)



Teile & Herrsche

Binäre Suche vs. lineare Suche

Laufzeit	**10**	**100**	**1,000**	**10,000**	**100,000**
n	10	100	1,000	10,000	100,000
log n	3	6	10	13	17

Beobachtung

- n wächst sehr viel stärker als log n
- Binäre Suche effizient für riesige Datenmengen
- In der Praxis ist log n fast wie eine Konstante

Teile & Herrsche

Integer Multiplikation

- Problem: Multipliziere zwei n-Bit Integer
- Eingabe: Zwei n-Bit Integer X,Y
- Ausgabe: 2n-Bit Integer Z mit Z=XY

Annahmen:

- Wir können n-Bit Integer in $\Theta(n)$ (worst case) Zeit addieren
- Wir können n-Bit Integer in $\Theta(n+k)$ (worst case) Zeit mit 2^k multiplizieren

Teile & Herrsche

Laufzeit Schulmethode

- n Multiplikationen mit 2^k für ein $k \leq n$
- n-1 Additionen im worst-case:

$$\underbrace{11...111}_{\text{n-Bit}} \cdot \underbrace{11...111}_{\text{n-Bit}}$$

- Jede Addition $\Theta(n)$ Zeit
- Insgesamt $\Theta(n^2)$ Laufzeit

Bessere Laufzeit mit Teile & Herrsche?

Teile & Herrsche

Integer Multiplikation (siehe Vollversion)

A **B** • **C** **D**

B · D

\+ **B · C**

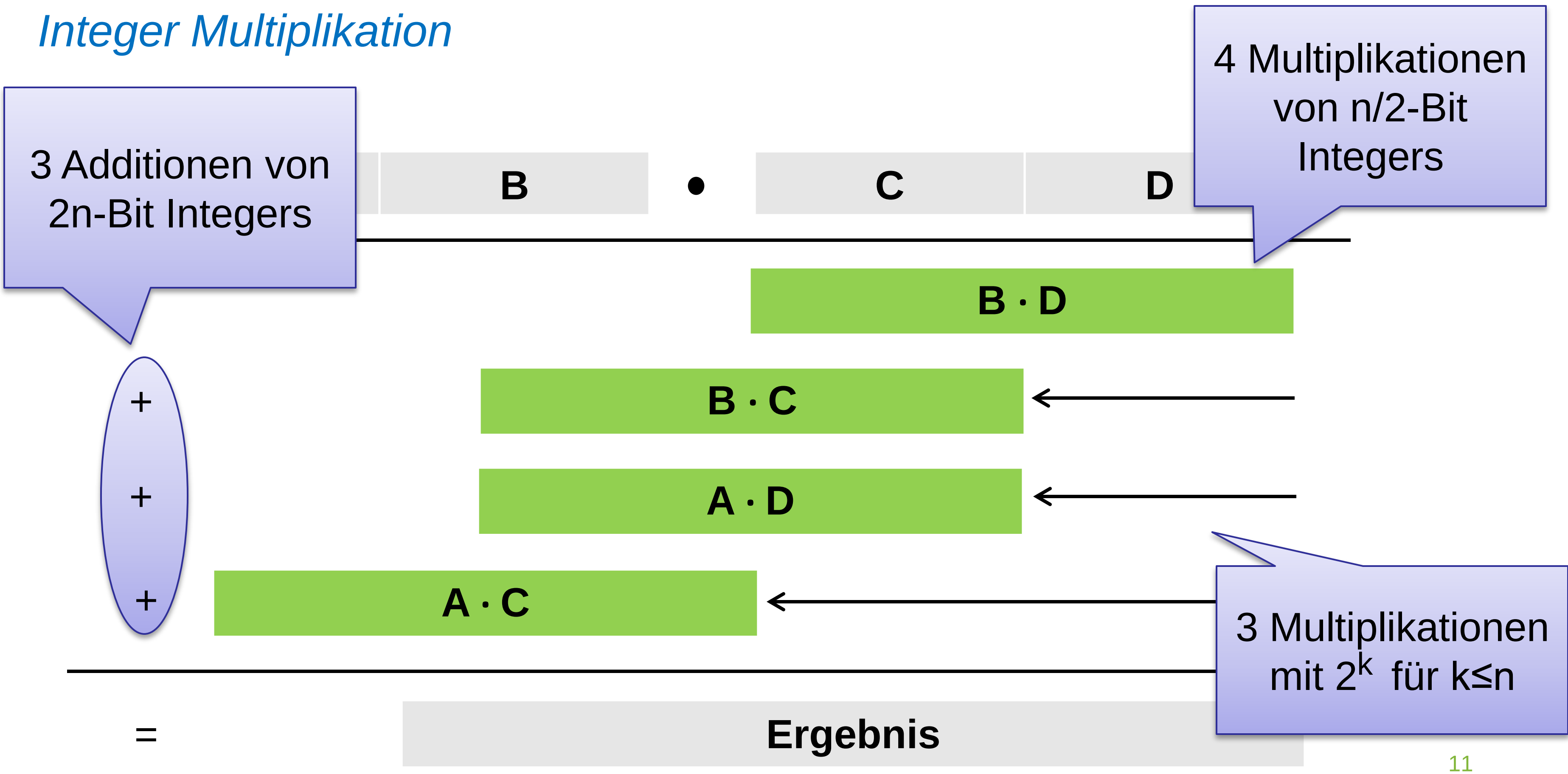
\+ **A · D**

\+ **A · C**

= **Ergebnis**

Teile & Herrsche

Integer Multiplikation

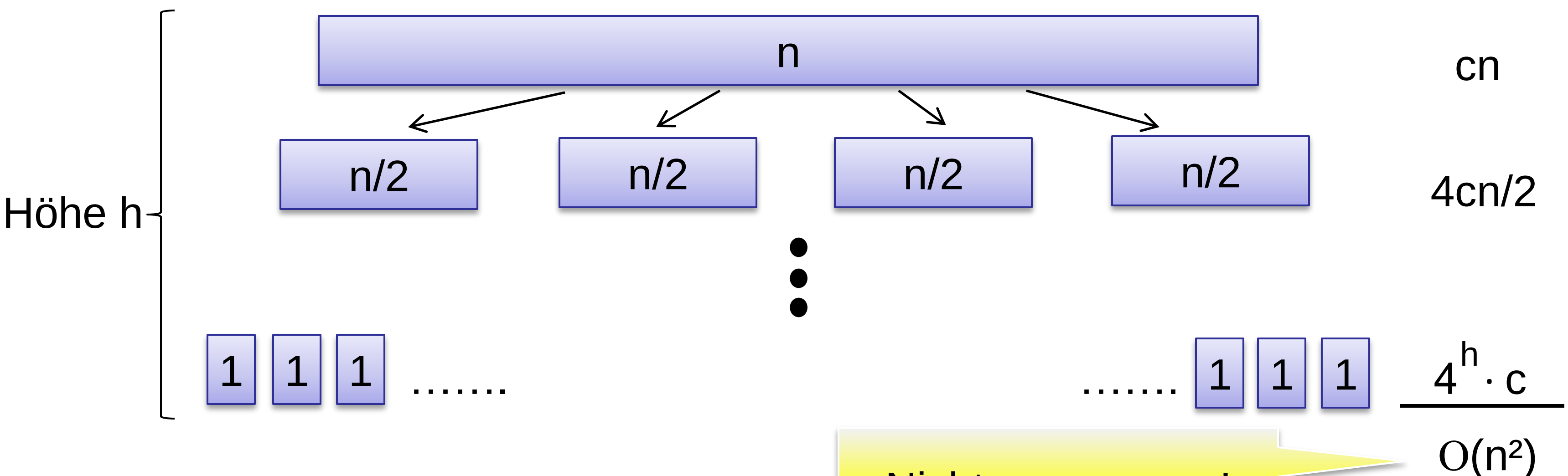


Teile & Herrsche

Laufzeit einfaches Teile & Herrsche

$$T(n) \le \begin{cases} 4\,T(n/2) + cn & , n>1 \\ c & , n=1 \end{cases}$$

c geeignete Konstante



Nichts gewonnen!

Höhe des Baums: $h = \log n$

Teile & Herrsche

Satz 11

Die Multiplikation zweier n-Bit Zahlen mit dem einfachen Teile & Herrsche Verfahren hat Laufzeit $O(n^2)$.

Beweis

- Wir nehmen an, dass n eine Zweierpotenz ist. Induktion über n. Wir zeigen $T(n) \le cn^2$.
- (I.A.) Die Laufzeit zur Multiplikation von zwei 1-Bit Zahlen ist höchstens c.
- (I.V.) Für jedes m<n ist die Laufzeit zur Multiplikation von zwei m-Bit Zahlen $c\,m^2$.
- (I.S.) Betrachte eine Multiplikation von zwei n-Bit Zahlen. Es gilt $T(n) \le 4\,T(n/2) + cn$. Nach (I.V.) gilt dann $T(n) \le 4\,c\,(n/2)^2 + cn = \boxed{cn^2+cn}$.

Funktioniert nicht !!!!!

Teile & Herrsche

Satz 11

Die Multiplikation zweier n-Bit Zahlen mit dem einfachen Teile & Herrsche Verfahren hat Laufzeit $O(n^2)$.

Beweis (neuer Versuch)

- Wir nehmen an, dass n eine Zweierpotenz ist. Induktion über n. O.b.d.A. sei $c \geq T(2)$. Wir zeigen $\boxed{T(n) \leq cn^2 - cn.}$

Trick: Die Funktion etwas verkleinern!!

Teile & Herrsche

Satz 11

Die Multiplikation zweier n-Bit Zahlen mit dem einfachen Teile & Herrsche Verfahren hat Laufzeit $O(n^2)$.

Beweis (neuer Versuch)

- Wir nehmen an, dass n eine Zweierpotenz ist. Induktion über n. O.b.d.A. sei $c \geq T(2)$. Wir zeigen $T(n) \leq cn^2 - cn$.
- (I.A.) Die Laufzeit zur Multiplikation von zwei 2-Bit Zahlen ist höchstens c.
- (I.V.) Für jedes $m<n$ ist die Laufzeit zur Multiplikation von zwei m-Bit Zahlen $c\, m^2 - cm$.
- (I.S.) Betrachte eine Multiplikation von zwei n-Bit Zahlen. Es gilt
  $T(n) \leq 4\, T(n/2) + cn$.
  Nach (I.V.) gilt dann $T(n) \leq 4\, c\, (n/2)^2 - 4\, c(n/2) + cn = cn^2 - cn$.

Teile & Herrsche

Verbesserte Integer Multiplikation

A	B	•	C	D

$$
\begin{array}{lr}
 & B \cdot D \\
+ & B \cdot C \\
+ & A \cdot D \\
+ & A \cdot C \\
\hline
= & \textbf{Ergebnis}
\end{array}
$$

Teile & Herrsche

Verbesserte Integer Multiplikation

A **B** • **C** **D**

B · D

\+ **B · C**

\+ **A · D**

\+ **A · C**

= **Ergebnis**

Berechne BC + AD
Mit einer Multiplikation

Trick:
(A+B) (C+D) – AC – BD
= BC + AD

Teile & Herrsche

Aufwand Verbesserte Integer Multiplikation

- 3 Multiplikationen der Länge n/2
- [AC, BD, (A+B) (C+D)]
- Konstant viele Additionen und Multiplikationen mit Zweierpotenzen

Laufzeit

$$T(n) = \begin{cases} 3\,T(n/2) + cn & , n>1 \\ c & , n=1 \end{cases}$$

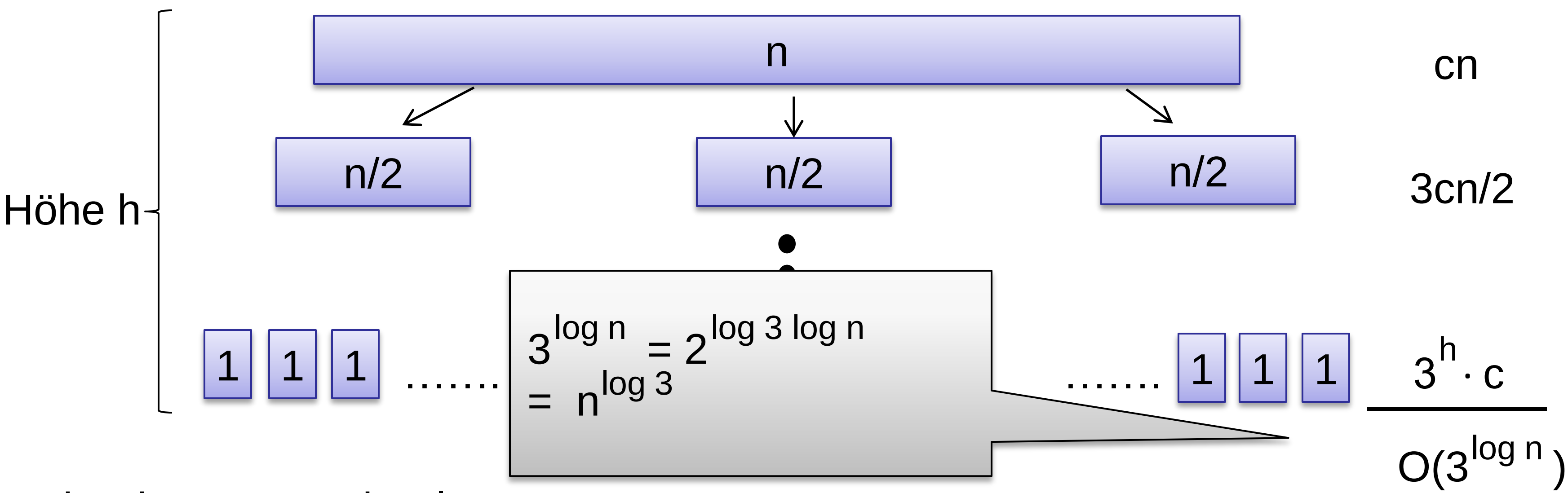
c geeignete Konstante

Teile & Herrsche

Laufzeit verbesserte Integer Multiplikation

$$T(n) \le \begin{cases} 3\,T(n/2) + cn & , n>1 \\ c & , n=1 \end{cases}$$

c geeignete Konstante



Höhe des Baums: h = log n

Teile & Herrsche

Satz 12

- Die Laufzeit der verbesserten Integer Multiplikation ist $O(3^{\log n}) = O(n^{\log 3})$.

Beweis

- Induktion über n. Sei $T(4) \leq c$. Wir zeigen $T(n) \leq c\ 3^{\log n} - 2cn$.
- (I.A.) Es gilt $T(4) \leq c = c \cdot 3^{\log 4} - 2 \cdot c \cdot 4$.
- (I.V.) Für jedes $m<n$ ist die Laufzeit für die Multiplikation zweier m-Bit Zahlen höchstens $c\ 3^{\log m} - 2cm$.
- (I.S.) Betrachte die Multiplikation von zwei n-Bit Zahlen. Es gilt
  $T(n) = 3\ T(n/2) + cn \leq c \cdot 3^{\log n} - 6\ (n/2) + cn = c \cdot 3^{\log n} - 2cn$.

Teile & Herrsche

Satz 13

- Zwei n-Bit Integer Zahlen können mit Hilfe des Teile & Herrsche Verfahrens in $O(n^{1.59})$ worst case Laufzeit multipliziert werden.

Beweis

- Die Laufzeit folgt aus Satz 12 wegen $1.59 \geq \log 3$. Korrektheit zeigen wir per Induktion über n.
- (I.A.) Die Multiplikation zweier 1-Bit Zahlen wird vom Rechner korrekt ausgeführt.
- (I.V.) Die Multiplikation von zweier m-Bit Zahlen für $m<n$ ist korrekt.
- (I.S.) Nach (I.V.) werden die Produkte AC, BD, (A+B)(C+D) korrekt berechnet. Damit folgt die Korrektheit des Algorithmus wegen $(A+B)(C+D) - AC - BD = BC + AD$ und aufgrund unserer Vorüberlegungen.

Teile & Herrsche

Matrixmultiplikation (siehe Vollversion)

$$\begin{pmatrix} 5 & 0 & 0 & 3 \\ 1 & 1 & 1 & 1 \\ 2 & 3 & 0 & 4 \\ 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 1 & 3 \\ 7 & 2 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 11 & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Zeile x Spalte

- $5 \cdot 1 + 0 \cdot 7 + 0 \cdot 0 + 3 \cdot 2 = 11$

Teile & Herrsche

Matrixmultiplikation

$$\begin{pmatrix} 5 & 0 & 0 & 3 \\ 1 & 1 & 1 & 1 \\ 2 & 3 & 0 & 4 \\ 0 & 0 & 1 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 2 & 1 & 3 \\ 7 & 2 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 11 & 10 & 17 & 15 \\ 10 & 5 & 7 & 3 \\ 31 & 10 & 24 & 6 \\ 2 & 1 & 4 & 0 \end{pmatrix}$$

Teile & Herrsche

Matrixmultiplikation

- Problem: Berechne das Produkt zweier n×n-Matrizen
- Eingabe: Matrizen X, Y
- Ausgabe: Matrix Z = X · Y

$$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{pmatrix} \bullet \begin{pmatrix} y_{1,1} & y_{1,2} & y_{1,3} & y_{1,4} \\ y_{2,1} & y_{2,2} & y_{2,3} & y_{2,4} \\ y_{3,1} & y_{3,2} & y_{3,3} & y_{3,4} \\ y_{4,1} & y_{4,2} & y_{4,3} & y_{4,4} \end{pmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} \\ z_{2,1} & z_{2,2} & z_{2,3} & z_{2,4} \\ z_{3,1} & z_{3,2} & z_{3,3} & z_{3,4} \\ z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} \end{pmatrix}$$

Teile & Herrsche

- MatrixMultiplikation (Array X,Y,n)

Laufzeit

1. **new array** Z[1..n][1..n] — $\Theta(n^2)$
2. **for** i $\leftarrow$ 1 **to** n **do**
3. **for** j $\leftarrow$ 1 **to** n **do**
4. Z[i][j] $\leftarrow$ 0
5. **for** k $\leftarrow$ 1 **to** n **do**
6. Z[i][j] $\leftarrow$ Z[i][j] + X[i][k] $\cdot$ Y[k][j]
7. **return** Z

<u>Ausnahme (Rechenmodell):</u>
Dynamische Initialisierung eines Feldes benötigt Zeit proportional zur Größe des Feldes

Teile & Herrsche

		Laufzeit
▪	MatrixMultiplikation (Array X,Y,n)	
1.	**new array** Z[1..n][1..n]	$\Theta(n^2)$
2.	**for** i $\leftarrow$ 1 **to** n **do**	$\Theta(n)$
3.	**for** j $\leftarrow$ 1 **to** n **do**	$\Theta(n^2)$
4.	Z[i][j] $\leftarrow$ 0	$\Theta(n^2)$
5.	**for** k $\leftarrow$ 1 **to** n **do**	$\Theta(n^3)$
6.	Z[i][j] $\leftarrow$ Z[i][j] + X[i][k] · Y[k][j]	$\Theta(n^3)$
7.	**return** Z	$\Theta(1)$
		$\Theta(n^3)$

Teile & Herrsche

Matrixmultiplikation

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \bullet \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

Aufwand

- 8 Multiplikationen von $n/2 \times n/2$ Matrizen
- 4 Additionen von $n/2 \times n/2$ Matrizen

- *Laufzeit*

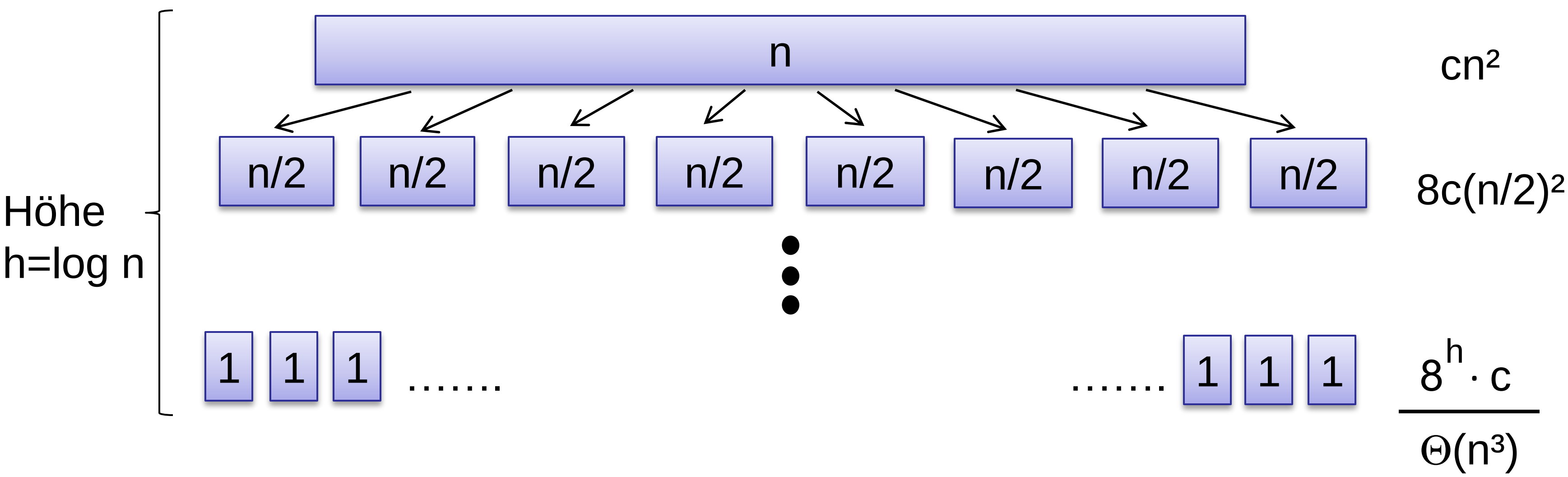
$$T(n) = \begin{cases} 8\,T(n/2) + cn^2 & \text{, für } n>1 \\ c & \text{, für } n=1 \end{cases}$$

Teile & Herrsche

Laufzeit einfache Matrixmultiplikation

$$T(n) = \begin{cases} 8\,T(n/2) + cn^2 & , n>1 \\ c & , n=1 \end{cases}$$

c geeignete Konstante



Teile & Herrsche

Satz 14

- Die einfache Matrizenmultiplikation hat Laufzeit $\Theta(n^3)$.

Beweis

- Wir nehmen an, dass n eine Zweierpotenz ist und $T(2) \le c$ gilt. Wir zeigen zunächst per Induktion über n, dass $T(n) \le cn^3 - cn^2$.
- (I.A.) Für $n=2$ ist $T(2) \le c \le c \cdot 2^3 - c \cdot 2^2$
- (I.V.) Für $m<n$ ist $T(m) \le cm^3 - cm^2$.
- (I.S.) Es gilt $T(n) = 8\, T(n/2) + cn^2$. Nach (I.V.) folgt
  $T(n) \le 8\, c(n/2)^3 - 8\, c(n/2)^2 + cn^2 \le cn^3 - cn^2$.

Teile & Herrsche

Matrixmultiplikation

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \bullet \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

Trick (wie bei Integer Multiplikation)

- $P_1 = A \cdot (F-H)$
- $P_2 = (A+B) \cdot H$
- $P_3 = (C+D) \cdot E$
- $P_4 = D \cdot (G-E)$

$P_5 = (A+D) \cdot (E+H)$

$P_6 = (B-D) \cdot (G+H)$

$P_7 = (A-C) \cdot (E+F)$

$AE+BG = P_4 + P_5 + P_6 - P_2$

$AF+BH = P_1 + P_2$

$CE+DG = P_3 + P_4$

$CF+DH = P_1 + P_5 - P_3 - P_7$

Nur 7 Multiplikationen !!!!!

Teile & Herrsche

Matrixmultiplikation

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \bullet \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

Trick (wie bei Integer Multiplikation)

- $P_1 = A \cdot (F-H)$
- $P_2 = (A+B) \cdot H$
- $P_3 = (C+D) \cdot E$
- $P_4 = D \cdot (G-E)$

$P_5 = (A+D) \cdot (E+H)$

$P_5 = (B-D) \cdot (G+H)$

$P_7 = (A-C) \cdot (E+F)$

$AE+BG = P_4 + P_5 + P_6 - P_2$

$AF+BH = P_1 + P_2$

$CE+DG = P_3 + P_4$

$CF+DH = P_1 + P_5 - P_3 - P_7$

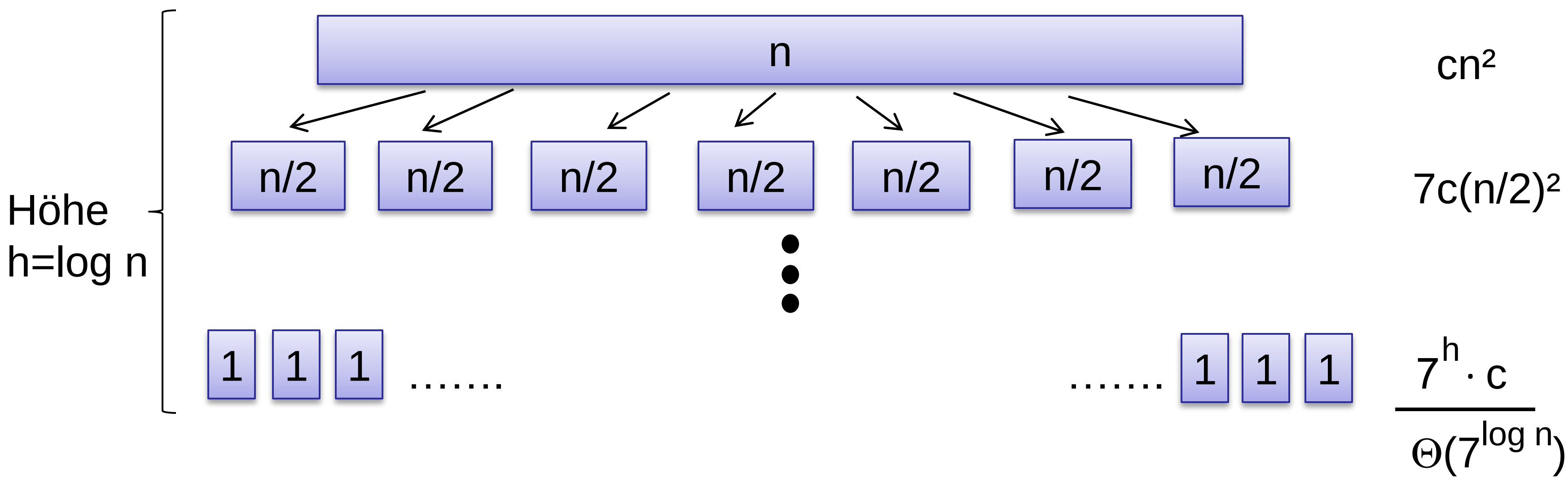
Nur 7 Multiplikationen !!!!!

Teile & Herrsche

Laufzeit einfache Matrixmultiplikation

$$T(n) = \begin{cases} 7\,T(n/2) + cn^2 & , n>1 \\ c & , n=1 \end{cases}$$

c geeignete Konstante

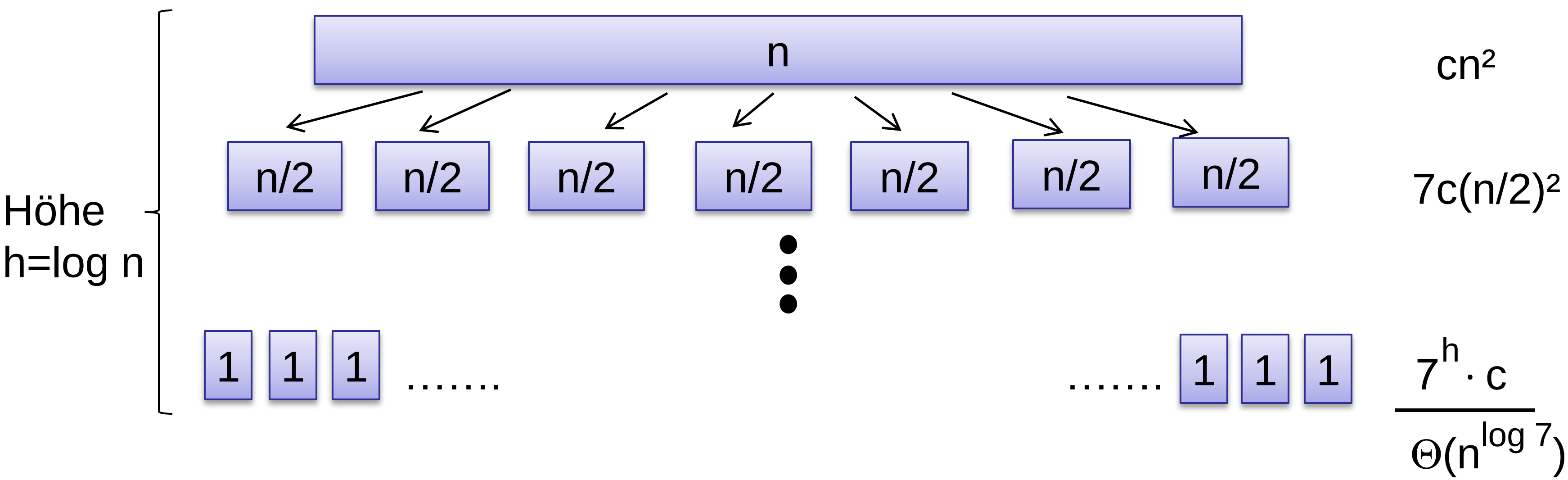


Teile & Herrsche

Laufzeit einfache Matrixmultiplikation

$$T(n) = \begin{cases} 7\,T(n/2) + cn^2 & , n>1 \\ c & , n=1 \end{cases}$$

c geeignete Konstante



Teile & Herrsche

Satz 14

- Die einfache Matrizenmultiplikation hat Laufzeit $\Theta(7^{\log n})$.

Beweis

- Wir nehmen an, dass n eine Zweierpotenz ist und $T(4) \leq c$ gilt. Wir zeigen zunächst per Induktion über n, dass $T(n) \leq c \cdot 7^{\log n} - 2cn^2$.
- (I.A.) Für $n=2$ ist $T(4) \leq c \leq c \cdot 49 - c \cdot 32$.
- (I.V.) Für $m<n$ ist $T(m) \leq c \cdot 7^{\log m} - cm^2$.
- (I.S.) Es gilt $T(n) = 7\, T(n/2) + cn^2$. Nach (I.V.) folgt $T(n) \leq 7 \cdot 7^{\log (n/2)} - 14\, c\, (n/2)^2 + cn^2 \leq 7^{\log n} - 2cn^2$.

Teile & Herrsche

Satz 15

- Zwei n-Bit Integer Zahlen können mit Hilfe des Teile & Herrsche Verfahrens in $O(n^{2.81})$ worst case Laufzeit multipliziert werden.

Beweis

- Die Laufzeit folgt aus Satz 14 wegen $2.81 \geq \log 7$. Korrektheit zeigen wir per Induktion über n.
- (I.A.) Die Multiplikation zweier 1×1 Matrizen ist korrekt.
- (I.V.) Die Multiplikation zweier $m \times m$ Matrizen für $m<n$ ist korrekt.
- (I.S.) Nach (I.V.) werden die Zwischenergebnisse P_1 bis P_7 korrekt berechnet. Damit folgt die Korrektheit des Algorithmus durch Nachrechnen der Teilformeln.

Teile & Herrsche

Zusammenfassung

- Multiplikation und Matrizenmultiplikation sind weitere Beispiel für Teile & Herrsche Algorithmen
- Faustregel: Je weniger rekursive Aufrufe desto schneller
- Trick bei Laufzeitbeweisen: Abziehen von Termen niedriger Ordnung