

Kapitel 4 - Algorithmen auf Zahlen

- Multiplikation von binären Zahlen
- Matrixmultiplikation

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b ($n=8$):

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb
xxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

```
aaaaaaaa * bbbbbb  
xxxxxxx  
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

```
aaaaaaaa * bbbbbb  
  xxxxxx  
   xxxxxx  
    xxxxxx
```

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxxxxxxxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger Binärzahlen a und b (n=8):

aaaaaaa * **b**bbbbbbb

XXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxxxxxxxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger Binärzahlen a und b (n=8):

aaaaaaaa * b**b**bbbbbb

[illegible]

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxxxxxxxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bb**b**bbbb
xxxxxxx
xxxxxxx
xxxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx

xxxxxxxxxxxxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxxxxxxxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation n-stelliger
Binärzahlen a und b (n=8):

aaaaaaaa * bbbbbbbb

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxxxxxxxxxxxxxxx

b=0: zugeh. Zeile
entfällt

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Lemma:

Die Schulmethode benötigt zur Multiplikation zweier binärer n -Bit Zahlen maximal

$$2n(n-1) = O(n^2)$$

einstellige Bitadditionen.

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

xxxxxxxxxx

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
xxxxxxx  
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
      xxxxxxxx
       xxxxxxxx
      -----
     xxxxxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
      xxxxxxxx
      xxxxxxxx
      -----
     xxxxxxxxxxx
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxxxxxxxx
      xxxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

```
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxx
      xxxxxxxxx
      -----
      xxxxxxxxx
```


Algorithmen auf Zahlen

Multiplikation - Schulmethode

Beweisskizze:

n-1 Bitstring-
Additionen

The diagram illustrates the school multiplication method using bitstrings. A large left square bracket groups a series of additions. Each addition consists of a bitstring of 'x's, a dashed line, and another bitstring of 'x's. The bitstrings are of increasing length from top to bottom, representing the partial products of a multiplication. The sequence ends with three dots, indicating further additions.

```
      xxxxxxxxx
      xxxxxxxxx
-----
xxxxxxx
      xxxxxxxxx
-----
xxxxxxxxxxxxx
      xxxxxxxxx
-----
xxxxxxxxxxxxx
      xxxxxxxxx
-----
xxxxxxxxxxxxx
      ...
```


Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

xxxxxxxx

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
xxxxxxx  
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

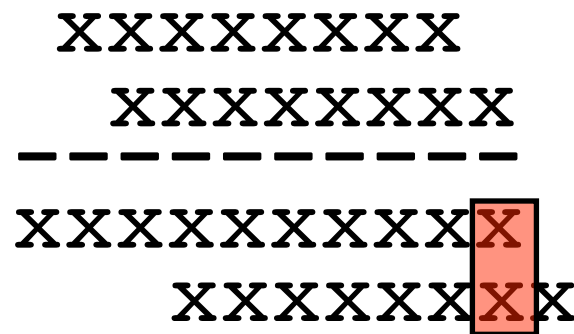
```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
 xxxxxxxxxxxxxx
       xxxxxxxxxx
```



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

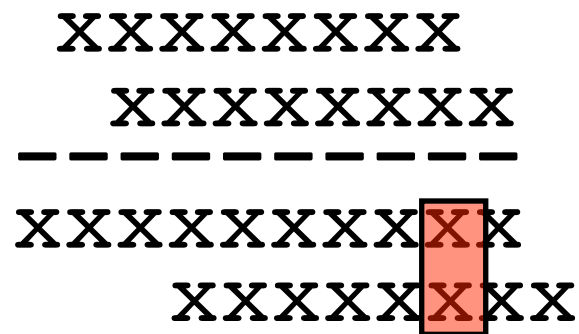
```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
 xxxxxxxxxxxxxx
       xxxxxxxxxx
```



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

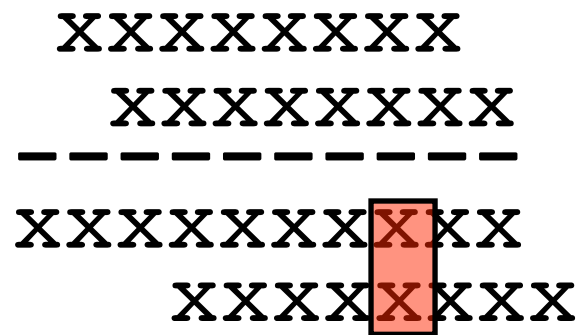
```
      xxxxxxxx
      xxxxxxxx
-----
xxxxxxx
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
 xxxxxxxxxx
  xxxxxxxxxx
```



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
      xxxxxxxx
-----
xxxxxxx
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

The diagram illustrates the school multiplication method for bit strings. It shows two 8-bit strings being multiplied. The first string is 'xxxxxxxx' and the second is 'xxxxxxxx'. A dashed line separates the two strings. Below the dashed line, the first string is shifted one position to the left, resulting in 'xxxxxxxxx' (where the last 'x' is highlighted in red). The second string is shifted two positions to the left, resulting in 'xxxxxxxxx' (where the 'x' at the third position from the left is highlighted in red). The red highlights indicate the carry-over process in the multiplication.

```
      xxxxxxxxxxx
        xxxxxxxxxxx
        -----
      xxxxxxxxxxxx
        xxxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

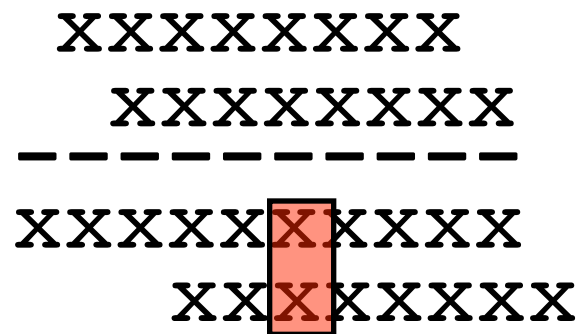
```
      xxxxxxxx
      xxxxxxxx
-----
xxxxxxx
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
     xxxxxx x xxxxx
           xx xxxxxxx
```



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

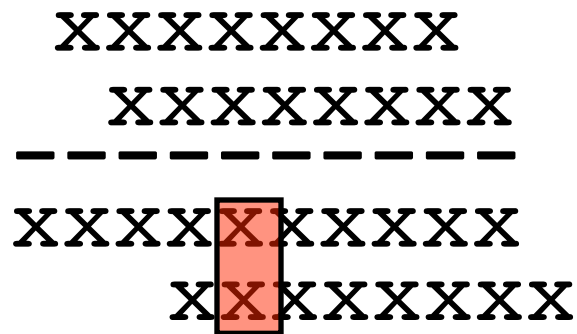
```
      xxxxxxxx
      xxxxxxxx
-----
xxxxxxx
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
     xxxxxxXxxxxx
           xxXxxxxxxx
```



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
      xxxxxxxx
```


Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
      xxxxxxxx
-----
xxxxxxx
xxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
       xxxxxxxx
      -----
xxx  xxxxxxxx
       xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxx
      xxxxxxxx
```

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx	
xxxxxxx	

xxxxxxxxxxx	
xxxxxxx	x
yyyyyyy	y

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

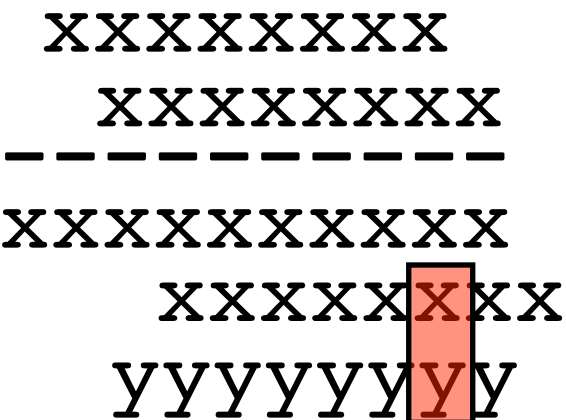
Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx	
xxxxxxx	

xxxxxxxxxxx	
xxxxxxx	
yyyyyy	Überträge



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

A diagram illustrating a 10x10 grid. The grid is composed of 10 rows and 10 columns. The first six rows are filled with 'x' characters. The seventh row is filled with 'x' characters, except for the seventh column, which is highlighted by a red vertical bar. The eighth row is filled with 'y' characters. The ninth and tenth rows are filled with 'y' characters. The red vertical bar is located at the seventh column, spanning from the seventh row to the eighth row.

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

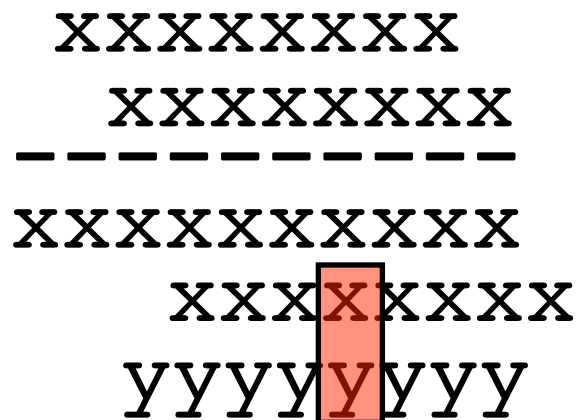
je Bitstring-Addition:

n Bit-Additionen

xxxxxxx	
xxxxxxx	

xxxxxxxxxxx	
xxx	xxxxx
yyyyy	yyy

Überträge



Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx	
xxxxxxx	

xxxxxxxxxxx	
xxx	xxxxxxx
yyy	yyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

```
      xxxxxxxx
      xxxxxxxx
      -----
      xxxxxxxxxx
      xx xxxxxx
      yy yyyyyy
```

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

A 10x10 grid of characters. The first row contains 10 'X's. The second row contains 8 'X's starting from the second column. The third row contains 10 dashes. The fourth row contains 10 'X's. The fifth row contains 10 'X's. The sixth row contains 10 'X's. The seventh row contains 10 'X's. The eighth row contains 10 'X's. The ninth row contains 10 'X's. The tenth row contains 10 'X's. A red vertical bar highlights the first column of the sixth row. The character 'X' in the sixth row, first column is red. The character 'Y' in the sixth row, second column is red.

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

	xxxxxxx	
	xxxxxxx	

	xxxxxxxxx	
	xxxxxxx	
	yyyyyyy	Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

n Bit-Additionen

xxxxxxx
xxxxxxx

xxxxxxxxxxx
xxxxxxx
yyyyyyy

Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

	xxxxxxx	
	xxxxxxx	

n Bit-Additionen	xxxxxxxxx	
	xxxxxxx	
n Bit-Additionen	yyyyyyy	Überträge

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

	xxxxxxx	
	xxxxxxx	

n Bit-Additionen	xxxxxxxxxxx	
	xxxxxxx	
n Bit-Additionen	yyyyyyyyy	Überträge

	xxxxxxxxxxx	

Algorithmen auf Zahlen

Multiplikation - Schulmethode

je Bitstring-Addition:

	xxxxxxx	
	xxxxxxx	

n Bit-Additionen	xxxxxxxxxxx	
	xxxxxxx	
n Bit-Additionen	yyyyyyyyy	Überträge

2n Bit-Additionen	xxxxxxxxxxx	

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Gesucht: wesentlich schnellere Methode
- Idee: **Divide and conquer**-Verfahren (teile und herrsche)
 - Zerlege das Problem in zwei Teile
 - löse das Problem für die beiden Teile
 - füge die Teilergebnisse zum Gesamtergebnis zusammen.

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Wie teilt man zwei n -stellige Binärzahlen $a=(a_{n-1}, \dots, a_0)$ und $b=(b_{n-1}, \dots, b_0)$?

- Setze

$$a^{(1)}=(a_{n-1}, \dots, a_{\lceil n/2 \rceil}), a^{(0)}=(a_{\lceil n/2 \rceil - 1}, \dots, a_0)$$

$$b^{(1)}=(b_{n-1}, \dots, b_{\lceil n/2 \rceil}), b^{(0)}=(b_{\lceil n/2 \rceil - 1}, \dots, b_0)$$

- Dann gilt:

$$a=a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}$$

$$b=b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) =$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= \boxed{a^{(1)} \cdot b^{(1)}} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) =$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= \boxed{a^{(1)} \cdot b^{(1)}} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + \boxed{a^{(1)} \cdot b^{(0)}} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + \boxed{a^{(1)} \cdot b^{(0)}} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ \boxed{a^{(0)} \cdot b^{(1)}} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ \boxed{a^{(0)} \cdot b^{(1)}} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + \boxed{a^{(0)} \cdot b^{(0)}}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) + O(n)$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned} a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\ &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\ &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)} \end{aligned}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) + O(n)$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$a \cdot b = (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)})$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}$$

- Laufzeit:

$$T(n) = T(n/2) + T(n/2) + T(n/2) + T(n/2) + O(n)$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned}
 a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\
 &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\
 &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

kostenlos:
Linksschieben
= *2

- Laufzeit:

$$\begin{aligned}
 T(n) &= T(n/2) + T(n/2) + T(n/2) + T(n/2) + \\
 &\quad O(n)
 \end{aligned}$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Dann gilt ferner:

$$\begin{aligned}
 a \cdot b &= (a^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)}) \cdot (b^{(1)} \cdot 2^{\lceil n/2 \rceil} + b^{(0)}) \\
 &= a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} \\
 &\quad + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

kostenlos:
Linksschieben
= *2

- Laufzeit:

$$\begin{aligned}
 T(n) &= T(n/2) + T(n/2) + T(n/2) + T(n/2) + O(n) \\
 &= 4T(n/2) + O(n)
 \end{aligned}$$

Algorithmen auf Zahlen

Multiplikation - Beschleunigung

- Leider gilt: Aus

$$T(n) = 4 * T(n/2) + O(n),$$

$$T(1) = 1$$

- folgt (selbst nachweisen!)

$$T(n) = O(n^2),$$

also nicht besser als die Schulmethode.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

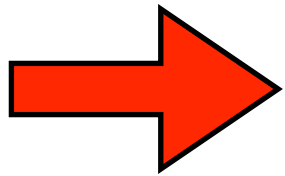
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

| Multiplikation und 3 Additionen.

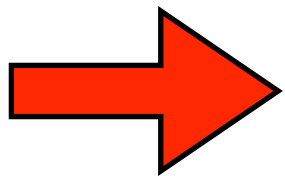
Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

hier:
2 Multiplikationen



$$+a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) = (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

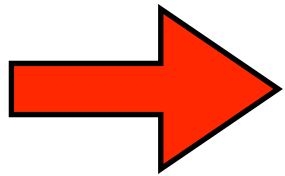
1 Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

| Multiplikation und 3 Additionen.

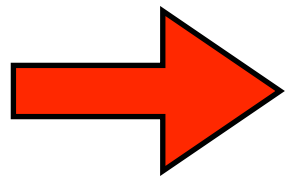
Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

hier:
1 Addition



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

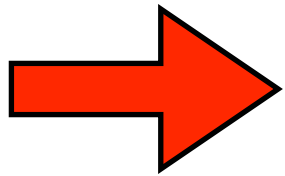
1 Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

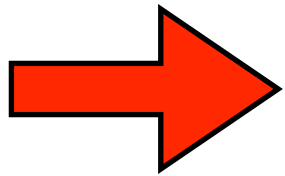
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

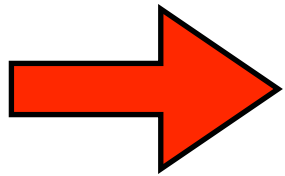
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

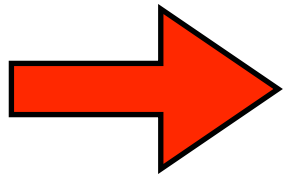
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

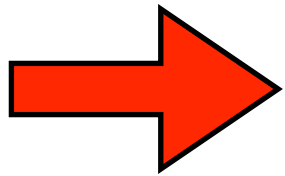
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

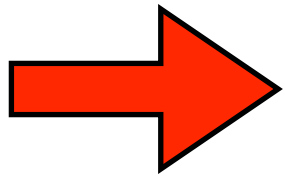
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = \boxed{a^{(1)} \cdot b^{(1)}} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

liegen
bereits vor

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) = \boxed{a^{(1)} \cdot b^{(1)}} + a^{(0)} \cdot b^{(0)}.$$

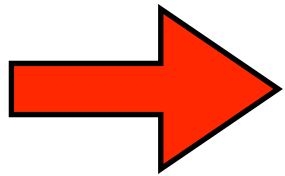
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

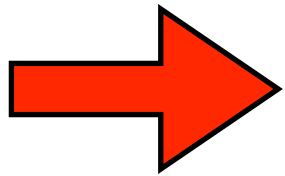
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) = (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

liegen
bereits vor

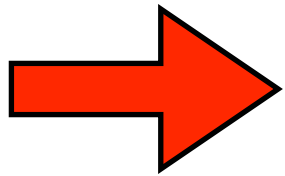
| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - geniale Idee

- Geniale Idee:

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$



$$+ a^{(1)} \cdot b^{(0)} \cdot 2^{\lceil n/2 \rceil} + a^{(0)} \cdot b^{(1)} \cdot 2^{\lceil n/2 \rceil}$$

$$+ a^{(0)} \cdot b^{(0)}$$

- umschreiben zu:

$$(a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

| Multiplikation und 3 Additionen.

Algorithmen auf Zahlen

Multiplikation - Ergebnis

- Insgesamt:
 - Gespart: 1 Multiplikation
 - Geopfert: 3 Additionen
 - Da Multiplikationen aufwendiger als Additionen insgesamt Operationen und Zeit gespart.

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ &+ a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$\begin{aligned} T(n) &= 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ &\leq 3 T(\lceil n/2 \rceil + 1) + O(n). \end{aligned}$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil} + (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ \leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ &+ a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$\begin{aligned} T(n) &= 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ &\leq 3 T(\lceil n/2 \rceil + 1) + O(n). \end{aligned}$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ &+ a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$\begin{aligned} T(n) &= 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ &\leq 3 T(\lceil n/2 \rceil + 1) + O(n). \end{aligned}$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} & \boxed{+} (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)}) \boxed{+} (a^{(0)} \cdot b^{(0)}) \\ & \boxed{+} a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \boxed{O(n)}$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ &+ a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+} (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ \leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+} (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n) \\ \leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:
 $2n +$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+}(a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:
 $2n +$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+}(a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:
 $2n + 2(n/2) +$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+}(a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:
 $2n + 2(n/2) +$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\boxed{+}(a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\ + a^{(0)} \cdot b^{(0)}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned}
 &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\
 &+ a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned}
 & \boxed{+} (a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\
 & \quad + a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned}
 & \boxed{+} (a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)} \boxed{+} a^{(0)} \cdot b^{(0)}) \\
 & \quad + a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned}
 &+ (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}) \\
 &+ a^{(0)} \cdot b^{(0)}
 \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n + 2n +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} & \boxed{+} (a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)} \boxed{+} a^{(0)} \cdot b^{(0)}) \\ & \boxed{+} a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n + 2n +$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} & \boxed{+} (a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)} \boxed{+} a^{(0)} \cdot b^{(0)}) \\ & \boxed{+} a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n + 2n + 2n$$

Algorithmen auf Zahlen

Multiplikation - Ergebnis

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2 \lceil n/2 \rceil}$$

$$\begin{aligned} & \boxed{+} (a^{(1)} \boxed{+} a^{(0)}) \cdot (b^{(1)} \boxed{+} b^{(0)}) \boxed{-} (a^{(1)} \cdot b^{(1)} \boxed{+} a^{(0)} \cdot b^{(0)}) \\ & \boxed{+} a^{(0)} \cdot b^{(0)} \end{aligned}$$

Laufzeit nun:

$$T(n) = 2 T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + O(n)$$

$$\leq 3 T(\lceil n/2 \rceil + 1) + O(n).$$

Konstante in $O(n)$:

$$2n + 2(n/2) + 2(n/2) + 2n + 2n + 2n = 10n$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$
$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

$$\leq 3^k T(n/2^k + 2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i) + 2)$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

$$\leq 3^k T(n/2^k + 2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i) + 2)$$

für $k = \log n$ gilt dann

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

$$\leq 3^k T(n/2^k + 2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i) + 2)$$

für $k = \log n$ gilt dann

$$\leq 3^{\log n} T(n/2^{\log n} + 2) + \sum_{i=0}^{\log n - 1} (3^i \cdot 10(n/2^i) + 2)$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

$$\leq 3^k T(n/2^k + 2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i) + 2)$$

für $k = \log n$ gilt dann

$$\leq 3^{\log n} T(n/2^{\log n} + 2) + \sum_{i=0}^{\log n - 1} (3^i \cdot 10(n/2^i) + 2)$$

$$\leq 2^{\log 3 \log n} T(3) + 10 \sum_{i=0}^{\log n - 1} (3^i \cdot (n/2^i) + 2)$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Löse Rekursionsgleichung (Annahme n
Zweierpotenz):

$$T(n) = 3 T(n/2 + 1) + O(n)$$

$$\leq 3^2 T(n/2^2 + 2) + 3 \cdot 10(n/2 + 1) + 2 + 10n$$

$$= 3^3 T(n/2^3 + 2) + 3^2 \cdot 10(n/2^2) + 2 \cdot 3 \cdot 10(n/2) + 2 + 10n$$

... nach k Iterationen

$$\leq 3^k T(n/2^k + 2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i) + 2)$$

für $k = \log n$ gilt dann

$$\leq 3^{\log n} T(n/2^{\log n} + 2) + \sum_{i=0}^{\log n - 1} (3^i \cdot 10(n/2^i) + 2)$$

$$\leq 2^{\log 3 \log n} T(3) + 10 \sum_{i=0}^{\log n - 1} (3^i \cdot (n/2^i) + 2)$$

$$\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i/2^i) + 2 \log n, \text{ da } T(3) \leq 9$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9$$
$$\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / (3/2 - 1) + 2 \log n$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned} &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\ &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\ &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \end{aligned}$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9$$

$$\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n$$

$$\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n$$

$$\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned}
 &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\
 &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\
 &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\
 &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\
 &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx 1.585).
 \end{aligned}$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned}
 &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\
 &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\
 &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\
 &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\
 &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx 1.585).
 \end{aligned}$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned} &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\ &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\ &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\ &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\ &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx \\ &1.585). \end{aligned}$$

Satz

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned} &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\ &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\ &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\ &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\ &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx 1.585). \end{aligned}$$

Satz

Zwei n-stellige Binärzahlen können mit

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned} &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\ &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\ &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\ &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\ &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx 1.585). \end{aligned}$$

Satz

Zwei n-stellige Binärzahlen können mit

$$29n^{\log 3} + 2 \log n = O(n^{\log 3})$$

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

$$\begin{aligned} &\leq 9n^{\log 3} + 10 \sum_{i=0}^{\log n - 1} (3^i / 2^i) + 2 \log n, \text{ da } T(3) \leq 9 \\ &\leq 9n^{\log 3} + 10n((3/2)^{\log n} - 1) / ((3/2) - 1) + 2 \log n \\ &\leq 9n^{\log 3} + 20n \cdot 2^{\log 1,5 \log n - 1} + 2 \log n \\ &\leq 9n^{\log 3} + 20n^{\log 3} + 2 \log n \\ &\leq 29n^{\log 3} + 2 \log n = O(n^{\log 3}) \quad (\log 3 \approx 1.585). \end{aligned}$$

Satz

Zwei n-stellige Binärzahlen können mit

$$29n^{\log 3} + 2 \log n = O(n^{\log 3})$$

vielen Bit-Operationen multipliziert werden.

Algorithmen auf Zahlen

Multiplikation - Drawback

$$T(n) = 29n^{\log 3} + 2 \log n = O(n^{\log 3})$$

- Wann ist dieser Ausdruck überhaupt besser als das traditionelle n^2 bei Schulmethode?
- $$29n^{\log 3} + 2 \log n < n^2$$

gilt erst ab etwa $n=500$ -stelligen Binärzahlen.
- Für $n < 500$ ist die Schulmethode besser.

Algorithmen auf Zahlen

Multiplikation - Drawback

- Verbesserung: Breche den Algorithmus bei Binärzahlen der Länge 2^m+2 ab und gehe von dort zur Schulmethode über.
- Wie findet man den optimalen Umschaltzeitpunkt:

...

$$\leq 3^k T(n/2^k+2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n/2^i)+2)$$

Rechne von hier für $k=\log n - m$ weiter:

...

- Ergebnis nach längerer Rechnerei: optimaler Umschaltzeitpunkt: Länge der Binärzahlen zwischen 10 und 18.
- Konstante sinkt dann auf etwa 10.

Algorithmen auf Zahlen

Multiplikation - Rekursionsgleichung

Satz

Zwei n -stellige Binärzahlen können mit
 $10n^{\log 3} + 2 \log n = O(n^{\log 3})$
vielen Bit-Operationen multipliziert werden.

Algorithmus schlägt Schulmethode ab etwa $n=50$.

35 Jahre lang bester Algorithmus:

Satz (A. Schönhage, V. Strassen, 1971)

Zwei n -stellige Binärzahlen können mit
 $O(n \log n \log \log n)$
vielen Bit-Operationen multipliziert werden.

Algorithmen auf Zahlen

Multiplikation - bestes Ergebnis zur Zeit

Aktuell bester Algorithmus:

Satz (M. Fürer, 2007)

Zwei n -stellige Binärzahlen können mit

$$O(n \log n 2^{O(\log^* n)})$$

vielen Bit-Operationen multipliziert werden.

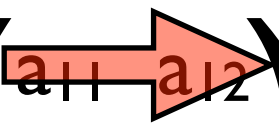
Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{cc} A & B \\ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} & \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \\ \\ \begin{pmatrix} & & \end{pmatrix} \\ C \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} A \qquad \qquad B \\ \left(\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right) \cdot \left(\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array} \right) = \\ \\ \left(\begin{array}{cc} & \\ & \end{array} \right) \\ C \end{array}$$


Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} \text{A} \qquad \qquad \text{B} \\ \left(\begin{array}{cc} \overrightarrow{a_{11} \quad a_{12}} \\ a_{21} \quad a_{22} \end{array} \right) \cdot \left(\begin{array}{cc} b_{11} \quad b_{12} \\ \downarrow b_{21} \quad b_{22} \end{array} \right) = \\ \left(\qquad \qquad \qquad \right) \\ \text{C} \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} \text{A} \qquad \qquad \text{B} \\ \left(\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right) \cdot \left(\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array} \right) = \\ \left(\begin{array}{c} a_{11}b_{11} + a_{12}b_{21} \\ \phantom{a_{11}b_{11} + a_{12}b_{21}} \end{array} \right) \\ \text{C} \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} \text{A} \qquad \qquad \text{B} \\ \left(\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right) \cdot \left(\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array} \right) = \\ \left(\begin{array}{c} a_{11}b_{11} + a_{12}b_{21} \\ \phantom{a_{11}b_{11} + a_{12}b_{21}} \end{array} \right) \\ \text{C} \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} \text{A} \qquad \qquad \text{B} \\ \left(\begin{array}{cc} \xrightarrow{a_{11}} & \xrightarrow{a_{12}} \\ \xrightarrow{a_{21}} & \xrightarrow{a_{22}} \end{array} \right) \cdot \left(\begin{array}{cc} \downarrow b_{11} & \downarrow b_{12} \\ \downarrow b_{21} & \downarrow b_{22} \end{array} \right) = \\ \left(\begin{array}{c} a_{11}b_{11} + a_{12}b_{21} \\ \phantom{a_{11}b_{11} + a_{12}b_{21}} \end{array} \right) \\ \text{C} \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{array}{c} \text{A} \qquad \qquad \text{B} \\ \left(\begin{array}{cc} \xrightarrow{a_{11}} & \xrightarrow{a_{12}} \\ \xrightarrow{a_{21}} & \xrightarrow{a_{22}} \end{array} \right) \cdot \left(\begin{array}{cc} \downarrow b_{11} & \downarrow b_{12} \\ \downarrow b_{21} & \downarrow b_{22} \end{array} \right) = \\ \left(\begin{array}{c} a_{11}b_{11}+a_{12}b_{21} \\ a_{21}b_{12}+a_{22}b_{22} \end{array} \right) \\ \text{C} \end{array}$$

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{matrix} & A & & B \\ \begin{pmatrix} \xrightarrow{a_{11}} a_{12} \\ \xrightarrow{a_{21}} a_{22} \end{pmatrix} & \cdot & \begin{pmatrix} \downarrow b_{11} \downarrow b_{12} \\ \downarrow b_{21} \downarrow b_{22} \end{pmatrix} & = \end{matrix}$$

$$\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

C

Algorithmen auf Zahlen

Matrixmultiplikation

$$\begin{matrix} & A & & B \\ \begin{pmatrix} \xrightarrow{a_{11}} a_{12} \\ \xrightarrow{a_{21}} a_{22} \end{pmatrix} & \cdot & \begin{pmatrix} \downarrow b_{11} \downarrow b_{12} \\ \downarrow b_{21} \downarrow b_{22} \end{pmatrix} & = \end{matrix}$$

$$\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}$$

C

Algorithmen auf Zahlen

Matrixmultiplikation

- Gegeben zwei $n \times n$ -Matrizen A und B.
- Gesucht: Matrix $C = A \cdot B$
- Anwendungen in allen Gebieten:
 - Wegeprobleme in Graphen
 - Physik (riesige Matrizen)
 - lineare Algebra

Algorithmen auf Zahlen

Matrixmultiplikation - Schulmethode

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \dots + a_{in-1}b_{n-1j} + a_{in}b_{nj}$$

- je Matrixelement c_{ij} nötig: n Multiplikationen und $n-1$ Additionen
- n^2 Matrixelemente sind zu berechnen:

$$n^2(n+n-1) = 2n^3 - n^2 = O(n^3)$$

Operationen.

Algorithmen auf Zahlen

Matrixmultiplikation - geniale Ideen

- Divide-and-conquer-Verfahren
- intelligente Berechnungsmethode, die Multiplikationen auf Kosten von Additionen spart
- Generalannahme: n Zweierpotenz, sonst Matrix auf nächste Zweierpotenz mit Nullen erweitern

Algorithmen auf Zahlen

Matrixmultiplikation - Vorgehen

Divide-and-conquer-Verfahren

- Teile A und B in je 4 gleichgroße $n/2 \times n/2$ Teilmatrizen auf:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

und berechne

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Vorgehen

Divide-and-conquer-Verfahren

- Teile A und B in je 4 gleichgroße $n/2 \times n/2$ Teilmatrizen auf:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

und berechne

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

8 Matrixmultiplikationen
von $n/2 \times n/2$ -Matrizen
und 4 Matrixadditionen

Algorithmen auf Zahlen

Matrixmultiplikation - Vorgehen

- Laufzeit:

$$T(n) = 8T(n/2) + n^2$$

$$T(1) = 1$$

- Lösung leider (Selbst nachweisen!):

$$T(n) = O(n^3)$$

- kein Gewinn gegenüber Schulmethode

Algorithmen auf Zahlen

Matrixmultiplikation - Strassen-Algor.

- Spare durch geschickte Zwischenrechnungen eine Multiplikation ein:
- Berechne:

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$M_5 = A_{11}(B_{12} - B_{22})$$

$$M_7 = (A_{21} + A_{22})B_{11}$$

und dann:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{21} = M_6 + M_7$$

$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_4 = (A_{11} + A_{12})B_{22}$$

$$M_6 = A_{22}(B_{21} - B_{11})$$

$$C_{12} = M_4 + M_5$$

$$C_{22} = M_2 - M_3 + M_5 - M_7.$$

Algorithmen auf Zahlen

Matrixmultiplikation - Strassen-Algor.

- Spare durch geschickte Zwischenrechnungen eine Multiplikation ein:

- Berechne:

7 Multiplikationen

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$M_5 = A_{11}(B_{12} - B_{22})$$

$$M_7 = (A_{21} + A_{22})B_{11}$$

und dann:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{21} = M_6 + M_7$$

$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_4 = (A_{11} + A_{12})B_{22}$$

$$M_6 = A_{22}(B_{21} - B_{11})$$

$$C_{12} = M_4 + M_5$$

$$C_{22} = M_2 - M_3 + M_5 - M_7.$$

Algorithmen auf Zahlen

Matrixmultiplikation - Strassen-Algor.

- Spare durch geschickte Zwischenrechnungen eine Multiplikation ein:
- Berechne:

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$M_5 = A_{11}(B_{12} - B_{22})$$

$$M_7 = (A_{21} + A_{22})B_{11}$$

und dann:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{21} = M_6 + M_7$$

$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_4 = (A_{11} + A_{12})B_{22}$$

$$M_6 = A_{22}(B_{21} - B_{11})$$

$$C_{12} = M_4 + M_5$$

$$C_{22} = M_2 - M_3 + M_5 - M_7.$$

Algorithmen auf Zahlen

Matrixmultiplikation - Strassen-Algor.

- Spare durch geschickte Zwischenrechnungen eine Multiplikation ein:

- Berechne:

18 Additionen

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$M_5 = A_{11}(B_{12} - B_{22})$$

$$M_7 = (A_{21} + A_{22})B_{11}$$

und dann:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{21} = M_6 + M_7$$

$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_4 = (A_{11} + A_{12})B_{22}$$

$$M_6 = A_{22}(B_{21} - B_{11})$$

$$C_{12} = M_4 + M_5$$

$$C_{22} = M_2 - M_3 + M_5 - M_7.$$

Algorithmen auf Zahlen

Matrixmultiplikation - Strassen-Algor.

- Spare durch geschickte Zwischenrechnungen eine Multiplikation ein:
- Berechne:

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$M_5 = A_{11}(B_{12} - B_{22})$$

$$M_7 = (A_{21} + A_{22})B_{11}$$

und dann:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{21} = M_6 + M_7$$

$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_4 = (A_{11} + A_{12})B_{22}$$

$$M_6 = A_{22}(B_{21} - B_{11})$$

$$C_{12} = M_4 + M_5$$

$$C_{22} = M_2 - M_3 + M_5 - M_7.$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

- Laufzeit:

$$T(n) = 7T(n/2) + 18(n/2)^2 = 7T(n/2) + 9n^2/2$$

$$\text{mit } T(1) = 1$$

- Lösung durch Abwicklung der Rekursion

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$T(n) = 7T(n/2) + 9n^2/2$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned} T(n) &= 7T(n/2) + 9n^2/2 \\ &= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$\begin{aligned}&= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i \\&= 2^{\log 7 \log n} + (9/2)n^2 \sum_{i=0}^{\log n - 1} (7/4)^i\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$\begin{aligned}&= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i \\&= 2^{\log 7 \log n} + (9/2)n^2 \sum_{i=0}^{\log n - 1} (7/4)^i \\&= n^{\log 7} + (9/2)n^2 (((7/4)^{\log n} - 1)/(7/4 - 1))\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$\begin{aligned}&= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i \\&= 2^{\log 7 \log n} + (9/2)n^2 \sum_{i=0}^{\log n - 1} (7/4)^i \\&= n^{\log 7} + (9/2)n^2 (((7/4)^{\log n} - 1)/(7/4 - 1)) \\&\leq n^{\log 7} + (9/2)n^2 (4/3) n^{\log 7 - \log 4}\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$\begin{aligned}&= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i \\&= 2^{\log 7 \log n} + (9/2)n^2 \sum_{i=0}^{\log n - 1} (7/4)^i \\&= n^{\log 7} + (9/2)n^2 (((7/4)^{\log n} - 1)/(7/4 - 1)) \\&\leq n^{\log 7} + (9/2)n^2 (4/3) n^{\log 7 - \log 4} \\&\leq n^{\log 7} + 6 n^{\log 7}\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Laufzeit

$$\begin{aligned}T(n) &= 7T(n/2) + 9n^2/2 \\&= 7 \cdot (7T(n/4) + (9/2)(n/2)^2) + 9n^2/2 \\&= 7^2 T(n/4) + 7 \cdot (9/2)(n/2)^2 + 9n^2/2\end{aligned}$$

Nach k Iterationen erhält man:

$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2)(n/2^i)^2$$

mit $k = \log n$ erhält man

$$\begin{aligned}&= 7^{\log n} T(1) + \sum_{i=0}^{\log n - 1} (7/4)^i \\&= 2^{\log 7 \log n} + (9/2)n^2 \sum_{i=0}^{\log n - 1} (7/4)^i \\&= n^{\log 7} + (9/2)n^2 (((7/4)^{\log n} - 1)/(7/4 - 1)) \\&\leq n^{\log 7} + (9/2)n^2 (4/3) n^{\log 7 - \log 4} \\&\leq n^{\log 7} + 6 n^{\log 7} \\&= 7 n^{\log 7} \approx 7 n^{2.81}\end{aligned}$$

Algorithmen auf Zahlen

Matrixmultiplikation - Ergebnis

Satz:

Der Algorithmus von Strassen benötigt zur Multiplikation zweier $n \times n$ -Matrizen $7n^{\log 7}$ viele arithmetische Operationen.

Nachrechnen ergibt: Strassen-Algorithmus schlägt Schulmethode ab $n \geq 700$.

Algorithmen auf Zahlen

Matrixmultiplikation - Verbesserungen

- 7 Multiplikationen und **15** Additionen $\Rightarrow$
Strassen-Algorithmus schlägt Schulmethode
ab $n \geq 310$

Algorithmen auf Zahlen

Matrixmultiplikation - Verbesserungen

- Gehe zur Schulmethode über, wenn diese effizienter ist als der Strassen-Algorithmus. Ab welcher Matrizengröße $2^m \times 2^m$ empfiehlt sich das?
- Wie oben rechnen:
$$= 7^k T(n/2^k) + \sum_{i=0}^{k-1} 7^i \cdot (9/2) (n/2^i)^2$$
mit $k = \log n - m$ weiterrechnen und bestmögliches m ermitteln
- Ergebnis: Umschaltzeitpunkt zwischen 8 und 16.
- Konstante sinkt dann auf etwa 4.

Algorithmen auf Zahlen

Matrixmultiplikation - Ergebnis

Satz:

Der Algorithmus von Strassen benötigt zur Multiplikation zweier $n \times n$ -Matrizen $4n^{\log 7}$ viele arithmetische Operationen, wenn man bei kleinen Matrizen auf die Schulmethode umsteigt.

Nachrechnen ergibt: Strassen-Algorithmus schlägt Schulmethode ab $n \geq 40$.

Algorithmen auf Zahlen

Matrixmultiplikation - state of the art

Jahr	Autoren	Laufzeit
1969	Strassen	$O(n^{2,808})$
1979	Pan	$O(n^{2,781})$
1979	Bin, Capovani, Lotti, Romani	$O(n^{2,7799})$
1979	Schönhage	$O(n^{2,548})$
1979	Pan	$O(n^{2,522})$
1982	Coppersmith, Winograd	$O(n^{2,496})$
1982	Strassen	$O(n^{2,47})$
1982	Coppersmith, Winograd	$O(n^{2,38})$

Algorithmen wegen der Konstanten
nur von theoretischem Interesse