Teile und Herrsche
Teil 2
binär Suchen und schnell Multiplizieren

Markus Fleck
Manuel Mauky

Hochschule Zittau/Görlitz

19. April 2009

Suchen in langen Listen

(0, 1, 2, 7, 8, 9, 9, 13, 13, 14, 14, 14, 16, 17, 21, 22, 22, 23, 23, 23, 25, 25, 27, 30, 31, 31, 32, 33, 39, 39, 42, 43, 45, 46, 46, 49, 52, 52, 52, 52, 53, 54, 55, 55, 57, 58, 59, 60, 68, 68, 69, 74, 76, 78, 78, 78, 78, 78, 79, 80, 82, 84, 84, 88, 88, 88, 90, 91, 92, 92, 93, 94, 94, 94, 95, 96, 97, 98, 99, 100, 101, 102, 102, 103, 103, 103, 104, 105, 105, 106, 107, 107, 107, 108, 108, 111, 111, 111, 112, 112, 114, 114, 115, 116, 116, 118, 118, 119, 119, 120, 123, 123, 126, 128, 129, 131, 131, 132, 133, 133, 134, 135, 136, 136, 136, 137, 138, 139, 141, 142, 143, 143, 145, 145, 147, 149, 151, 156, 158, 159, 160, 161, 163, 164, 164, 165, 167, 172, 172, 172, 172, 172, 173, 173, 175, 178, 178, 178, 178, 179, 180, 181, 181, 182, 182, 183, 183, 184, 185, 185, 185, 186, 186, 187, 190, 191, 192, 193, 193, 194, 196, 196, 197, 198, 199, 200, 200, 201, 202, 204, 204, 205, 207, 215, 215, 215, 215, 215, 215, 219, 219, 219, 221, 221, 221, 223, 225, 226, 227, 227, 228, 229, 229, 230, 231, 232, 232, 234, 234, 234, 235, 236, 236, 237, 237, 239, 240, 242, 244, 246, 248, 248, 249, 249, 249, 249, 252, 253, 253, 254, 256, 257, 257, 258, 259, 260, 261, 263, 265, 266, 267, 267, 268, 270, 271, 273, 273, 277, 277, 277, 277, 278, 284, 285, 285, 287, 287, 289, 289, 290, 290, 292, 292, 295, 296, 297, 297, 299, 299, 300, 302, 302, 304, 304, 304, 305, 305, 305, 308, 309, 309, 310, 312, 312, 313, 313, 315, 315, 316, 317, 317, 318, 318, 319, 319, 320, 320, 321, 323, 323, 324, 324, 324, 327, 329, 330, 330, 333, 334, 335, 336, 337, 339, 342, 342, 342, 343, 348, 348, 349, 349, 350, 351, 351, 351, 353, 353, 354, 355, 355, 356, 356, 357, 358, 359, 359, 360, 360, 360, 363, 363, 364, 364, 365, 369, 371, 373, 374, 375, 376, 378, 381, 381, 381, 381, 382, 382, 382, 382, 382, 383, 383, 384, 385, 386, 386, 386, 387, 388, 389, 389, 391, 392, 392, 395, 398, 398, 399, 400, 401, 401, 402, 403, 404, 406, 407, 409, 410, 410, 411, 412, 416, 416, 417 ...)

Multiplikation langer Zahlen

$$1253435647448758968152532544646376485578698799686$$

$$*$$

$$1234123453226536345756896789679868967874674585 76$$

Matrizenmultiplikation

$$\begin{pmatrix} 125 & 124 & 123 & 122 & 119 & 119 & 119 & 113 \\ 111 & 110 & 109 & 107 & 106 & 106 & 105 & 97 \\ 93 & 89 & 88 & 88 & 87 & 87 & 85 & 84 \\ 84 & 83 & 83 & 81 & 78 & 77 & 76 & 73 \\ 70 & 62 & 62 & 61 & 60 & 59 & 57 & 54 \\ 52 & 51 & 45 & 44 & 40 & 40 & 37 & 36 \\ 34 & 32 & 27 & 26 & 24 & 19 & 18 & 18 \\ 15 & 11 & 10 & 5 & 5 & 5 & 4 & 4 \end{pmatrix}$$

$$*$$

$$\begin{pmatrix} 125 & 125 & 122 & 120 & 117 & 117 & 111 & 109 \\ 106 & 105 & 105 & 103 & 99 & 94 & 88 & 88 \\ 86 & 84 & 82 & 79 & 79 & 78 & 78 & 77 \\ 75 & 74 & 73 & 73 & 66 & 64 & 63 & 61 \\ 59 & 56 & 52 & 49 & 46 & 44 & 44 & 42 \\ 41 & 39 & 38 & 38 & 35 & 29 & 28 & 25 \\ 25 & 23 & 20 & 20 & 20 & 20 & 19 & 15 \\ 10 & 8 & 8 & 5 & 5 & 4 & 2 & 1 \end{pmatrix}$$

Binäre Suche

Problem:

Eine Liste mit sehr vielen Elementen nach der Position eines Elements zu durchsuchen.

Lösungs-Ideen

dummer Ansatz

jedes Element der Reihe nach durchprobieren!?

intelligenterer Ansatz

Suchraum einschränken!?

Entwurf nach Divide & Conquer

Divide

Der Suchraum a wird halbiert.
In $a_1 \ldots a_k$ und $a_{k+1} \ldots a_n$

Vorraussetzung

Die Liste ist sortiert!

Conquer

Divide & Conquer

Entscheidung

- wenn $x = a_{k+1}$ dann gib $k + 1$ zurück
- ist $x < a_{k+1}$ dann muss sich x in $a_1 \ldots a_k$ befinden!
- ist $x > a_{k+1}$ dann muss sich x in $a_{k+1} \ldots n$
- wenn eine Teilliste leer ist bzw. die Liste nicht mehr teilbar ist wird abgebrochen

Conquer

Divide & Conquer

Entscheidung

- wenn $x = a_{k+1}$ dann gib $k + 1$ zurück
- ist $x < a_{k+1}$ dann muss sich x in $a_1 \ldots a_k$ befinden!
- ist $x > a_{k+1}$ dann muss sich x in $a_{k+1} \ldots n$
- wenn eine Teilliste leer ist bzw. die Liste nicht mehr teilbar ist wird abgebrochen

Rekursion

Wir wenden unseren Algorithmus auf eine der beiden neuen Suchräume an.

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
5      (cond
6        ((null? rls) "Wert nicht enthalten")
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
5      (cond
6        ((null? rls) "Wert nicht enthalten")
7        ((= x (car rls)) "Wert wurde gefunden")
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
5      (cond
6        ((null? rls) "Wert nicht enthalten")
7        ((= x (car rls)) "Wert wurde gefunden")
8        ((< x (car rls))
9          (if (null? lls)
10            "Wert nicht enthalten"
11            (binaersuche x lls)))
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
5      (cond
6        ((null? rls) "Wert nicht enthalten")
7        ((= x (car rls)) "Wert wurde gefunden")
8        ((< x (car rls))
9          (if (null? lls)
10            "Wert nicht enthalten"
11            (binaersuche x lls)))
12       (else
13         (if (null? (cdr rls))
14           "Wert nicht enthalten"
15           (binaersuche x rls)))))))))
```

Binäre Suche in Scheme

```
0 (define binaersuche
1 (lambda (x ls)
2   (call-with-values
3     (lambda () (teillisten2 ls))
4     (lambda (lls rls)
5      (cond
6        ((null? rls) "Wert nicht enthalten")
7        ((= x (car rls)) "Wert wurde gefunden")
8        ((< x (car rls))
9          (if (null? lls)
10            "Wert nicht enthalten"
11            (binaersuche x lls)))
12        (else
13          (if (null? (cdr rls))
14            "Wert nicht enthalten"
15            (binaersuche x rls)))))))))
```

Aufwandsanalyse

Rekursiongleichung

$$T(n) =$$

Aufwandsanalyse

Rekursiongleichung

$$T(n) = T(\frac{n}{2}) + c$$

Aufwandsanalyse

Rekursiongleichung

$$T(n) = T(\frac{n}{2}) + c$$

$$n = 2^k \curvearrowright k = \log_2 n$$

Aufwandsanalyse

Rekursiongleichung

$$T(n) = T(\frac{n}{2}) + c$$

$$n = 2^k \curvearrowright k = \log_2 n$$

asymptotische Ordnung - $\mathcal{O}$

$$\mathcal{O}(log_2 n)$$

Aufwandsanalyse

sequenzielle Suche

$$\mathcal{O}(n)$$

Das heißt im Worst-Case für 1.000.000 Elemente braucht die Suche 1.000.000 Schritte.

binäre Suche

$$\mathcal{O}(\log_2 n)$$

Für etwa 1.000.000 Elemente braucht dieses Verfahren ungefähr 20 Schritte.

Karatsubas Multiplikation

Problem

Die Multiplikation von riesigen Zahlen

$$1253435647448758968152532544646376485578698799686$$

$$*$$

$$5447655345322653634575689678967986896787467458576$$

Schulbuchmethode

Einführendes Beispiel

Wieviel ist 2876 $\times$ 5839?

Wie gehts?

	2	8	7	6	$*$	5	8	3	9
		1	4	3	8	0			
$+$			2	3	0	0	8		
$+$					8	6	2	8	
$+$					2	5	8	8	4
$+$				1	1	1	1		
$=$		1	6	7	9	2	9	6	4

Schulbuchmethode

Aufwand

$$T(n) = 4 \cdot n^2 - 2 \cdot n$$

Für Zahlen der selben Stelligkeit n!

$$\mathcal{O}(n^2)$$

Vorüberlegung

Divide & Conquer

Divide

Im einfachsten Fall werden zwei Faktoren a und b mit $n = 2$ Stellen in ihre Ziffernbestandteile zerlegt:

$$a = a_1 \cdot 10 + a_0$$

$$b = b_1 \cdot 10 + b_0$$

Vorüberlegung

Divide

$$\begin{aligned} a \cdot b &= (a_1 \cdot 10 + a_0) \cdot (b_1 \cdot 10 + b_0) \\ &= (a_1 \cdot b_1) \cdot 100 + (a_0 \cdot b_1 + b_0 \cdot a_1) \cdot 10 + a_0 \cdot b_0 \end{aligned}$$

Vorüberlegung

Divide

$$\begin{aligned} a \cdot b &= (a_1 \cdot 10 + a_0) \cdot (b_1 \cdot 10 + b_0) \\ &= (a_1 \cdot b_1) \cdot 100 + (a_0 \cdot b_1 + b_0 \cdot a_1) \cdot 10 + a_0 \cdot b_0 \end{aligned}$$

Für $n > 2$

Wir verallgemeinern die oben gezeigte Formel auf folgende Form:

$$a = a_1 \cdot B^{\frac{n}{2}} + a_0$$

$$b = b_1 \cdot B^{\frac{n}{2}} + b_0$$

B ist die Basis des Zahlensystems, in unserem Beispiel 10

Vorüberlegung

Rekursion

Wie im Beispiel gezeigt, kann man dies rekursiv beschreiben.

Ist das nun ein Gewinn?

$$\underbrace{(a_1 \cdot b_1)}_{1} \cdot B^n + (\underbrace{a_0 \cdot b_1}_{2} + \underbrace{b_0 \cdot a_1}_{3}) \cdot B^{\frac{n}{2}} + \underbrace{a_0 \cdot b_0}_{4}$$

Vorüberlegung

Rekursion

Wie im Beispiel gezeigt, kann man dies rekursiv beschreiben.

Ist das nun ein Gewinn?

$$\underbrace{(a_1 \cdot b_1)}_{1} \cdot B^n + (\underbrace{a_0 \cdot b_1}_{2} + \underbrace{b_0 \cdot a_1}_{3}) \cdot B^{\frac{n}{2}} + \underbrace{a_0 \cdot b_0}_{4}$$

$$T(n) = 4 \cdot T(\frac{n}{2}) + \Theta(n)$$

Es sind nach wie vor 4 Multiplikationen zu tätigen!

$$T(n) = \mathcal{O}(n^2)$$

Karatsubas Idee

Ausgangsformel

$$(a_1 \cdot b_1) \cdot B^n + (a_0 \cdot b_1 + b_0 \cdot a_1) \cdot B^{\frac{n}{2}} + a_0 \cdot b_0$$

Karatsubas Trick

$$a \cdot b = a_1 \cdot b_1 \cdot B^n + (a_1 \cdot b_1 + a_0 \cdot b_0 - (a_0 - a_1) \cdot (b_0 - b_1)) \cdot B^{\frac{n}{2}} + a_0 \cdot b_0$$

Karatsubas Multiplikation

Karatsubas Trick

$$(a_1 \cdot b_1) \cdot B^n + ((a_1 \cdot b_1) + (a_0 \cdot b_0) - (a_0 - a_1) \cdot (b_0 - b_1)) \cdot B^{\frac{n}{2}} + a_0 \cdot b_0$$

So siehts einfacher aus:

Wir führen folgende Werte ein:

$$\begin{aligned} r &= a_1 \cdot b_1 \\ s &= (a_0 - a_1) \cdot (b_0 - b_1) \\ t &= a_0 \cdot b_0 \end{aligned}$$

Substituierte Gleichung

$$a \cdot b = r \cdot B^n + (r + t - s) \cdot B^{\frac{n}{2}} + t$$

Karatsubas Idee

Divide & Conquer

Substituierte Gleichung

$$a \cdot b = r \cdot B^n + (r + t - s) \cdot B^{\frac{n}{2}} + t$$

Karatsubas Idee

Divide & Conquer

Substituierte Gleichung

$$a \cdot b = r \cdot B^n + (r + t - s) \cdot B^{\frac{n}{2}} + t$$

Effiziensanalyse

3 Multiplikationen deren Ergebnisse mehrfach verwendet werden und deren Addition (bzw. Subtraktion)

$$T(n) = 3 \cdot T(\frac{n}{2}) + \Theta(n)$$

Asymptotische Ordnung

Schulbuch

$$T(n) = 4 \cdot T(\frac{n}{2}) + \Theta(n)$$

Entspricht unter Anwendung der Meistermethode

$$\Theta(n^{\log_2 4}) = \Theta(n^2)$$

$$\mathcal{O}(n^2)$$

Asymptotische Ordnung

Schulbuch

$$T(n) = 4 \cdot T(\frac{n}{2}) + \Theta(n)$$

Entspricht unter Anwendung der Meistermethode

$$\Theta(n^{\log_2 4}) = \Theta(n^2)$$

$$\mathcal{O}(n^2)$$

Karatsubas Methode

$$T(n) = 3 \cdot T(\frac{n}{2}) + \Theta(n)$$

Ebenso durch Andwendung der Meistermethode

$$\Theta(n^{\log_2 3}) = \Theta(n^{1,585})$$

$$\mathcal{O}(n^{1,585})$$

Vergleich der Effiziens

Schulbuch

$$\mathcal{O}(n^2)$$

Das heißt bei einer $n = 100$ stelligen Zahl brauchen wir mehr als 10.000 Grundoperationen

Vergleich der Effiziens

Schulbuch

$$\mathcal{O}(n^2)$$

Das heißt bei einer $n = 100$ stelligen Zahl brauchen wir mehr als 10.000 Grundoperationen

Karatsubas Methode

$$\mathcal{O}(n^{1,585})$$

Für $n = 100$ stellige Zahlen benötigen wir weniger als 2.000 Operationen

Matrizen Multiplizieren

Beispiel

$$C = A * B = \begin{pmatrix} 2 & 4 & 6 & 5 \\ 3 & 6 & 9 & 3 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 3 & 7 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 5 & 7 \\ 5 & 4 & 3 & 2 \\ 8 & 2 & 6 & 4 \\ 3 & 2 & 6 & 9 \end{pmatrix}$$

Matrizen Multiplizieren

Beispiel

$$C = A * B = \begin{pmatrix} 2 & 4 & 6 & 5 \\ 3 & 6 & 9 & 3 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 3 & 7 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 5 & 7 \\ 5 & 4 & 3 & 2 \\ 8 & 2 & 6 & 4 \\ 3 & 2 & 6 & 9 \end{pmatrix}$$

Ausrechnen mit der Schulmethode?

$$\begin{pmatrix} (2*1+4*5+6*8+5*3) & (2*3+4*4+6*2+5*2) & (2*5+4*3+6*6+5*6) & \dots \\ (3*1+6*5+9*8+3*3) & (3*3+6*4+9*2+3*2) & (3*5+6*3+9*6+3*6) & \dots \\ (1*1+2*5+3*8+4*3) & (1*3+2*4+3*2+4*2) & (1*5+2*3+3*6+4*6) & \dots \\ (4*1+5*5+3*8+7*3) & (4*3+5*4+3*2+7*2) & (4*5+5*3+3*6+7*6) & \dots \end{pmatrix}$$

$$\mathcal{O}(n^3)$$

Vorüberlegung

Divide & Conquer

$$C = A * B = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} * \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Vorüberlegung

Divide

$$C = A * B = \left(\begin{array}{cc|cc} 2 & 4 & 6 & 5 \\ 3 & 6 & 9 & 3 \\ \hline 1 & 2 & 3 & 4 \\ 4 & 5 & 3 & 7 \end{array}\right) * \left(\begin{array}{cc|cc} 1 & 3 & 5 & 7 \\ 5 & 4 & 3 & 2 \\ \hline 8 & 2 & 6 & 4 \\ 3 & 2 & 6 & 9 \end{array}\right)$$

Mit Zahlen aus dem Beispiel

$$A_{1,1} = \begin{pmatrix} 2 & 4 \\ 3 & 6 \end{pmatrix} A_{1,2} = \begin{pmatrix} 6 & 5 \\ 9 & 3 \end{pmatrix} A_{2,1} = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix} A_{2,2} = \begin{pmatrix} 3 & 4 \\ 3 & 7 \end{pmatrix}$$

$$B_{1,1} = \begin{pmatrix} 1 & 3 \\ 5 & 4 \end{pmatrix} B_{1,2} = \begin{pmatrix} 5 & 7 \\ 3 & 2 \end{pmatrix} B_{2,1} = \begin{pmatrix} 8 & 2 \\ 3 & 2 \end{pmatrix} B_{2,2} = \begin{pmatrix} 6 & 4 \\ 6 & 9 \end{pmatrix}$$

Lösungsmatrix

Conquer

$$
\begin{aligned}
C_{1,2} &= A_{1,1} * B_{1,1} + A_{1,2} * B_{2,1} \\
C_{1,2} &= A_{1,1} * B_{1,2} + A_{1,2} * B_{2,2} \\
C_{2,1} &= A_{2,1} * B_{1,1} + A_{2,2} * B_{2,1} \\
C_{2,2} &= A_{2,1} * B_{1,2} + A_{2,2} * B_{2,2}
\end{aligned}
$$

Lösungsmatrix

Conquer

$$
\begin{aligned}
C_{1,2} &= A_{1,1} * B_{1,1} + A_{1,2} * B_{2,1} \\
C_{1,2} &= A_{1,1} * B_{1,2} + A_{1,2} * B_{2,2} \\
C_{2,1} &= A_{2,1} * B_{1,1} + A_{2,2} * B_{2,1} \\
C_{2,2} &= A_{2,1} * B_{1,2} + A_{2,2} * B_{2,2}
\end{aligned}
$$

Aufwand

$$T(n) = 8 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$$

Lösungsmatrix

Conquer

$$
\begin{aligned}
C_{1,2} &= A_{1,1} * B_{1,1} + A_{1,2} * B_{2,1} \\
C_{1,2} &= A_{1,1} * B_{1,2} + A_{1,2} * B_{2,2} \\
C_{2,1} &= A_{2,1} * B_{1,1} + A_{2,2} * B_{2,1} \\
C_{2,2} &= A_{2,1} * B_{1,2} + A_{2,2} * B_{2,2}
\end{aligned}
$$

Aufwand

$$T(n) = 8 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$$

$$\mathcal{O}(n^3)$$

Strassen's Idee

Hilfsmatrizen

$$
\begin{aligned}
M_1 &= (A_{1,2} - A_{2,2}) * (B_{2,1} + B_{2,2}) \\
M_2 &= (A_{1,1} + A_{2,2}) * (B_{1,1} + B_{2,2}) \\
M_3 &= (A_{1,1} - A_{2,1}) * (B_{1,1} + B_{1,2}) \\
M_4 &= (A_{1,1} + A_{1,2}) * B_{2,2} \\
M_5 &= A_{1,1} * (B_{1,2} + B_{2,2}) \\
M_6 &= A_{2,2} * (B_{2,1} + B_{1,2}) \\
M_7 &= (A_{2,1} + A_{2,2}) * B_{1,1}
\end{aligned}
$$

Conquer

$$
\begin{aligned}
C_{1,1} &= M_1 + M_2 - M_4 + M_6 \\
C_{1,2} &= M_4 + M_5 \\
C_{2,1} &= M_6 + M_7 \\
C_{2,2} &= M_2 - M_3 + M_5 - M_7
\end{aligned}
$$

Effiziensanalyse

Rekursionsgleichung

Effiziensanalyse

Rekursionsgleichung

$$T(n) = 7 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$$

Effiziensanalyse

Rekursionsgleichung

$$T(n) = 7 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$$

Asymptotische Ordnung

$$T(n) = \mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2,81})$$

Wir sind am Ende

Danke für eure Aufmerksamkeit, viel Spass bei den Übungen!