Algorithmen & Datenstrukturen

Herbst 2018

Vorlesung 14

Interessante Divide-and-Conquer Algorithmen

Multiplikation ganzer Zahlen

Schulmethode: $ab \cdot cd$

$62 \cdot 37$

14	bd
42	ad
6	bc
18	ac
2294	

4 Multiplikationen von Ziffern:

$(10a+b)(10c+d) =$
$100ac + 10bc + 10ad + bd$

Anzahl Ziffernmultiplikationen: $\Theta(n^2)$
(beide Zahlen n Ziffern)

Karatsuba/Ofman (1962):

$ab \quad cd$

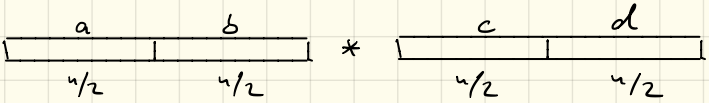
$62 \cdot 37$

14	bd
14	$=$
16	$(a-b)(d-c)$
18	ac
18	ac
2294	

3 Multiplikationen!

$(10a+b)(10c+d) =$
$100ac + 10ac + 10(a-b)(d-c)$
$+ 10bd + bd$

Lässt sich auf beliebige Zahlen verallgemeinern:



$a \cdot 10^{n/2} + b$ $\qquad$ $c \cdot 10^{n/2} + d$

$$(a\,10^{n/2} + b)(c\,10^{n/2} + d) = ac\,10^{n} + ac\,10^{n/2} + bd\,10^{n/2} + bd + 10^{n/2}(a-b)(d-c)$$

Ergibt rekursiven Algorithmus: Anzahl Ziffernops:

$T(n) = 3\,T(n/2) + x \cdot n$ ← Additionen, x Konstant

$= 3^2\,T(n/4) + \frac{3}{2}xn + xn$

$= 3^3\,T(n/8) + \left(\frac{3}{2}\right)^2 xn + \frac{3}{2}xn + xn$

. . . .

$= 3^{\log_2 n}\,T(1) + \Theta(n)$ ⊗ $\log_2 n = \frac{\log_3 n}{\log_3 2} = \log_3 n \cdot \log_2 3$

$= \Theta(n^{\log_2 3}) \approx \Theta(n^{1,584\ldots})$

Geht auch mit Polynommultiplikation:

$$p(x) = \sum_{i=0}^{n-1} a_i x^i = p_1(x) \cdot x^{n/2} + p_2(x)$$

$$q(x) = \sum_{i=0}^{n-1} b_i x^i = q_1(x) \cdot x^{n/2} + q_2(x)$$

Beide Probleme gehen noch schneller:

$O(n \log n \log\log n)$ bzw. $O(n \log n)$

mit Fouriertransformation

Matrixmultiplikation: A, B $n \times n$ Matrizen

$$C = AB \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \Theta(n^3)$$

Rekursiv: Blöcken

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

alle Blöcke sind $n/2 \times n/2$

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

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

etc.

8 Mults von $n/2 \times n/2$ Matrizen

$T(n) = 8\, T(\frac{n}{2}) + a \cdot n^2$ ← Matrix additionen

$\Rightarrow T(n) = \Theta(n^3)$ (keine Verbesserung)

Strassen (1969):

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22}) B_{11}$$
$$M_3 = A_{11} (B_{12} - B_{22})$$
$$M_4 = A_{22} (B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12}) B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

7 Multiplikationen von $n/2 \times n/2$ Matrizen!

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

$T(n) = 7\, T(\frac{n}{2}) + a \cdot n^2$

$\Rightarrow T(n) = \Theta(n^{\log_2 7})$

$\simeq \Theta(n^{2.8\ldots})$

Anzahl Ops besser als $n = 654$

Aber hat ein paar praktische Nachteile:
- numerische Stabilität
- schlechte Struktur für Prozessoren

Es geht besser (aber Riesenkonstanten):

$\Theta(n^{2.37\ldots})$ untere Schranke: $\Omega(n^2 \log n)$

Auswahlproblem (selection problem)

gegeben: Array $A[1], \ldots, A[n]$

gesucht: i-kleinstes Element Spezialfall: Median $= \lceil n/2 \rceil$-kleinstes

Idee 1: i mal Minimum entfernen $\Theta(in)$

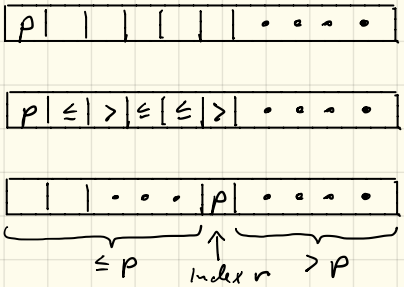
Idee 2: Sortieren, i-tes ausgeben $\Theta(n \log n)$

Idee 3: Pivotieren, rekursiv

$O(1)$ 1.) wähle Pivot (z.B. erstes)

$O(n)$ 2.) zähle Elemente $\leq p$ ($= r$ viele)

$O(n)$ 3.) teile A



4.) $i = r$: return p
$i < r$: suche i-tes links
$i > r$: suche $(i-r)$-tes rechts

Heißt "quickselect": wie quicksort aber Rekursion nur auf einer Hälfte

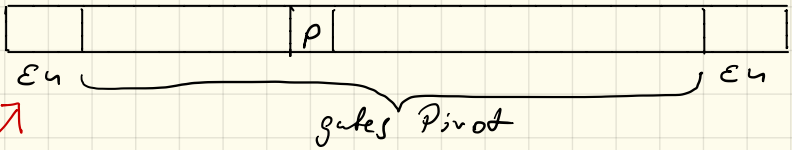
Laufzeit: worst case $T(n) = T(n-1) + \Theta(n)$

$\Rightarrow T(n) = \Theta(n^2)$

aber average case $T(n) = \Theta(n)$ (ohne Beweis)

Was ist ein gutes Pivotelement?
Antwort: reduziert Länge des Arrays in jeder Iteration mindestens um Faktor $q < 1$

Bild: Nach Aufteilen:



$0 < \varepsilon < \frac{1}{2}$ fix!

Dann dann: $T(n) \leq T(qn) + c \cdot n$, c Konstant
$(q = 1 - \varepsilon)$

teleskopieren: $\leq T(q^2 n) + cqn + cn$

$\cdots$ ($\log_q(\frac{1}{n})$ Schritte)

$\leq T(1) + n \cdot c \cdot$ geometrische Reihe in q

$= \Theta(n)$

Ein gutes Pivot verkleinert das Array in jedem Schritt mindestens um einen fixen Anteil ε

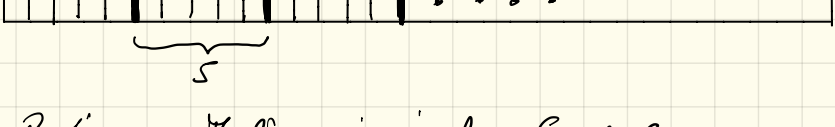
Wie schaffen wir das deterministisch?
(d.h. immer, nicht nur im average case)

Algorithmus "Median der Medians"
(Blum, Floyd, Pratt, Rivest, Tarjan 1973)

Idee: clevere Pivotwahl (neuer Schritt 1.)

1.) wähle Pivot wie folgt:

O(1) a.) Betrachte Array in Fünfergruppen:



O(n) b.) Bestimme Median in jeder Gruppe

O(n) c.) A' = Array der Gruppenmediane (Länge $\lceil \frac{n}{5} \rceil$)

d.) Bestimme Median von A': Median = Pivot p

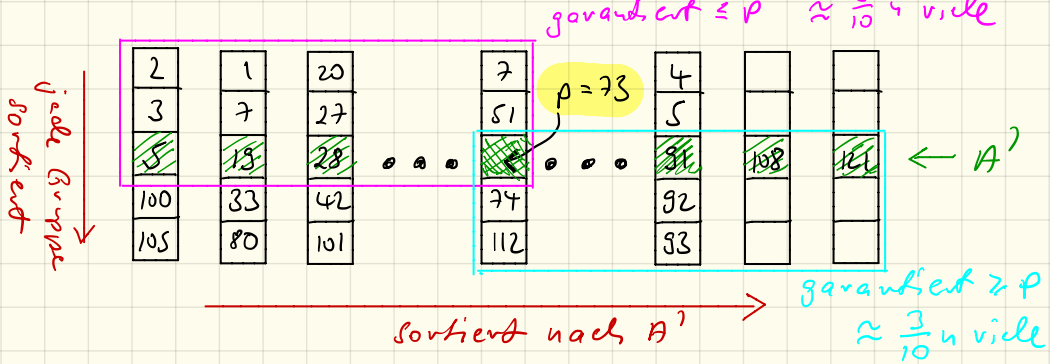
wie? Rekursion: $\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \rceil$ - tes Element von A'

2.) - 4.) wie vorher (= Aufteilen, Rekursion links oder rechts)

Also zwei rekursive Aufrufe!
p ist Median der Gruppenmediane

Wie gut ist p? (= wieviele Elemente in A sind garantiert kleiner und größer?
= was ist ε von voriger Seite?)

Bild: wir malen A so: (A ist nicht so sortiert)



Laufzeit: $T(n) \leq T(\frac{n}{5}) + T(\frac{7}{10}n) + cn$ (Skizze)

$n - \frac{3}{10}n$

Zeige $T(n) \leq an$ per Induktion, a konstant

Anfang: $T(n) \leq a \cdot n \quad \forall\, n \leq n_0$ (z.B. $n_0 = 100$)
wähle a so daß das gilt und $a > 10c$

Schritt: Annahme $T(k) \leq ak \quad \forall\, k < n$

$$\Rightarrow T(n) \leq a\frac{n}{5} + a\frac{7}{10}n + cn$$

$$= a \cdot \frac{9}{10}n + cn$$

$$\leq an \qquad \square$$

Insbesondere: Medianen geht in $O(n)$

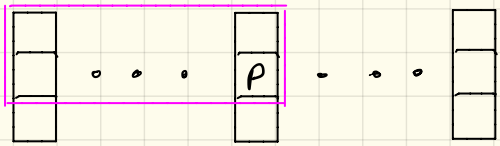
In der Praxis: zufälliges Pivot funktioniert gut

Warum Fünfergruppen?

$$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + cn$$

Summe $= \frac{9}{10}n < n$ Wichtig im Beweis!

Dreiergruppen: $\approx \frac{1}{3}n$



Rekurrenz $T(n) \le T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$

Summe $= n$

Viergruppen: $T(n) \le T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + cn$

Ab $g \ge 5$ klappt es, g konstant!