Divide-and-Conquer / Teile und Herrsche

Wichtige Methode für den Entwurf von Algorithmen

Allgemeiner Ansatz:

0.	Wenn Problem klein, löse es direkt	
1.	Zerlege Problem geeignet in Teilprobleme	*divide*
2.	Löse jedes Teilproblem rekursiv	*conquer*
3.	Vereinige die Teillösungen zu einer Gesamtlösung	*„marry“*

Divide-and-Conquer / Teile und Herrsche

Wichtige Methode für den Entwurf von Algorithmen

Allgemeiner Ansatz:

0. Wenn Problem klein, löse es direkt
1. Zerlege Problem geeignet in Teilprobleme — *divide*
2. Löse jedes Teilproblem rekursiv — *conquer*
3. Vereinige die Teillösungen zu einer Gesamtlösung — *„marry“*

Beispiel: MergeSort(L : Liste von Zahlen)

0. if $|L| \leq 1$ then return L
1. Teile L in Teillisten L_1 und L_2 möglichst gleicher Größe (~ n/2) — *divide*
2. E_1 = MergeSort(L_1) E_2 = MergeSort(L_2) — *conquer*
3. E = Merge(E_1 , E_2) — *„marry“*
   return E

Divide-and-Conquer Laufzeitanalyse

$T(n)$ worst case Laufzeit von MergeSort wenn $|L| = n$

Beispiel: MergeSort(L : Liste von Zahlen)

0.	if $\|L\| \leq 1$ then return L	$O(1)$
1.	Teile L in Teillisten L_1 und L_2 möglichst gleicher Größe (~ $n/2$)	$O(n)$
2.	E_1 = MergeSort(L_1) E_2 = MergeSort(L_2)	$2 \cdot T(n/2)$
3.	E = Merge(E_1 , E_2) return E	$O(n)$

Divide-and-Conquer Recurrence

$T(n) \leq d$ wenn $n \leq 1$

$T(n) \leq \beta n + 2 \cdot T(n/2)$ sonst

Lösung: $T(n) = O(n \cdot \log n)$

Divide-and-Conquer Recurrences

("Mastertheorem")

$a>0, b>1, c>0$

$T(n) \leq d$ wenn $n \leq 1$

$T(n) \leq a \cdot T(n/b) + \beta \cdot n^c$ sonst

Lösung: $T(n) = O(n^c)$ wenn $a < b^c$

$T(n) = O(n^c \cdot \log n)$ wenn $a = b^c$

$T(n) = O(n^\lambda)$ mit $\lambda = \log_b a$ wenn $a > b^c$

Divide-and-Conquer Recurrences 2

("Mastertheorem")

$a>0, b>1, c>0, k \geq 0$

$T(n) \leq d$ wenn $n \leq 1$

$T(n) \leq a \cdot T(n/b) + \beta \cdot n^c \cdot \log^k n$ sonst

Lösung: $T(n) = O(n^c \cdot \log^k n)$ wenn $a < b^c$

$T(n) = O(n^c \cdot \log^{k+1} n)$ wenn $a = b^c$

$T(n) = O(n^\lambda)$ mit $\lambda = \log_b a$ wenn $a > b^c$

Beispiel: Polynommultiplikation

Polynome vom Grad n:

$p(x) = p_0 + p_1x^1 + p_2x^2 + \cdots + p_nx^n$ dargestellt durch Feld $P[0..n] = \langle p_0,p_1,p_2,\cdots,p_n \rangle$

$q(x) = q_0 + q_1x^1 + q_2x^2 + \cdots + q_nx^n$ dargestellt durch Feld $Q[0..n] = \langle q_0,q_1,q_2,\cdots,q_n \rangle$

Summenpolynom $(p+q)$ und Differenzpolynom $(p-q)$ lassen sich offensichtlih in $O(n)$ Zeit berechnen (addiere, bzw. subtrahiere die entsprechenden Koeffizienten).

Produktpolynom $r(x) = p(x)\cdot q(x)$ dargestellt durch Feld $R[0..2n] = \langle r_0,r_1,r_2,\cdots,r_{2n} \rangle$

Lässt sich naiv in $O(n^2)$ Zeit berechnen:

```
for i = 0 to 2n do R[i] = 0
for i = 0 to n do
  for j = 0 to n do
        R[i+j] += P[i]·Q[j]
```

Schneller mit Divide-and-Conquer?

Beispiel: Polynommultiplikation via Divide-and-Conquer, einfältig

$$p(x) = p_0 + p_1x^1 + p_2x^2 + \cdots + p_nx^n = p_0 + \cdots + p_{n/2-1}x^{n/2-1} + x^{n/2}[\, p_{n/2} + \cdots + p_nx^{n/2} \,]$$

$$= p'(x) + x^{n/2}\, p''(x) \qquad \text{grad}(p') = \text{grad}(p'') = n/2$$

Analog

$$q(x) = q_0 + q_1x^1 + q_2x^2 + \cdots + q_nx^n = q'(x) + x^{n/2}\, q''(x) \qquad \text{grad}(q') = \text{grad}(q'') = n/2$$

Produktpolynom

$$r(x) = p(x)\cdot q(x) = p'(x)\cdot q'(x) + x^{n/2}[\, p'(x)\cdot q''(x) + p''(x)\cdot q'(x) \,] + x^n[p''(x)\cdot q''(x)]$$

Eine Multiplikation von Grad-n-Polynomen wird zurückgeführt auf 4 Multiplikationen von Grad-(n/2) Polynomen plus ein paar Additionen und „Shifts“ (Multiplikation mit x^k)

Laufzeit: $T(n) = O(n) + 4\cdot T(n/2)$

Mastertheorem $\Rightarrow$ $T(n) = O(n^2)$, keine Verbesserung!

(a=4 , b=2 , c=1)

Beispiel: Polynommultiplikation via Divide-and-Conquer, schlau (Karatsuba-Ofman)

$p(x) = p_0 + p_1x^1 + p_2x^2 + \cdots + p_nx^n = p'(x) + x^{n/2}\, p''(x)$ $\quad \text{grad}(p') = \text{grad}(p'') = n/2$

$q(x) = q_0 + q_1x^1 + q_2x^2 + \cdots + q_nx^n = q'(x) + x^{n/2}\, q''(x)$ $\quad \text{grad}(q') = \text{grad}(q'') = n/2$

Berechne $U = (p'+p'') \cdot (q'+q'')$ $\quad V = p' \cdot q'$ $\quad W = p'' \cdot q''$

$r(x) = p(x) \cdot q(x) = p'(x) \cdot q'(x) + x^{n/2}[\, p'(x) \cdot q''(x) + p''(x) \cdot q'(x)\,] + x^n [p''(x) \cdot q''(x)]$

$= \quad V \quad + x^{n/2}[\, U - V - W\,] + x^n [\, W\,]$

Eine Multiplikation von Grad-n-Polynomen wird zurückgeführt auf **3** Multiplikationen von Grad-(n/2) Polynomen plus ein paar Additionen und „Shifts“ (Multiplikation mit x^k)

Laufzeit: $T(n) = O(n) + 3 \cdot T(n/2)$

Mastertheorem $\Rightarrow$ $T(n) = O(n^{1.59})$, klare Verbesserung!

(a=3 , b=2 , c=1)

Weitere Verbesserungen sind möglich.

Multiplikation und Division von Polynomen, Ausblick

Satz:

$A(x)$, $B(x)$ Polynome vom Grad höchstens n

Produkt $P(x) = A(x) \cdot B(x)$

Quotient $Q(x) = A(x)$ div $B(x)$

Rest $R(x) = A(x)$ mod $B(x)$

können in Zeit $O(n \cdot \log n)$ berechnet werden

$Q(x)$ und $R(x)$ definiert durch eindeutige Darstellung von $A(x)$ als

$$A(x) = B(x) \cdot Q(x) + R(x)$$

mit $\text{grad}(R) < \text{grad}(B)$

Mit Hilfe der schnellen Diskreten Fourier Transformation (eine Divide-and-Conquer Methode)

sowie iterativer Approximation

Simultane Auswertung eines Polynoms für viele Argumente

Geg.: Polynom $f(x)$ vom Grad n sowie n Zahlen $a_1,\cdots,a_n$

Ges.: $f(a_1),\cdots,f(a_n)$

Naive Lösung benötigt $O(n^2)$ Zeit.

Lemma:
Sei $R(x) = A(x) \bmod B(x)$ und a eine Nullstelle von B, also $B(a)=0$.

Dann gilt $A(a) = R(a)$.

Folgt direkt aus Definition von
$Q(x) = A(x) \text{ div } B(x)$ und
$R(x) = A(x) \bmod B(x)$

definiert durch eindeutige Darstellung von $A(x)$ als

$$A(x) = B(x)\cdot Q(x) + R(x)$$

mit grad(R)<grad(B)

Divide-and-Conquer Algorithmus zur Simultanauswertung

Teile $\{a_1,\cdots,a_n\}$ in $A1 = \{a_1,\cdots,a_{n/2}\}$ und $A2 = \{a_{n/2+1},\cdots,a_n\}$

Berechne $B_1(x) = \Pi_{a\in A1}(x-a)$ $B_2(x) = \Pi_{a\in A2}(x-a)$ $O(n\cdot\log^2 n)$

Berechne $f1 = f \bmod B1$ und $f2 = f \bmod B2$ $O(n\cdot\log n)$

Löse rekursiv die Simultanauswertung von $f1$ für $A1$ $T(n/2)$
Löse rekursiv die Simultanauswertung von $f2$ für $A2$ $T(n/2)$

Beachte: $f1$ und $f2$ haben Grad < $n/2$

Divide-and-Conquer Algorithmus zur Simultanauswertung

Teile $\{a_1,\cdots,a_n\}$ in $A1 = \{a_1,\cdots,a_{n/2}\}$ und $A2 = \{a_{n/2+1},\cdots,a_n\}$

Berechne $B_1(x) = \Pi_{a\in A1}(x-a)$ $B_2(x) = \Pi_{a\in A2}(x-a)$ $O(n\cdot\log^2 n)$

Berechne $f1 = f \bmod B1$ und $f2 = f \bmod B2$ $O(n\cdot\log n)$

Löse rekursiv die Simultanauswertung von $f1$ für $A1$ $T(n/2)$
Löse rekursiv die Simultanauswertung von $f2$ für $A2$ $T(n/2)$

Beachte: $f1$ und $f2$ haben Grad $< n/2$

Laufzeit: $T(n) = 2T(n/2) + O(n\cdot\log^2 n)$

Aus dem Mastertheorem ergibt sich $T(n) = O(n\cdot\log^3 n)$

Verbesserung der Laufzeit auf $O(n\cdot\log^2 n)$ ist möglich.