

Algorithmen und Datenstrukturen

Wintersemester 2023/24

4. Vorlesung

Laufzeitanalyse – Beispiele

Analyse von Aktienkursen



Wichtig:
Es genügt *nicht*,
nur Minimum
und Maximum
zu suchen!

Problem. Gegeben: Folge $A[1..n]$ von Aktienkursen in Euro.

Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$,
so dass $A[j] - A[i]$ maximal.

Verkaufskurs

Profit pro Aktie

Einkaufskurs

Analyse von Aktienkursen

Problem: Gegeben: Folge $A[1..n]$ von ganzen Zahlen
 Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$,
 so dass $A[j] - A[i]$ maximal. } MAX-DIFF

Lösung: per „roher Gewalt“

Übung:

Schreiben Sie
Pseudocode!

- für alle erlaubten Paare (i, j) berechne $A[j] - A[i]$
- gib Maximum zurück

Laufzeit \approx Anzahl der berechneten Differenzen
 $=$ Anzahl erlaubter Paare
 $= (n - 1) + (n - 2) + \dots + 2 + 1 = \frac{n^2 - n}{2}$
 $\in \Theta(n^2)$

Ein ähnliches Problem

Problem: Gegeben: Folge $A[1..n]$ von **ganzen Zahlen**
 Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$,
 so dass $\sum_{k=i}^j A[k]$ maximal. } **MAX-SUM**

7	4	-9	1	-3	3	1	12	0	-2	0	4	2	-8	2
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$\Rightarrow (6, 13)$
oder $(1, 13)$

Lösung: per „roher Gewalt“

Übung:
Schreiben Sie
Pseudocode!

– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
 – gib Maximum zurück

Laufzeit \approx Anzahl der Additionen

Obere Schranke dafür: $O(n^2) \cdot O(n) = O(n^3)$

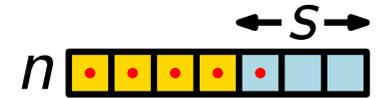
Untere Schranke (Anz. Paare) $= \Omega(n^2)$

Wo ist die Wahrheit?

Genauere Analyse

Laufzeit \approx Anzahl der Additionen des Rohe-Gewalt-Algos:

- für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
- gib Maximum zurück



Beob.

- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \end{aligned}$$

$$= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \in \Omega(n^3)$$

(falls $4|n$)

$\frac{n}{2} + 1$ Terme der Größe mindestens $\frac{n}{4} \cdot \frac{n}{4}$

Übung:

Berechnen Sie diese Summe *genau* und beweisen Sie Ihr Ergebnis per Induktion!

\Rightarrow Der Rohe-Gewalt-Alg. läuft in $O(n^3) \cap \Omega(n^3) = \Theta(n^3)$ Zeit.

Can we do better?

Wie berechnen?

Add. = $an^3 + bn^2 + cn + d$
Wertetabelle für $n = 1, 2, 3, 4$.
LGS aufstellen + lösen!

Eine schnellere Lösung

Problem: Gegeben: Folge $A[1..n]$ von ganzen Zahlen
 Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$,
 so dass $S_{ij} = \sum_{k=i}^j A[k]$ maximal.

Idee:

Für $i = 1, \dots, n$

berechne $S_{ii}, S_{i,i+1}, S_{i,i+2}, S_{i,i+3}, \dots, S_{i,n}$
 $=$ $A[i] + A[i+1] + A[i+2] + A[i+3] + \dots + A[n]$
Wie?

$n - i$ Additionen

Insgesamt

$$\sum_{i=1}^n$$

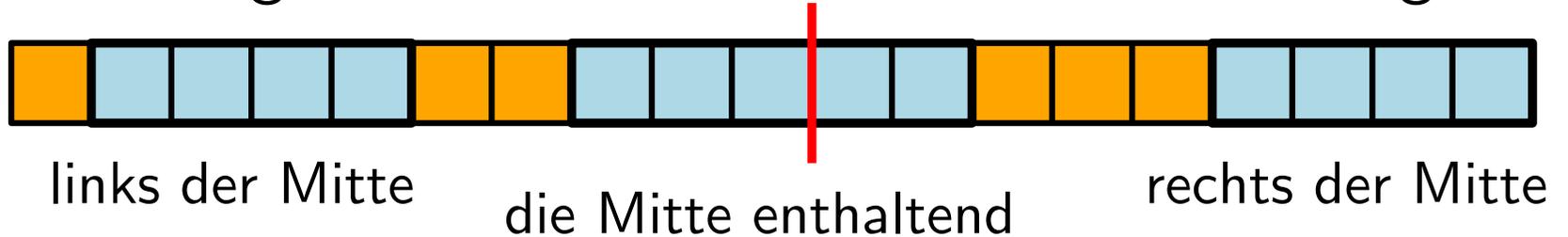
$n - i$

$$= \sum_{j=0}^{n-1} j = \frac{n \cdot (n-1)}{2} \in \Theta(n^2)$$

Additionen

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte enthalten

Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$

Einsicht: Wenn die *maximale* Teilsumme die Mitte enthält,
dann muss ihr linker Teil (bis zur Mitte) maximal sein
und dann muss ihr rechter Teil (ab der Mitte) maximal sein.
⇒ Können li. u. re. Teil *unabhängig* voneinander berechnen!

Teile & Herrsche

```

MaxTeilfeld(int[] A, int beginn = 1, int ende = A.length)
  if beginn == ende then
    | return (beginn, ende, A[beginn]) herrsche (in kleinen Teilinstanzen)
  else
    | mitte = ⌊(beginn + ende)/2⌋ teile herrsche
    | (L-beginn, L-ende, L-summe) = MaxTeilfeld(A, beginn, mitte)
    | (R-beginn, R-ende, R-summe) = MaxTeilfeld(A, mitte + 1, ende)
    | (M-beginn, M-ende, M-summe) =
      | MaxMittleresTeilfeld(A, beginn, mitte, ende) ← kombiniere
    | return (Tripel mit größter Summe)
  
```

Laufzeit: $T_{MT}(1) = \Theta(1)$

für $n > 1$: $T_{MT}(n) = T_{MT}(\lfloor n/2 \rfloor) + T_{MT}(\lceil n/2 \rceil) + T_{MMT}(n)$
 $\approx 2 \cdot T_{MT}(n/2) + T_{MMT}(n)$

$T_{MMT}(n) = ?$

Kombiniere

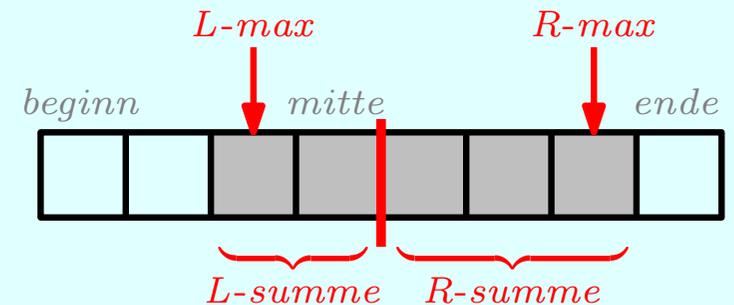
MaxMittleresTeilfeld(int[] A, int *beginn*, int *mitte*, int *ende*)

$L\text{-summe} = -\infty$

$summe = 0$

for $i = mitte$ **downto** $beginn$ **do**

Vervollständigen Sie
den Algorithmus!



$R\text{-summe} = -\infty$

$summe = 0$

for $i = mitte + 1$ **to** $ende$ **do**

└ // analog zu oben

return ($L\text{-max}$, $R\text{-max}$, $L\text{-summe} + R\text{-summe}$)

Kombiniere

MaxMittleresTeilfeld(int[] A, int *beginn*, int *mitte*, int *ende*)

$L\text{-summe} = -\infty$

$summe = 0$

for $i = mitte$ **downto** $beginn$ **do**

$summe = summe \oplus A[i]$

if $summe > L\text{-summe}$ **then**

$L\text{-summe} = summe$

$L\text{-max} = i$

$R\text{-summe} = -\infty$

$summe = 0$

for $i = mitte + 1$ **to** $ende$ **do**

 // analog zu oben \oplus

return ($L\text{-max}, R\text{-max}, L\text{-summe} + R\text{-summe}$)

Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i, mitte}$ und

$L\text{-summe} =$

$\max_{i \leq k \leq mitte} S_{k, mitte}$

Laufzeit? ✓

$:=_{\text{hier}}$ Anz. Additionen

$mitte - beginn + 1$

$ende - mitte$

$ende - beginn + 1$

$= n$

Putting Things Together

Laufzeit von MaxTeilfeld:

$$T_{\text{MT}}(1) = \Theta(1)$$

für $n > 1$: $T_{\text{MT}}(n) \approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{MMT}}(n)$

$$= 2 \cdot T_{\text{MT}}(n/2) + n$$

$$= V_{\text{MS}}(n) = O(n \log_2 n) \quad \text{[für } n = \text{Zweierpotenz}]$$

Warum die Einschränkung
wegfällt, sehen wir noch...

Denn für $a, b \geq 2$ gilt:
 $\Theta(\log_a n) = \Theta(\log_b n)$.



Denkaufgaben:

- Lösen Sie MAXSUM in $O(n)$ – also in linearer – Zeit!
- Was hat MAXSUM mit MAXDIFF (vom Anfang der VL) zu tun?

Und wenn...? $T(n) = 2 \cdot T(n/2) + 4n$ (und $T(1) = \Theta(1)$)

Gilt dann auch $T(n) = O(n \log n)$?

Übersicht

- Anzahl der Additionen ^{*}
- geschätzte Rechenzeit ^{**}

Algorithmus	Laufzeit	$n = 1\,000$		$n = 1\,000\,000$	
Rohe Gewalt	$O(n^3)$	10^9	1 s	10^{18}	31,7 y
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}	1000 s 17 m
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	3 μ s	$6 \cdot 10^6$	6 ms
Linearer Scan	$O(n)$	10^3	1 μ s	10^6	1 ms

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

^{*}) Wir setzen die Konstante c in der $O()$ -Notation auf 1.

^{**}) für einen Kern (CPU) à 1 GHz, d.h. 10^9 Add./s