Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität

Institut für Informatik

Algorithmen und Datenstrukturen

Wintersemester 2020/21

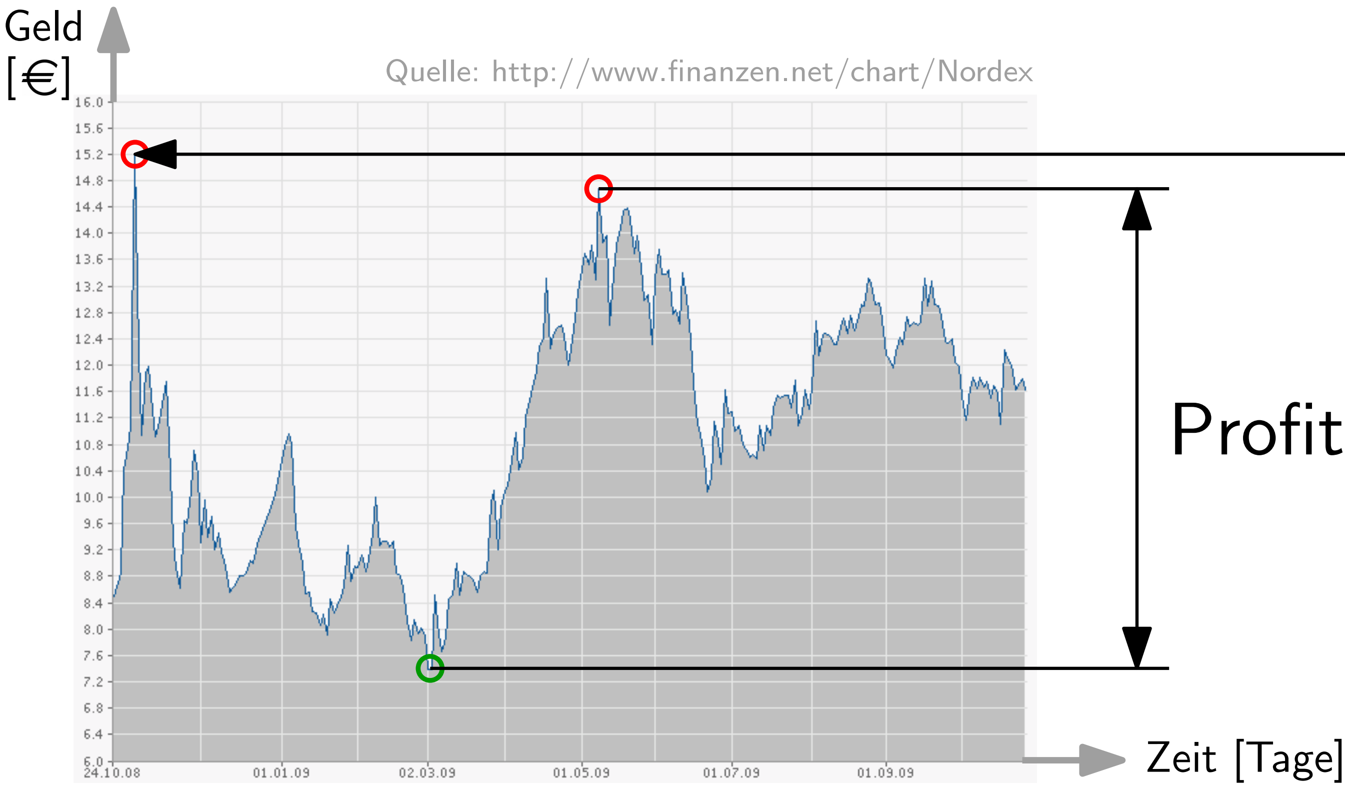
4. Vorlesung

Laufzeitanalyse – Beispiele

Prof. Dr. Alexander Wolff

Lehrstuhl für Informatik I

Analyse von Aktienkursen



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

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

Gesucht: Paar (i, j) mit $1 \le i < j \le n$,

so dass $\underbrace{\overset{\text{Verkaufskurs}}{A[j]} - \overset{\text{Einkaufskurs}}{A[i]}}_{\text{Profit pro Aktie}}$ maximal.

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.

$\Big\}$ 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

$$= (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n^2 - n}{2}$$

$$\in \Theta(n^2)$$

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!

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

Laufzeit $\approx$ Anzahl der Additionen
Obere Schranke dafür: $\boxed{O(n^2)} \cdot \boxed{O(n)} = O(n^3)$
Untere Schranke (Anz. Paare) $= \Omega(n^2)$

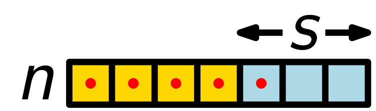
Wo ist die Wahrheit?

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.

$$\Rightarrow \text{Anz. Add.} = \sum_{s=1}^{n} (n-s+1)\cdot(s-1)$$

$$= n\cdot 0 + (n-1)\cdot 1 + (n-2)\cdot 2 + \ldots + 2\cdot(n-2) + 1\cdot(n-1)$$

$$= \cdots + \underbrace{\frac{3n}{4}\cdot\frac{n}{4} + \cdots + \frac{n}{2}\cdot\frac{n}{2} + \cdots + \frac{n}{4}\cdot\frac{3n}{4}}_{\frac{n}{2}+1 \text{ Terme der Größe mindestens } \frac{n}{4}\cdot\frac{n}{4}} + \cdots \quad \in \Omega(n^3)$$

(falls $4\,|\,n$)

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 $1 \le i \le j \le n$,
so dass $S_{ij} = \sum_{k=i}^{j} A[k]$ maximal.

Idee: Für $\boxed{i = 1, \dots, n}$

berechne $S_{ii},\ S_{i,i+1},\ S_{i,i+2},\ S_{i,i+3},\ \dots,\ S_{i,n}$

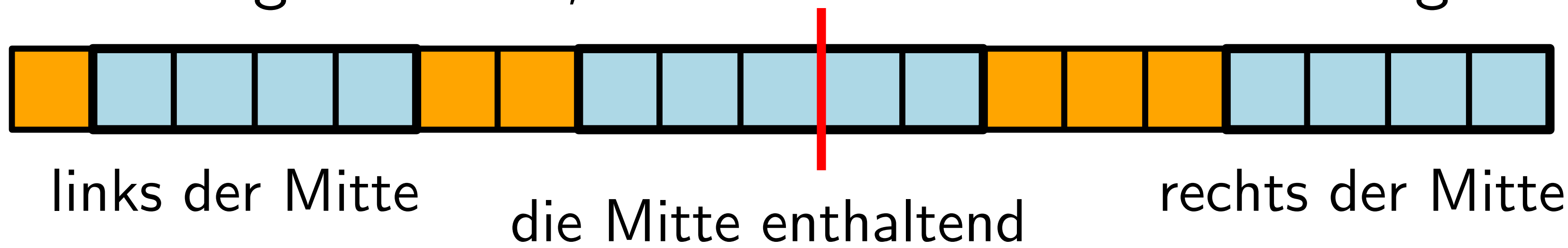
$= \quad + \quad + \quad + \quad \cdots \quad +$

Wie? $A[i] \quad \underbrace{A[i+1] \quad A[i+2] \quad A[i+3] \quad \cdots \quad A[n]}_{\boxed{n-i}\ \text{Additionen}}$

Insgesamt $\boxed{\sum_{i=1}^{n}} \boxed{n-i} = \sum_{j=n-1}^{0} j = \sum_{j=1}^{n-1} j \in \Theta(n^2)$ Add.

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.
$\Rightarrow$ 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
  $mitte = \lfloor (beginn + ende)/2 \rfloor$ *teile*
  $(L\text{-}beginn, L\text{-}ende, L\text{-}summe) = \text{MaxTeilfeld}(A, beginn, mitte)$ *herrsche*
  $(R\text{-}beginn, R\text{-}ende, R\text{-}summe) = \text{MaxTeilfeld}(A, mitte + 1, ende)$
  $(M\text{-}beginn, M\text{-}ende, M\text{-}summe) =$
    MaxMittleresTeilfeld$(A, beginn, mitte, ende)$
  return (Tripel mit größter Summe) *kombiniere*

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

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

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

$$T_{\text{MMT}}(n) = ?$$

Kombiniere

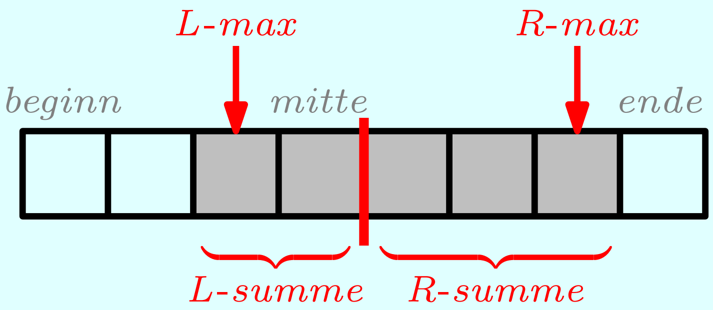
MaxMittleresTeilfeld(int[] A, int $beginn$, int $mitte$, int $ende$)

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

$summe = 0$

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-summe = −∞
summe = 0
for i = mitte downto beginn do
    summe = summe ⊕ A[i]
    if summe > L-summe then
        L-summe = summe
        L-max = i

R-summe = −∞
summe = 0
for i = mitte + 1 to ende do
    // analog zu oben ⊕

return (L-max, R-max, L-summe + R-summe)
```

Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

$L\text{-}summe = \max_{i \le k \le mitte} S_{k,mitte}$

Laufzeit? ✓

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

$mitte - beginn + 1$

$ende - mitte$

$ende - beginn + 1$

$= n$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

$$\begin{aligned} T_{\mathrm{MT}}(n) &\approx 2 \cdot T_{\mathrm{MT}}(n/2) + T_{\mathrm{MMT}}(n) \\ &= 2 \cdot T_{\mathrm{MT}}(n/2) + n \\ &= V_{\mathrm{MS}}(n) \;=\; O(n \log_{\cancel{2}} n) \end{aligned}$$

~~[für n = 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	$O(n\ \log_{10} n)$	$3{\cdot}10^3$	$3\ \mu s$	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	$1\ \mu s$	10^6	1 ms

*) Wir setzen die Konstante c in der $O()$-Notation auf 1.
**) für einen Kern (CPU) à 1 GHz, d.h. 10^9 Add./s