

Algorithmen und Datenstrukturen

Wintersemester 2022/23

4. Vorlesung

Laufzeitanalyse – Beispiele

Analyse von Aktienkursen

Quelle: <http://www.finanzen.net/chart/Nordex>



Analyse von Aktienkursen



Analyse von Aktienkursen



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

Analyse von Aktienkursen



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**.

Analyse von Aktienkursen



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

Analyse von Aktienkursen



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

Einkaufskurs

Analyse von Aktienkursen



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 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 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 Aktienkursen in Euro.

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

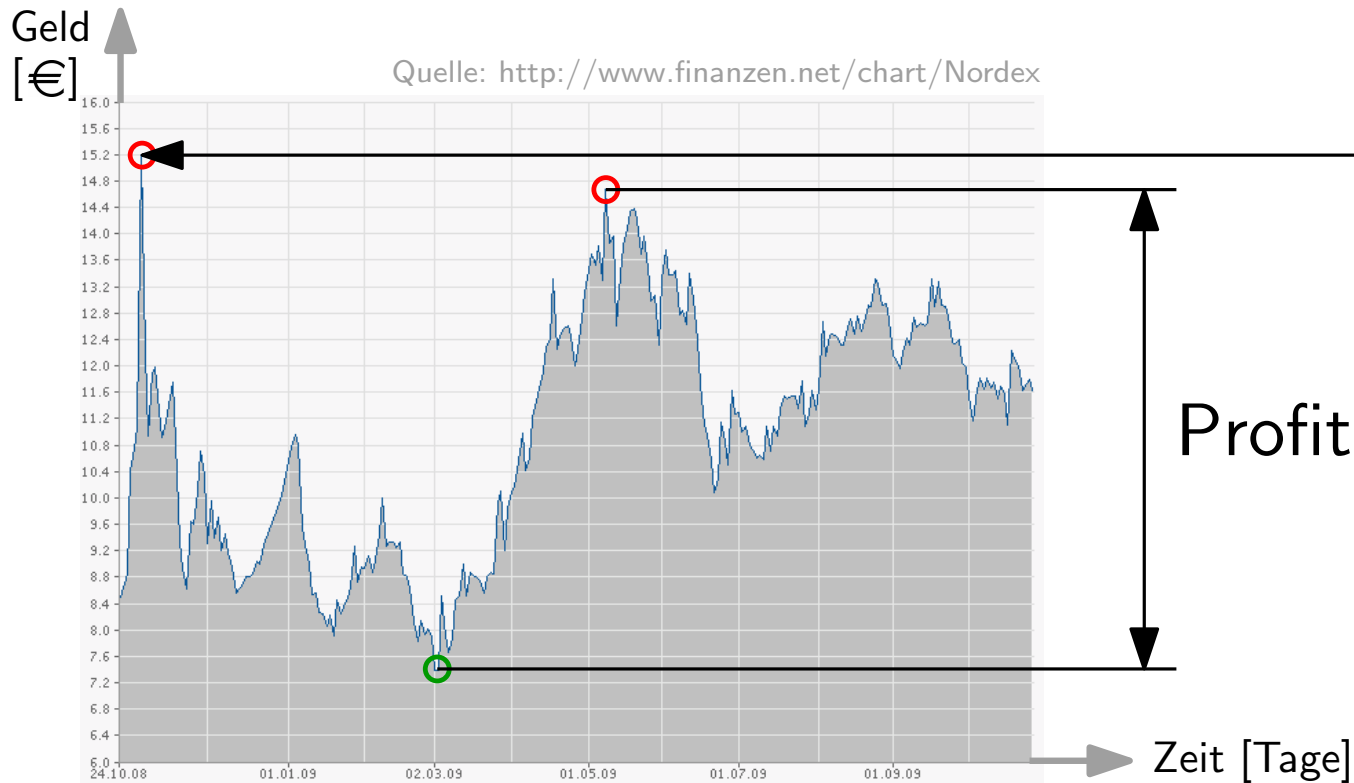
so dass $A[j] - A[i]$ maximal.

Verkaufskurs

Profit pro Aktie

Einkaufskurs

Analyse von Aktienkursen



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.

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

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“

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“
– für alle erlaubten Paare (i, j) berechne $A[j] - A[i]$

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“

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

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

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

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

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

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
 $= (n - 1) + (n - 2) + \dots + 2 + 1$

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

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.

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

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



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
---	---	---	---	---	---	---	----	---	---	---	---	---	---	---

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
---	---	---	---	---	---	---	----	---	---	---	---	---	---	---

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
---	---	----	---	----	---	---	----	---	----	---	---	---	----	---

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

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

$\Rightarrow (6, 13)$

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 $(1, 13)$

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*“

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

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

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

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:

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:



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!

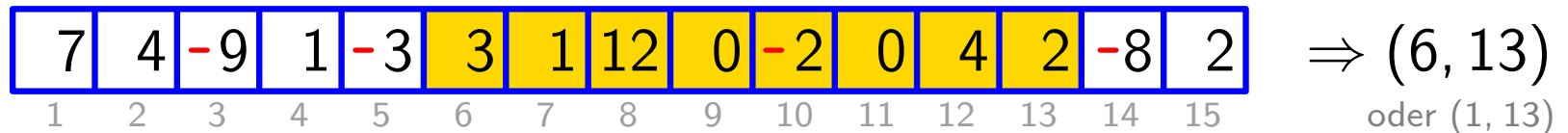
- für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
- gib Maximum zurück

Laufzeit \approx Anzahl der Additionen
 Obere Schranke dafür:

$O(n^2)$.

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**



Lösung: per „roher Gewalt“

Übung:
Schreiben Sie
Pseudocode!

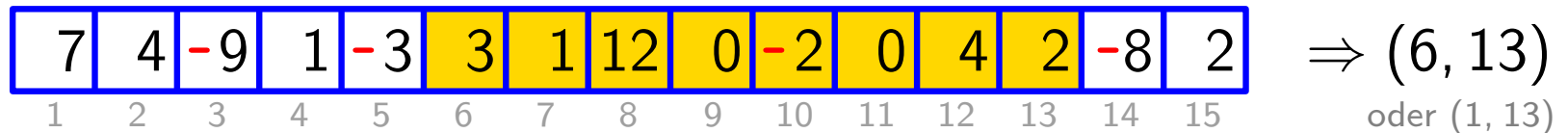
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
 – gib Maximum zurück

Laufzeit \approx Anzahl der Additionen
 Obere Schranke dafür:

$O(n^2)$ ·

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**



Lösung: per „roher Gewalt“

Übung:
Schreiben Sie
Pseudocode!

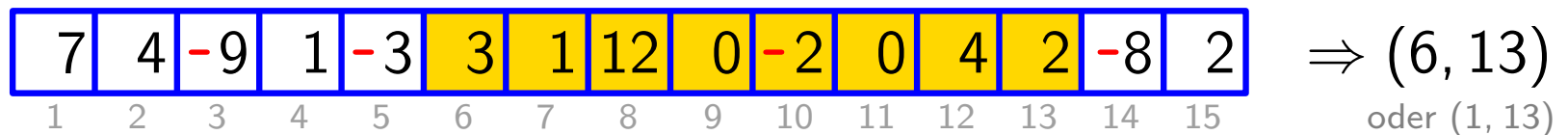
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
 – gib Maximum zurück

Laufzeit \approx Anzahl der Additionen
 Obere Schranke dafür:

$$O(n^2) \cdot O(n)$$

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**



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

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:
 Untere Schranke

$$O(n^2) \cdot O(n) = O(n^3)$$

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)

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

Genauere Analyse

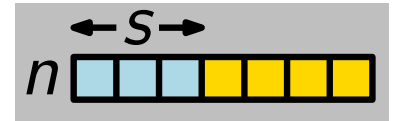
- Laufzeit** \approx Anzahl der Additionen des Rohe-Gewalt-Algos:
- für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^j A[k]$
 - gib Maximum zurück

Beob.

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

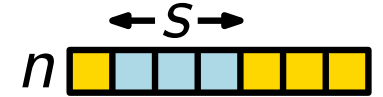


Beob. ● Anz. der Summen mit s Summanden ist

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

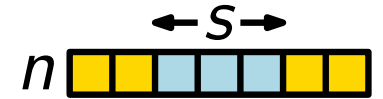


Beob. ● Anz. der Summen mit s Summanden ist

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

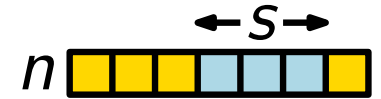


Beob. ● Anz. der Summen mit s Summanden ist

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

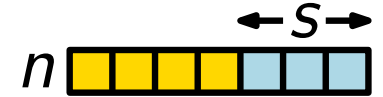


Beob. ● Anz. der Summen mit s Summanden ist

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

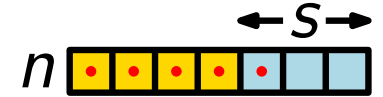


Beob. ● Anz. der Summen mit s Summanden ist

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

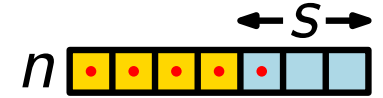


Beob. ● Anz. der Summen mit s Summanden ist

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

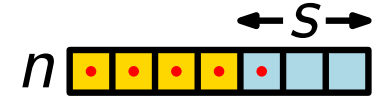


Beob. ● Anz. der Summen mit s Summanden ist $n - s + 1$.

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

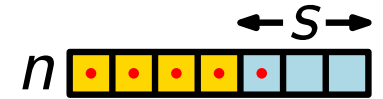


- Beob.**
- Anz. der Summen mit s Summanden ist $n - s + 1$.
 - s Summanden benötigen ? Additionen.

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.

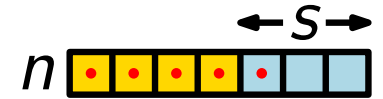
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

\Rightarrow Anz. Add. =

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.

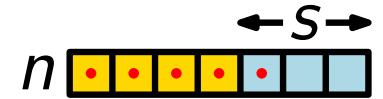
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

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

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.

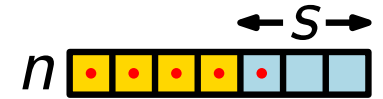
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

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

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.

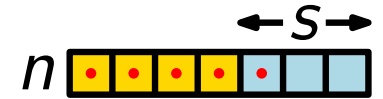
$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \end{aligned}$$

(falls $4|n$)

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.

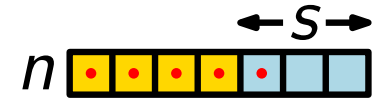
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \underbrace{\frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4}}_{\text{(falls } 4|n)} + \dots \end{aligned}$$

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.

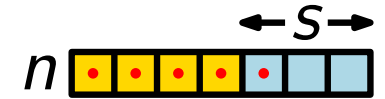
$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \\ &\quad \underbrace{\hspace{10em}} \\ &\frac{n}{2} + 1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4} \end{aligned}$$

(falls $4|n$)

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.

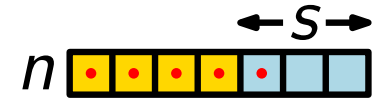
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \in \Omega(n^3) \\ &\quad \underbrace{\hspace{10em}} \\ &\frac{n}{2} + 1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4} \end{aligned}$$

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.

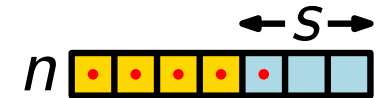
$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \in \Omega(n^3) \\ &\quad \underbrace{\hspace{10em}} \\ &\quad \frac{n}{2} + 1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4} \end{aligned}$$

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

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.

$$\begin{aligned} \Rightarrow \text{Anz. Add.} &= \sum_{s=1}^n (n - s + 1) \cdot (s - 1) \\ &= n \cdot 0 + (n - 1) \cdot 1 + (n - 2) \cdot 2 + \dots + 2 \cdot (n - 2) + 1 \cdot (n - 1) \\ &= \dots + \frac{3n}{4} \cdot \frac{n}{4} + \dots + \frac{n}{2} \cdot \frac{n}{2} + \dots + \frac{n}{4} \cdot \frac{3n}{4} + \dots \in \Omega(n^3) \\ &\quad \underbrace{\hspace{10em}} \\ &\quad \frac{n}{2} + 1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4} \end{aligned}$$

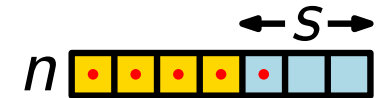
\Rightarrow Der Rohe-Gewalt-Alg. läuft in $O(n^3) \cap \Omega(n^3) = \Theta(n^3)$ Zeit.

Can we do better?

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!

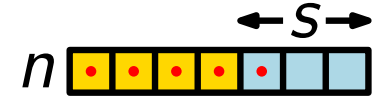
\Rightarrow Der Rohe-Gewalt-Alg. läuft in $O(n^3) \cap \Omega(n^3) = \Theta(n^3)$ Zeit.

Can we do better?

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.

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

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

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

(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.

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.

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 $i = 1, \dots, n$

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

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

Wie?

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



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

The diagram illustrates the calculation of the first two terms of the sequence of partial sums for a fixed i . It shows S_{ii} and $S_{i,i+1}$ above an equals sign. Below the equals sign are $A[i]$ and $A[i+1]$ separated by a plus sign. An arrow points from $A[i]$ to S_{ii} , and another arrow points from $A[i+1]$ to $S_{i,i+1}$. A curved arrow also points from $A[i+1]$ to S_{ii} , indicating that S_{ii} is simply $A[i]$.

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

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.

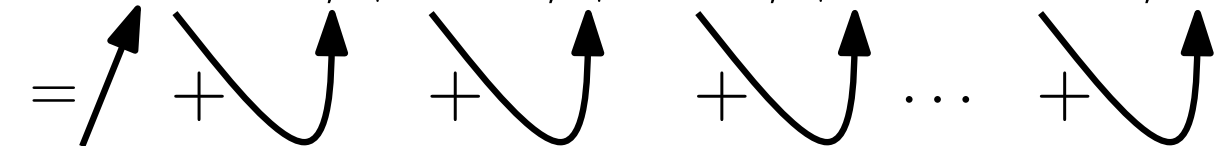
Idee: Für $i = 1, \dots, n$

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

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.

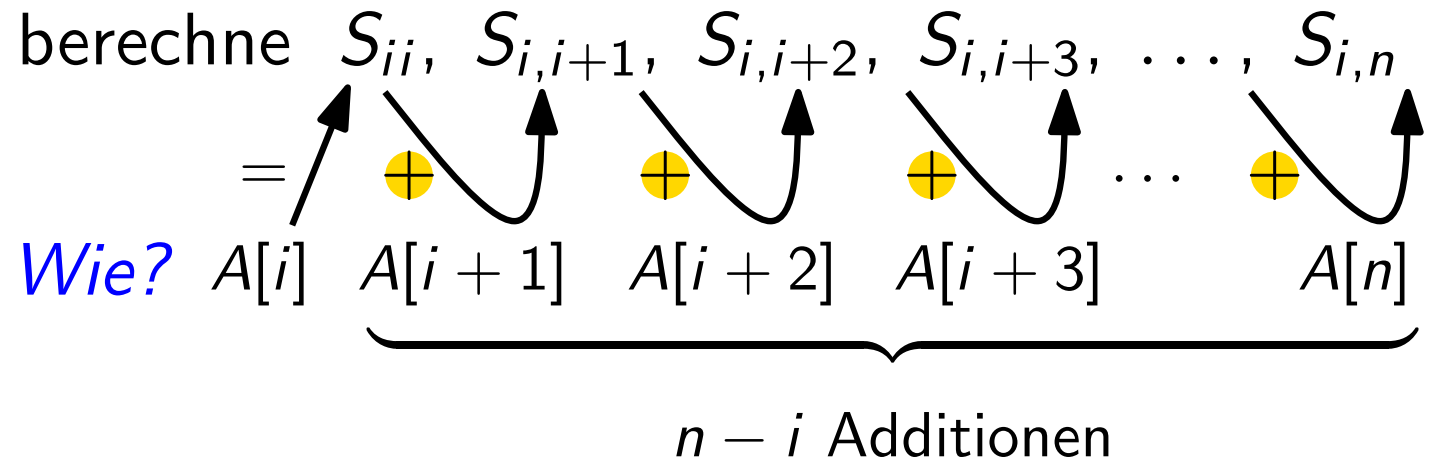
Idee: Für $i = 1, \dots, n$

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

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.

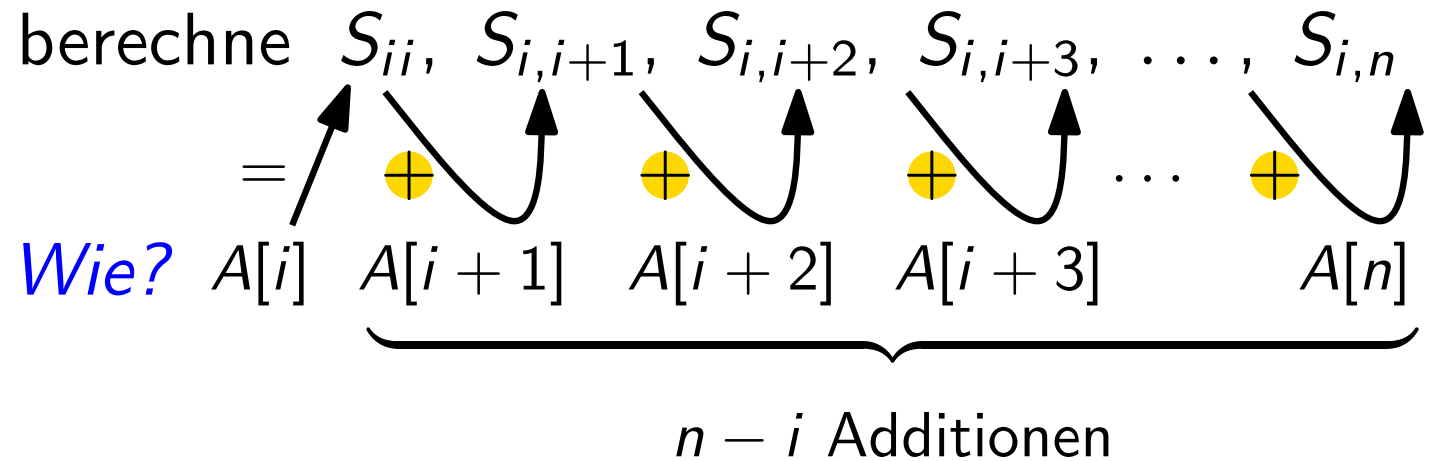
Idee: Für $i = 1, \dots, n$



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 $i = 1, \dots, n$



Insgesamt

Eine schnellere Lösung

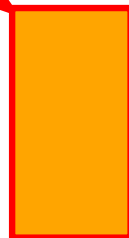
Problem: Gegeben: Folge $A[1..n]$ von ganzen Zahlen
 Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$,
 so dass $S_{ij} = \sum_{k=i}^j A[k]$ maximal.

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? $A[i] \quad A[i+1] \quad A[i+2] \quad A[i+3] \quad \dots \quad A[n]$
 $n - i$ Additionen

Insgesamt



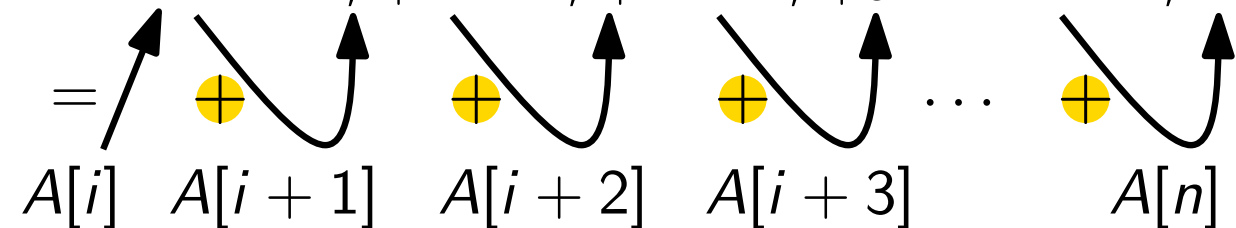
Eine schnellere Lösung

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

$=$ 

Wie? $A[i]$ $A[i+1]$ $A[i+2]$ $A[i+3]$ \dots $A[n]$

$n - i$ Additionen

Insgesamt

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

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

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

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

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

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

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

$n - i$

$$= (n - 1) + (n - 2) + \dots + 1 + 0$$

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

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

$n - i$

$$= 0 + 1 + \dots + (n - 2) + (n - 1)$$

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

$$\sum_{i=1}^n (n - i) = \sum_{j=0}^{n-1} j$$

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

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

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

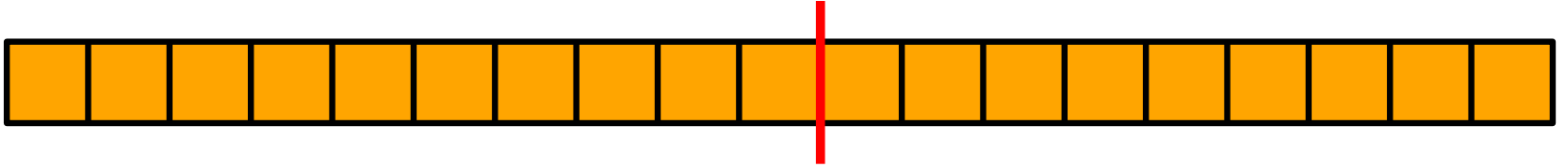
Eine noch schnellere Lösung?

Idee:



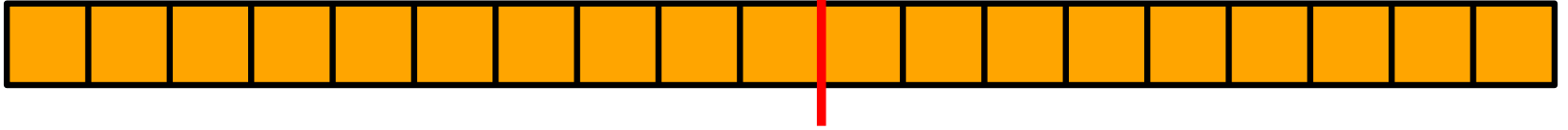
Eine noch schnellere Lösung?

Idee:



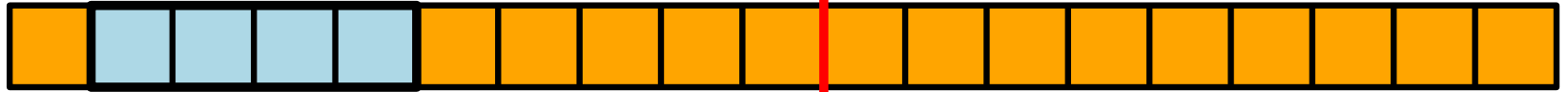
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

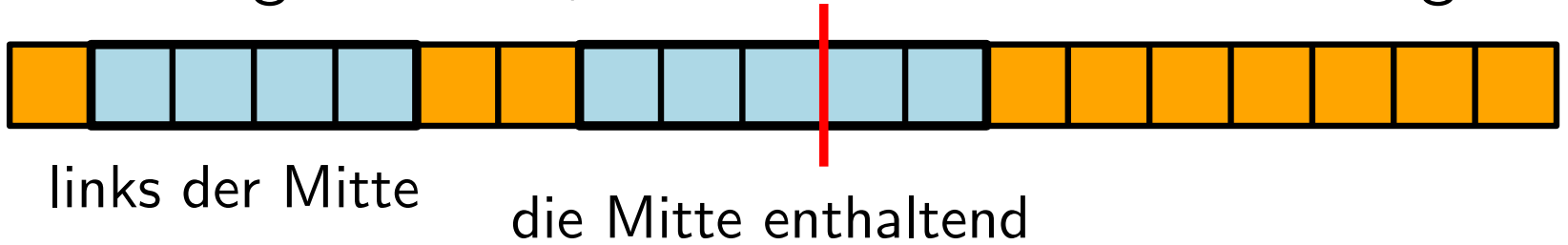
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



links der Mitte

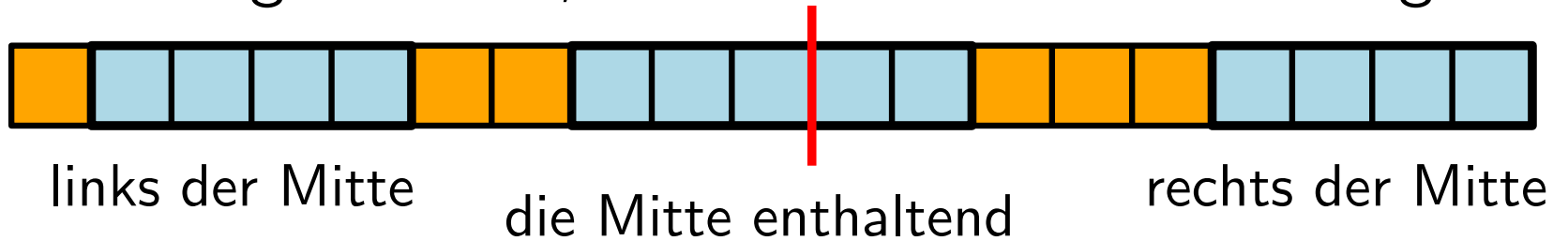
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



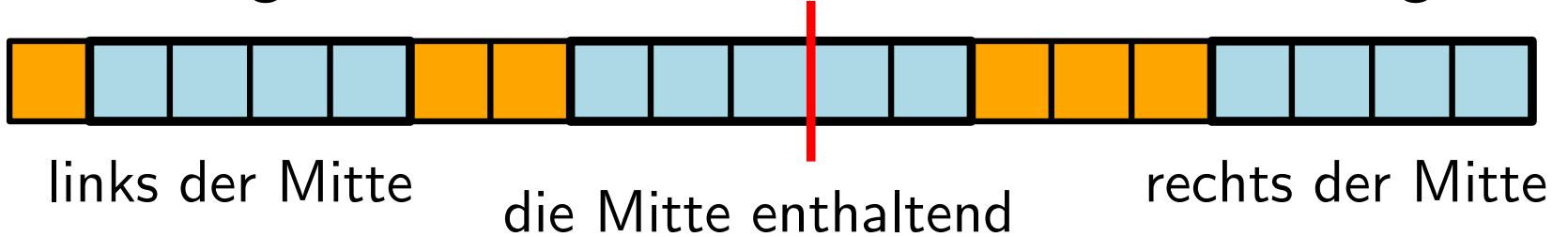
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

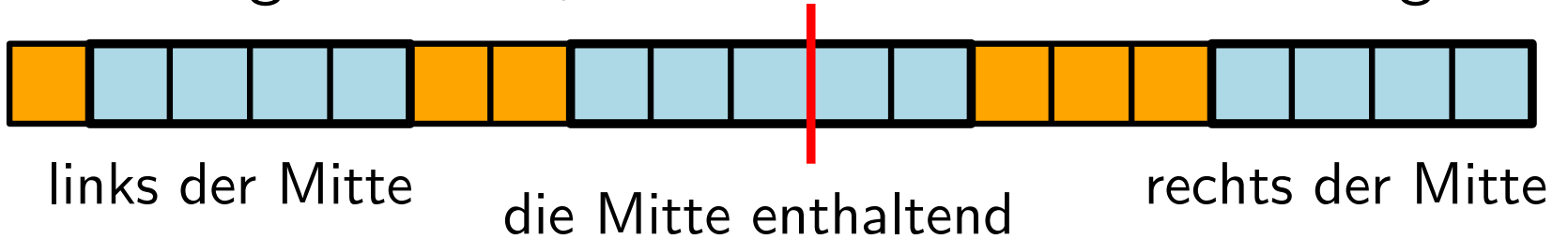
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

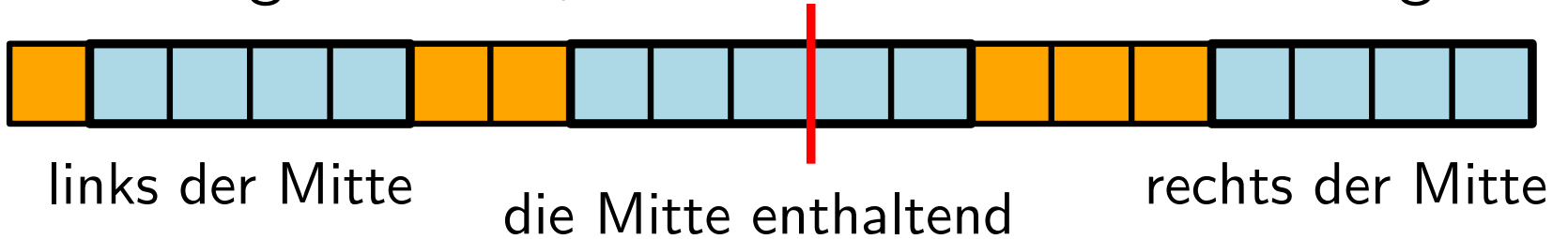


Nimm Entwurfstechnik *Teile & Herrsche!*

- *teile:*
- *herrsche:*
- *kombiniere:*

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

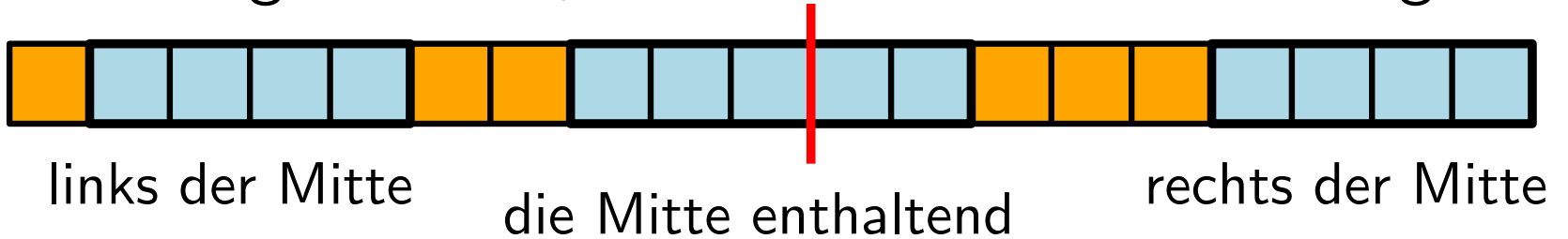


Nimm Entwurfstechnik *Teile & Herrsche!*

- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:*
- *kombiniere:*

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

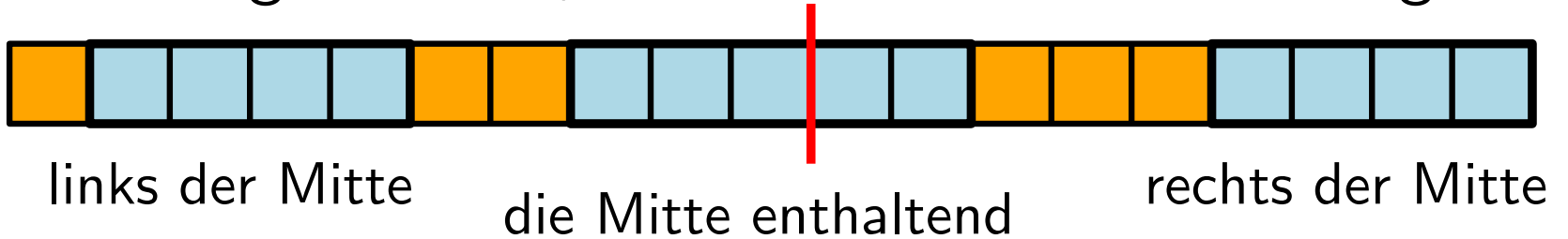


Nimm Entwurfstechnik *Teile & Herrsche!*

- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:*

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

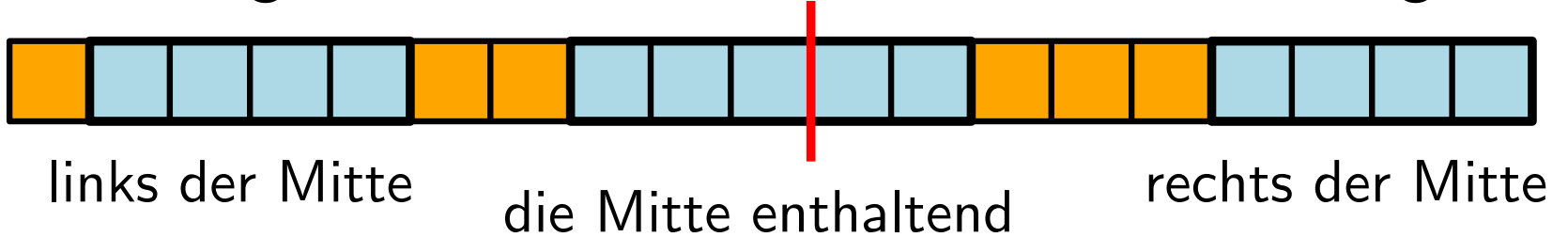


Nimm Entwurfstechnik *Teile & Herrsche!*

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

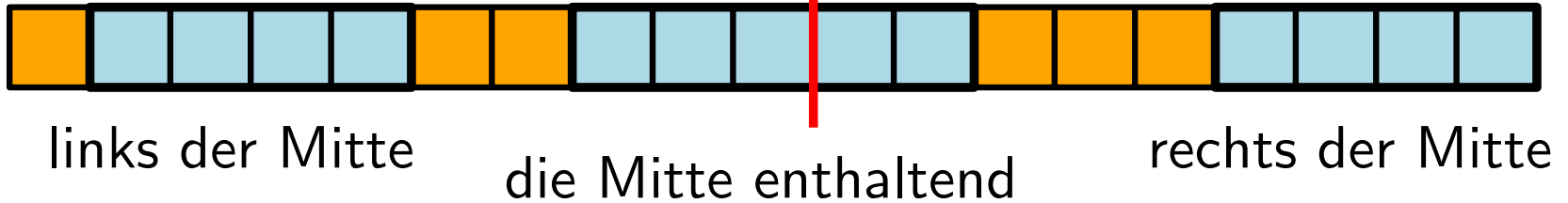


Nimm Entwurfstechnik *Teile & Herrsche!*

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

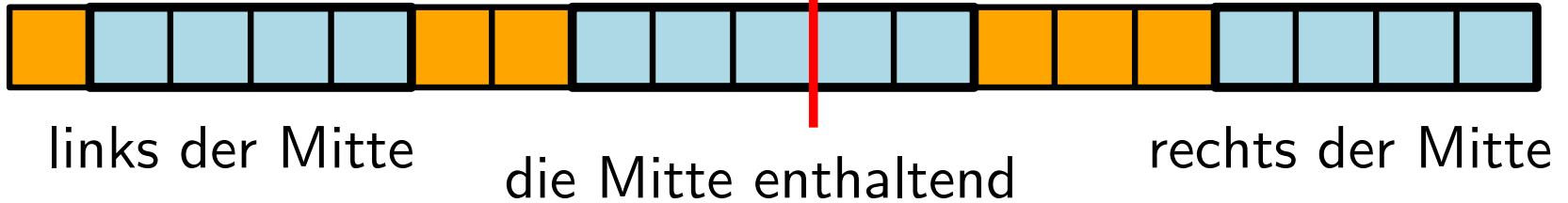


Nimm Entwurfstechnik *Teile & Herrsche!*

- *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*

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



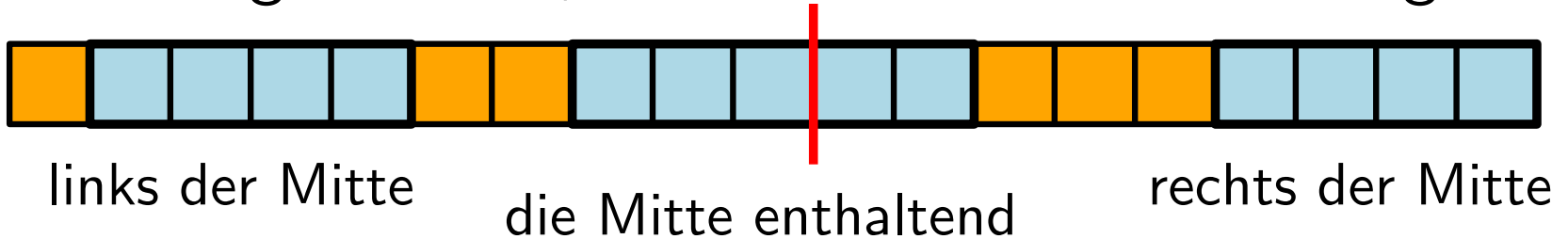
Nimm Entwurfstechnik *Teile & Herrsche!*

- *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 $\frac{n}{2} \cdot \frac{n}{2}$

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



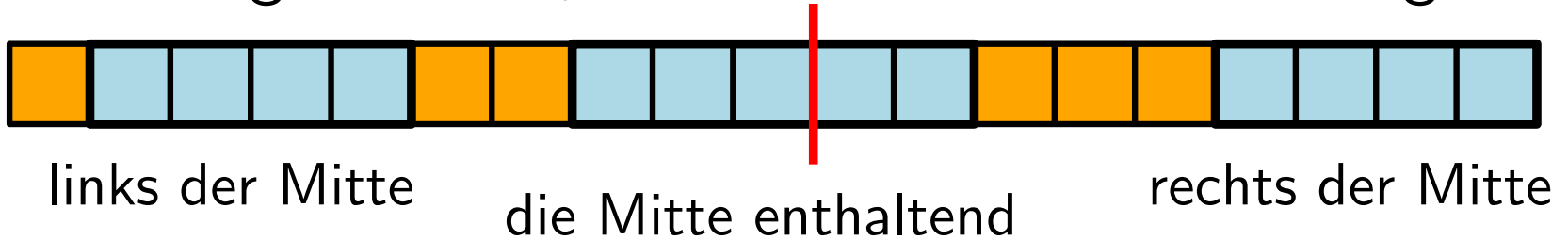
Nimm Entwurfstechnik *Teile & Herrsche!*

- *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 $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

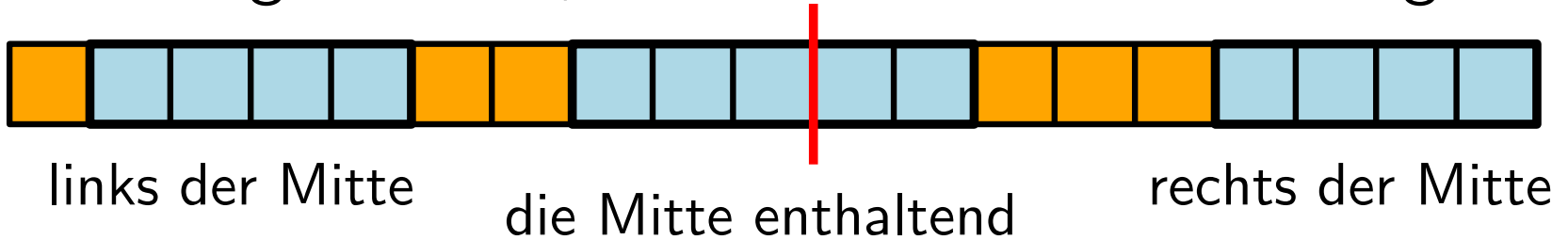
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte
enthalten

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



Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

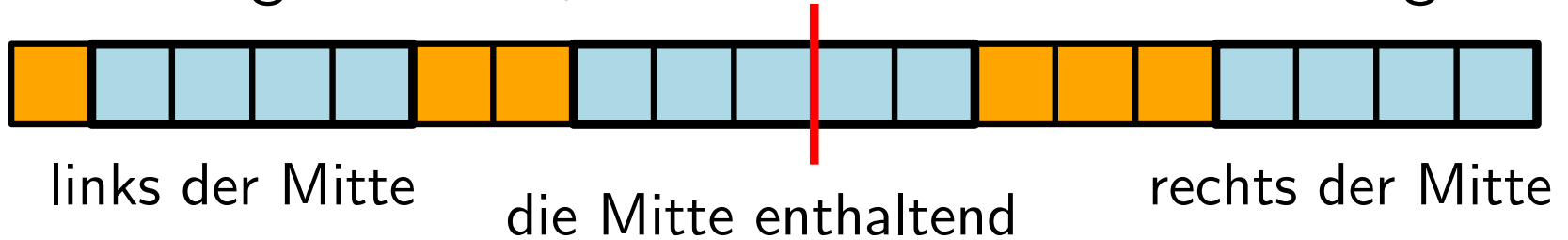
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte enthalten

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

Einsicht:

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

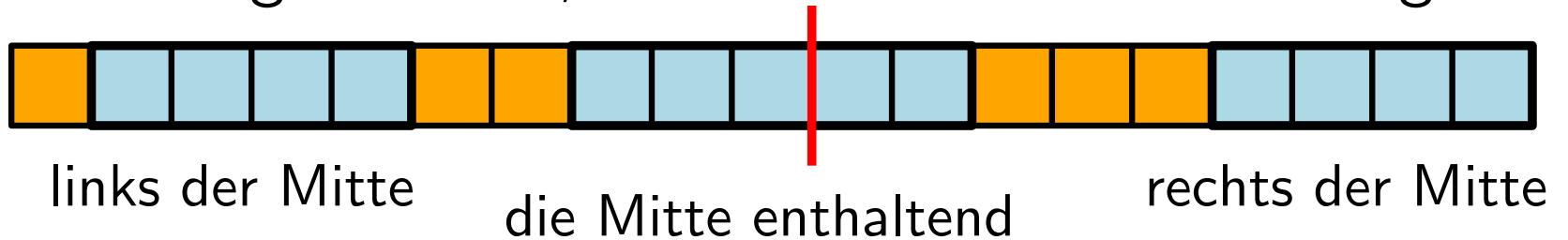
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte
enthalten

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

Einsicht: Wenn die *maximale* Teilsumme die Mitte enthält,

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

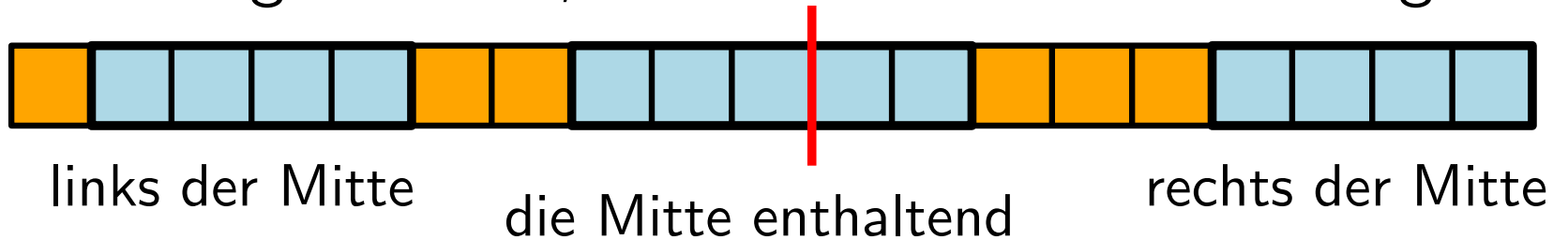
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte
enthalten

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

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

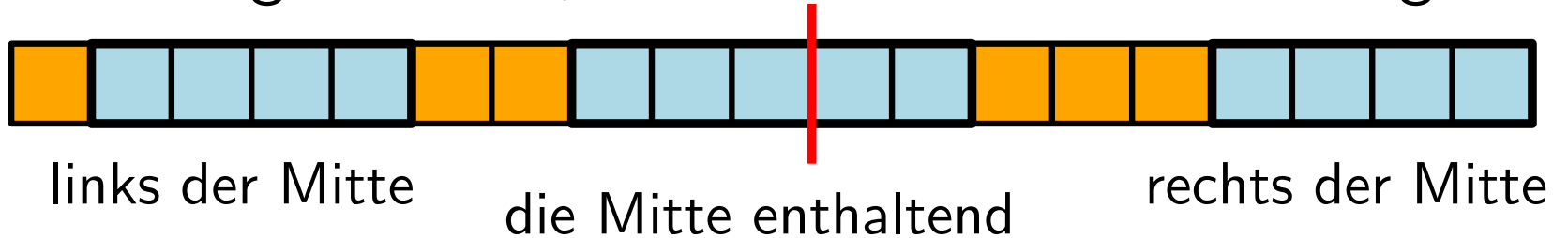
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$

Einsicht: Wenn die *maximale* Teilsumme die Mitte enthält, dann muss ihr linker Teil (bis zur Mitte) maximal sein

und

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche!*

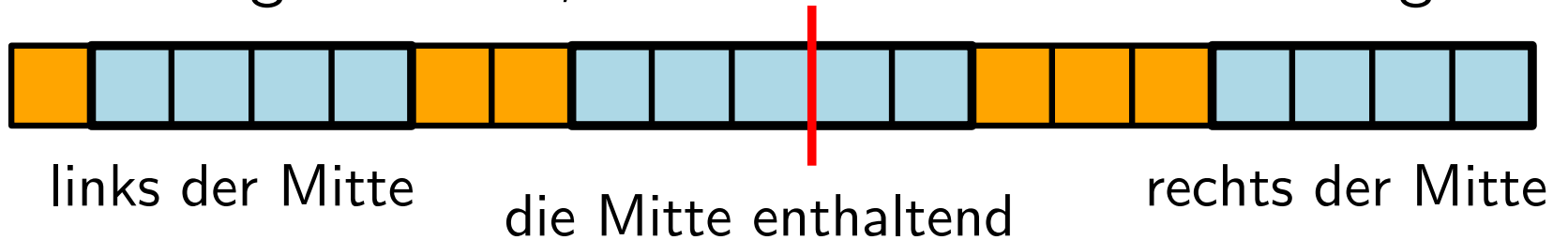
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere **alle** Teilsummen, die die Mitte enthalten

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.

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])
  else
    mitte = ⌊(beginn + ende)/2⌋
    (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)
    return (Tripel mit größter Summe)
```

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 =  $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$ 
    | (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)
    | return (Tripel mit größter Summe)
  
```

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 =  $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$  teile
    | (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)
    | return (Tripel mit größter Summe)
  
```


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 = $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$ *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*) =

Max**Mittleres**Teilfeld(A, *beginn*, *mitte*, *ende*)

return (Tripel mit größter Summe)

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 = $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$ *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*)

return (Tripel mit größter Summe) *kombiniere*

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 = $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$ *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*) =

↓
 Max**Mittleres**Teilfeld(A, *beginn*, *mitte*, *ende*)

return (Tripel mit größter Summe)

kombiniere

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 =  $\lfloor (\textit{beginn} + \textit{ende}) / 2 \rfloor$  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)
  | return (Tripel mit größter Summe) kombiniere

```

Laufzeit:

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)
    | return (Tripel mit größter Summe) kombiniere

```

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

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

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

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*)

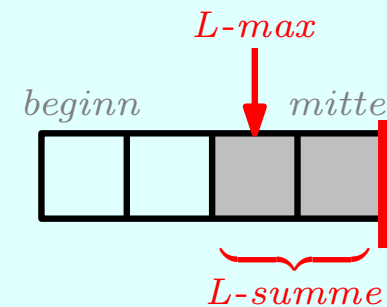
Kombiniere

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

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

$summe = 0$

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



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

Kombiniere

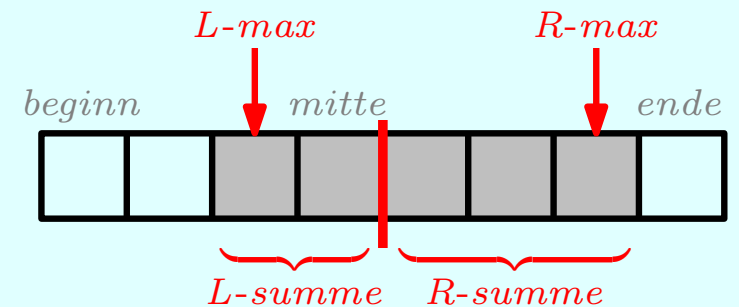
MaxMittleresTeilfeld(int[] A, int *beginn*, int *mitte*, int *ende*)

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

$summe = 0$

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

└



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

$summe = 0$

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

└ // analog zu oben

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

Kombiniere

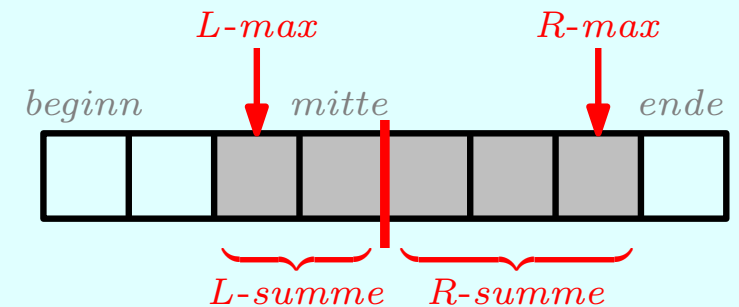
MaxMittleresTeilfeld(int[] A, int *beginn*, int *mitte*, int *ende*)

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

$summe = 0$

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

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**

$summe = summe + A[i]$

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

$L\text{-summe} = summe$

$L\text{-max} = i$

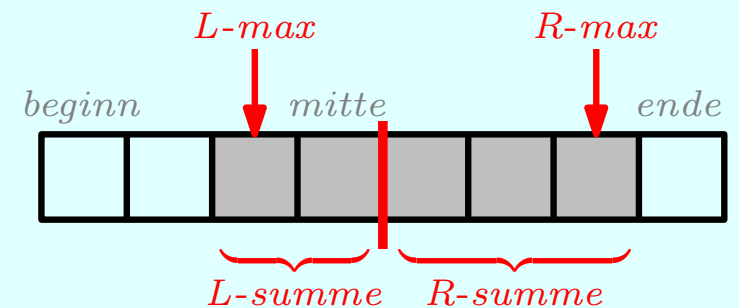
$R\text{-summe} = -\infty$

$summe = 0$

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

 // analog zu oben

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**

┌ $summe = summe + A[i]$

└ **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

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

Korrektheit?

Kombiniere

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

L-summe = $-\infty$

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 = $-\infty$

summe = 0

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

┌ // analog zu oben

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

Korrektheit?

Schleifeninvariante:

Kombiniere

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

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

$summe = 0$

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

$summe = summe + A[i]$

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

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

Korrektheit?

Schleifeninvariante:

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

Korrektheit?

Schleifeninvariante:

$summe = S_{i, mitte}$

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

Korrektheit?

Schleifeninvariante:

$summe = S_{i, mitte}$ und

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

Korrektheit?

Schleifeninvariante:

$summe = S_{i, mitte}$ und

$L\text{-summe} =$

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

Korrektheit?

Schleifeninvariante:

$summe = S_{i, mitte}$ und

$L\text{-summe} =$

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

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i, mitte}$ und

$L\text{-summe} =$

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

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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?

Kombiniere

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

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

$summe = 0$

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

└ $summe = summe + A[i]$

└ **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

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

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

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

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

→ $mitte - beginn + 1$

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

$mitte - beginn + 1$

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

$\rightarrow mitte - beginn + 1$

$\rightarrow ende - mitte$

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

$mitte - beginn + 1$

$ende - mitte$

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

$mitte - beginn + 1$

$ende - mitte$

$ende - beginn + 1$

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

$mitte - beginn + 1$

$ende - mitte$

$ende - beginn + 1$

$= n$

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

$$\begin{aligned} \text{für } n > 1: \quad T_{\text{MT}}(n) &\approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{MMT}}(n) \\ &= 2 \cdot T_{\text{MT}}(n/2) + n \end{aligned}$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

$$\begin{aligned} \text{für } n > 1: \quad T_{\text{MT}}(n) &\approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{MMT}}(n) \\ &= 2 \cdot T_{\text{MT}}(n/2) + n \\ &= V_{\text{MS}}(n) \end{aligned}$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

$$\begin{aligned} \text{für } n > 1: \quad T_{\text{MT}}(n) &\approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{MMT}}(n) \\ &= 2 \cdot T_{\text{MT}}(n/2) + n \\ &= V_{\text{MS}}(n) = n \log_2 n \quad [\text{für } n = \text{Zweierpotenz}] \end{aligned}$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

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

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:
 $\Theta(\log_a n) = \Theta(\log_b n)$.

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:
 $\Theta(\log_a n) = \Theta(\log_b n)$.



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!

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

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

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?

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...?

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 $T(1) = \Theta(1)$)

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

Algorithmus

Rohe Gewalt

Reihenfolge der
Summen ändern

Teile & Herrsche

Linearer Scan

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus

Laufzeit

Rohe Gewalt

Reihenfolge der
Summen ändern

Teile & Herrsche

Linearer Scan

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus

Laufzeit

Rohe Gewalt

$O(n^3)$

Reihenfolge der
Summen ändern

Teile & Herrsche

Linearer Scan

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus

Laufzeit

Rohe Gewalt

$O(n^3)$

Reihenfolge der
Summen ändern

$O(n^2)$

Teile & Herrsche

Linearer Scan

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus

Laufzeit

Rohe Gewalt

$O(n^3)$

Reihenfolge der
Summen ändern

$O(n^2)$

Teile & Herrsche

$O(n \log n)$

Linearer Scan

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus

Laufzeit

Rohe Gewalt

$O(n^3)$

Reihenfolge der
Summen ändern

$O(n^2)$

Teile & Herrsche

$O(n \log n)$

Linearer Scan

$O(n)$

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

– Anzahl der Additionen

<i>Algorithmus</i>	<i>Laufzeit</i>
Rohe Gewalt	$O(n^3)$
Reihenfolge der Summen ändern	$O(n^2)$
Teile & Herrsche	$O(n \log n)$
Linearer Scan	$O(n)$

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

– Anzahl der Additionen [★]

<i>Algorithmus</i>	<i>Laufzeit</i>
Rohe Gewalt	$O(n^3)$
Reihenfolge der Summen ändern	$O(n^2)$
Teile & Herrsche	$O(n \log n)$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$

[★]) Wir setzen die Konstante c in der $O()$ -Notation auf 1.

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log n)$	
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	

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

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log n)$	
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	10^3

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

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log_{10} n)$	
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	10^3

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

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$
Linearer Scan <small>(siehe Buch [CLRS], Ü-Aufgabe 4.1-5)</small>	$O(n)$	10^3

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

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	10^6
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	10^3

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

Übersicht

– Anzahl der Additionen [★]

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>
Rohe Gewalt	$O(n^3)$	10^9
Reihenfolge der Summen ändern	$O(n^2)$	10^6
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	10^3

★) Wir setzen die Konstante c in der $O()$ -Notation auf 1.

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>	<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9	
Reihenfolge der Summen ändern	$O(n^2)$	10^6	
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	
Linearer Scan <small>(siehe Buch [CLRS], Ü-Aufgabe 4.1-5)</small>	$O(n)$	10^3	

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

Übersicht

– Anzahl der Additionen ^{*}

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>	<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9	10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6	10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	10^6

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

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

Übersicht

- Anzahl der Additionen ^{*}
- geschätzte Rechenzeit

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>	<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9	10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6	10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	10^6

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>	<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9	10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6	10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	10^6

(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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>		<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9		10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6		10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$		$6 \cdot 10^6$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$O(n)$	10^3	$1 \mu s$	10^6

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

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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>		<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9		10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6		10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$3 \mu s$	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	$1 \mu s$	10^6

(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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>		<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9		10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$3 \mu s$	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	$1 \mu s$	10^6

(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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>		<i>n = 1 000 000</i>
Rohe Gewalt	$O(n^3)$	10^9	1 s	10^{18}
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	3 μ s	$6 \cdot 10^6$
Linearer Scan	$O(n)$	10^3	1 μ s	10^6

(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

Übersicht

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

<i>Algorithmus</i>	<i>Laufzeit</i>	<i>n = 1 000</i>		<i>n = 1 000 000</i>	
Rohe Gewalt	$O(n^3)$	10^9	1 s	10^{18}	
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}	
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$3 \mu\text{s}$	$6 \cdot 10^6$	
Linearer Scan	$O(n)$	10^3	$1 \mu\text{s}$	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

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}	
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}	
Teile & Herrsche	$O(n \log_{10} n)$	$3 \cdot 10^3$	$3 \mu\text{s}$	$6 \cdot 10^6$	6 ms
Linearer Scan	$O(n)$	10^3	$1 \mu\text{s}$	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

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}	
Reihenfolge der Summen ändern	$O(n^2)$	10^6	1 ms	10^{12}	1000 s
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

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}	
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

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

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