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

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

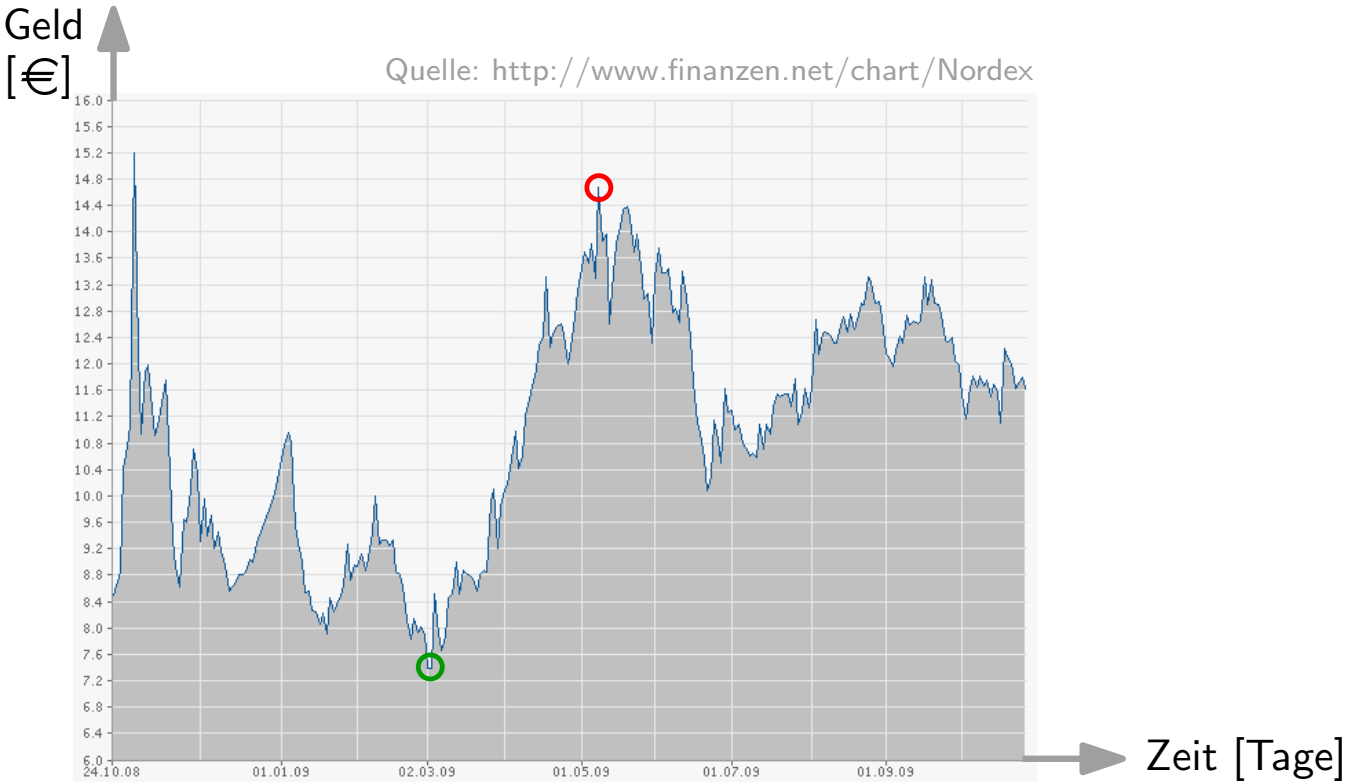
Problem. Gegeben: Folge $A[1..n]$ von Aktienkursen in Euro.

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

Verkaufskurs *Einkaufskurs*

Profit pro Aktie

Analyse von Aktienkursen



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

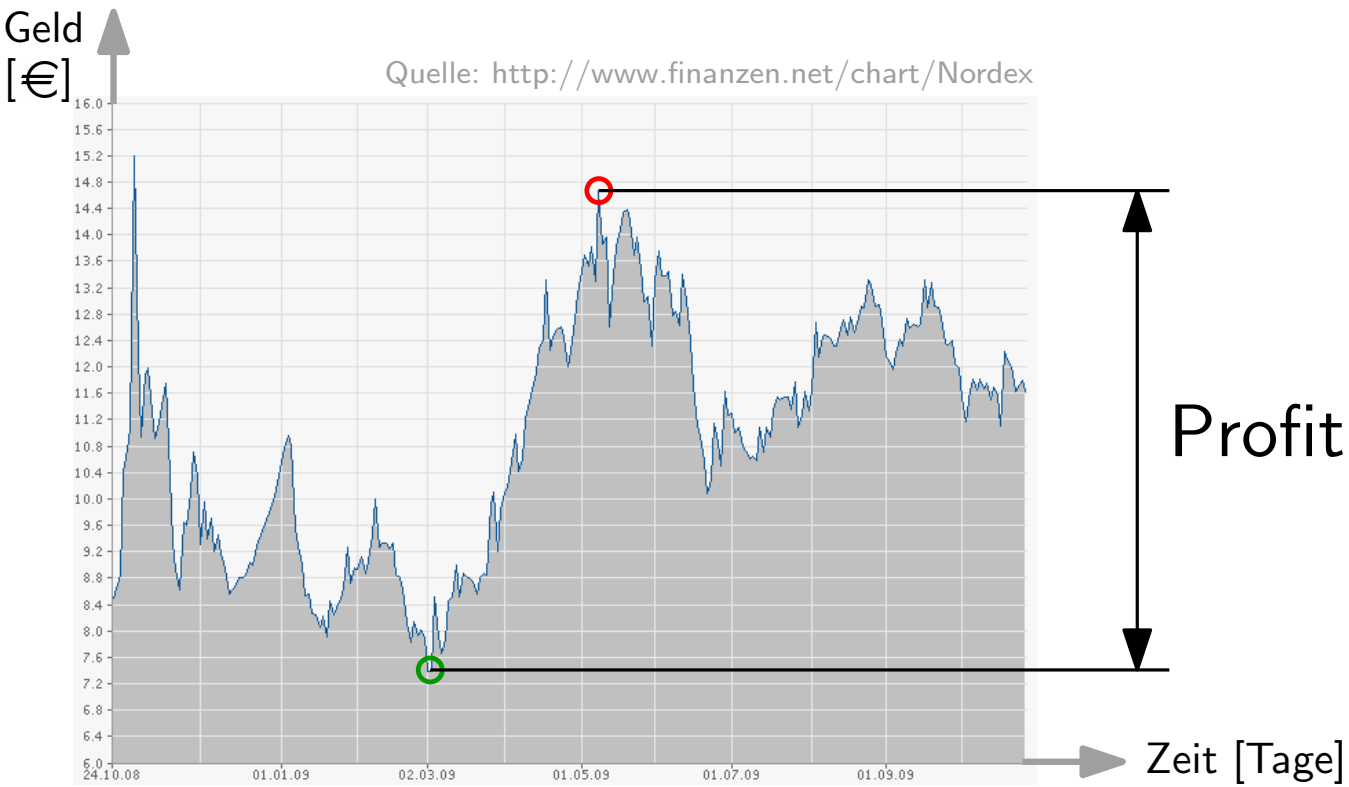
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$,
so dass $A[j] - A[i]$ maximal.

Verkaufskurs — $A[j]$

Einkaufskurs — $A[i]$

Profit pro Aktie — $A[j] - A[i]$

Analyse von Aktienkursen



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

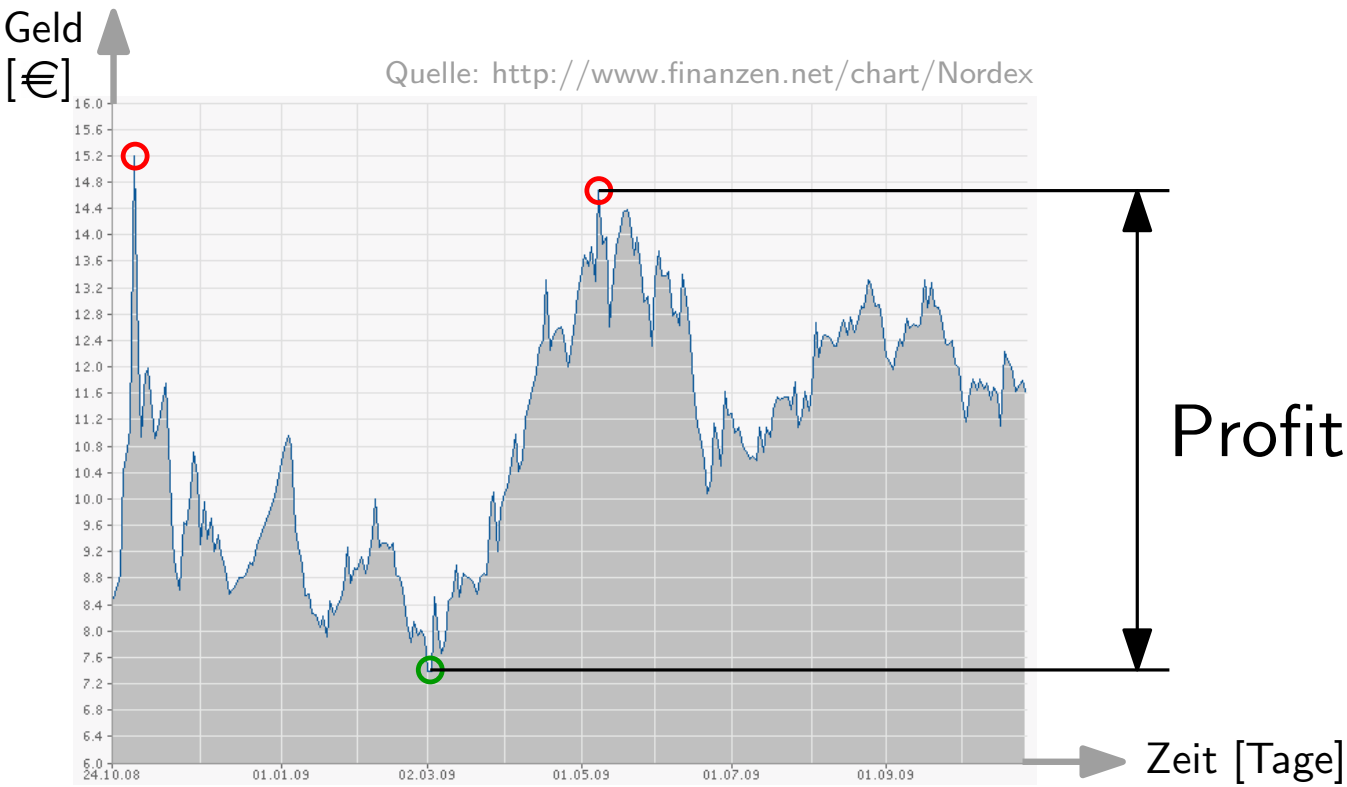
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$,
so dass $A[j] - A[i]$ maximal.

Verkaufskurs — $A[j]$

Einkaufskurs — $A[i]$

Profit pro Aktie — $A[j] - A[i]$

Analyse von Aktienkursen



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

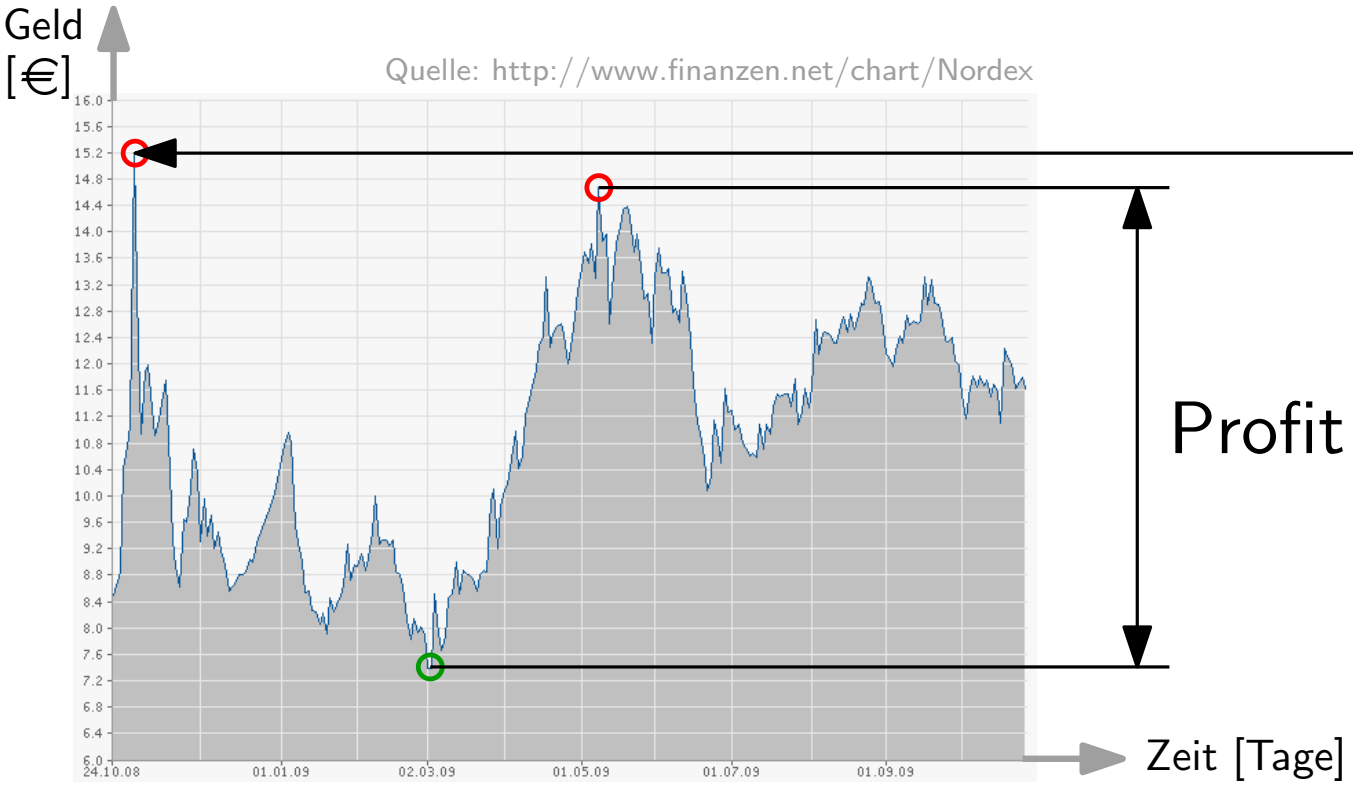
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$,
so dass $A[j] - A[i]$ maximal.

Verkaufskurs — $A[j]$

Einkaufskurs — $A[i]$

Profit pro Aktie — $A[j] - A[i]$

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 $1 \leq i < j \leq n$,
so dass $A[j] - A[i]$ maximal.

Verkaufskurs — $A[j]$

Einkaufskurs — $A[i]$

Profit pro Aktie — $A[j] - A[i]$

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.

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“*

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

– gib Maximum zurück

Übung:
Schreiben Sie Pseudocode!

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

Übung: Schreiben Sie Pseudocode!

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“*

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

– gib Maximum zurück

Übung:
Schreiben Sie Pseudocode!

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“*

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

– gib Maximum zurück

Übung: Schreiben Sie Pseudocode!

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

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“*

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

– gib Maximum zurück

Übung: Schreiben Sie Pseudocode!

Laufzeit $\approx$ Anzahl der berechneten Differenzen

$=$ Anzahl erlaubter Paare

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

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“*

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

– gib Maximum zurück

Übung: Schreiben Sie Pseudocode!

Laufzeit

$$\begin{aligned} &\approx \text{Anzahl der berechneten Differenzen} \\ &= \text{Anzahl erlaubter Paare} \\ &= (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n^2 - n}{2} \\ &\in \Theta(n^2) \end{aligned}$$

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

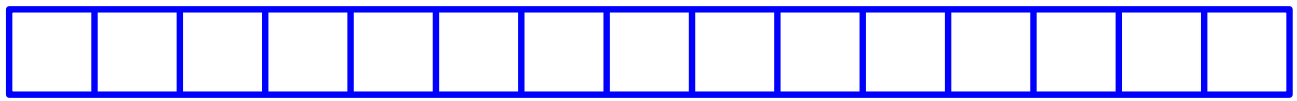
Problem: Gegeben: Folge $A[1..n]$ von ganzen Zahlen
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$,
so dass $\sum_{k=i}^{j} A[k]$ maximal.

} 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.

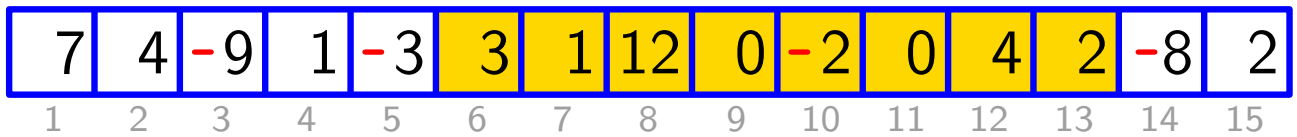
} MAX-SUM

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

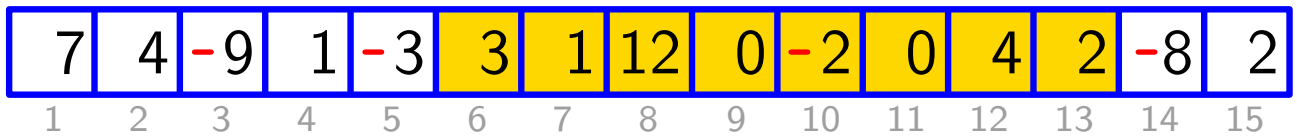


$\Rightarrow (6, 13)$

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

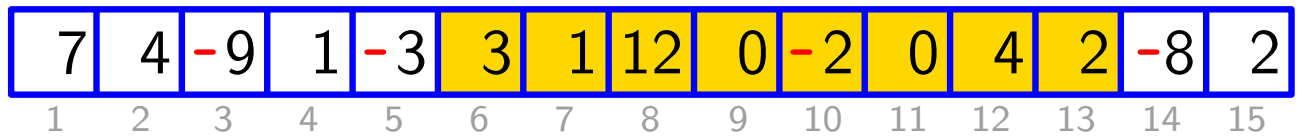


$\Rightarrow (6, 13)$
oder $(1, 13)$

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



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

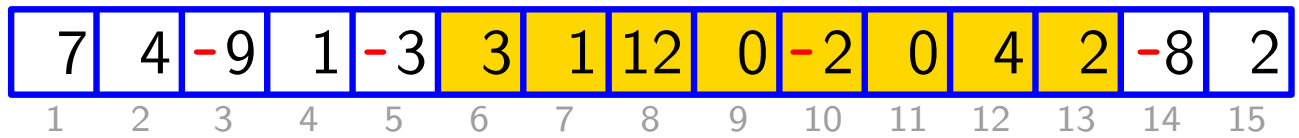
Lösung: per *„roher Gewalt“*

- 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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

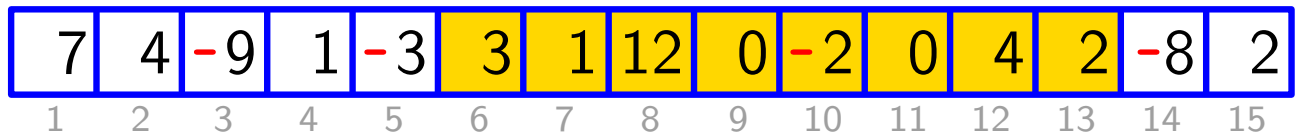
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$

– 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



$\Rightarrow (6, 13)$
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

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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

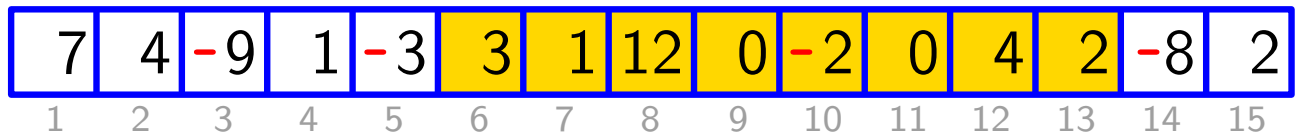
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

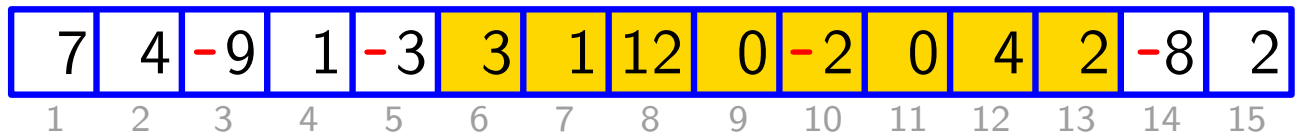
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

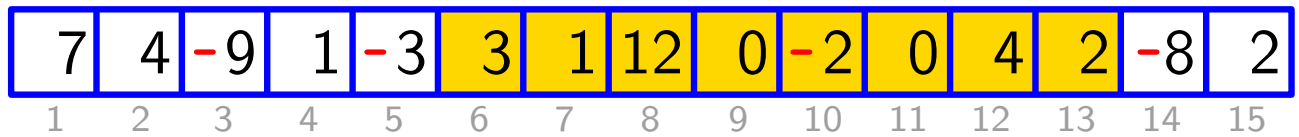
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

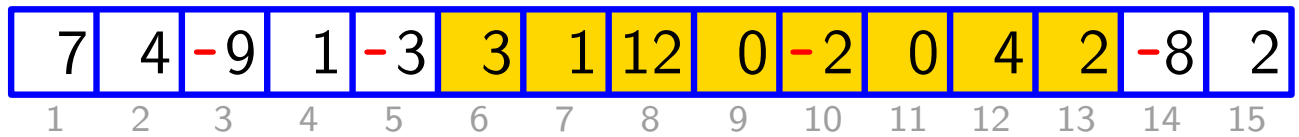
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

Laufzeit $\approx$ Anzahl der Additionen
Obere Schranke dafür: $O(n^2) \cdot$ [ ]

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

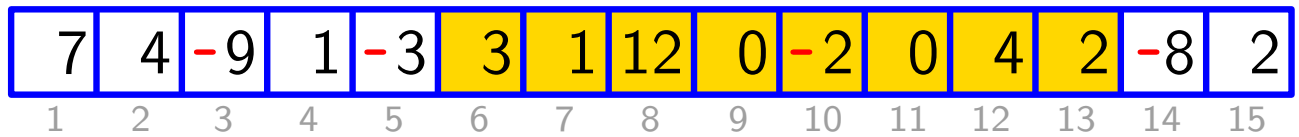
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

Laufzeit $\approx$ Anzahl der Additionen
Obere Schranke dafür: $O(n^2) \cdot O(n)$

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

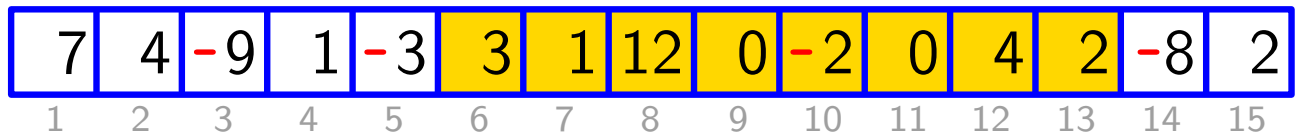
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

Laufzeit $\approx$ Anzahl der Additionen
Obere Schranke dafür: $O(n^2) \cdot O(n) = O(n^3)$

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

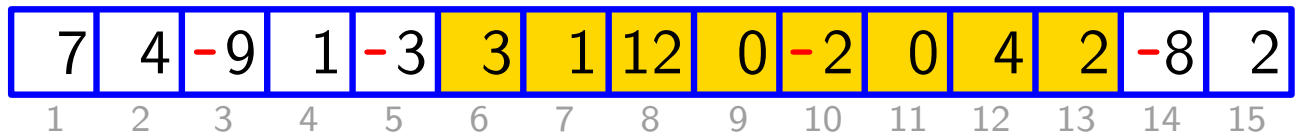
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

Laufzeit $\approx$ Anzahl der Additionen
Obere Schranke dafür: $O(n^2) \cdot O(n) = O(n^3)$
Untere Schranke

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



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

Lösung: per *„roher Gewalt“*

Übung: Schreiben Sie Pseudocode!

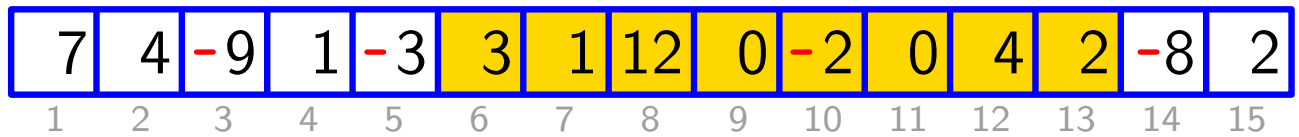
– für alle erlaubten Paare (i, j) berechne $\sum_{k=i}^{j} A[k]$
– gib Maximum zurück

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



$\Rightarrow (6, 13)$
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

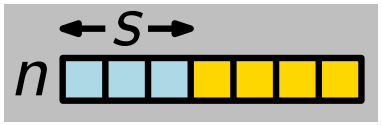
Beob.

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



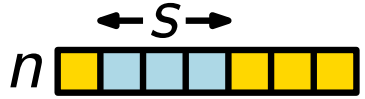
Beob.

- 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.

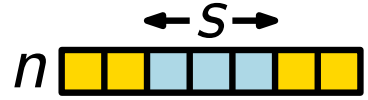
- Anz. der Summen mit s Summanden ist

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



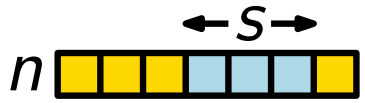
Beob.

- 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



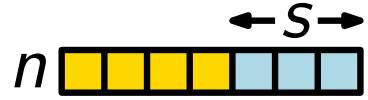
Beob.

- 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



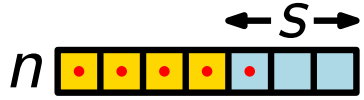
Beob.

- 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



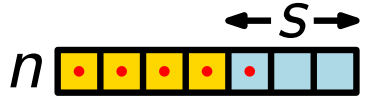
Beob.

- 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.

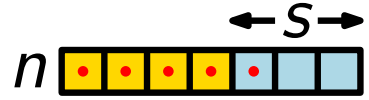
- Anz. der Summen mit s Summanden ist $n - s + 1$.

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



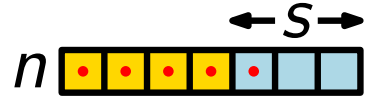
Beob.

- 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.

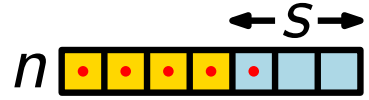
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

$\Rightarrow$ 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



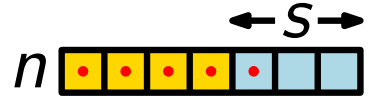
Beob.

- 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

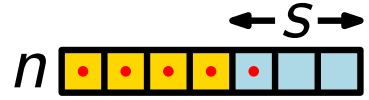


Beob.
- Anz. der Summen mit s Summanden ist $n - s + 1$.
- s Summanden benötigen $s - 1$ Additionen.

$$\Rightarrow \text{Anz. Add.} = \sum_{s=1}^{n} (n - s + 1) \cdot (s - 1)$$
$$= n \cdot 0 + (n-1) \cdot 1 + (n-2) \cdot 2 + \ldots + 2 \cdot (n-2) + 1 \cdot (n-1)$$

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.

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

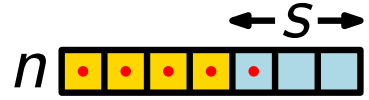
$$= n \cdot 0 + (n-1) \cdot 1 + (n-2) \cdot 2 + \ldots + 2 \cdot (n-2) + 1 \cdot (n-1)$$

$$= \ldots + \tfrac{3n}{4} \cdot \tfrac{n}{4} + \cdots + \tfrac{n}{2} \cdot \tfrac{n}{2} + \cdots + \tfrac{n}{4} \cdot \tfrac{3n}{4} + \ldots$$

(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.

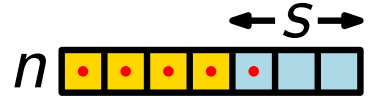
$$\Rightarrow \text{Anz. Add.} = \sum_{s=1}^{n} (n - s + 1) \cdot (s - 1)$$

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

$$= \ldots + \underbrace{\tfrac{3n}{4} \cdot \tfrac{n}{4} + \cdots + \tfrac{n}{2} \cdot \tfrac{n}{2} + \cdots + \tfrac{n}{4} \cdot \tfrac{3n}{4}}_{\text{(falls } 4 \mid n\text{)}} + \ldots$$

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.

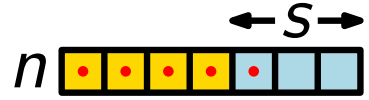
$$\Rightarrow \text{Anz. Add.} = \sum_{s=1}^{n} (n - s + 1) \cdot (s - 1)$$

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

$$\underset{(\text{falls } 4 \mid n)}{=} \ldots + \underbrace{\tfrac{3n}{4} \cdot \tfrac{n}{4} + \cdots + \tfrac{n}{2} \cdot \tfrac{n}{2} + \cdots + \tfrac{n}{4} \cdot \tfrac{3n}{4}}_{\frac{n}{2} + 1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4}} + \ldots$$

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.

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

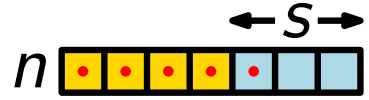
$$= n \cdot 0 + (n-1) \cdot 1 + (n-2) \cdot 2 + \ldots + 2 \cdot (n-2) + 1 \cdot (n-1)$$

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

(falls $4 \mid n$)

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.

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

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

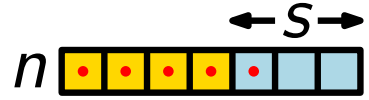
$$= \ldots + \underbrace{\tfrac{3n}{4} \cdot \tfrac{n}{4} + \cdots + \tfrac{n}{2} \cdot \tfrac{n}{2} + \cdots + \tfrac{n}{4} \cdot \tfrac{3n}{4}}_{\frac{n}{2}+1 \text{ Terme der Größe mindestens } \frac{n}{4} \cdot \frac{n}{4}} + \ldots \quad \in \Omega(n^3)$$

(falls $4 \mid n$)

$\Rightarrow$ 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.

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

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

$$= \ldots + \underbrace{\frac{3n}{4} \cdot \frac{n}{4} + \cdots + \frac{n}{2} \cdot \frac{n}{2} + \cdots + \frac{n}{4} \cdot \frac{3n}{4}}_{\text{(falls } 4 \mid n\text{)}} + \ldots \quad \in \Omega(n^3)$$

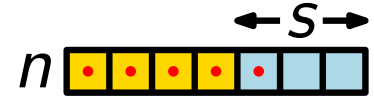
$\frac{n}{2} + 1$ Terme der Größe mindestens $\frac{n}{4} \cdot \frac{n}{4}$

$\Rightarrow$ 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.

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

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

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

(falls $4 \mid n$)

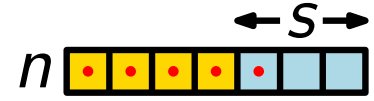
Übung:
Berechnen Sie diese Summe *genau* und beweisen Sie Ihr Ergebnis per Induktion!

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

Can we do better?

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.

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

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

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

(falls $4 \mid n$)

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 $1 \leq i \leq j \leq n$,
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 $i = 1, \ldots, n$

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

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 $i = 1, \ldots, n$

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

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 $i = 1, \ldots, n$

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

$=$

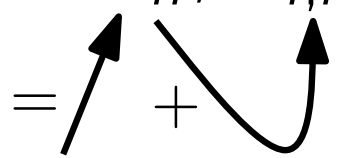
Wie? $A[i]$

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 $i = 1, \ldots, n$

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



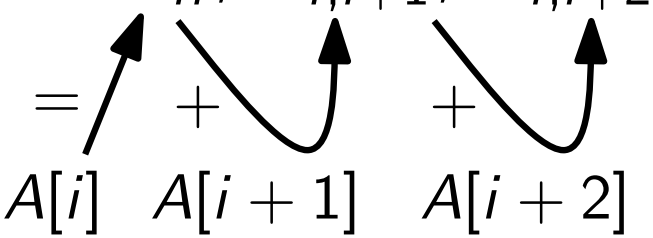
Wie? $A[i]$ $A[i+1]$

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 $i = 1, \ldots, n$

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

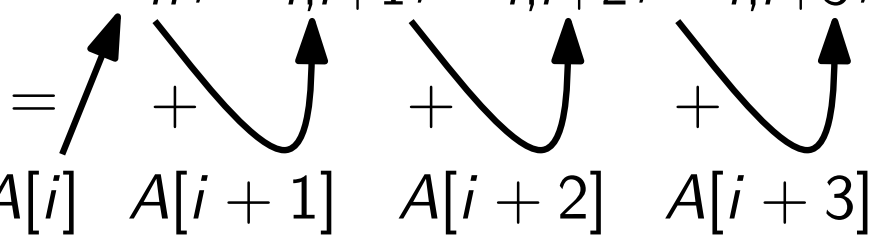


Eine schnellere Lösung

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 $i = 1, \ldots, n$

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

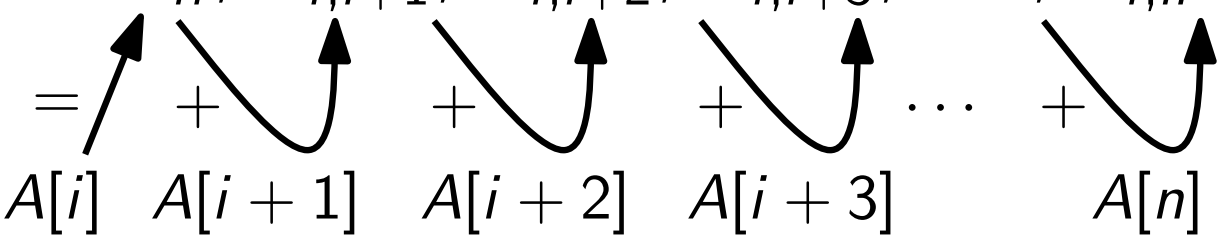


Eine schnellere Lösung

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 $i = 1, \ldots, n$

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

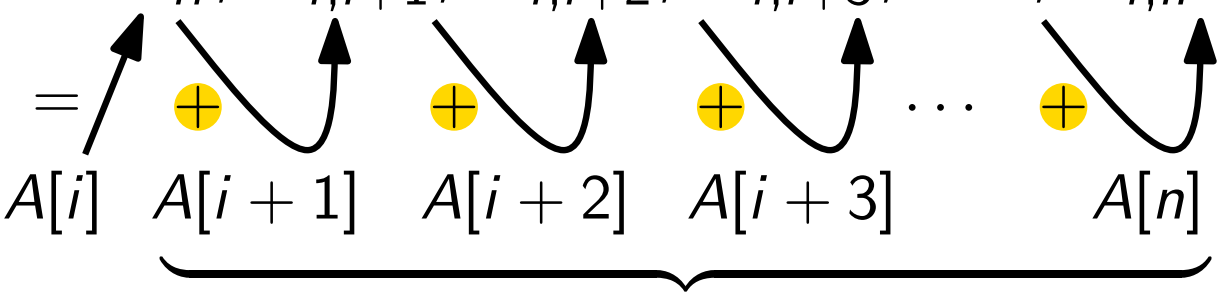


Eine schnellere Lösung

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 $i = 1, \ldots, n$

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

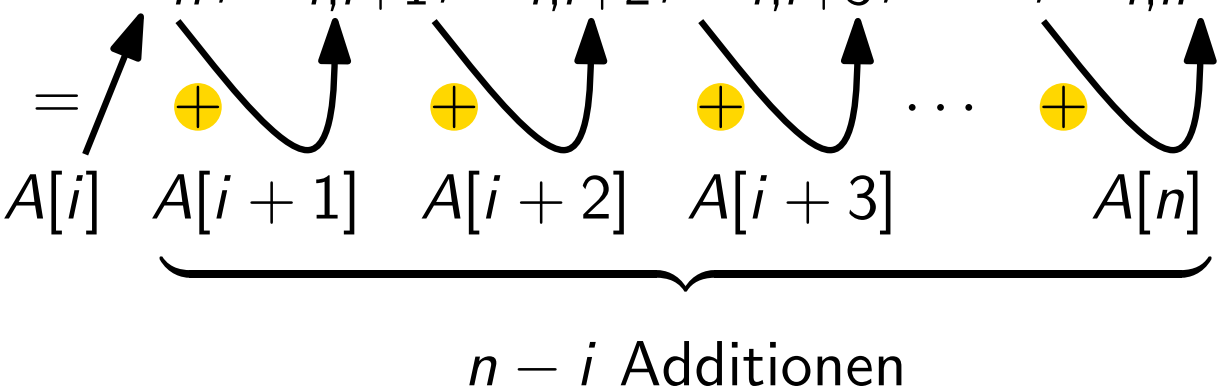


Eine schnellere Lösung

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 $i = 1, \ldots, n$

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

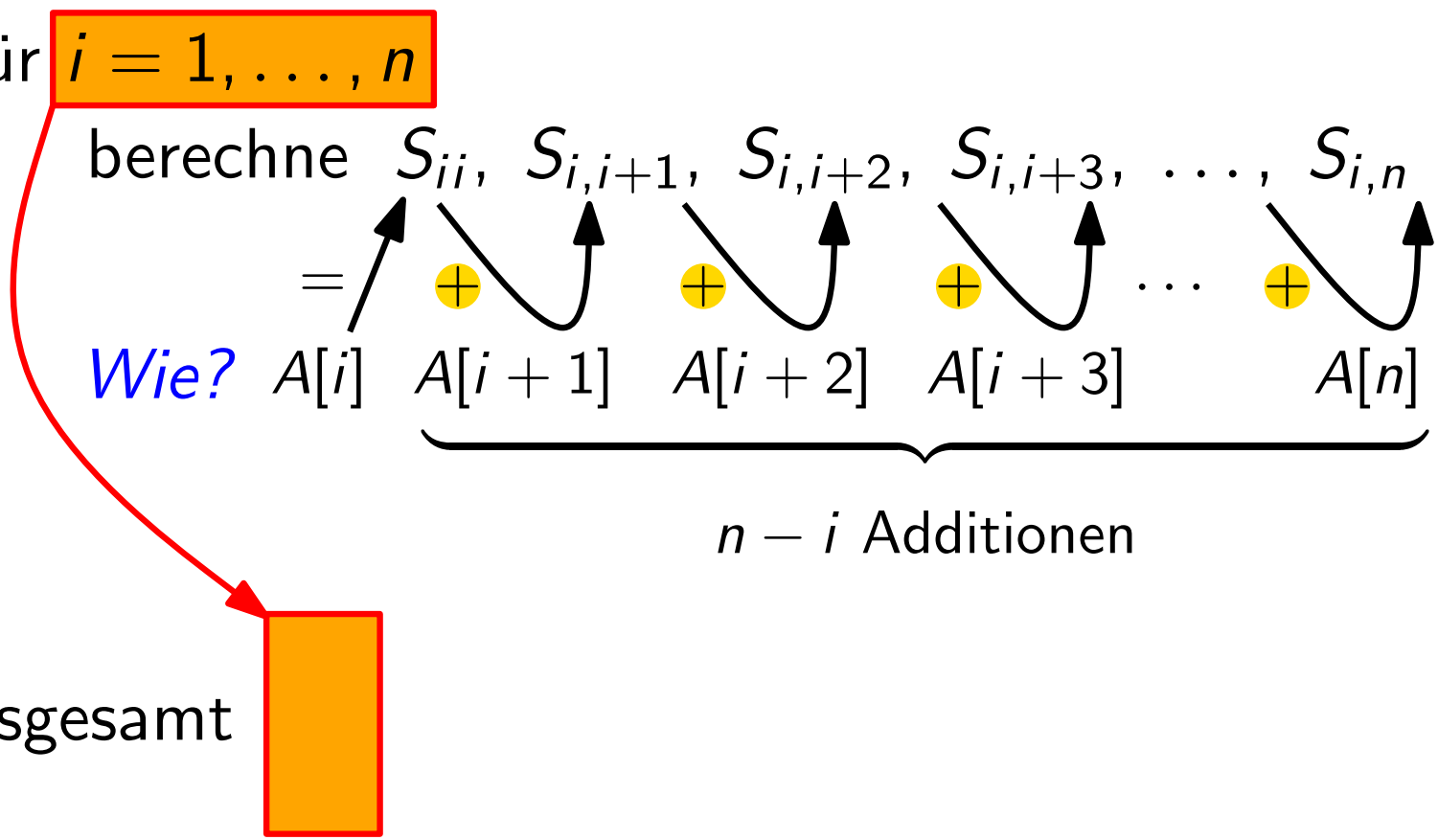


Insgesamt

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.

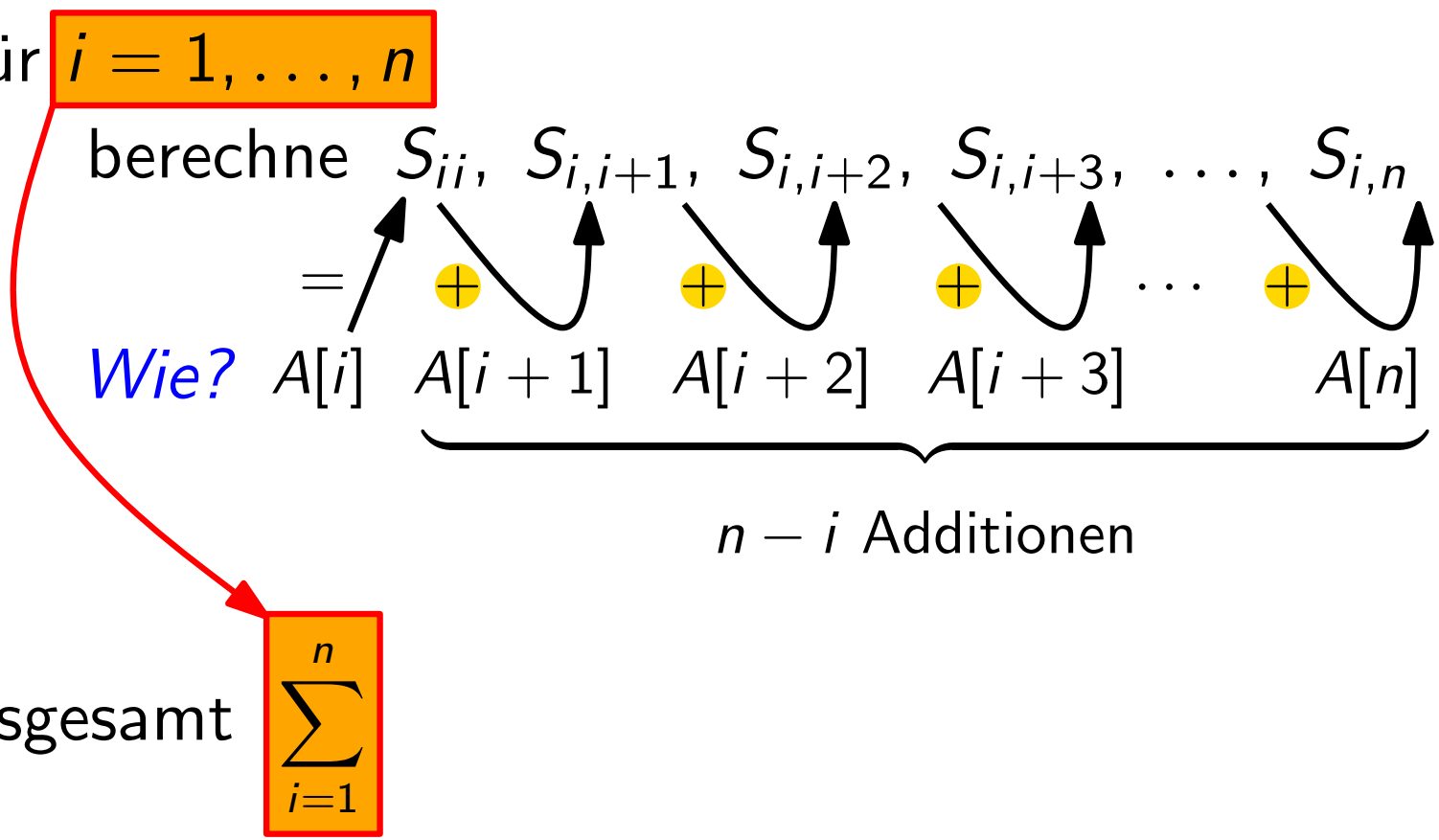
Idee: Für $i = 1, \ldots, n$



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.

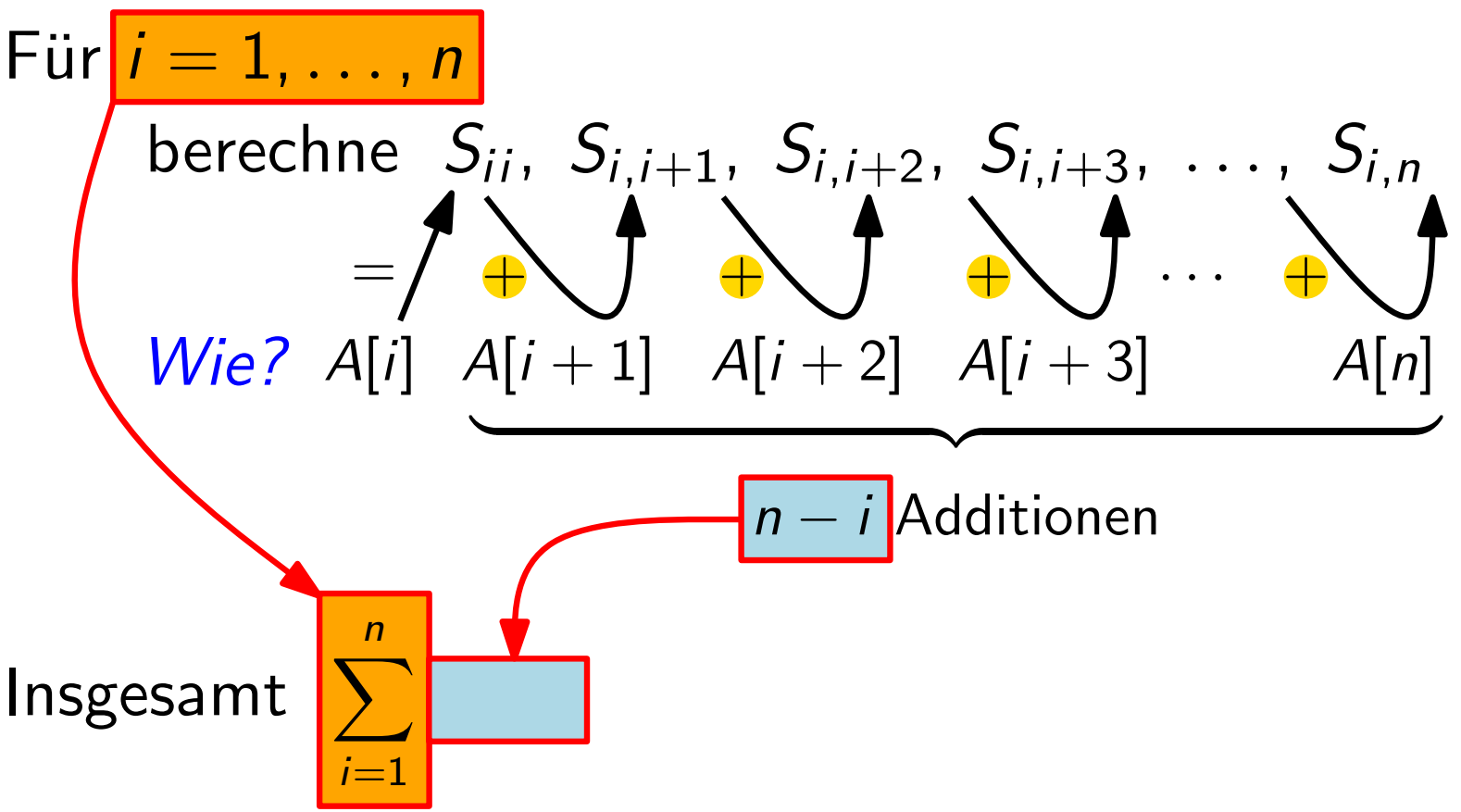
Idee: Für $i = 1, \ldots, n$



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.

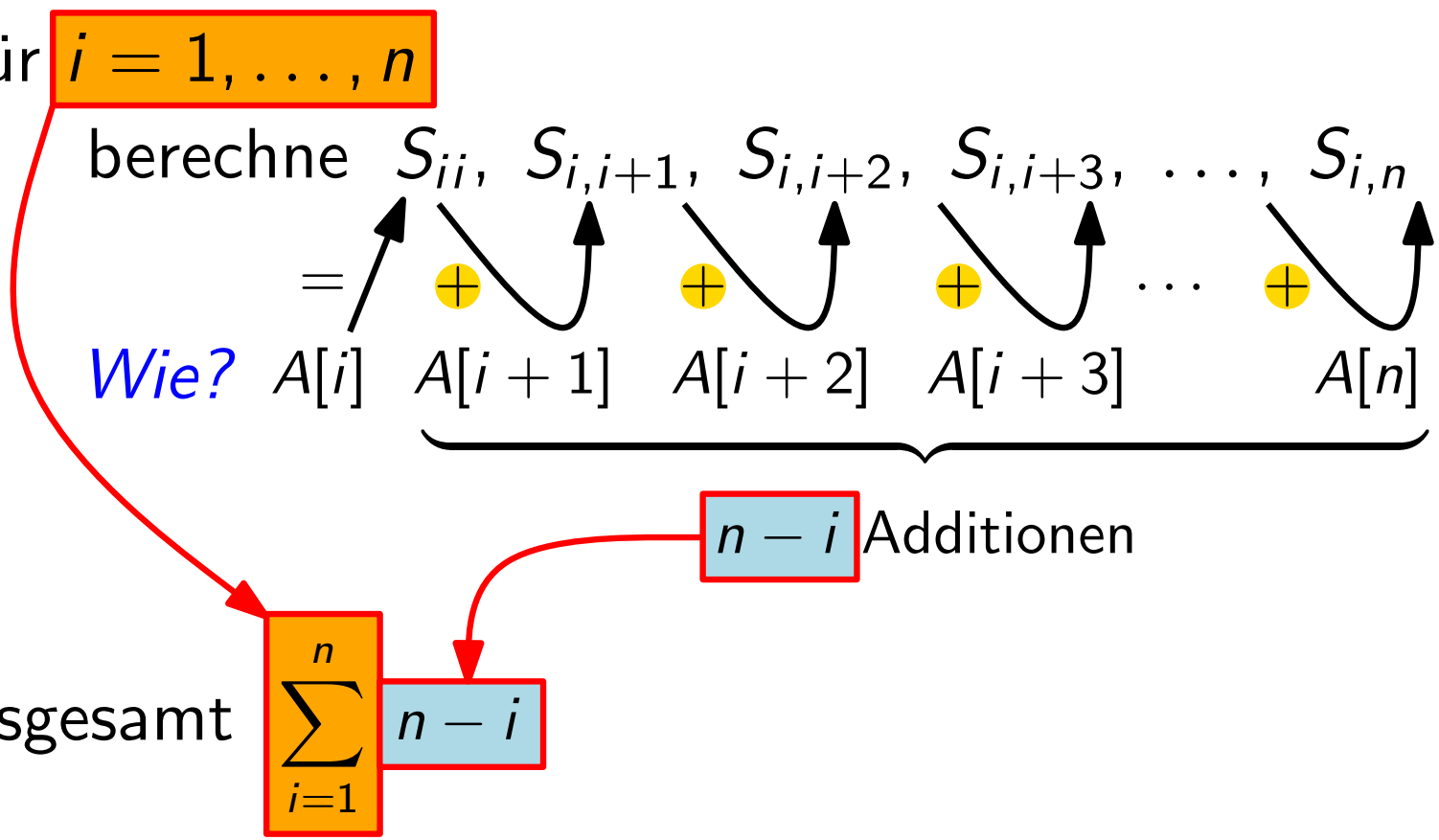
Idee:



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.

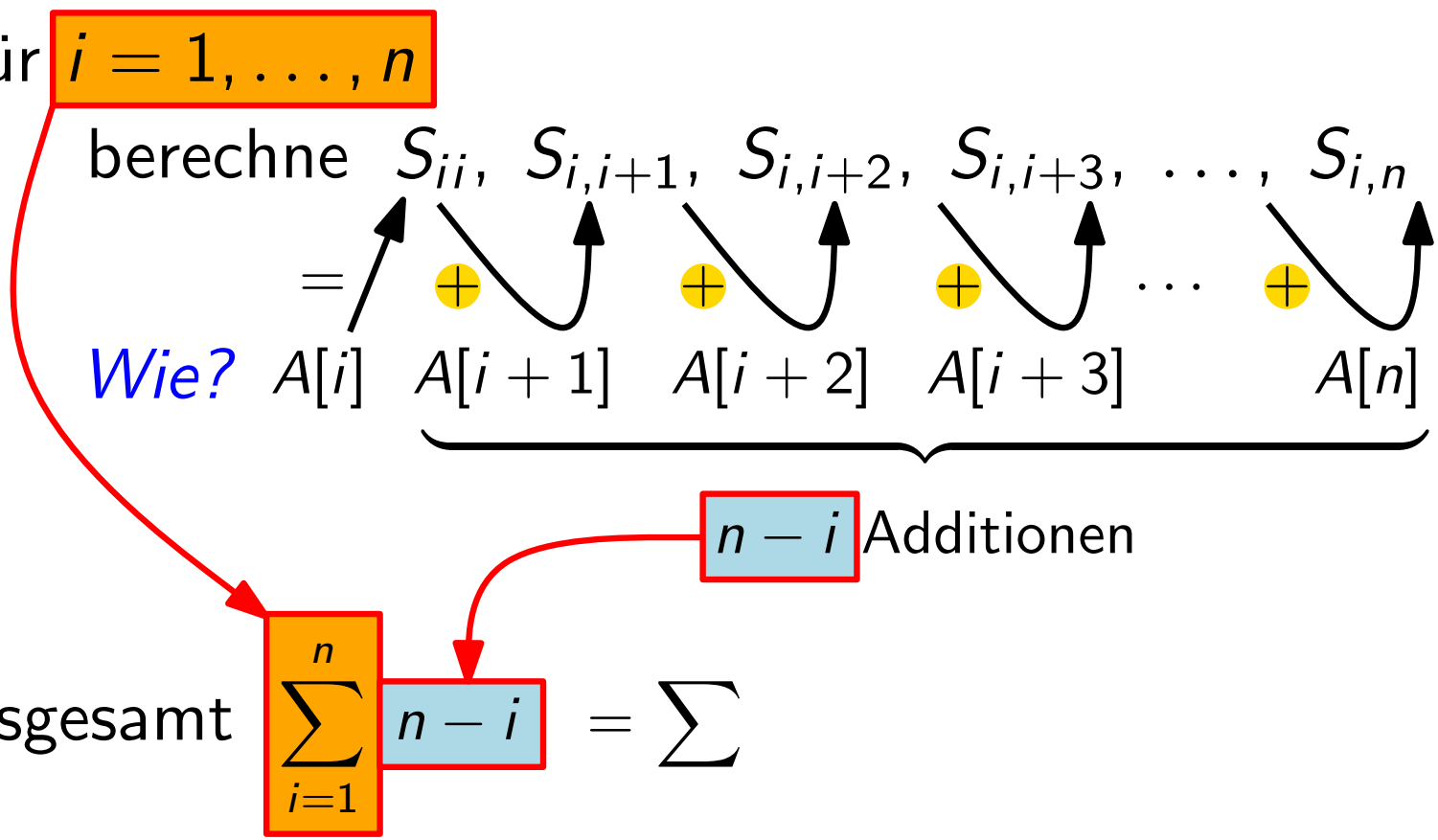
Idee: Für $i = 1, \ldots, n$



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 $i = 1, \ldots, n$



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 $i = 1, \ldots, n$

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

$=$ $A[i]$ $+$ $A[i+1]$ $+$ $A[i+2]$ $+$ $A[i+3]$ $\ldots$ $+$ $A[n]$

Wie?

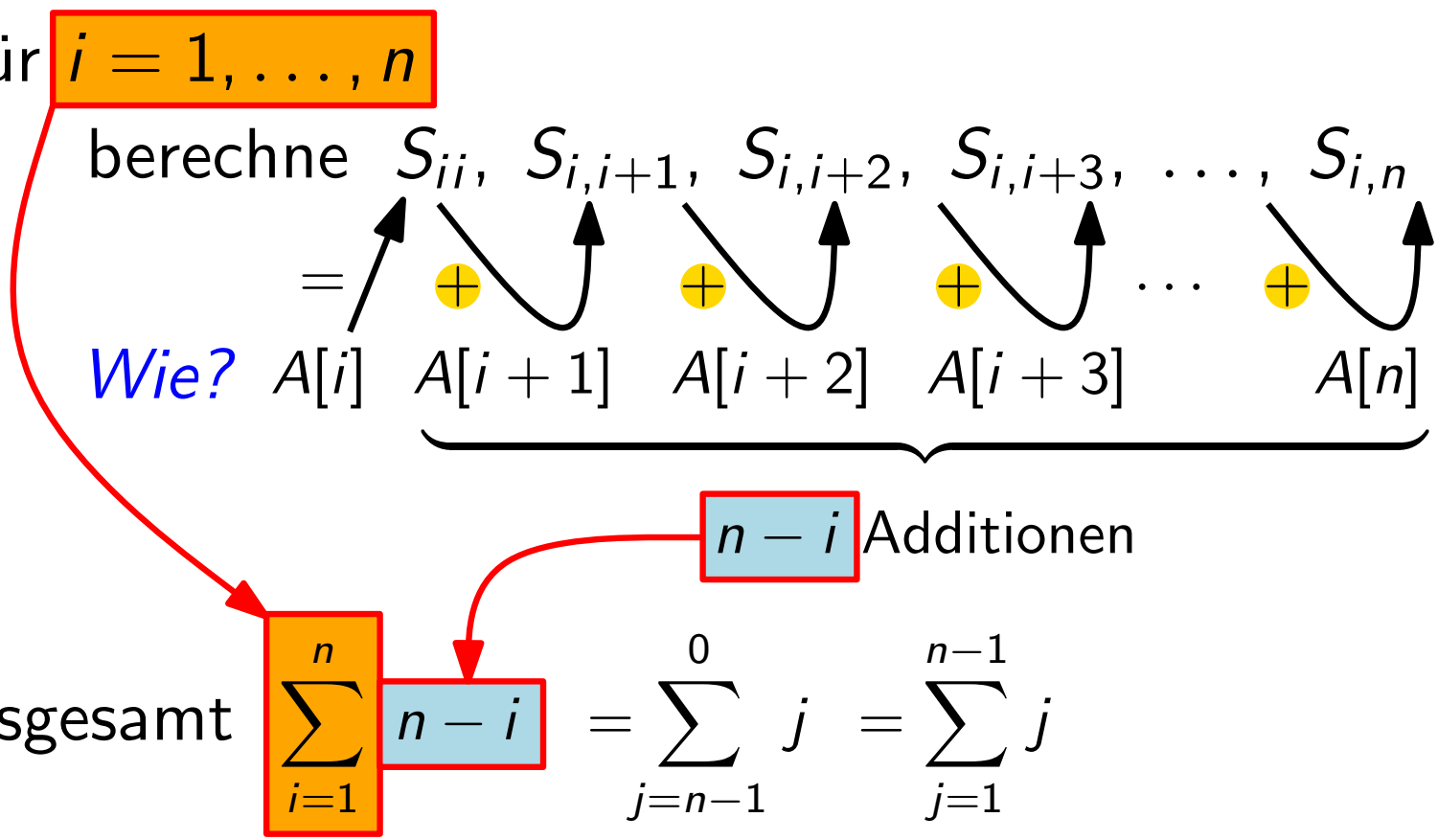
$n - i$ Additionen

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

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.

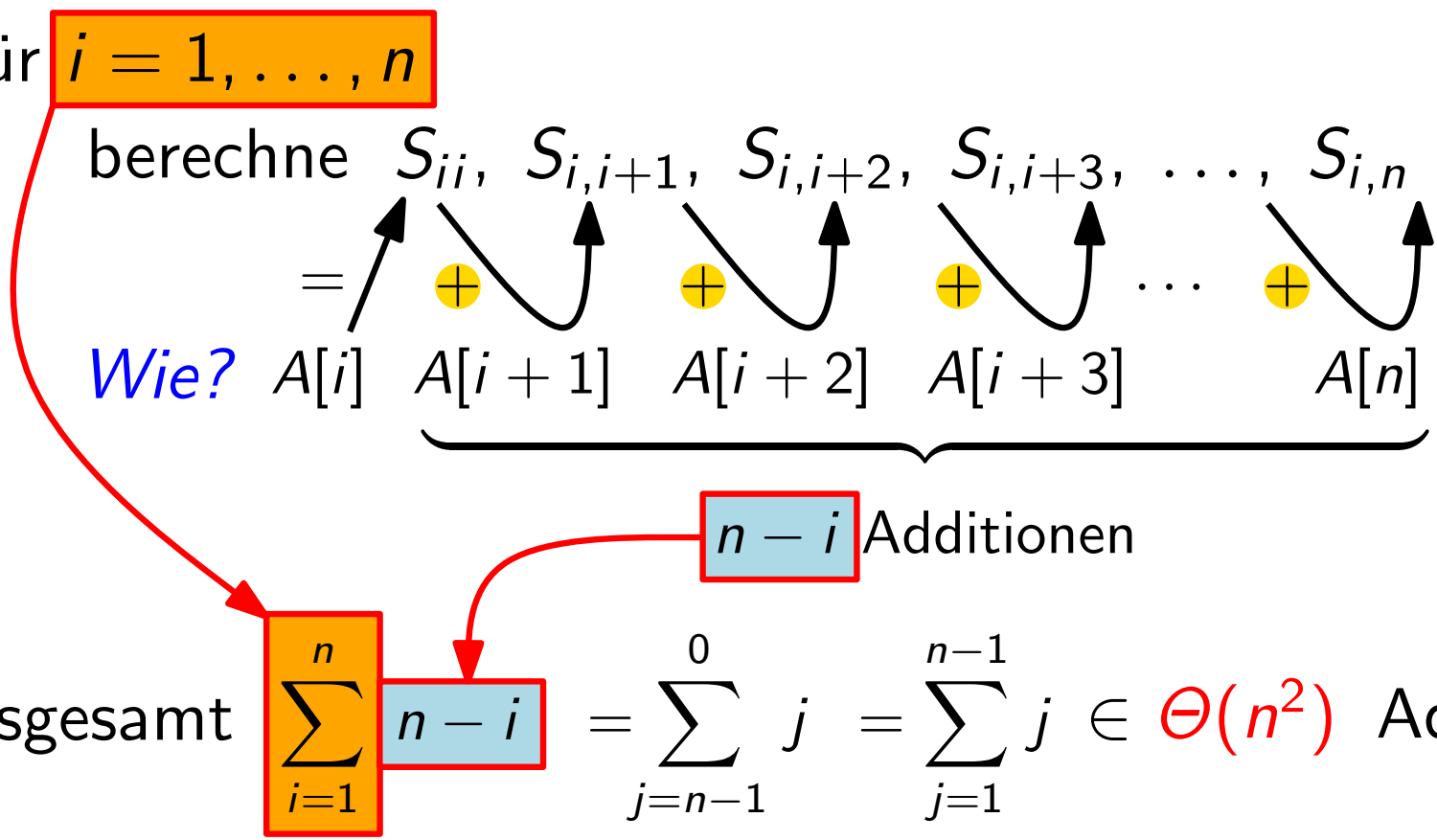
Idee: Für $i = 1, \ldots, n$



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 $i = 1, \ldots, n$

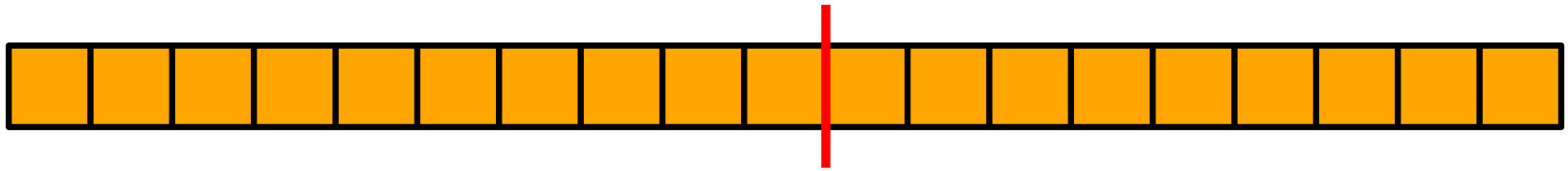


Eine noch schnellere Lösung?

Idee:

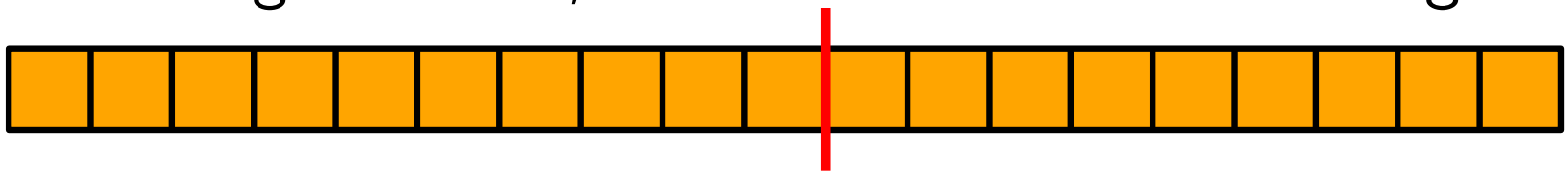
Eine noch schnellere Lösung?

Idee:



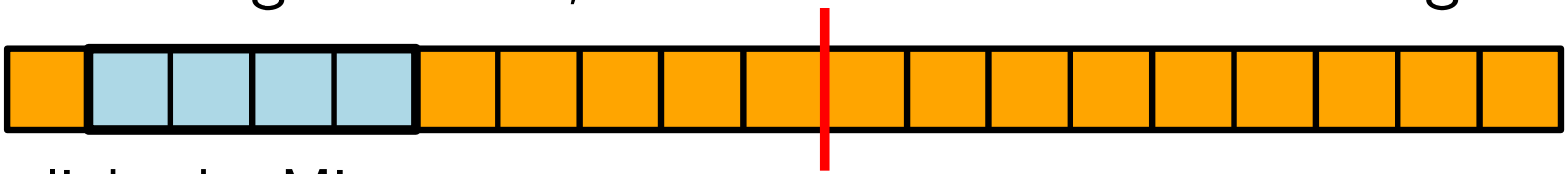
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

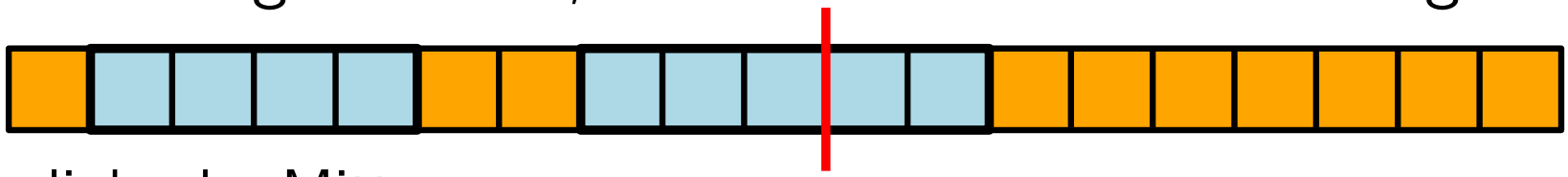
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



links der Mitte

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

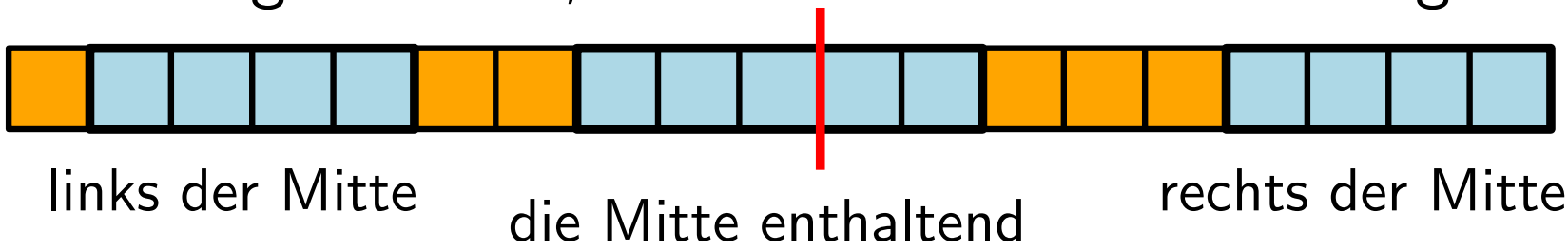


links der Mitte

die Mitte enthaltend

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



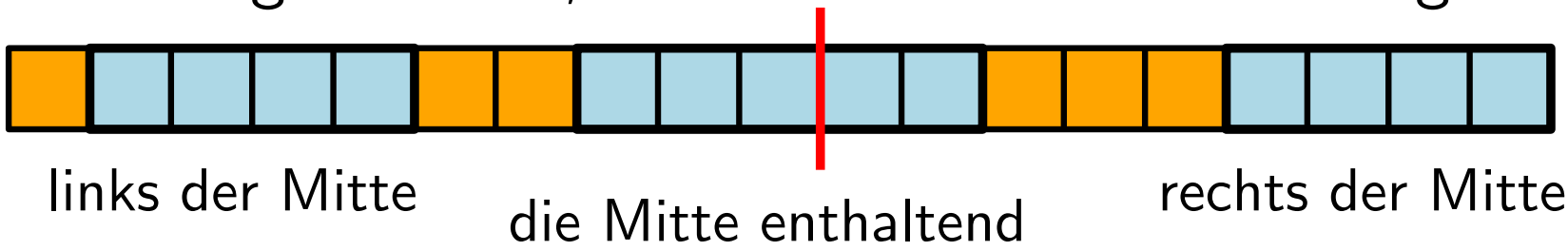
links der Mitte

die Mitte enthaltend

rechts der Mitte

Eine noch schnellere Lösung?

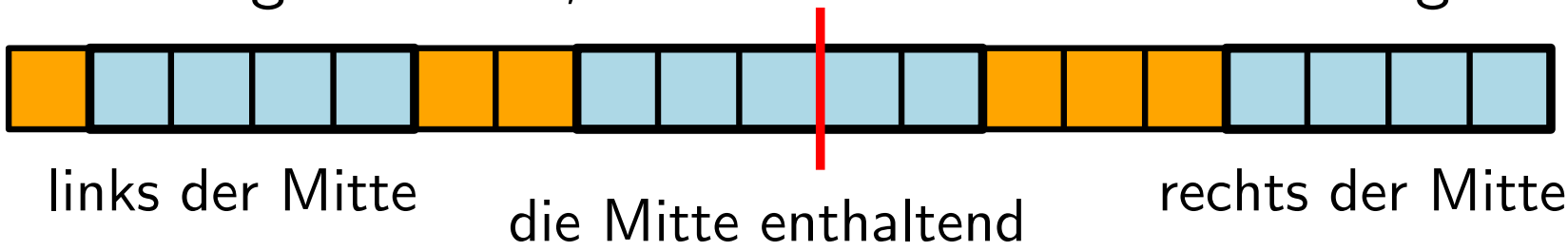
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik *Teile & Herrsche*!

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

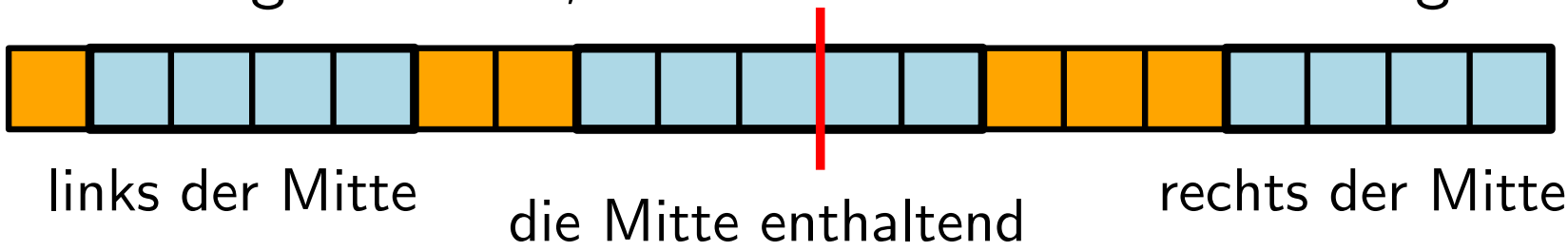


Nimm Entwurfstechnik *Teile & Herrsche*!

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

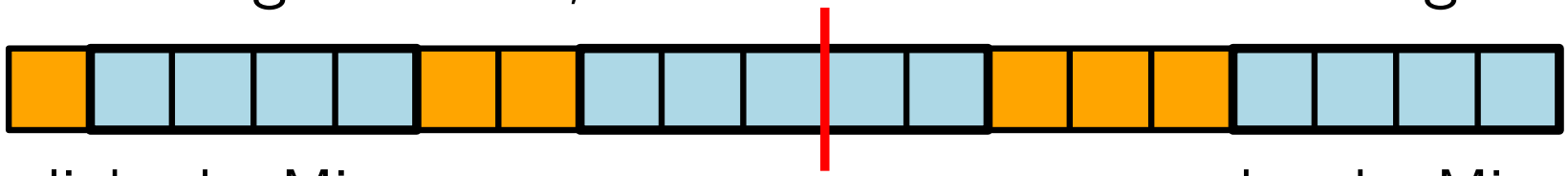


Nimm Entwurfstechnik *Teile & Herrsche*!

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



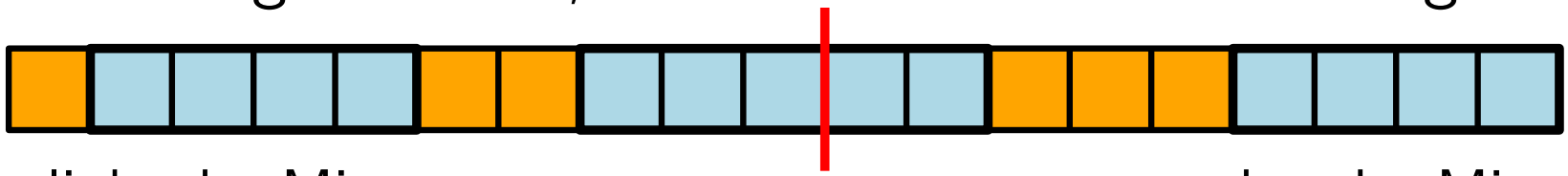
links der Mitte — die Mitte enthaltend — rechts der Mitte

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:



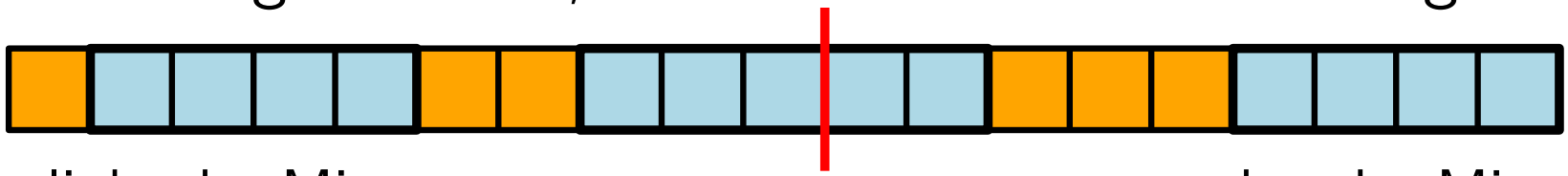
links der Mitte — die Mitte enthaltend — rechts der Mitte

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:



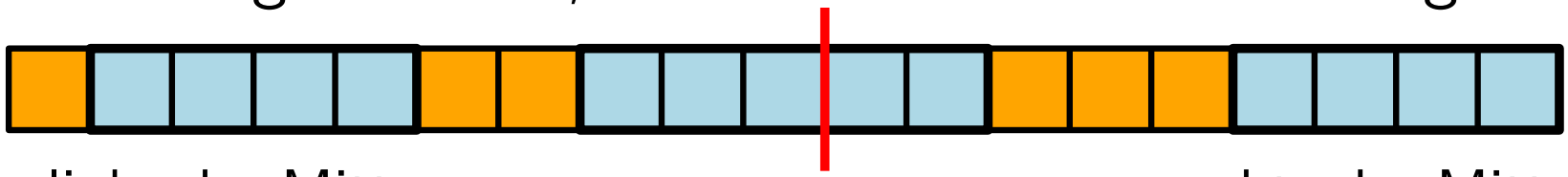
links der Mitte | die Mitte enthaltend | rechts der Mitte

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:



links der Mitte | die Mitte enthaltend | rechts der Mitte

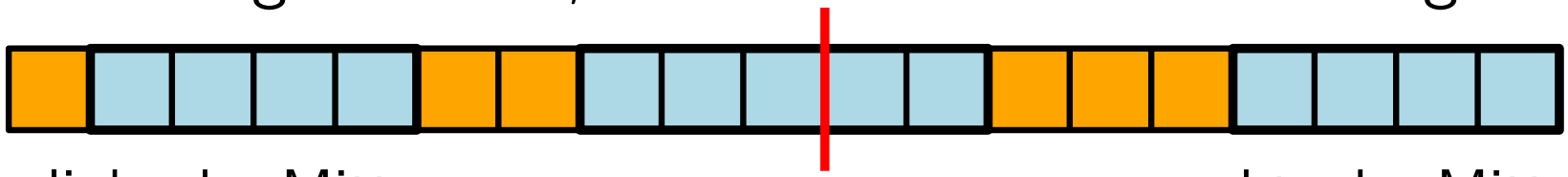
Nimm Entwurfstechnik *Teile & Herrsche*!

- *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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

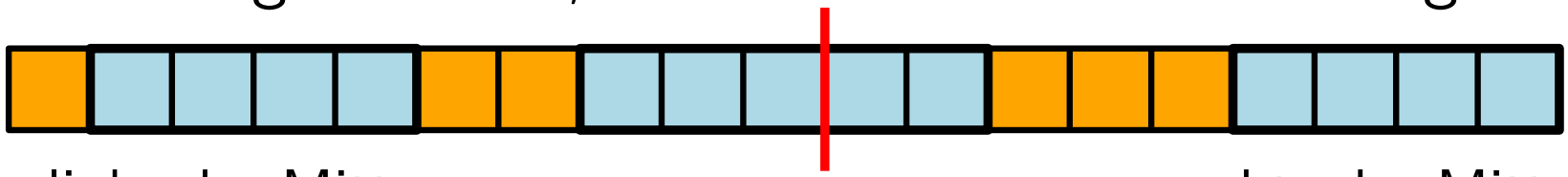
Nimm Entwurfstechnik *Teile & Herrsche*!

- *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:



links der Mitte | die Mitte enthaltend | rechts der Mitte

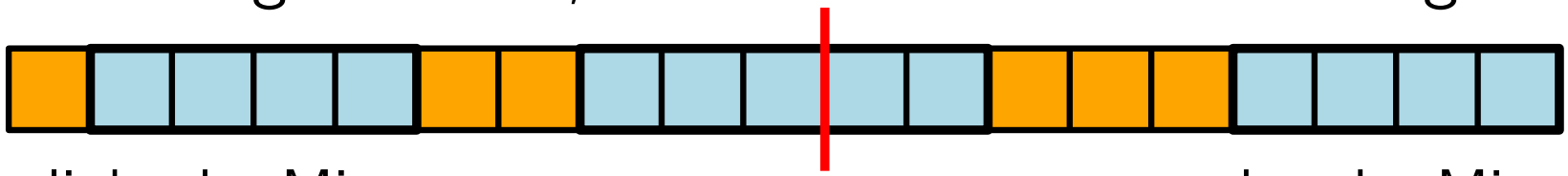
Nimm Entwurfstechnik *Teile & Herrsche*!

- *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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

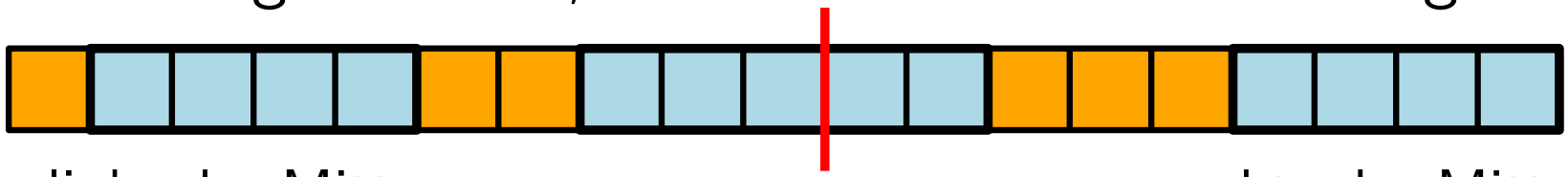
Nimm Entwurfstechnik *Teile & Herrsche*!

- *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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

Nimm Entwurfstechnik *Teile & Herrsche*!

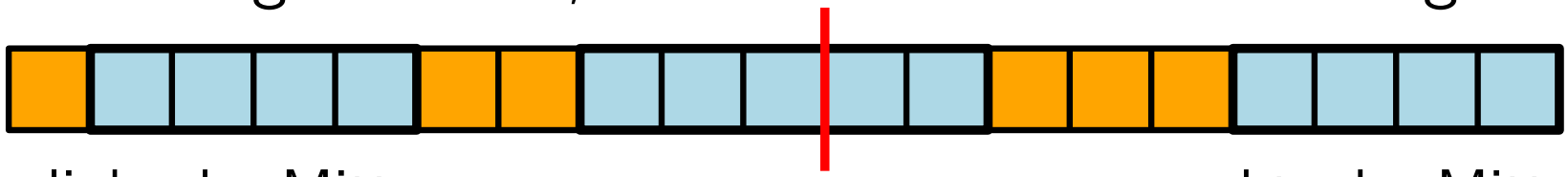
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere alle Teilsummen, die die Mitte enthalten

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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

Nimm Entwurfstechnik *Teile & Herrsche*!

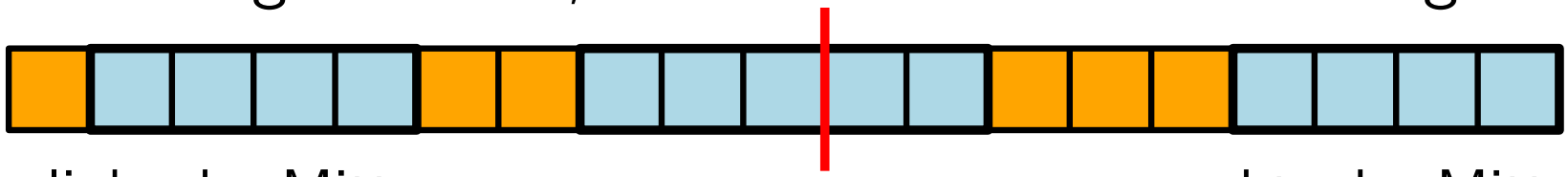
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere alle Teilsummen, die die Mitte enthalten

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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

Nimm Entwurfstechnik *Teile & Herrsche*!

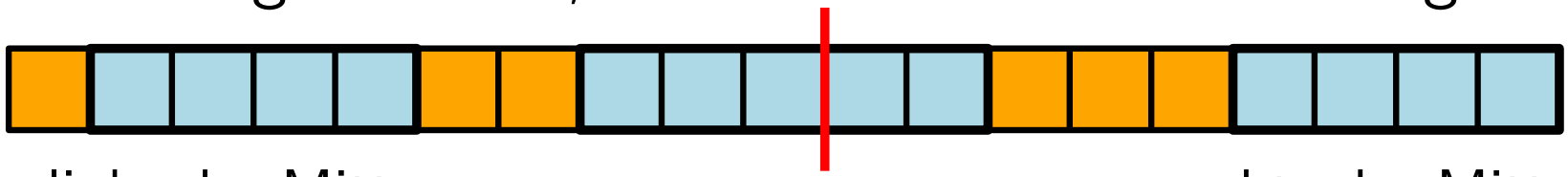
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere alle Teilsummen, die die Mitte enthalten

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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

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

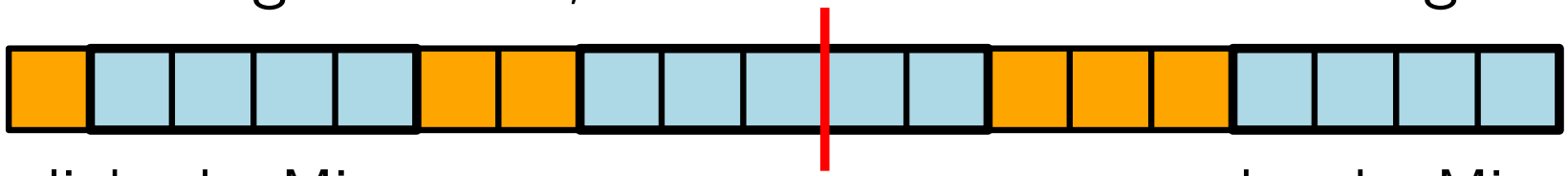
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$

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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

Nimm Entwurfstechnik *Teile & Herrsche*!

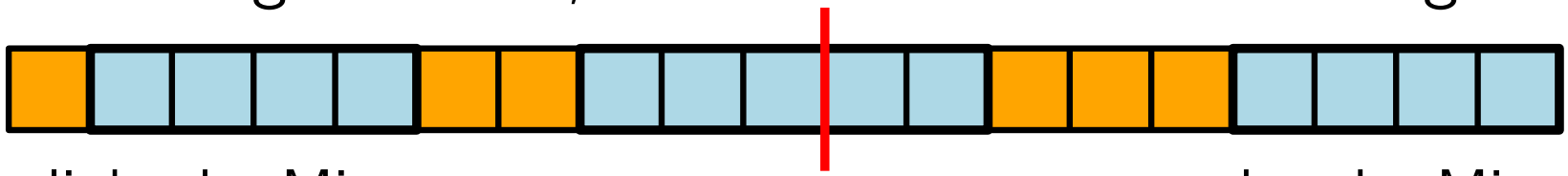
- *teile:* in zwei ungefähr gleichgroße Hälften
- *herrsche:* durch rekursive Aufrufe für li. u. re. Hälfte
- *kombiniere:* kontrolliere alle Teilsummen, die die Mitte enthalten

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:



links der Mitte — die Mitte enthaltend — rechts der Mitte

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])
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
  $mitte = \lfloor (beginn + ende)/2 \rfloor$
  $(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)
  $(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)
  $(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ =
    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
  $mitte = \lfloor (beginn + ende)/2 \rfloor$ — *teile*
  $(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)
  $(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)
  $(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ =
    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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ =
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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ =
MaxMittleresTeilfeld(A, $beginn$, $mitte$, $ende$)

kombiniere

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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ =

MaxMittleresTeilfeld(A, $beginn$, $mitte$, $ende$)

kombiniere

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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe) = \text{MaxTeilfeld}(A, beginn, mitte)$

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe) = \text{MaxTeilfeld}(A, mitte + 1, ende)$

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe) =$
$\text{MaxMittleresTeilfeld}(A, beginn, mitte, ende)$

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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe) =$ MaxTeilfeld$(A, beginn, mitte)$

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe) =$ MaxTeilfeld$(A, mitte + 1, ende)$

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe) =$
MaxMittleresTeilfeld$(A, beginn, mitte, ende)$

kombiniere

return (Tripel mit größter Summe)

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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe)$ = MaxTeilfeld(A, $beginn$, $mitte$)

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe)$ = MaxTeilfeld(A, $mitte + 1$, $ende$)

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe)$ = 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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe) = \text{MaxTeilfeld}(A, beginn, mitte)$

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe) = \text{MaxTeilfeld}(A, mitte + 1, ende)$

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe) =$
$\text{MaxMittleresTeilfeld}(A, beginn, mitte, ende)$

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

teile

$mitte = \lfloor (beginn + ende)/2 \rfloor$

herrsche

$(L\text{-}beginn, L\text{-}ende, L\text{-}summe) = \text{MaxTeilfeld}(A, beginn, mitte)$

$(R\text{-}beginn, R\text{-}ende, R\text{-}summe) = \text{MaxTeilfeld}(A, mitte + 1, ende)$

$(M\text{-}beginn, M\text{-}ende, M\text{-}summe) =$
$\text{MaxMittleresTeilfeld}(A, beginn, mitte, ende)$

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 $ende$)

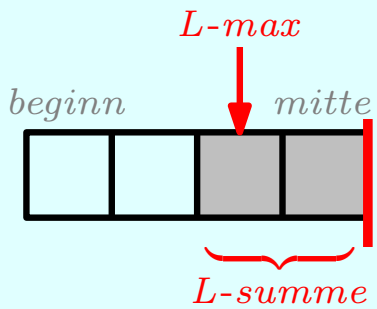
Kombiniere

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

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

$summe = 0$

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



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

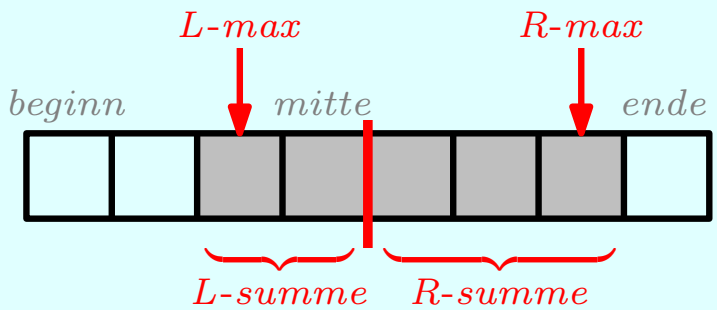
Kombiniere

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

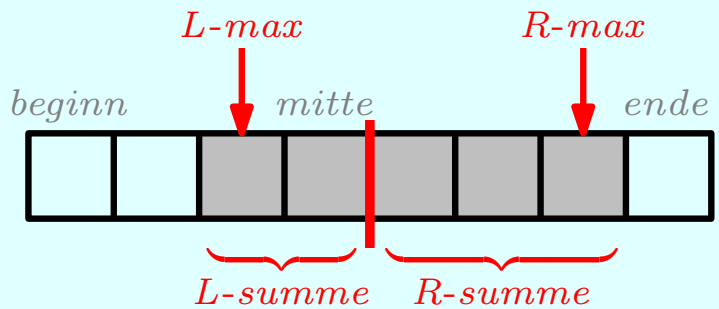
MaxMittleresTeilfeld(int[] A, int $beginn$, int $mitte$, int $ende$)

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

$summe = 0$

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

Vervollständigen Sie den Algorithmus!



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

$summe = 0$

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

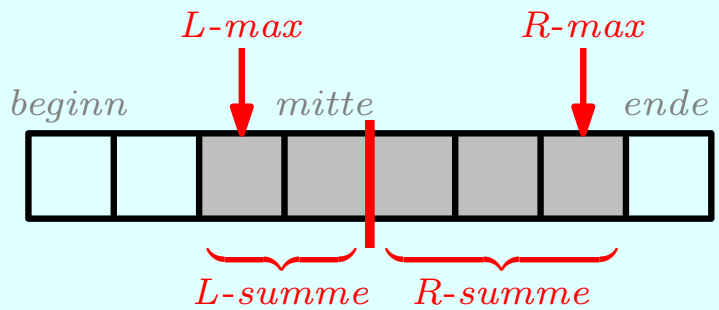
// analog zu oben

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 $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
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:

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

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

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

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

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

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

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

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

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

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

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**
  $summe = summe$ ⊕ $A[i]$
  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

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

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**
  $summe = summe + A[i]$
  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

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

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**
  $summe = summe + A[i]$
  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

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

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**
  $summe = summe + A[i]$
  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

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

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

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

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

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

Laufzeit?

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

$$\frac{\begin{array}{l} mitte - beginn + 1 \\ ende - mitte \end{array}}{ende - beginn + 1}$$

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

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

Laufzeit?

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

$$\frac{\overbrace{mitte - beginn + 1}^{ende - mitte}}{ende - beginn + 1} = n$$

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

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

Laufzeit? ✓

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

$$\frac{\begin{array}{c} mitte - beginn + 1 \\ ende - mitte \end{array}}{ende - beginn + 1} = n$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

$$\begin{aligned} T_{\mathsf{MT}}(n) &\approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n) \\ &= 2 \cdot T_{\mathsf{MT}}(n/2) + n \end{aligned}$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

$$\begin{aligned} T_{\mathsf{MT}}(n) &\approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n) \\ &= 2 \cdot T_{\mathsf{MT}}(n/2) + n \\ &= V_{\mathsf{MS}}(n) \end{aligned}$$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

$$\begin{aligned} T_{\mathsf{MT}}(n) &\approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n) \\ &= 2 \cdot T_{\mathsf{MT}}(n/2) + n \\ &= V_{\mathsf{MS}}(n) = n \log_2 n \end{aligned}$$

[für n = Zweierpotenz]

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

$$\begin{aligned} T_{\mathsf{MT}}(n) &\approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n) \\ &= 2 \cdot T_{\mathsf{MT}}(n/2) + n \\ &= V_{\mathsf{MS}}(n) = O(n \log_2 n) \end{aligned}$$

~~[für n = Zweierpotenz]~~

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

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

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

$$= V_{\mathsf{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch...

~~[für n = Zweierpotenz]~~

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

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

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

$$= V_{\mathsf{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch...

~~[für n = Zweierpotenz]~~

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$: $$T_{\mathsf{MT}}(n) \approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n)$$
$$= 2 \cdot T_{\mathsf{MT}}(n/2) + n$$
$$= V_{\mathsf{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch. . .

~~[für n = Zweierpotenz]~~

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:

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

für $n > 1$: $$T_{\mathsf{MT}}(n) \approx 2 \cdot T_{\mathsf{MT}}(n/2) + T_{\mathsf{MMT}}(n)$$
$$= 2 \cdot T_{\mathsf{MT}}(n/2) + n$$
$$= V_{\mathsf{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch. . .

~~[für n = Zweierpotenz]~~

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:

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

für $n > 1$: $$T_{\mathrm{MT}}(n) \approx 2 \cdot T_{\mathrm{MT}}(n/2) + T_{\mathrm{MMT}}(n)$$
$$= 2 \cdot T_{\mathrm{MT}}(n/2) + n$$
$$= V_{\mathrm{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch...

~~[für n = Zweierpotenz]~~

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:

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

für $n > 1$: $$T_{\mathrm{MT}}(n) \approx 2 \cdot T_{\mathrm{MT}}(n/2) + T_{\mathrm{MMT}}(n)$$
$$= 2 \cdot T_{\mathrm{MT}}(n/2) + n$$
$$= V_{\mathrm{MS}}(n) = O(n \log_2 n)$$

Warum die Einschränkung wegfällt, sehen wir noch...

~~[für n = Zweierpotenz]~~

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:

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

für $n > 1$: $$T_{\mathrm{MT}}(n) \approx 2 \cdot T_{\mathrm{MT}}(n/2) + T_{\mathrm{MMT}}(n)$$
$$= 2 \cdot T_{\mathrm{MT}}(n/2) + n$$
$$= V_{\mathrm{MS}}(n) = O(n \log_2 n)$$ ~~[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

Algorithmus

Rohe Gewalt

Reihenfolge der Summen ändern

Teile & Herrsche

Linearer Scan

(siehe Buch [CLRS?])

Übersicht

Algorithmus	*Laufzeit*
Rohe Gewalt	
Reihenfolge der Summen ändern	
Teile & Herrsche	
Linearer Scan (siehe Buch [CLRS?])	

Übersicht

Algorithmus	*Laufzeit*
Rohe Gewalt	$O(n^3)$
Reihenfolge der Summen ändern	
Teile & Herrsche	
Linearer Scan (siehe Buch [CLRS?])	

Übersicht

Algorithmus	*Laufzeit*
Rohe Gewalt	$O(n^3)$
Reihenfolge der Summen ändern	$O(n^2)$
Teile & Herrsche	
Linearer Scan (siehe Buch [CLRS?])	

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?])

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?])

Übersicht

– *Anzahl der Additionen*

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?])	$O(n)$

Übersicht

– *Anzahl der Additionen* *

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?])	$O(n)$

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log n)$	
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log n)$	
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
Rohe Gewalt	$O(n^3)$	
Reihenfolge der Summen ändern	$O(n^2)$	
Teile & Herrsche	$O(n \log_{10} n)$	
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
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 (siehe Buch [CLRS?])	$O(n)$	10^3

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
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?])	$O(n)$	10^3

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$
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?])	$O(n)$	10^3

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$	$n = 1\,000\,000$
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?])	$O(n)$	10^3	

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

Übersicht

– *Anzahl der Additionen* *

Algorithmus	*Laufzeit*	$n = 1\,000$	$n = 1\,000\,000$
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?])	$O(n)$	10^3	10^6

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

Übersicht

– *Anzahl der Additionen* *
– *geschätzte Rechenzeit*

Algorithmus	*Laufzeit*	$n = 1\,000$	$n = 1\,000\,000$
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?])	$O(n)$	10^3	10^6

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

Übersicht

– *Anzahl der Additionen* *
– *geschätzte Rechenzeit* **

Algorithmus	*Laufzeit*	$n = 1\,000$	$n = 1\,000\,000$
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?])	$O(n)$	10^3	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* **

Algorithmus	*Laufzeit*	$n = 1\,000$	$n = 1\,000\,000$
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?])	$O(n)$	10^3 $\quad 1\,\mu s$	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* **

Algorithmus	*Laufzeit*	$n = 1\,000$		$n = 1\,000\,000$
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 (siehe Buch [CLRS?])	$O(n)$	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* **

Algorithmus	*Laufzeit*	$n = 1\,000$		$n = 1\,000\,000$
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 μs	$6{\cdot}10^6$
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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* **

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 μs	$6{\cdot}10^6$
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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* **

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 μs	$6{\cdot}10^6$	
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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

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 μs	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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

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 μs	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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

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 μs	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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

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 μs	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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

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 μs	$6{\cdot}10^6$	6 ms
Linearer Scan (siehe Buch [CLRS?])	$O(n)$	10^3	1 μs	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