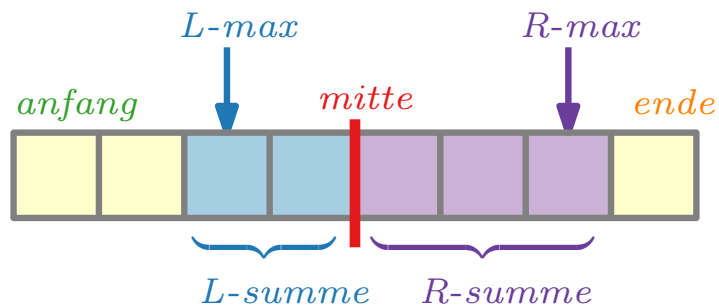




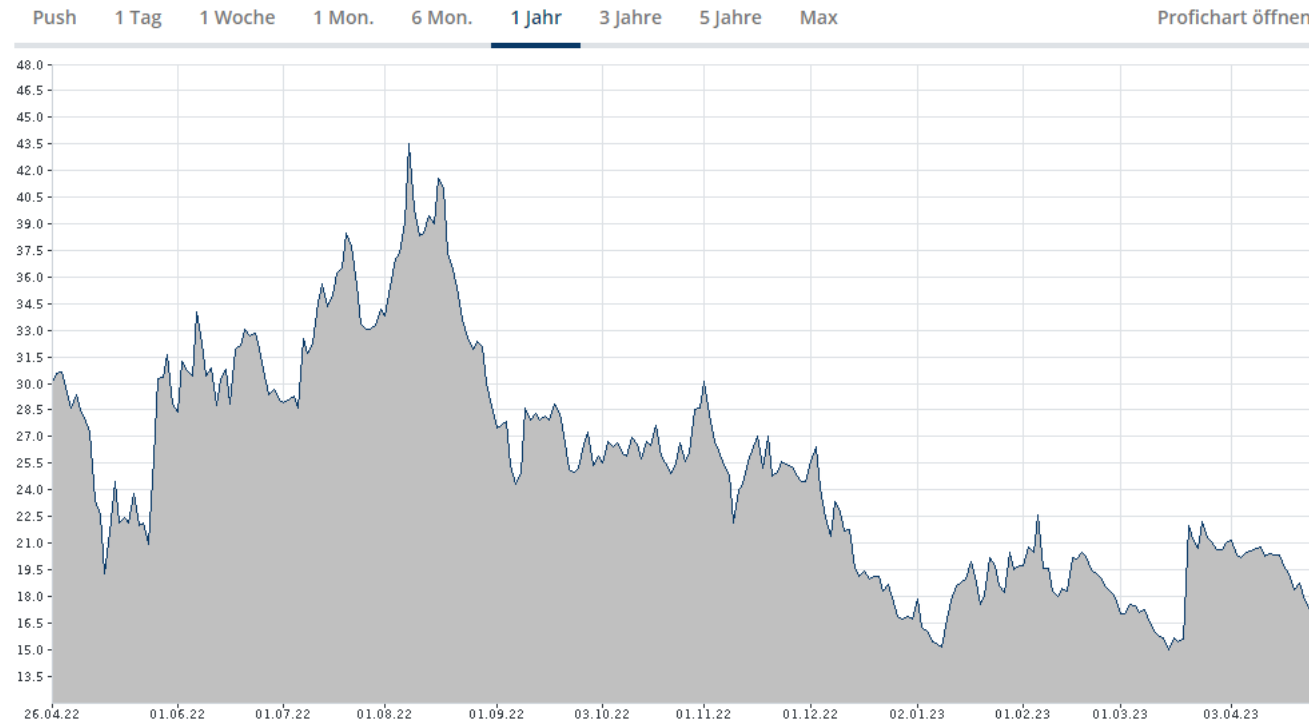
Algorithmen und Datenstrukturen

Vorlesung 4: Maximales Teilfeld



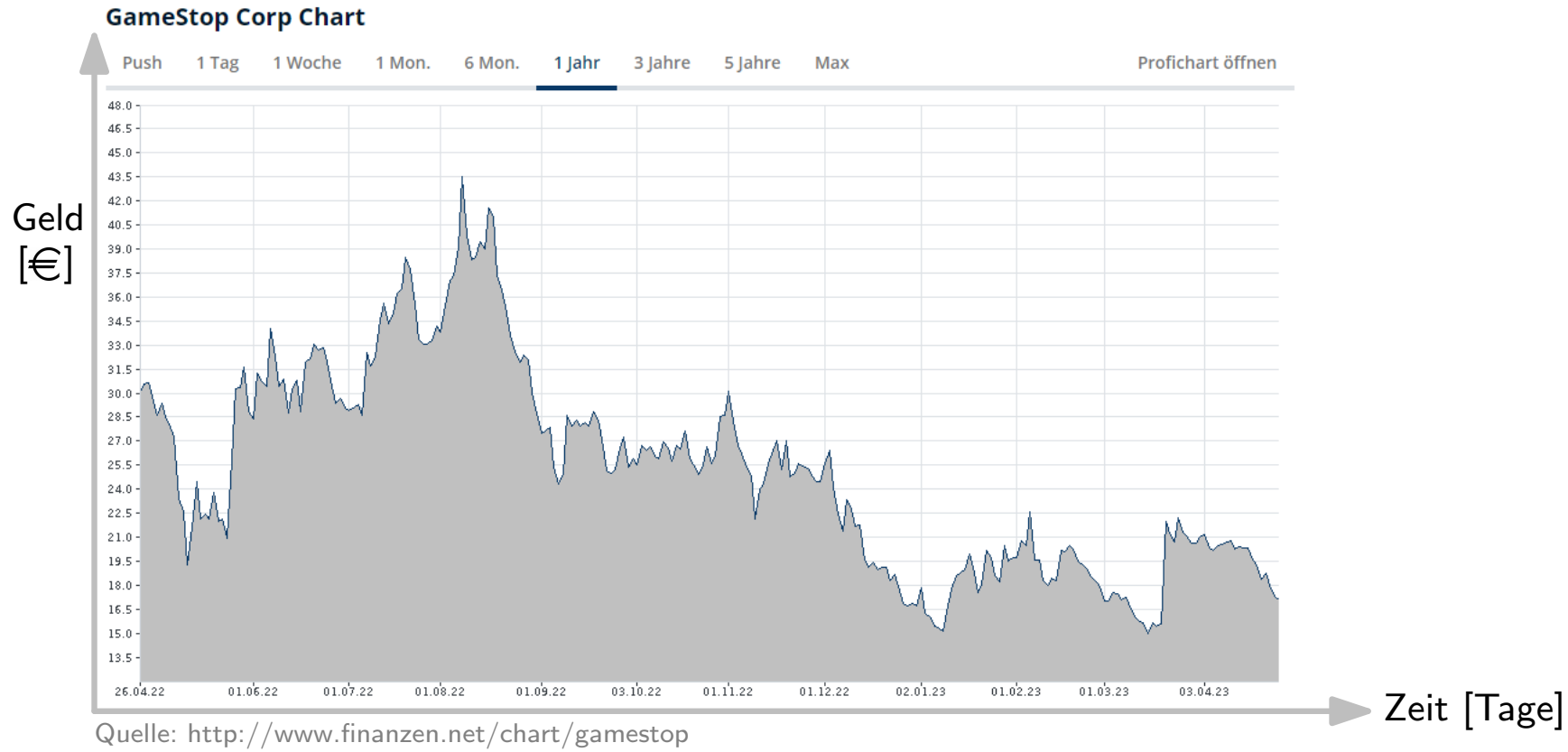
Analyse von Aktienkursen

GameStop Corp Chart

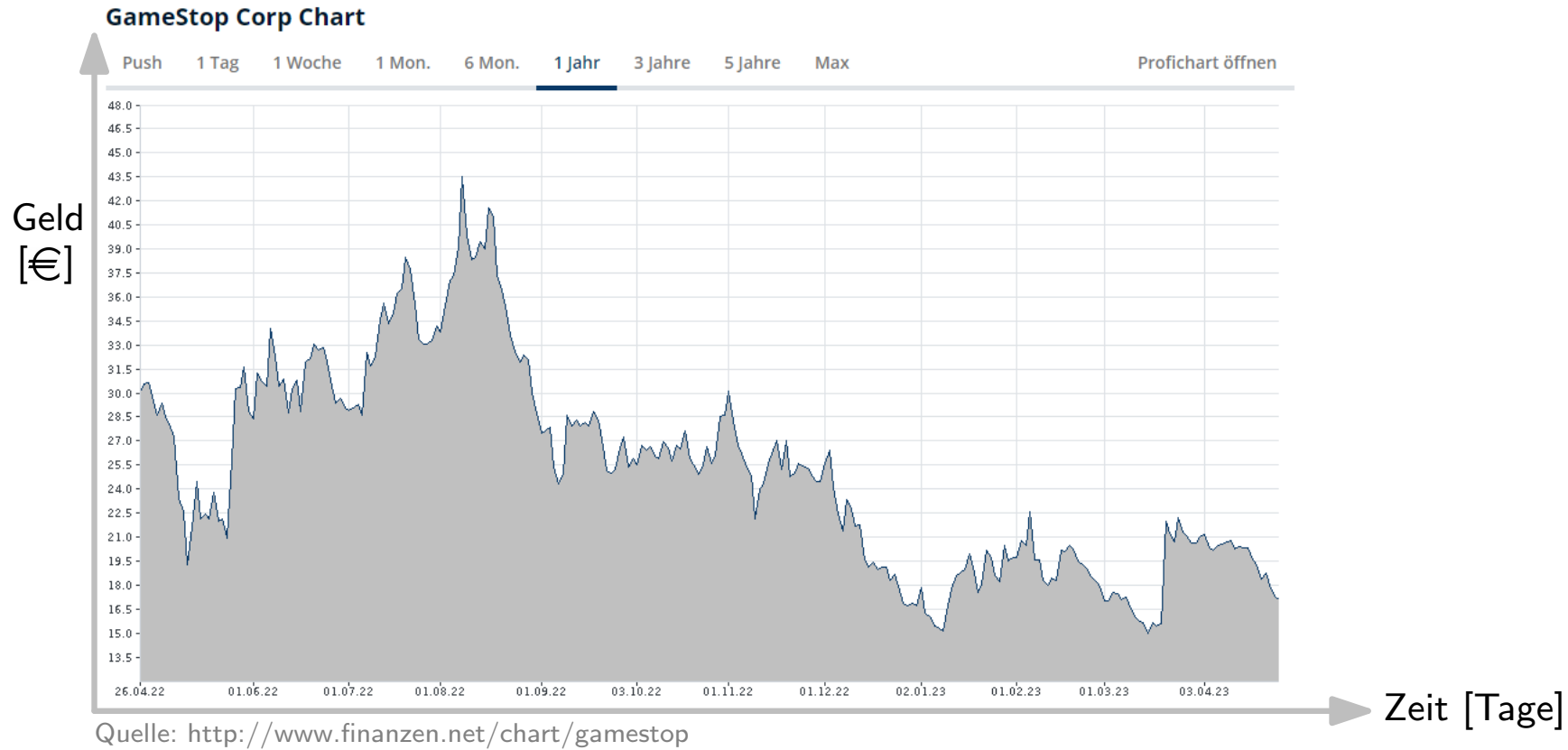


Quelle: <http://www.finanzen.net/chart/gamestop>

Analyse von Aktienkursen

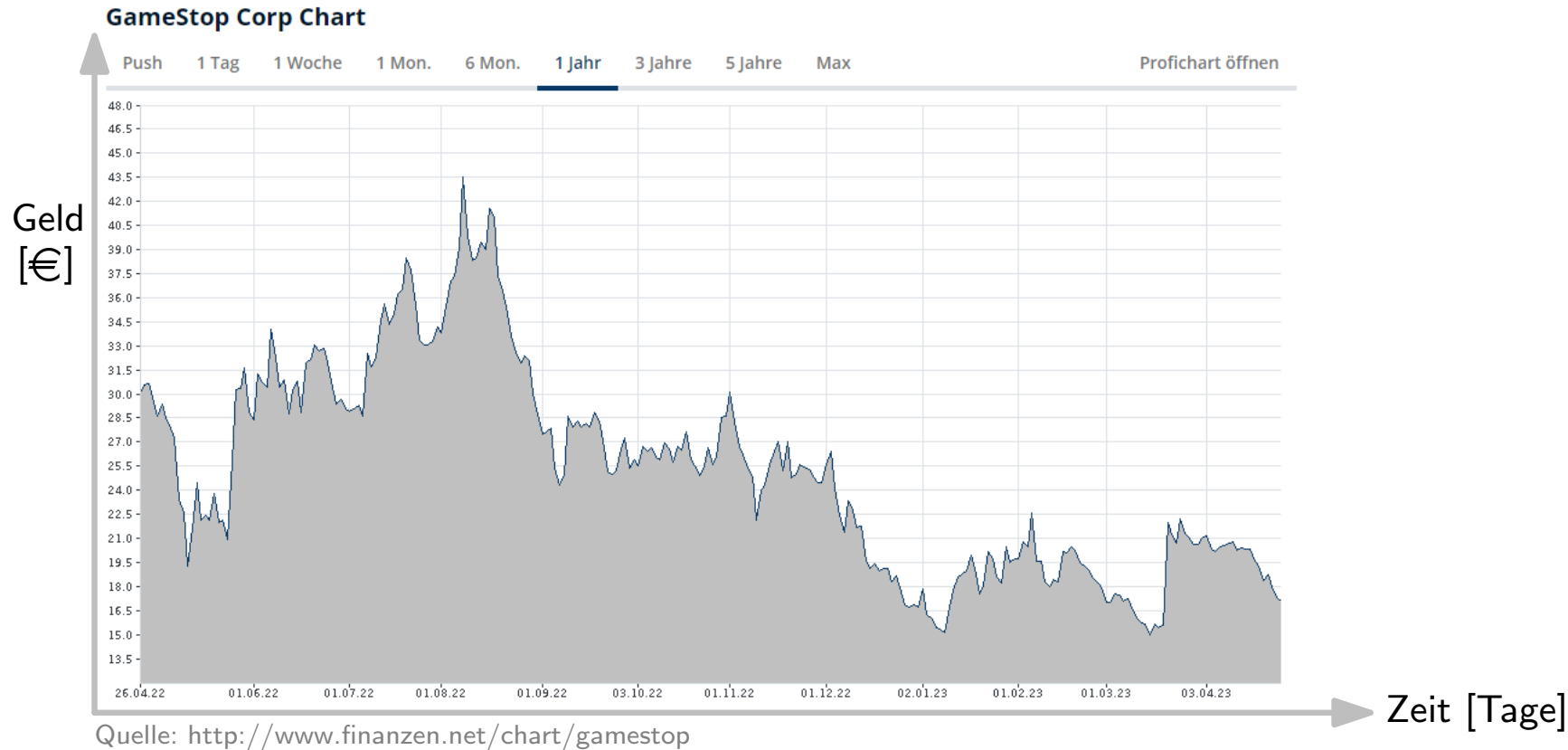


Analyse von Aktienkursen



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

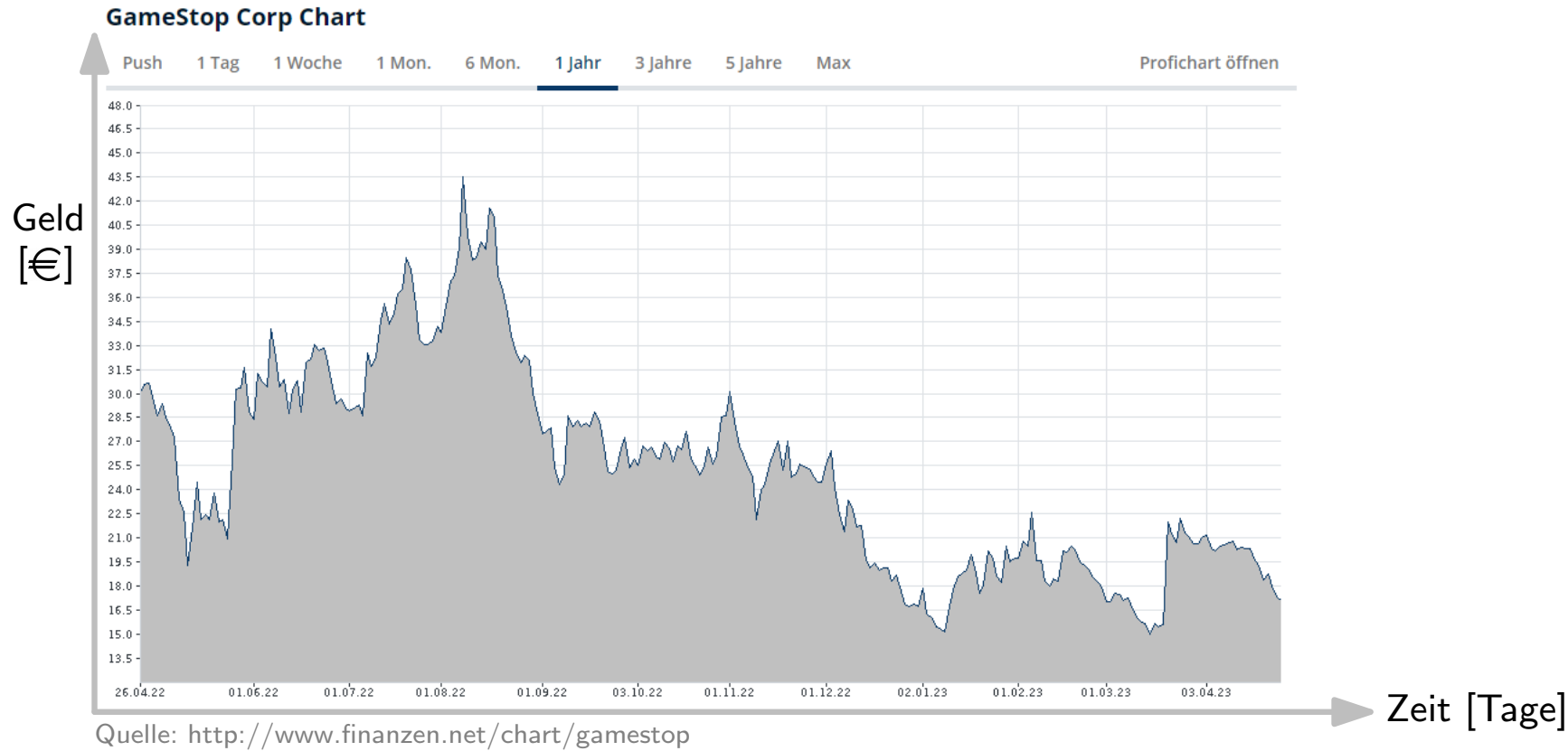
Analyse von Aktienkursen



Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

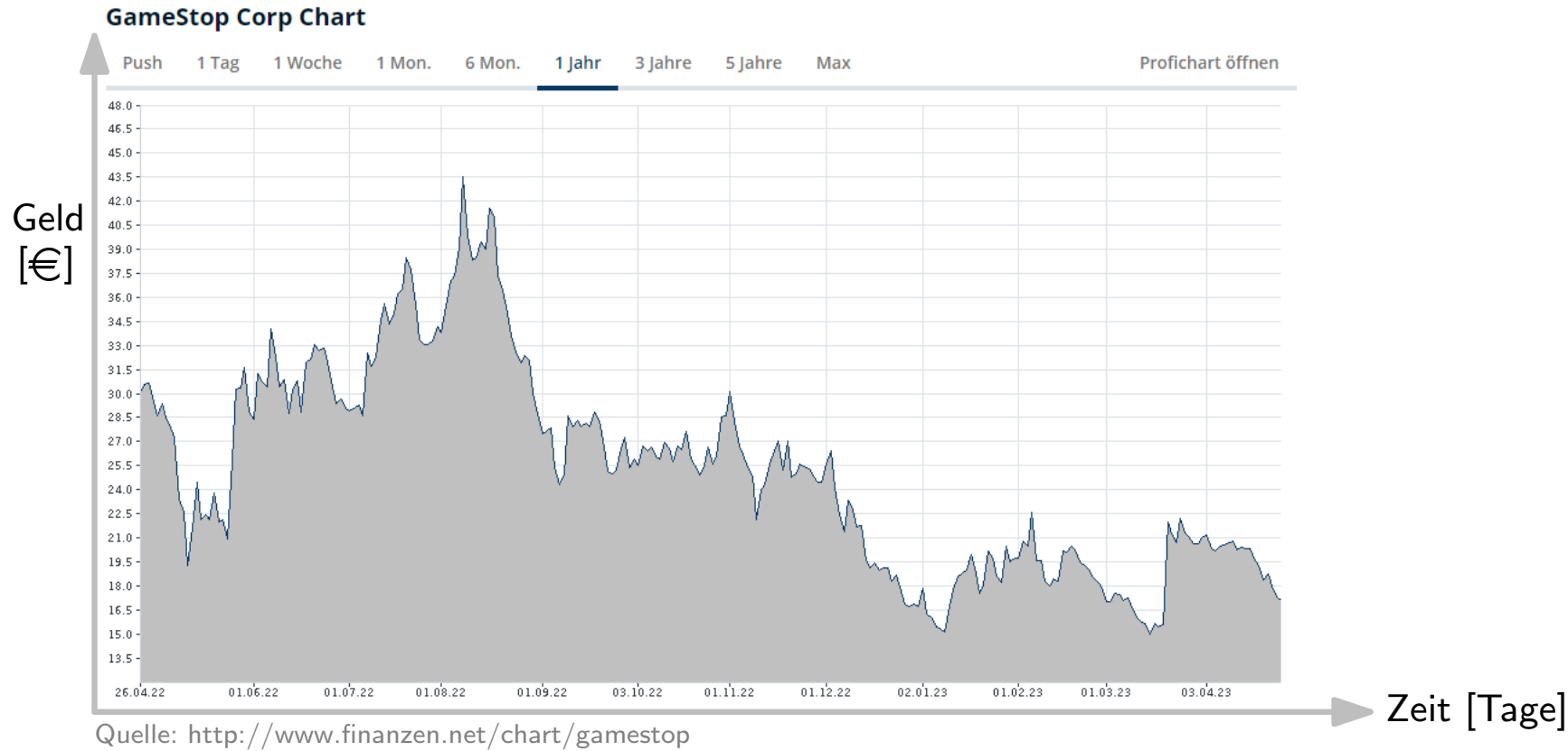
Analyse von Aktienkursen



Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ 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

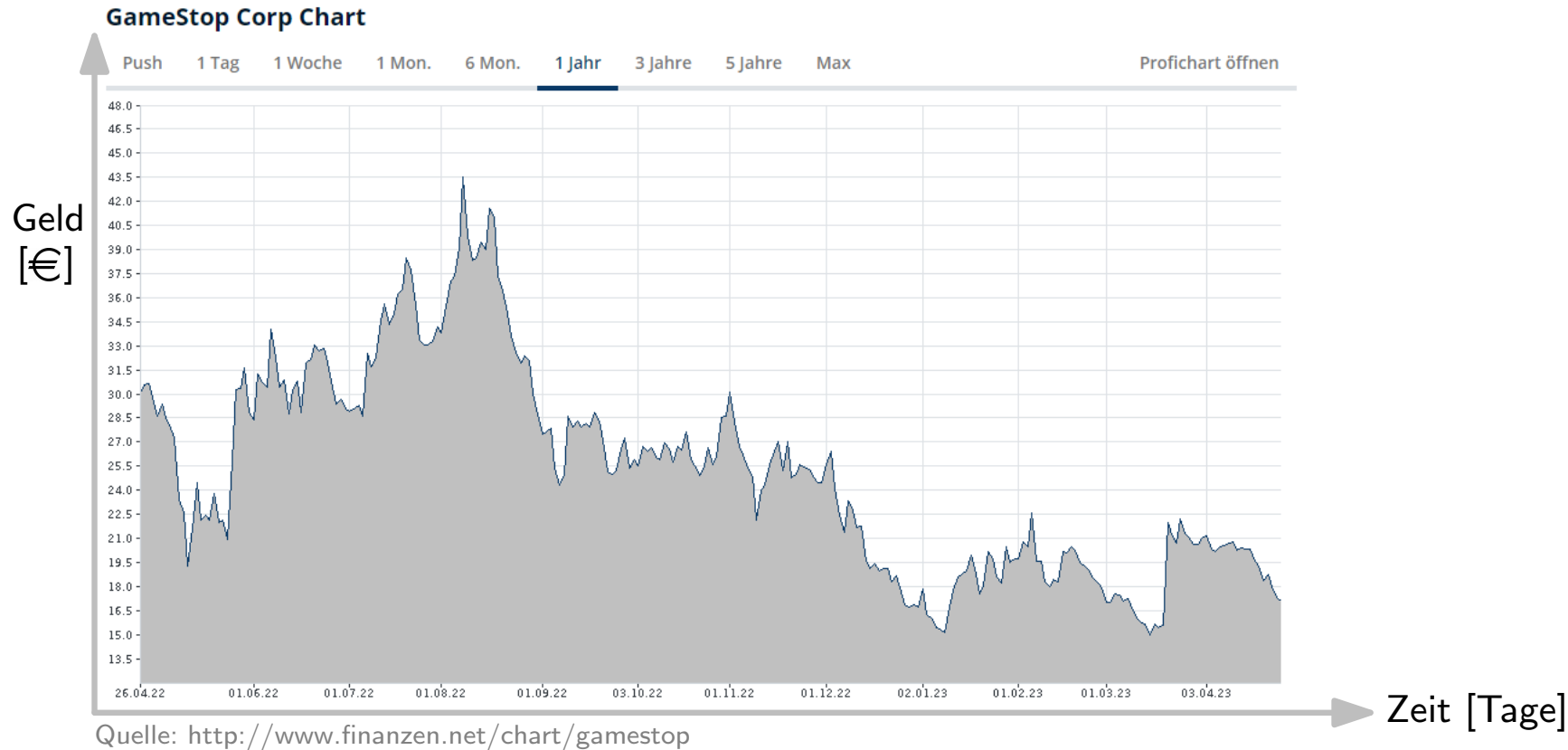


Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

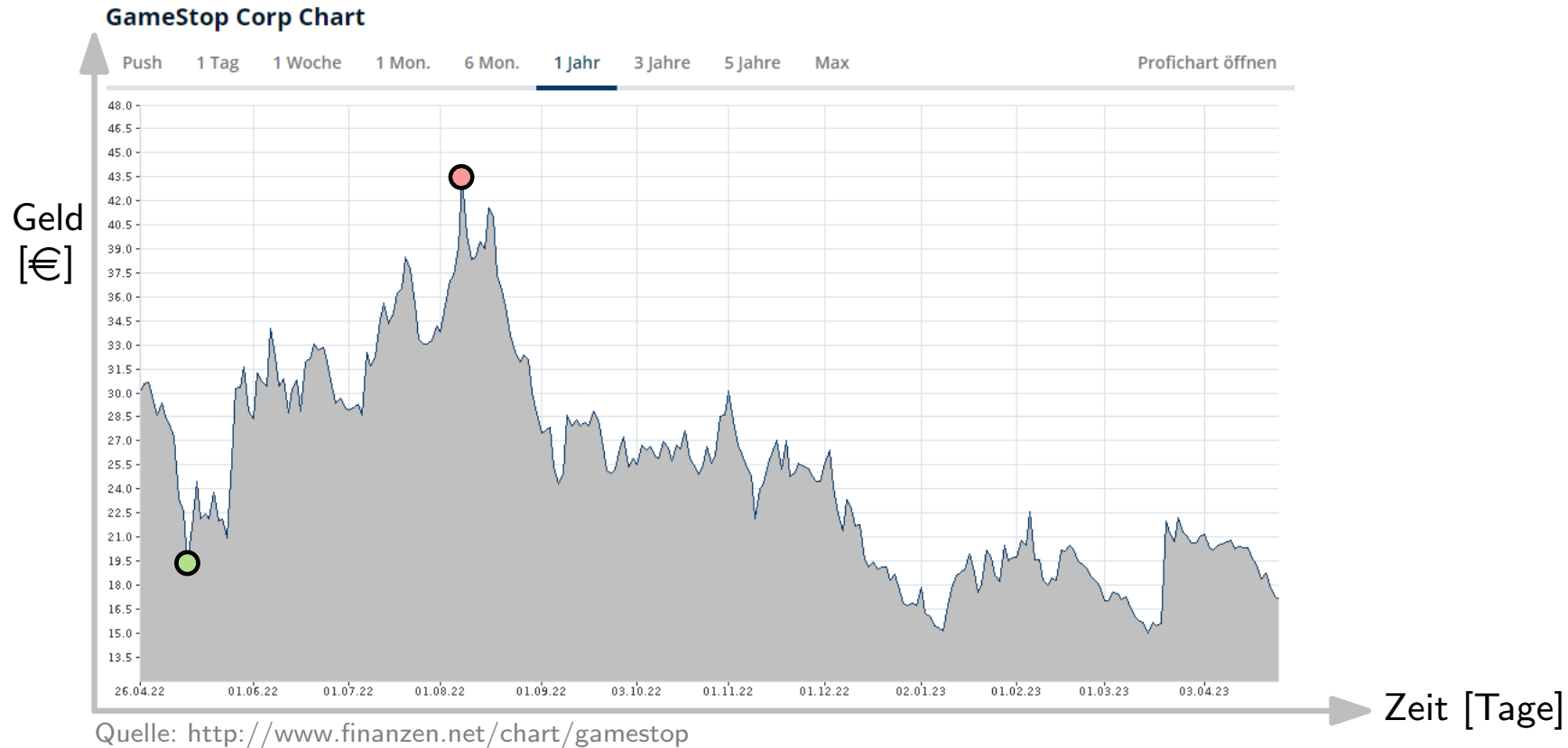
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.

Verkaufskurs

Einkaufskurs

Profit pro Aktie

Analyse von Aktienkursen



Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

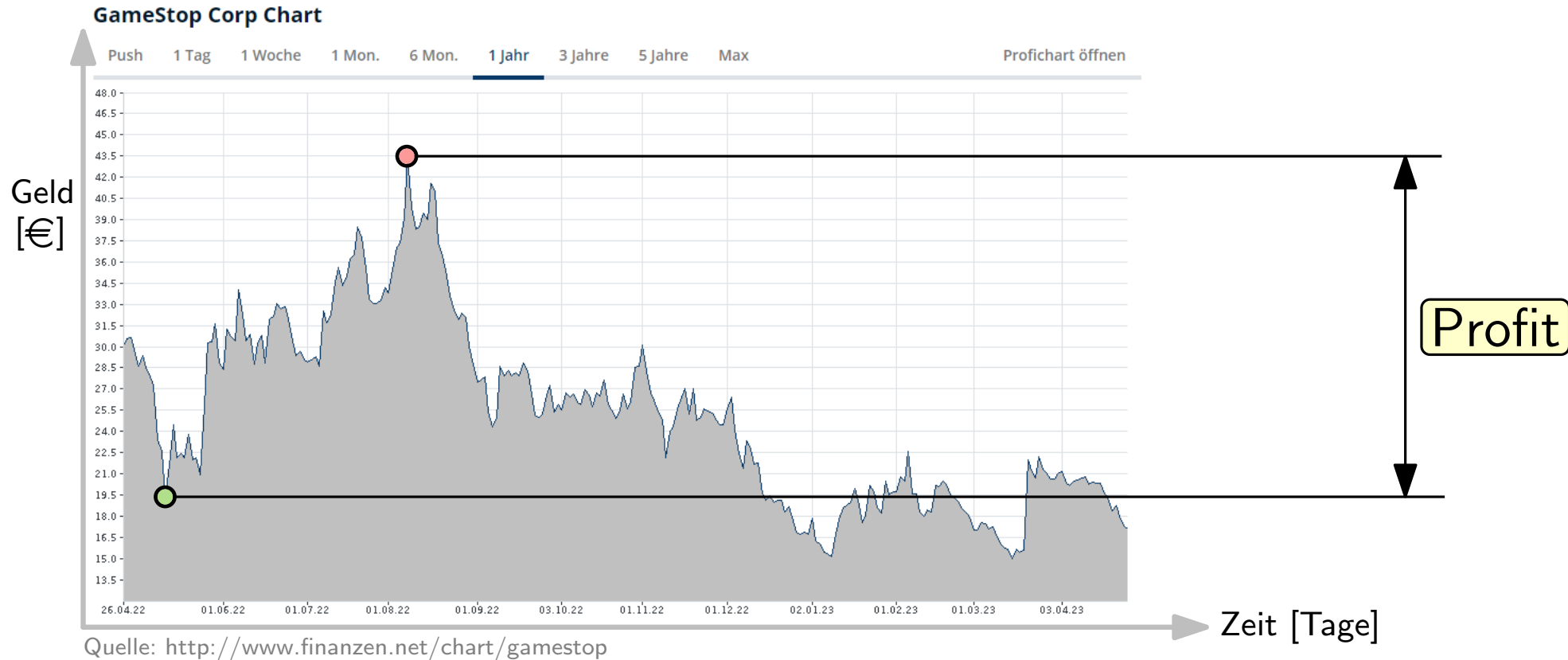
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.

Verkaufskurs

Einkaufskurs

Profit pro Aktie

Analyse von Aktienkursen



Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

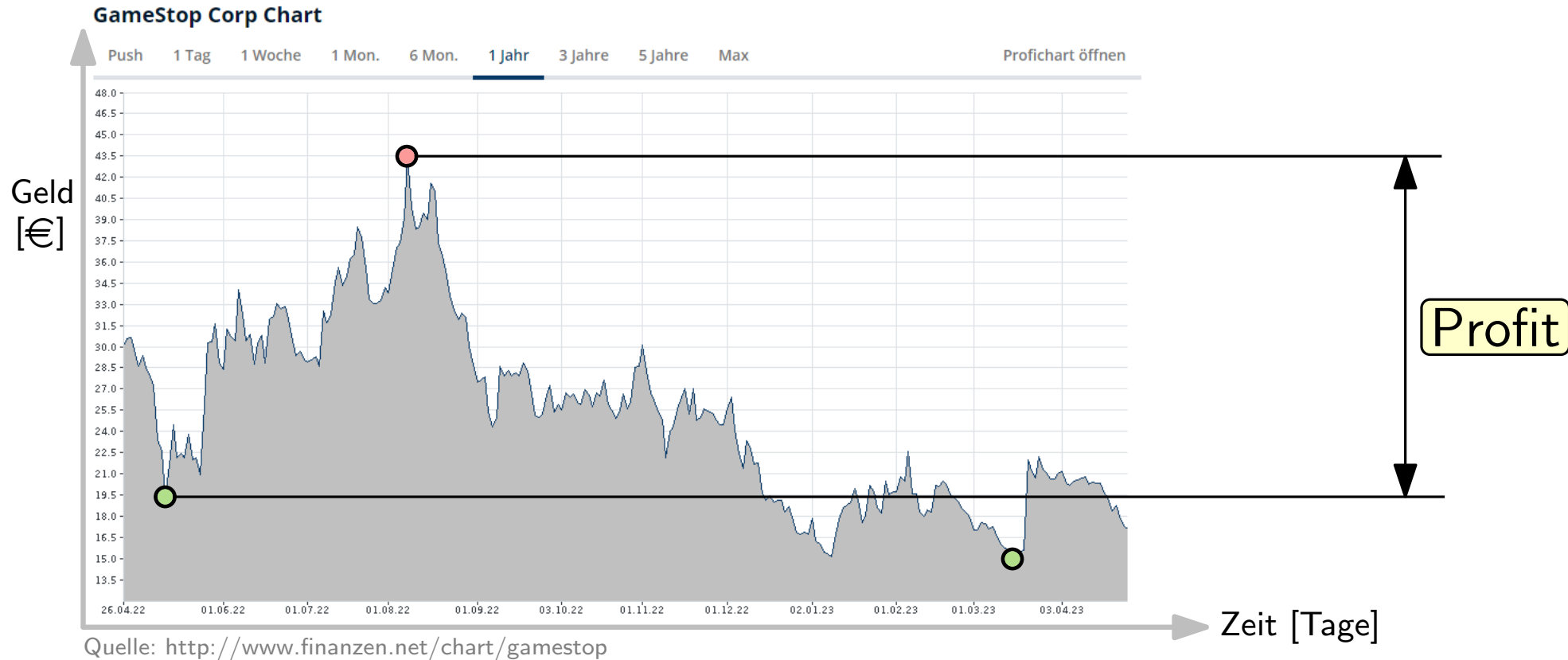
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.

Verkaufskurs

Einkaufskurs

Profit pro Aktie

Analyse von Aktienkursen



Problem. **Gegeben:** Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

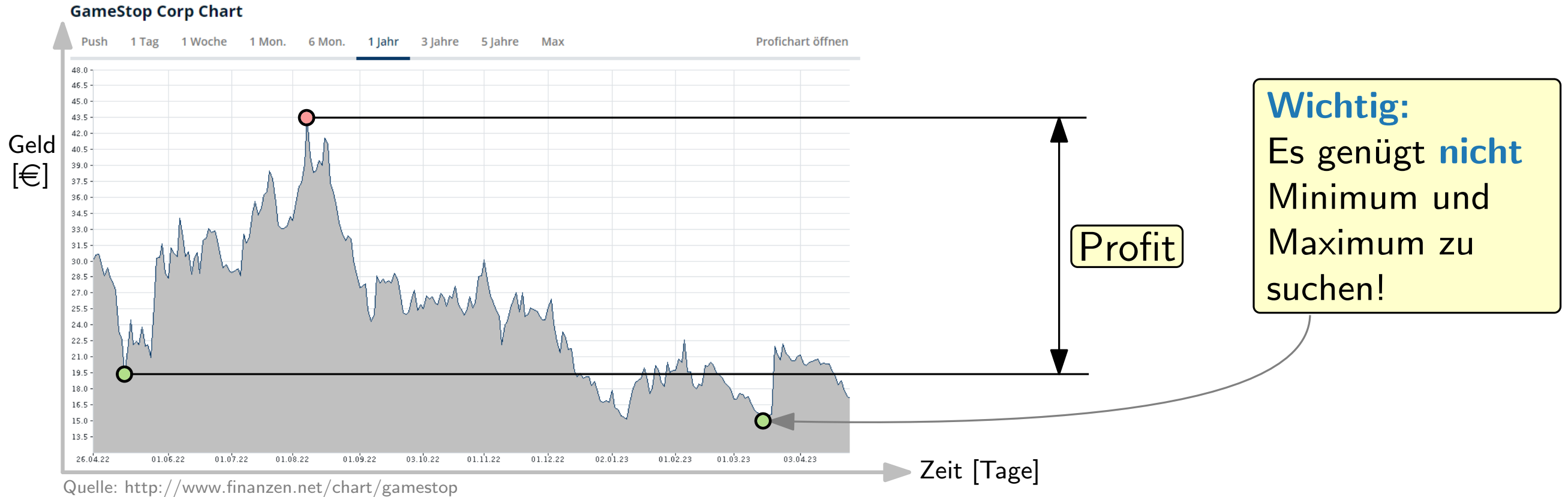
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.

Verkaufskurs

Einkaufskurs

Profit pro Aktie

Analyse von Aktienkursen



Problem. Gegeben: Eine Folge $A[1 \dots n]$ 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: Eine Folge $A[1 \dots n]$ 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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ 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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Lösung: per **roher Gewalt**

Analyse von Aktienkursen

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Lösung: per **roher Gewalt**

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

Analyse von Aktienkursen

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Lösung: per **roher Gewalt**

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

Analyse von Aktienkursen

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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

$$1) \sum_{i=1}^n i = \frac{n(n+1)}{2} \text{ arithmetische Reihe}$$

Analyse von Aktienkursen

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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

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

$$1) \sum_{i=1}^n i = \frac{n(n+1)}{2} \quad \text{arithmetische Reihe}$$

Analyse von Aktienkursen

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

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

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

$$1) \sum_{i=1}^n i = \frac{n(n+1)}{2} \text{ arithmetische Reihe}$$

Ein ähnliches Problem

Problem. MAXSUM

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

MAXDIFF

0	7	11	2	3	0	3	4	16	16	14	14	18	20	12	14
---	---	----	---	---	---	---	---	----	----	----	----	----	----	----	----

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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

MAXSUM

MAXDIFF

0	7	11	2	3	0	3	4	16	16	14	14	18	20	12	14
---	---	----	---	---	---	---	---	----	----	----	----	----	----	----	----

Ein ähnliches Problem

Problem. MAXSUM

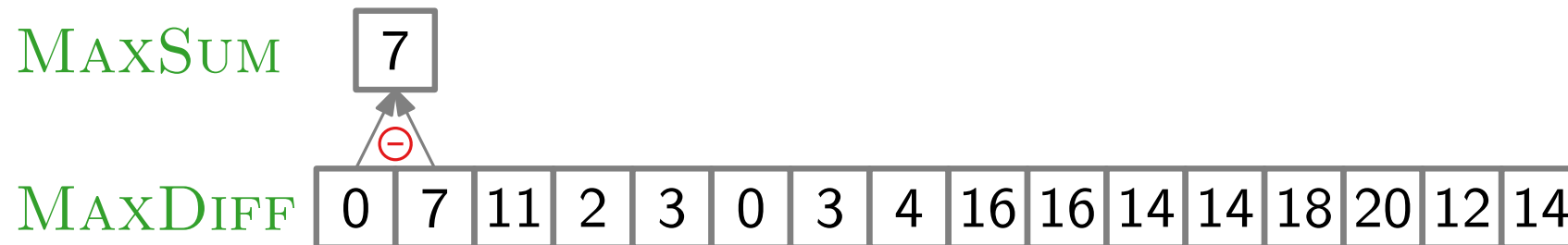
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

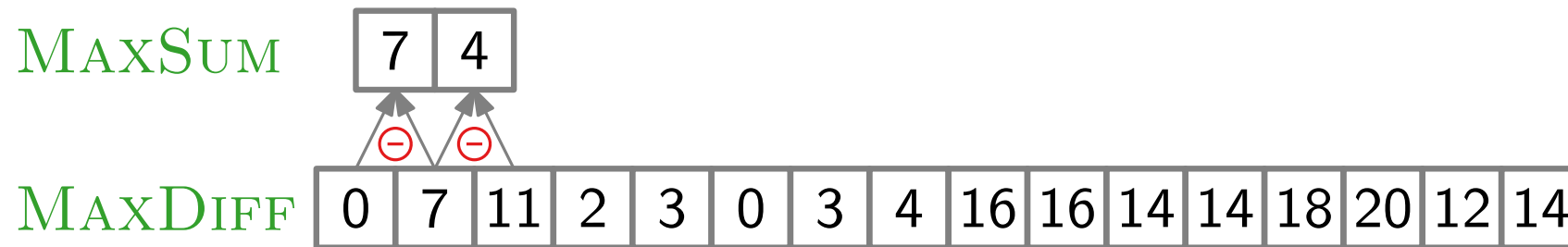
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

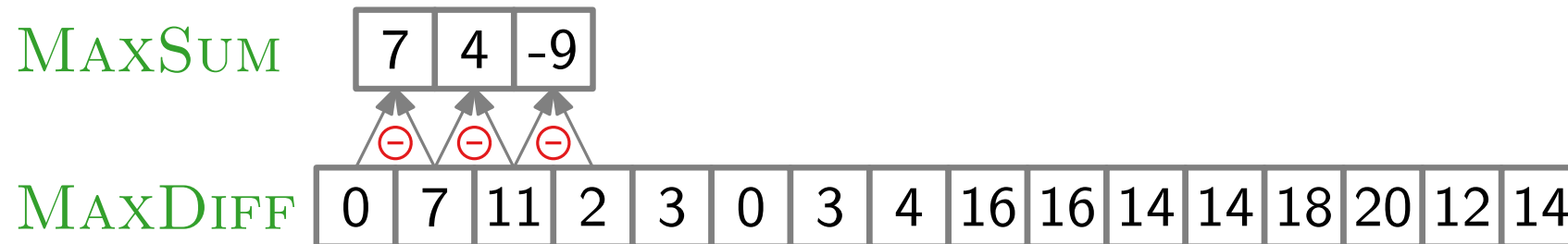
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $\sum_{k=i}^j A[k]$ **maximal**.

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

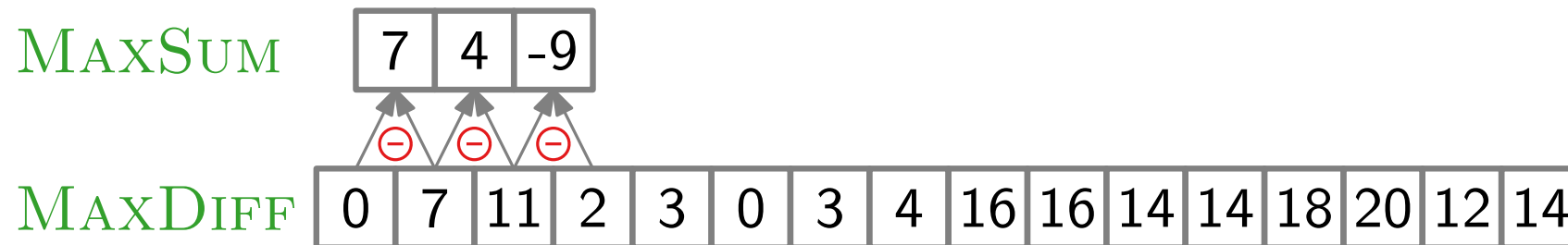
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

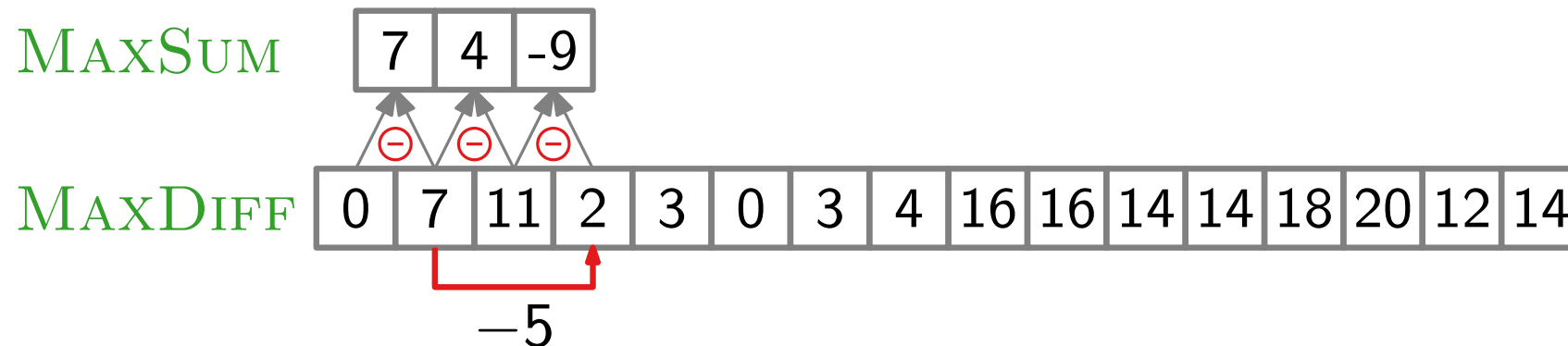
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

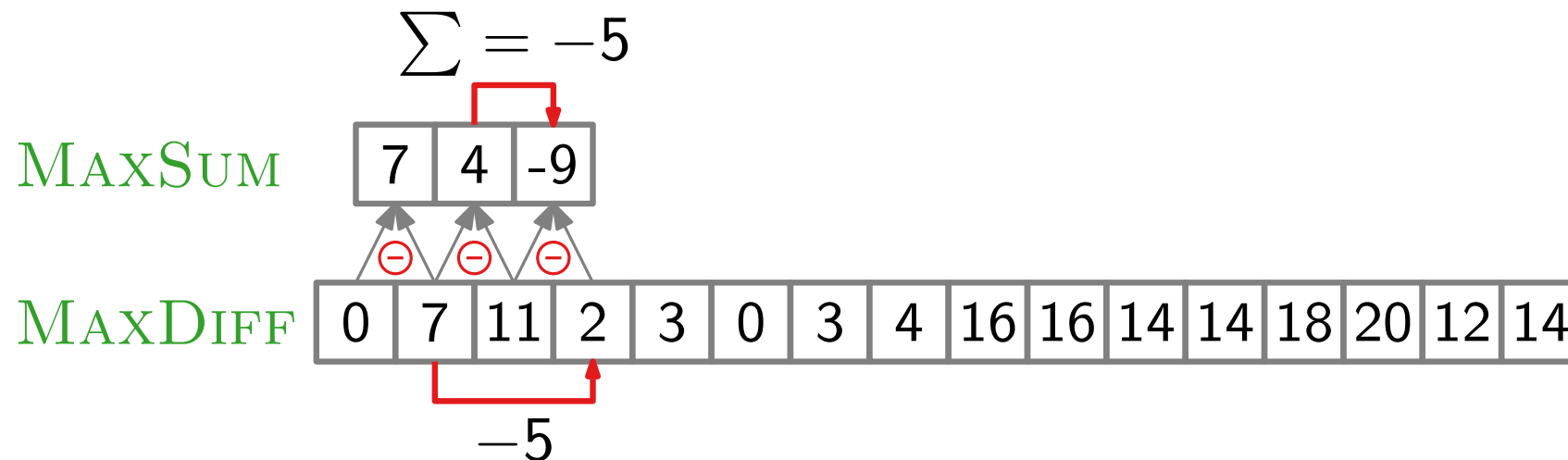
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

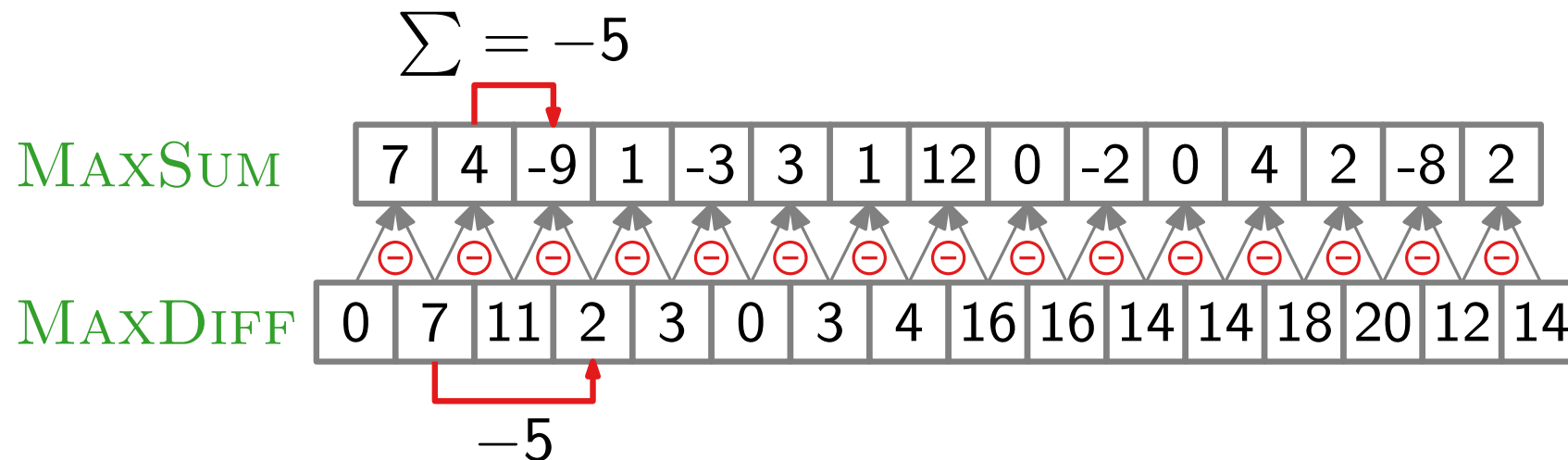
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

Problem. MAXDIFF

Gegeben: Eine Folge $A[1 \dots n]$ von Aktienkursen in Euro.

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



Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

7	4	-9	1	-3	3	1	12	0	-2	0	4	2	-8	2
---	---	----	---	----	---	---	----	---	----	---	---	---	----	---

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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 $\sum_{k=i}^j A[k]$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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

Demo.

<https://algo.uni-trier.de/demos/maxsum.html>

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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.

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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:

$\mathcal{O}(n^2)$.

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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:

$\mathcal{O}(n^2)$



Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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:

$$\mathcal{O}(n^2) \cdot \mathcal{O}(n)$$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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:

$$\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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

$$\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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 (Anz. Paare)

$$\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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 (Anz. Paare)

$$\begin{aligned} \mathcal{O}(n^2) \cdot \mathcal{O}(n) &= \mathcal{O}(n^3) \\ &= \Omega(n^2) \end{aligned}$$

Ein ähnliches Problem

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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

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 (Anz. Paare)

$$\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$$

$$= \Omega(n^2)$$

Wo ist die Wahrheit?

Genauere Analyse

Laufzeit.

- ≈ 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.

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

Beobachtung.

Genauere Analyse

Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist

Genauere Analyse

Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist

n 

Genauere Analyse

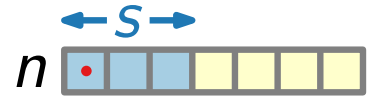
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist



Genauere Analyse

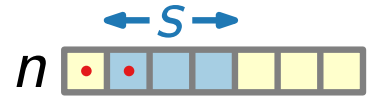
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist



Genauere Analyse

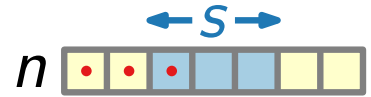
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist



Genauere Analyse

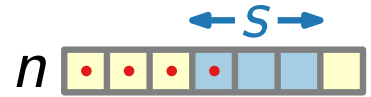
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist



Genauere Analyse

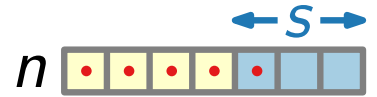
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- Anz. der Summen mit s Summanden ist



Genauere Analyse

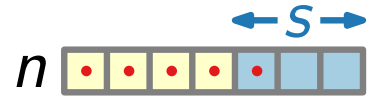
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



Genauere Analyse

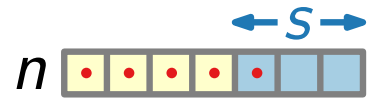
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



Genauere Analyse

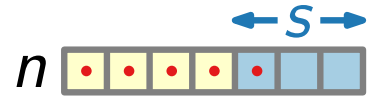
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



Genauere Analyse

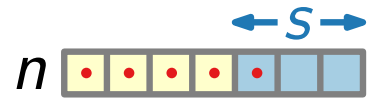
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



⇒ Anz. Add. =

Genauere Analyse

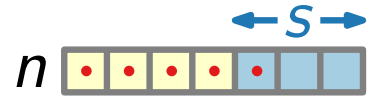
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- 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

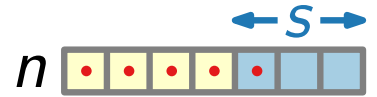
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

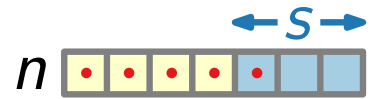
Laufzeit.

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

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

Beobachtung.

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



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

Genauere Analyse

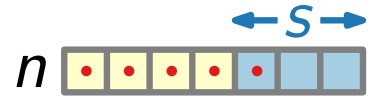
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

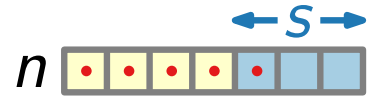
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

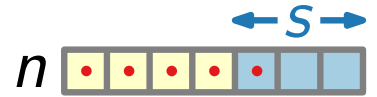
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

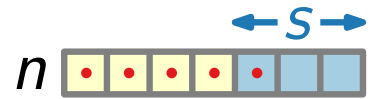
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

- 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

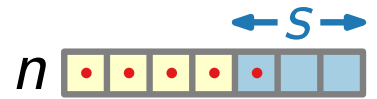
Laufzeit.

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

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

Beobachtung.

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



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

Genauere Analyse

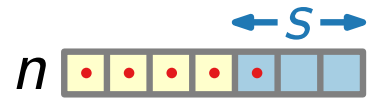
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

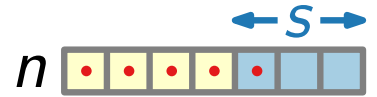
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

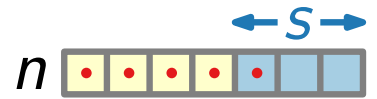
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

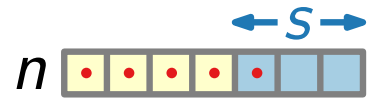
Laufzeit.

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

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

Beobachtung.

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



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

Genauere Analyse

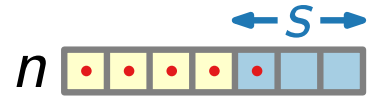
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

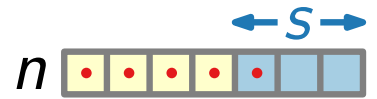
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Genauere Analyse

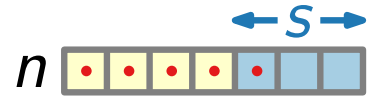
Laufzeit.

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

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

Beobachtung.

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



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

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

Genauere Analyse

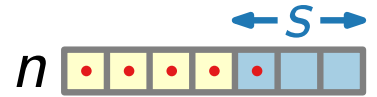
Laufzeit.

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

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

Beobachtung.

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



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

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

Geht das besser?

Genauere Analyse

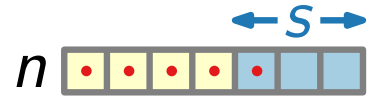
Laufzeit.

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

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

Beobachtung.

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



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

Übung.

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

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

Geht das besser?

Genauere Analyse

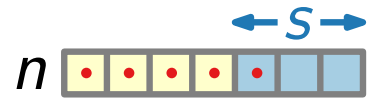
Laufzeit.

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

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

Beobachtung.

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



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

Übung.

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

Wie berechnen?

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

Geht das besser?

Genauere Analyse

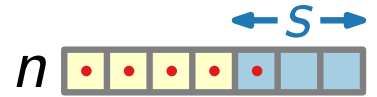
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

Übung.

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

Wie berechnen?

$$\# \text{ Add.} = an^3 + bn^2 + cn + d$$

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

Geht das besser?

Genauere Analyse

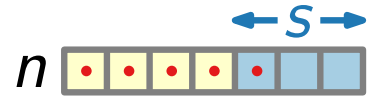
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

(falls $4 \mid n$)

Übung.

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

Wie berechnen?

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

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

Geht das besser?

Genauere Analyse

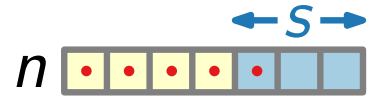
Laufzeit.

≈ Anzahl der Additionen des Rohe-Gewalt-Algos:

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

Beobachtung.

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



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

(falls $4 \mid n$)

Übung.

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

Wie berechnen?

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

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

Geht das besser?

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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

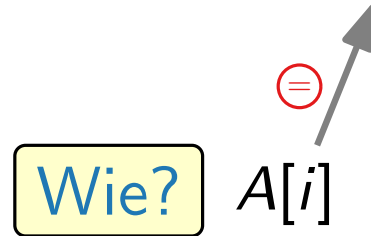
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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 $S_{ii}, S_{i,i+1}, S_{i,i+2}, S_{i,i+3}, \dots, S_{i,n}$

 $A[i]$

Eine schnellere Lösung

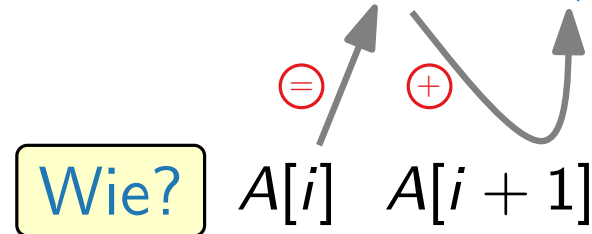
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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

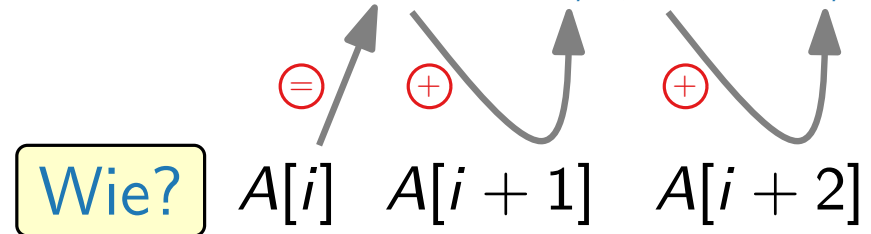
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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

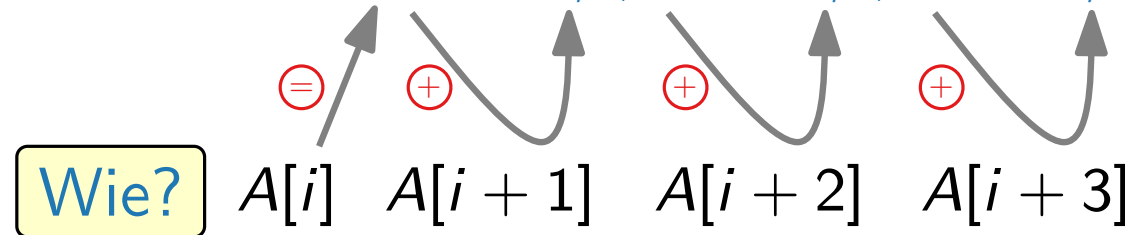
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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

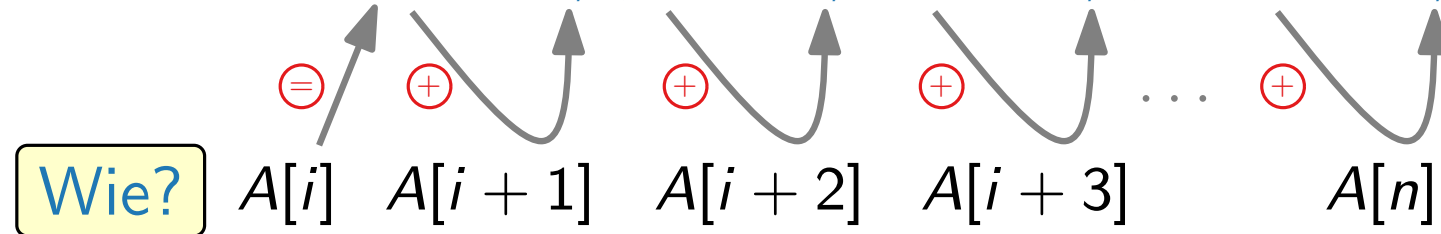
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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 $S_{ii}, S_{i,i+1}, S_{i,i+2}, S_{i,i+3}, \dots, S_{i,n}$



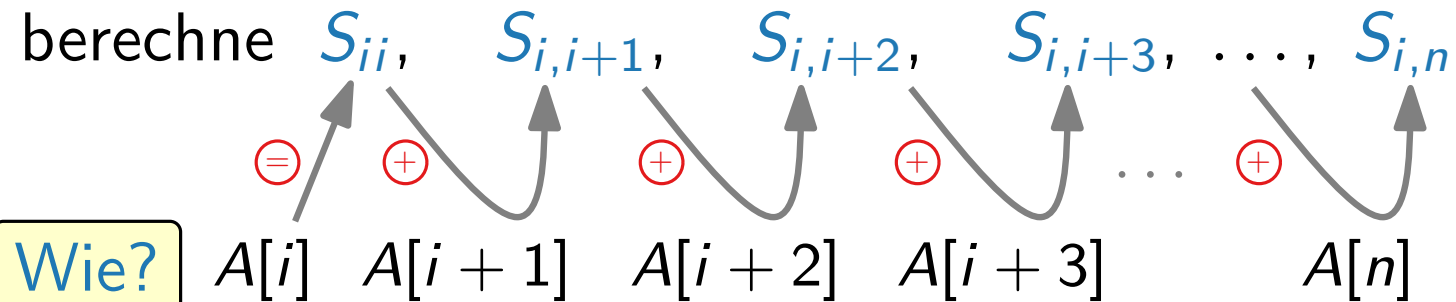
Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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 $i = 1, \dots, n$



Demo.

<https://algo.uni-trier.de/demos/maxsum.html>

Eine schnellere Lösung

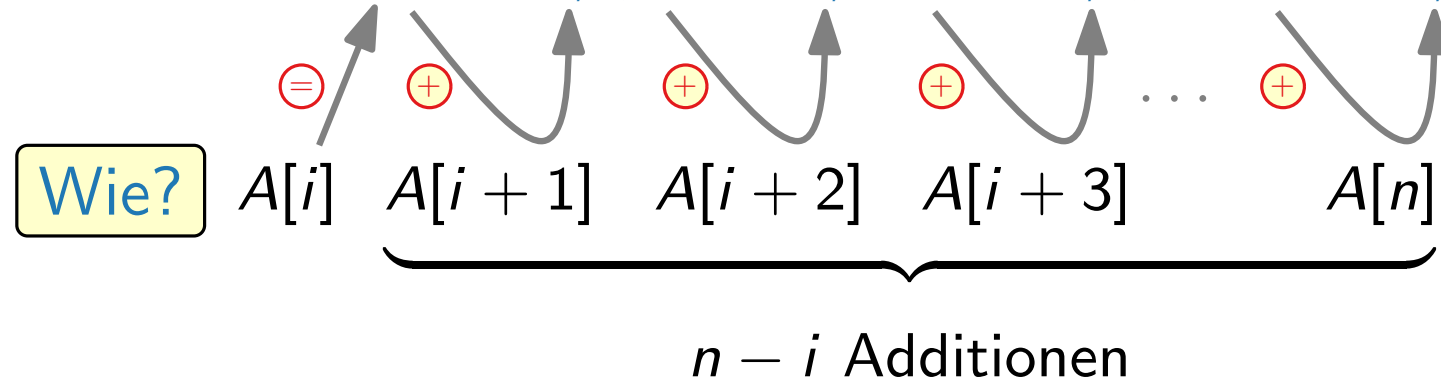
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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

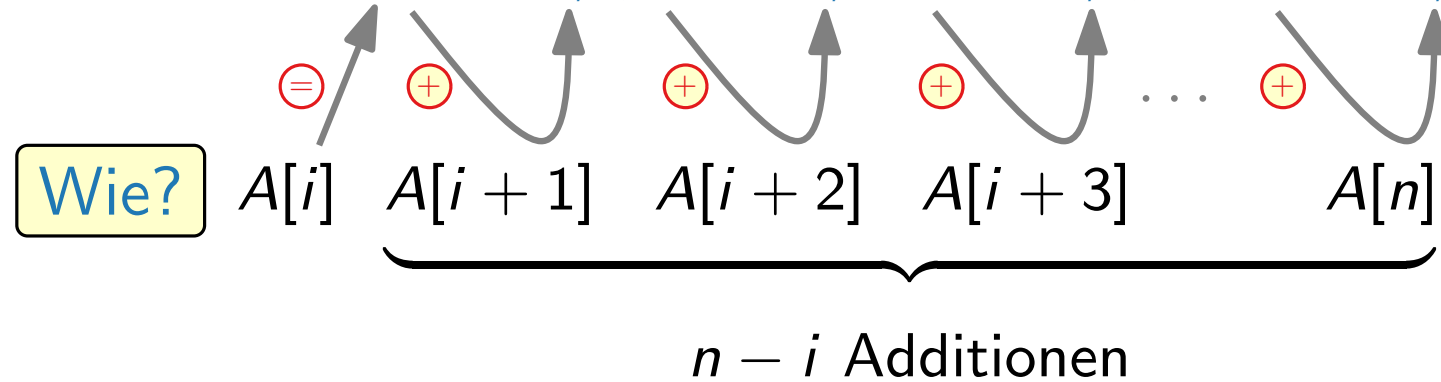
Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ 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 $S_{ii}, S_{i,i+1}, S_{i,i+2}, S_{i,i+3}, \dots, S_{i,n}$



Insgesamt

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

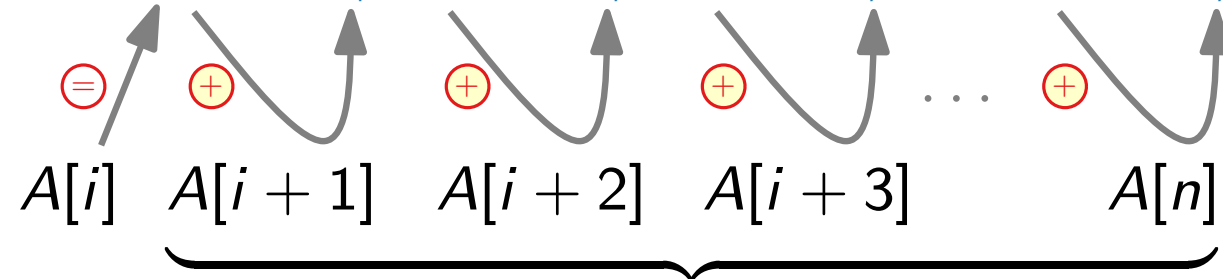
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt



Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

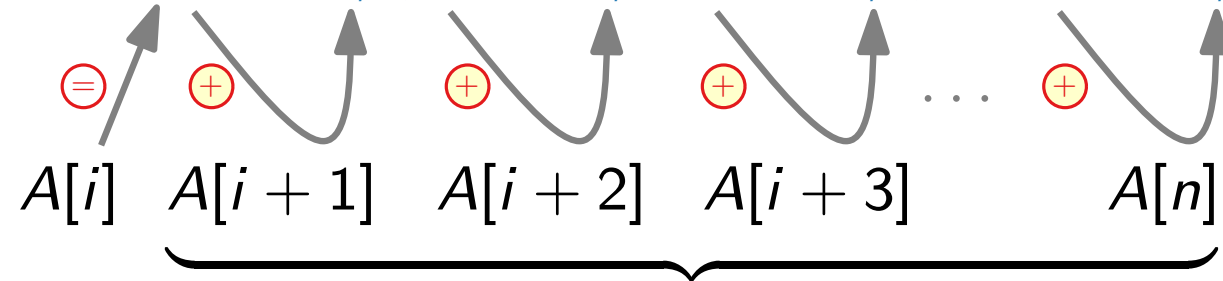
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

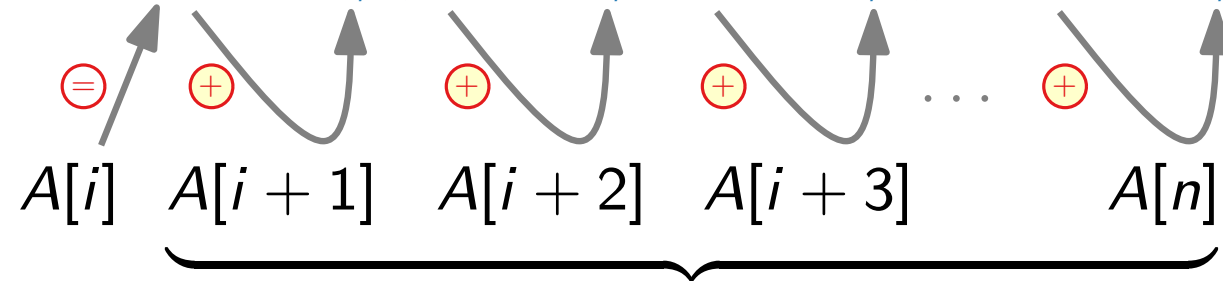
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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



Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

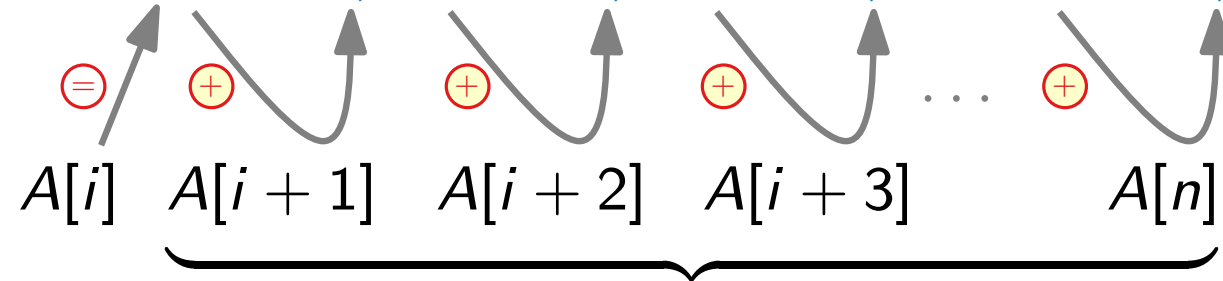
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

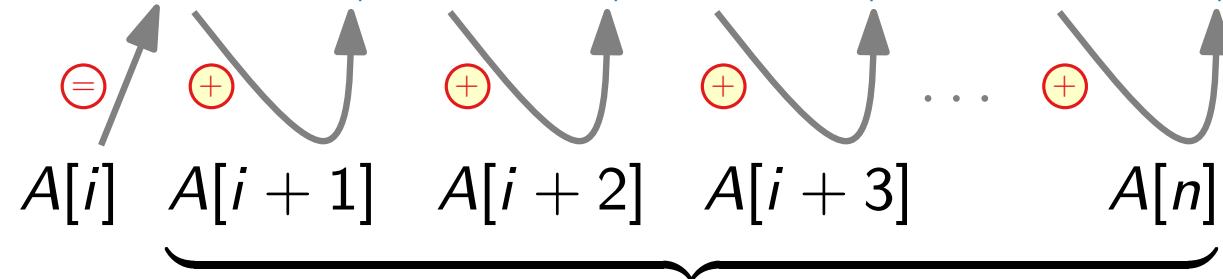
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

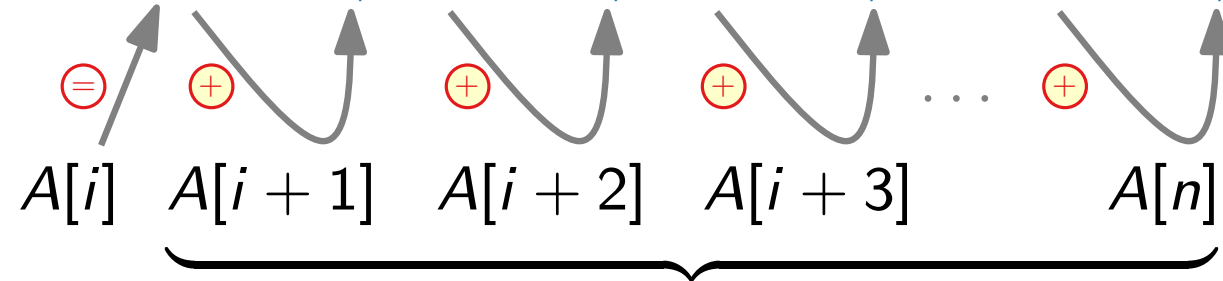
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

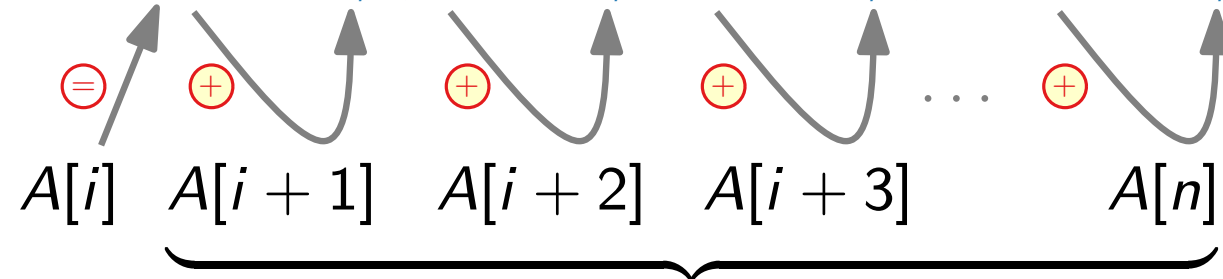
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

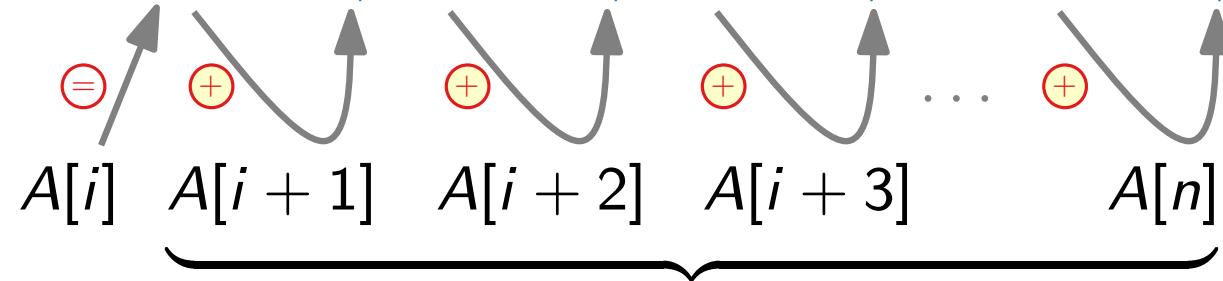
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

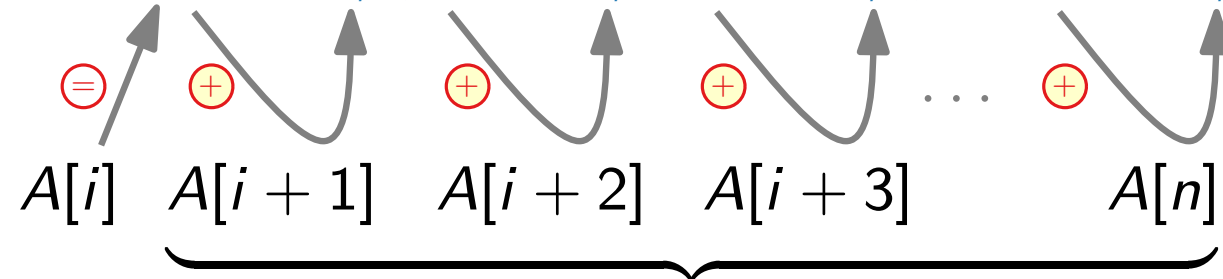
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

Insgesamt

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

$$1) \sum_{i=1}^n i = \frac{n(n+1)}{2} \text{ arithmetische Reihe}$$

Eine schnellere Lösung

Problem. MAXSUM

Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

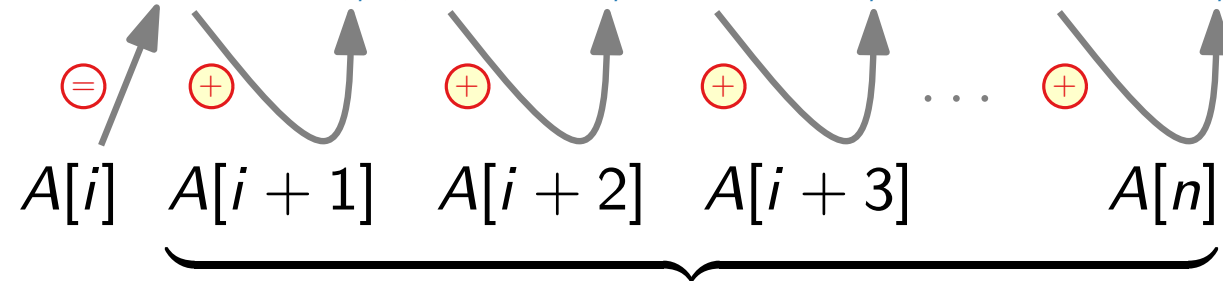
Gesucht: Paar (i, j) mit $1 \leq i \leq j \leq n$, so dass $S_{ij} = \sum_{k=i}^j A[k]$ **maximal**.

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?



$n - i$ Additionen

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



1) $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ arithmetische Reihe

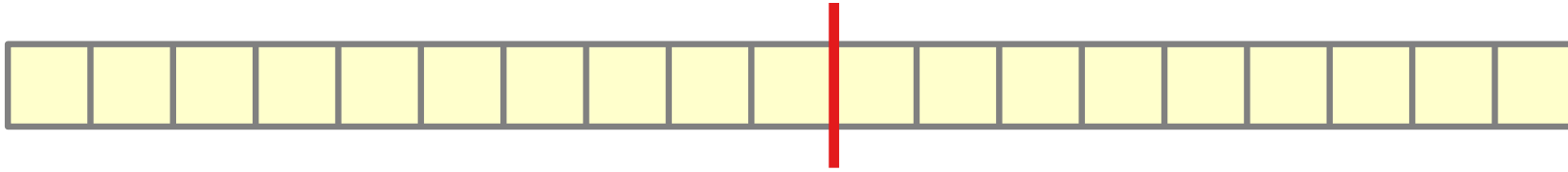
Eine noch schnellere Lösung?

Idee:



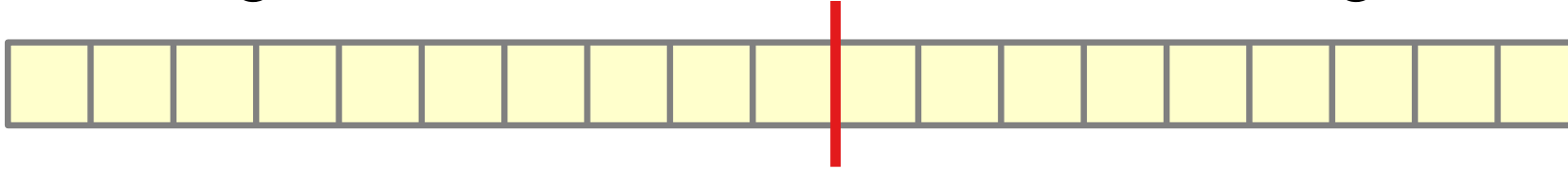
Eine noch schnellere Lösung?

Idee:



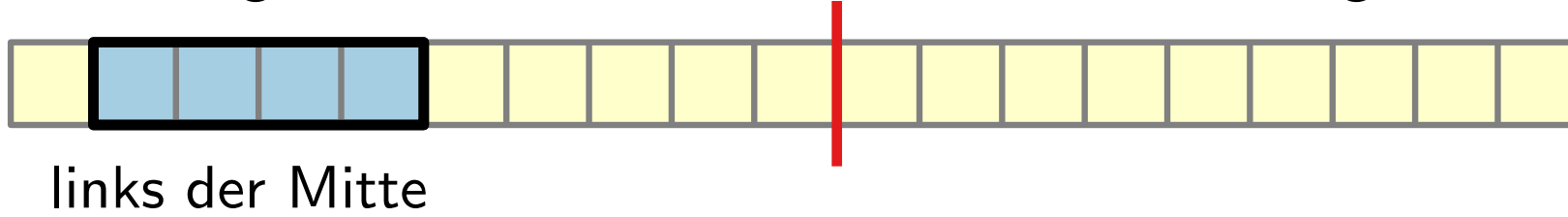
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



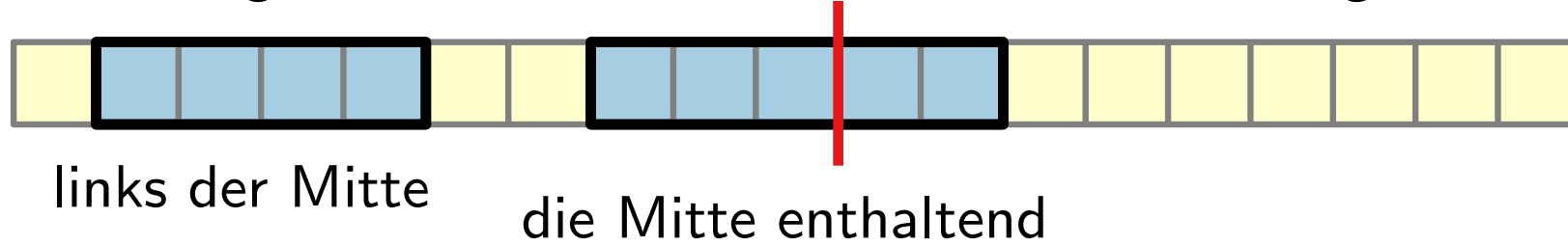
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



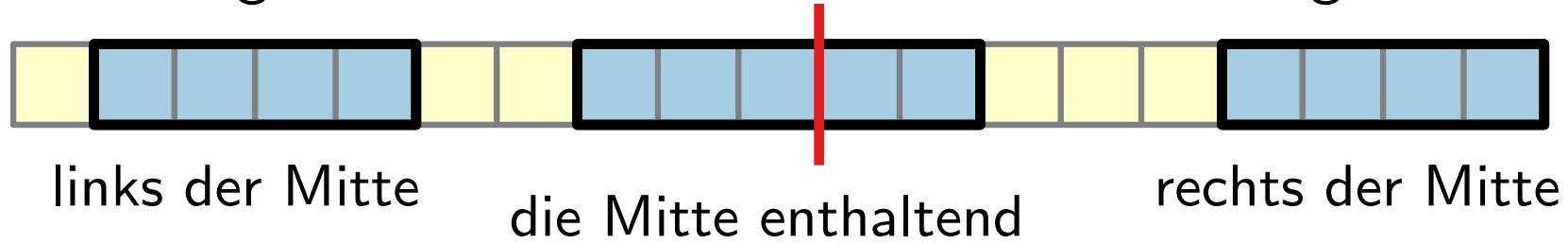
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



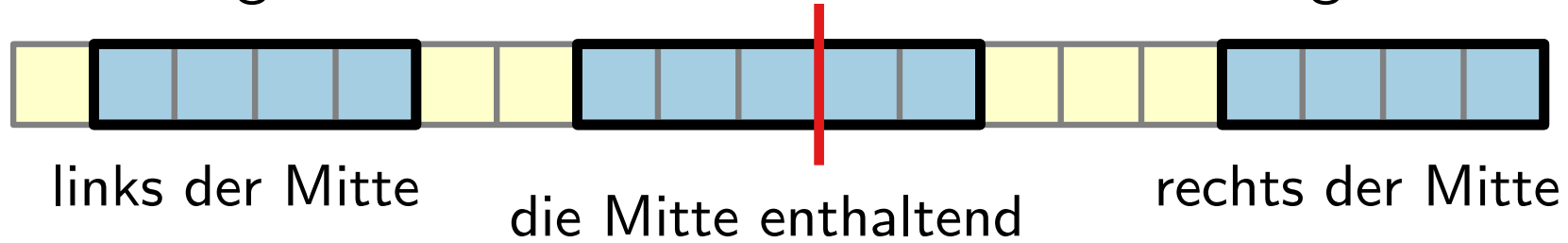
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

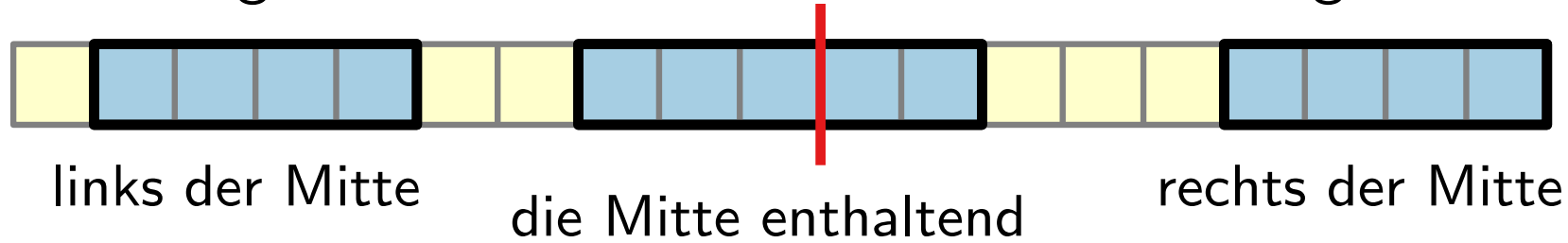
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik **Teile & Herrsche!**

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

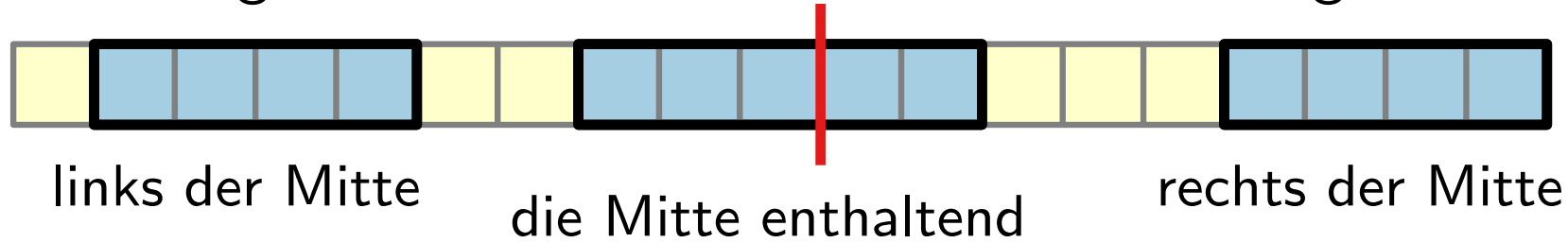


Nimm Entwurfstechnik **Teile & Herrsche!**

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

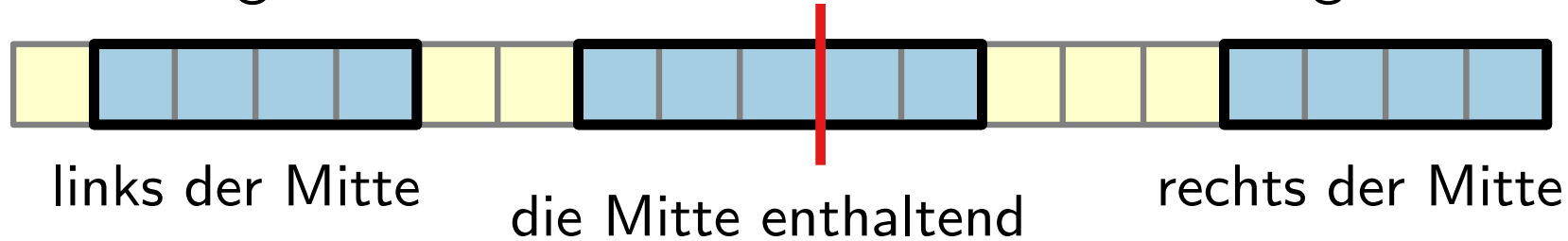


Nimm Entwurfstechnik **Teile & Herrsche!**

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

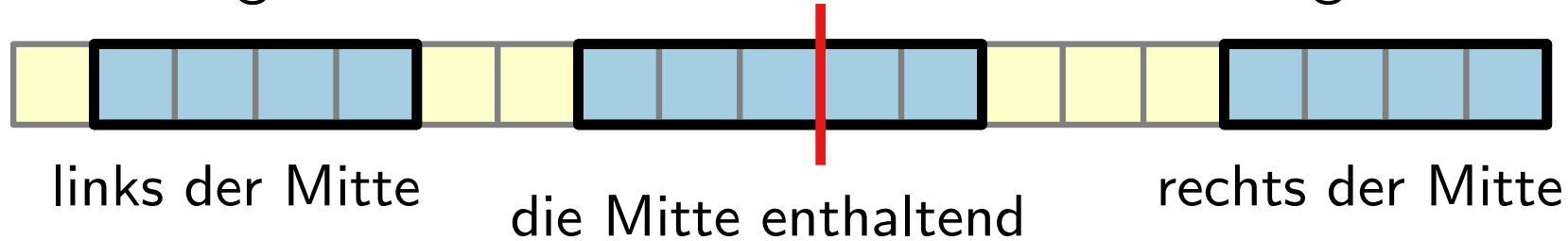


Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:**

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

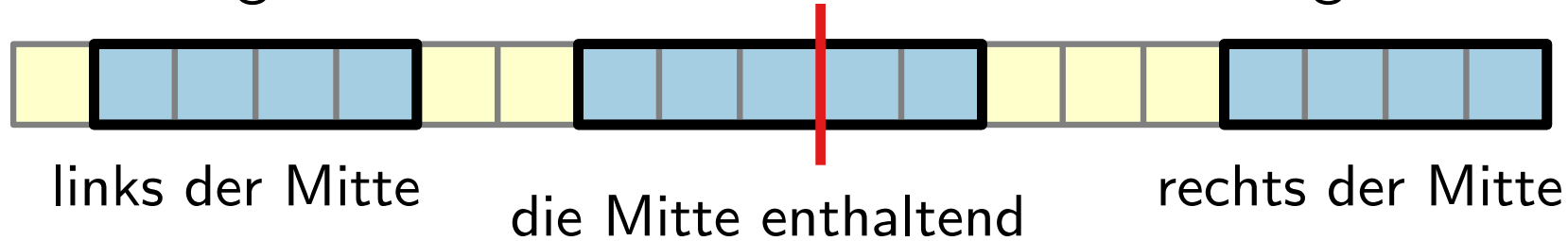


Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere alle Teilsummen, die die Mitte enthalten

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

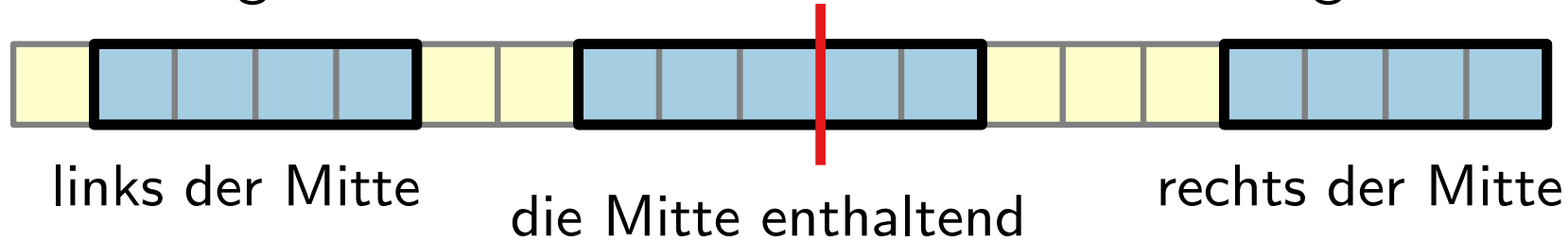


Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



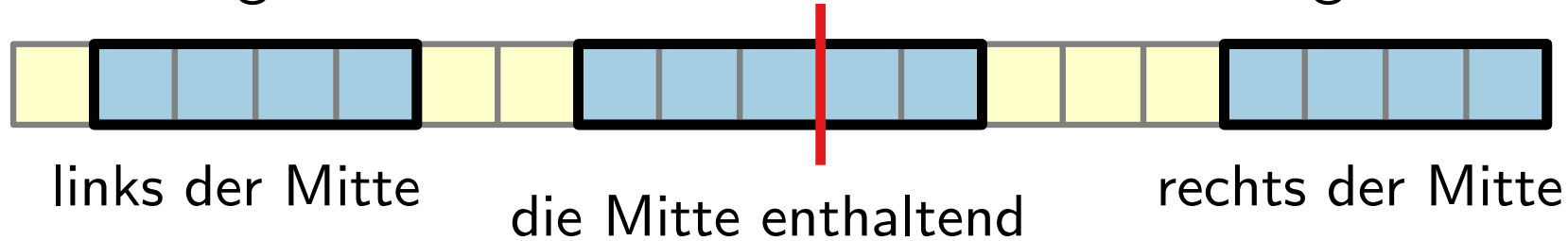
Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

Davon gibt's

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



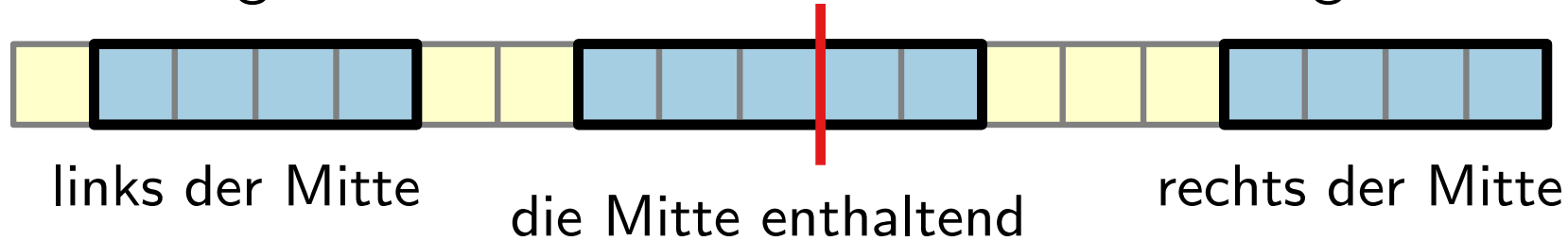
Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte 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:



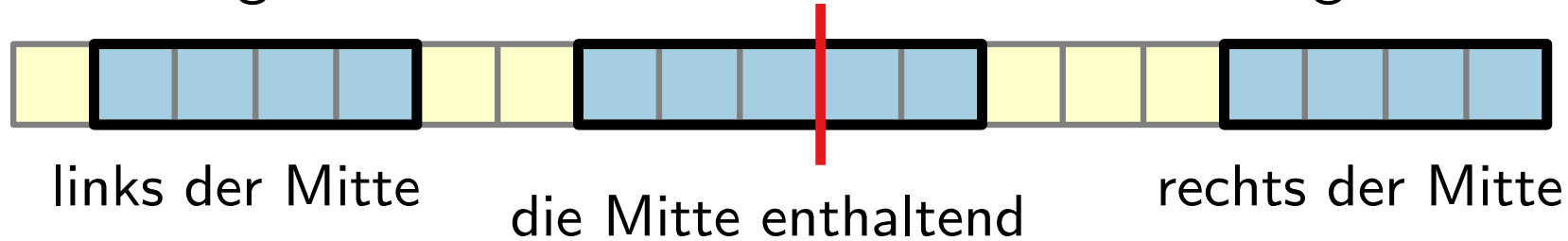
Nimm Entwurfstechnik **Teile & Herrsche!**

- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte 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!**

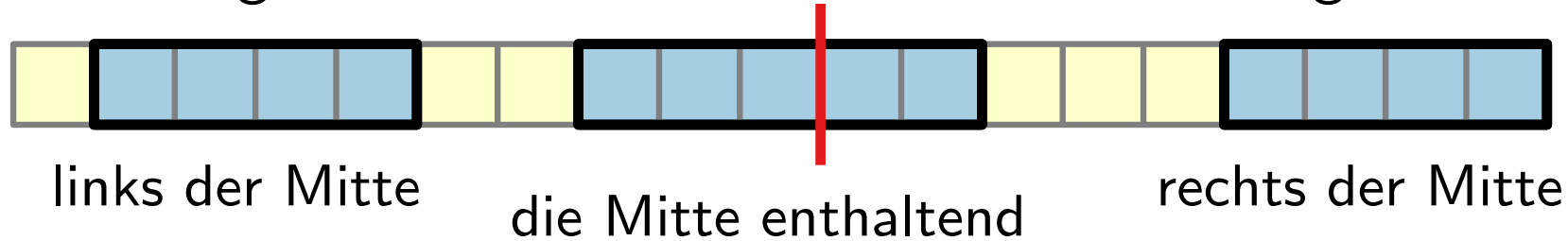
- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

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 linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

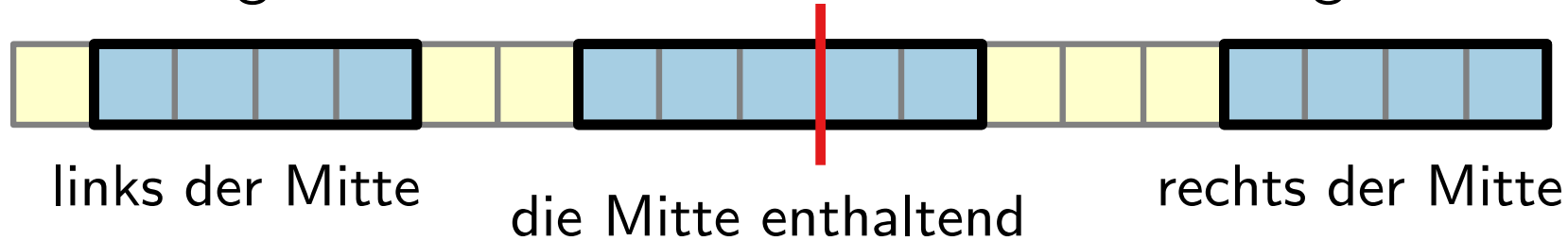
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$



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 linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

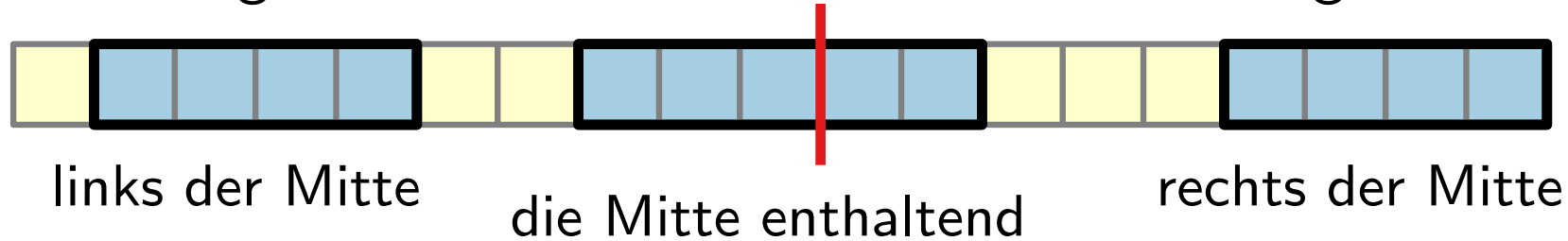
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$



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 linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

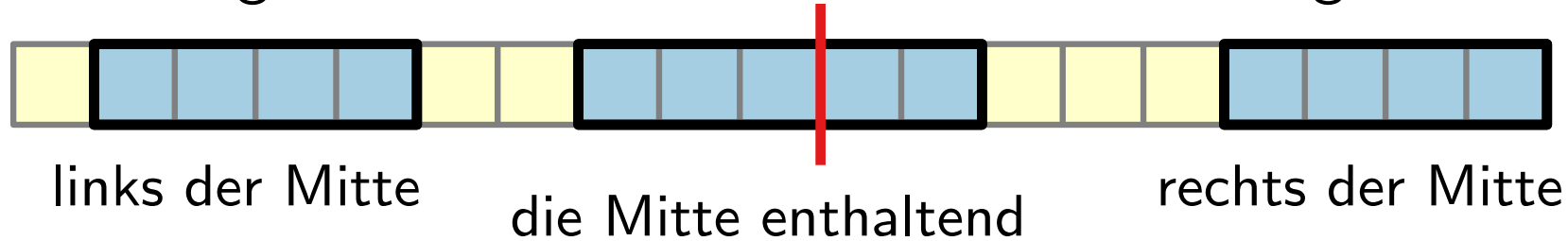
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$



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 linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

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

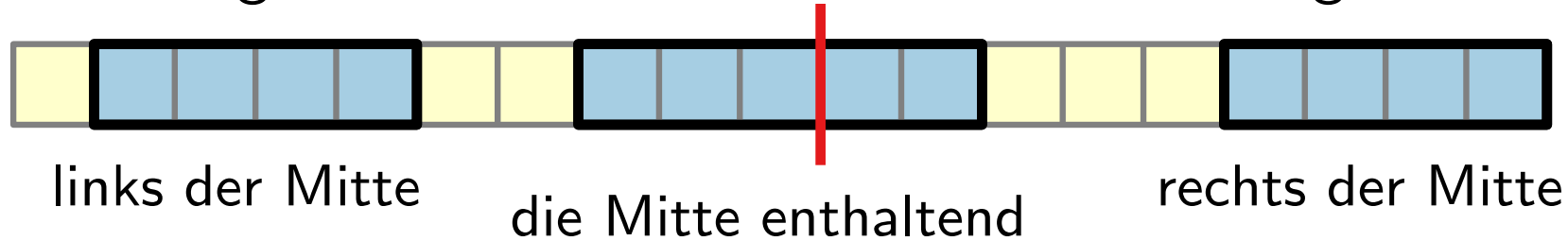


Einsicht: Wenn die **maximale** Teilsumme die Mitte enthält,
dann muss ihr linker Teil (bis zur Mitte) maximal sein

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 linke und rechte Hälfte
- **kombiniere:** kontrolliere **alle** Teilsummen, die die Mitte enthalten

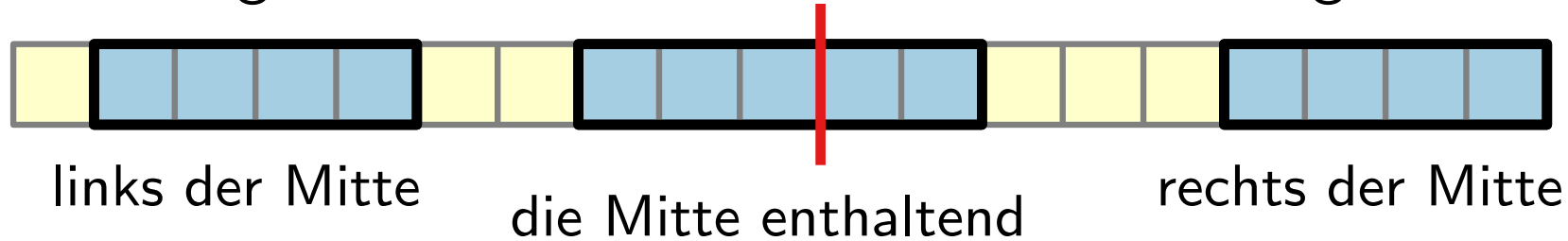
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$



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 linke und rechte 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 linken und rechten Teil **unabhängig** voneinander berechnen!

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
```

```
    |
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
```

```
    | return (anfang, ende, A[anfang])
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
  else
    |
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
  else
    | mitte =  $\lfloor (\textit{anfang} + \textit{ende}) / 2 \rfloor$ 
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then  
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else  
    | mitte =  $\lfloor (\textit{anfang} + \textit{ende}) / 2 \rfloor$  } teile
```

Teile & Herrsche

```

MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
  else
    | mitte = ⌊(anfang + ende)/2⌋
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } teile

```

Teile & Herrsche

MAXTEILFELD(int[] A, int *anfang* = 1, int *ende* = A.length)

if *anfang* == *ende* then
 | return (*anfang*, *ende*, A[*anfang*]) } herrsche (in kleinen Teilinstanzen)

else

mitte = $\lfloor (\textit{anfang} + \textit{ende}) / 2 \rfloor$

(L-*anfang*, L-*ende*, L-summe) = MAXTEILFELD(A, *anfang*, *mitte*)

(R-*anfang*, R-*ende*, R-summe) = MAXTEILFELD(A, *mitte* + 1, *ende*)

} teile

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende)
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende)
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende)
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
```

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
```

```
    mitte = ⌊(anfang + ende)/2⌋
```

```
    (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte)
```

```
    (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende)
```

```
    (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende)
```

} teile

} herrsche

} kombiniere

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
```

```
    mitte = ⌊(anfang + ende)/2⌋
```

```
    (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte)
```

```
    (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende)
```

```
    (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende)
```

```
    return (Tripel mit größter Summe)
```

} teile

} herrsche

} kombiniere

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

Laufzeit:

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

für $n > 1$: $T_{MT}(n) =$

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

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

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

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

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
```

```
    mitte = ⌊(anfang + ende)/2⌋
```

```
    (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte)
```

```
    (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende)
```

```
    (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende)
```

```
    return (Tripel mit größter Summe)
```

} teile

} herrsche

} kombiniere

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

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

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

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

Teile & Herrsche

```
MAXTEILFELD(int[] A, int anfang = 1, int ende = A.length)
```

```
  if anfang == ende then
    | return (anfang, ende, A[anfang]) } herrsche (in kleinen Teilinstanzen)
```

```
  else
    | mitte = ⌊(anfang + ende)/2⌋ } teile
    | (L-anfang, L-ende, L-summe) = MAXTEILFELD(A, anfang, mitte) } herrsche
    | (R-anfang, R-ende, R-summe) = MAXTEILFELD(A, mitte + 1, ende) }
    | (M-anfang, M-ende, M-summe) = MAXMITTEILFELD(A, anfang, mitte, ende) } kombiniere
    | return (Tripel mit größter Summe)
```

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

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

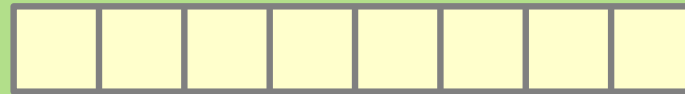
$T_{MMT}(n) = ?$

Kombiniere

```
MAXMITTTEILFELD(int[] A, int anfang, int mitte, int ende)
```

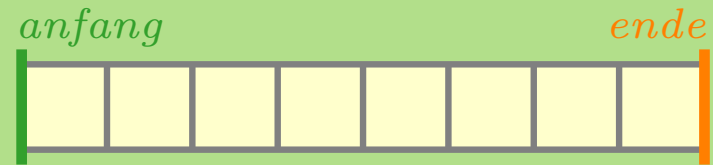
Kombiniere

MAX**MITT**TEILFELD(int[] *A*, int *anfang*, int *mitte*, int *ende*)



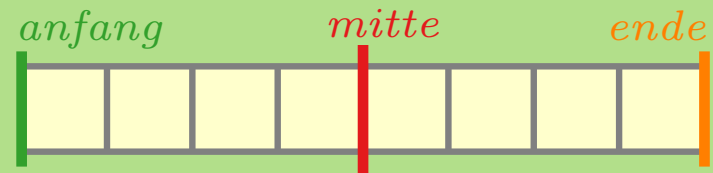
Kombiniere

MAX**MITT**TEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)



Kombiniere

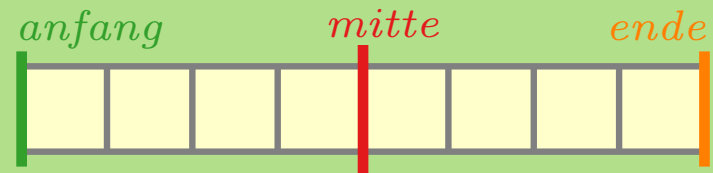
MAX**MITT**TEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)



Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

L-summe = $-\infty$

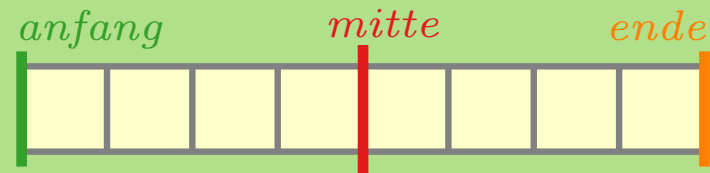


Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

L-summe = $-\infty$

summe = 0



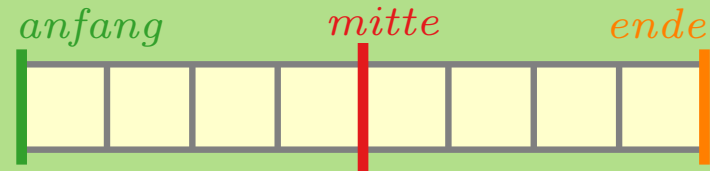
Kombiniere

```
MAXMITTEILFELD(int[] A, int anfang, int mitte, int ende)
```

```
  L-summe =  $-\infty$ 
```

```
  summe = 0
```

```
  for i = mitte downto anfang do
```



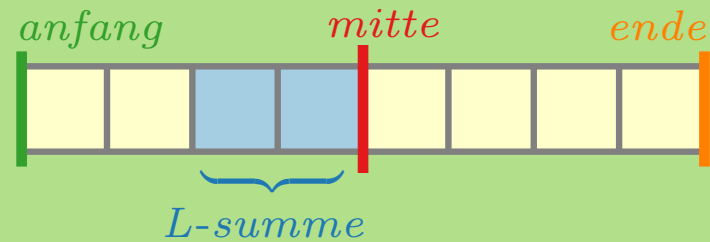
Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

L-summe = $-\infty$

summe = 0

for *i* = *mitte* downto *anfang* do



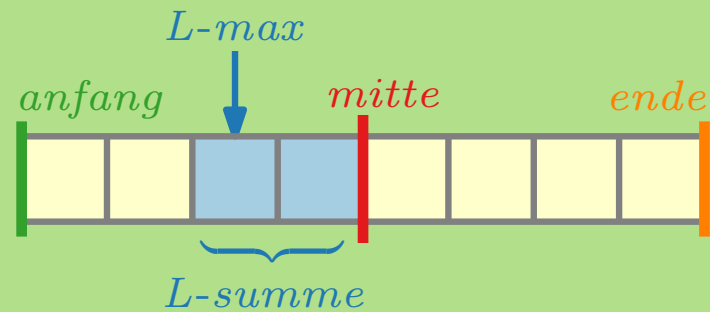
Kombiniere

MAXMITTELFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**



Kombiniere

MAXMITTELFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

L-summe = $-\infty$

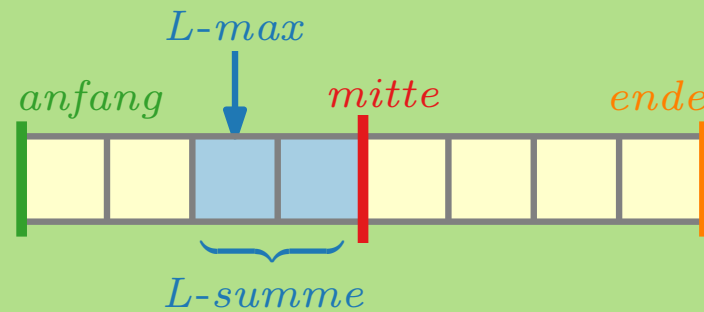
summe = 0

for *i* = *mitte* **downto** *anfang* **do**

|

R-summe = $-\infty$

summe = 0



Kombiniere

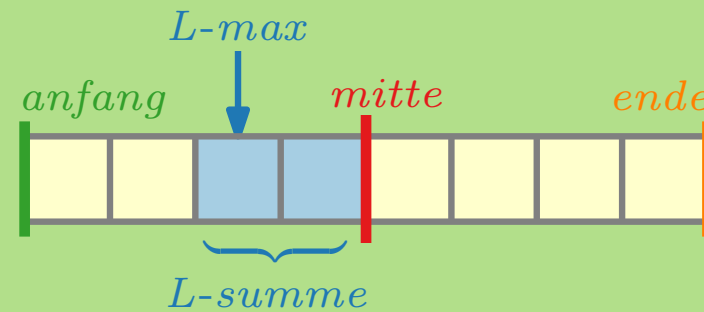
MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

|



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

| // analog zu oben

Kombiniere

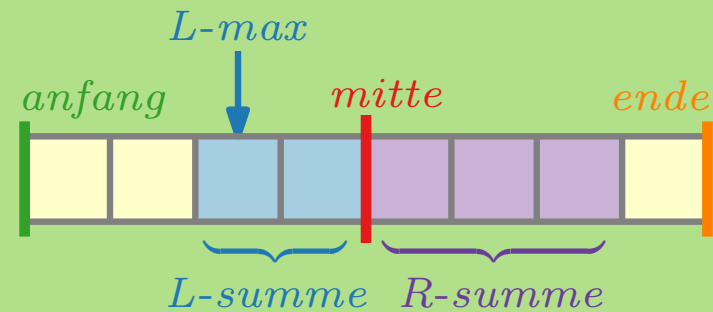
MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

|



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

| // analog zu oben

Kombiniere

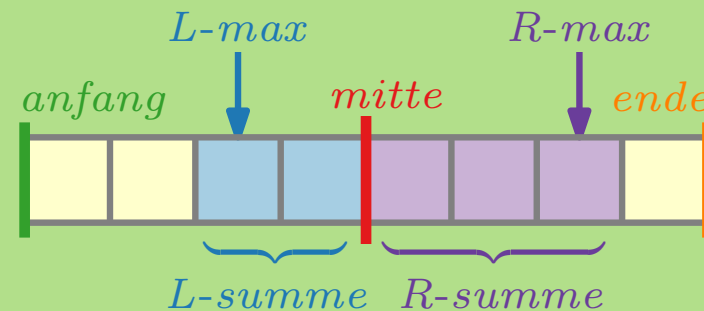
MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

|



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

| // analog zu oben

Kombiniere

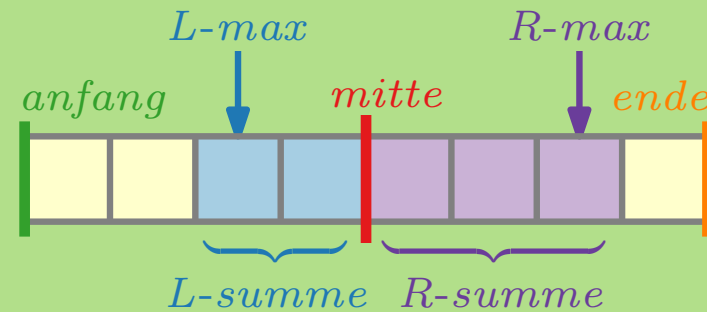
MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

|



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

| // analog zu oben

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

Kombiniere

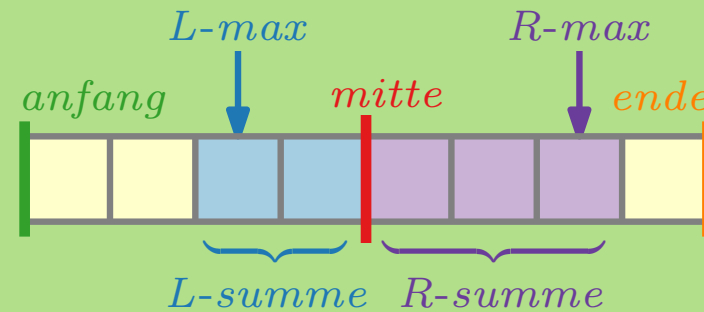
MAXMITTELFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

Vervollständigen Sie
den Algorithmus!



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

└ // analog zu oben

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

Kombiniere

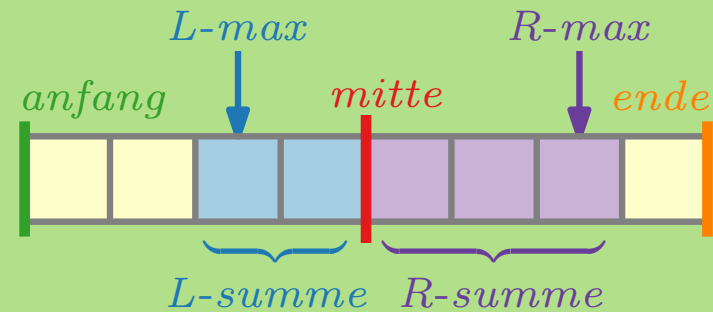
MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

L-summe = $-\infty$

summe = 0

for *i* = *mitte* **downto** *anfang* **do**

summe = *summe* + *A*[*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*)

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

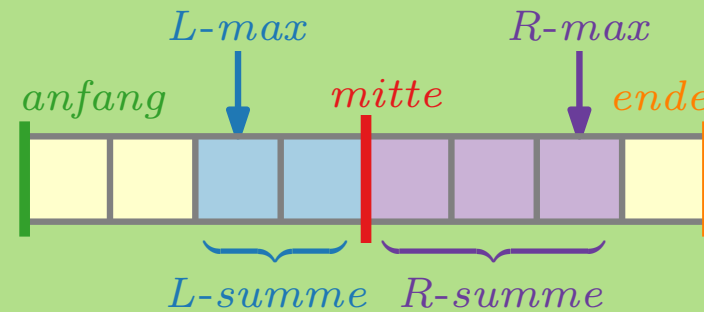
$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

 └



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

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 └ // analog zu oben

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

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

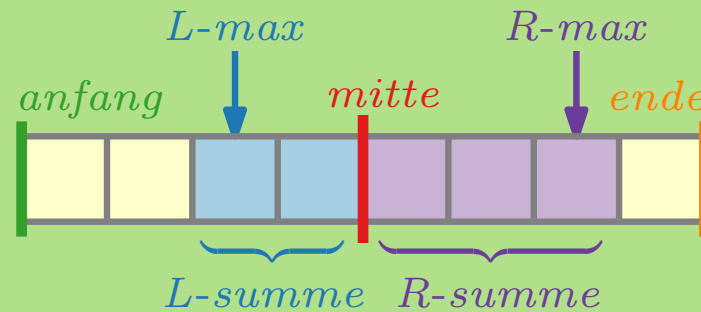
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

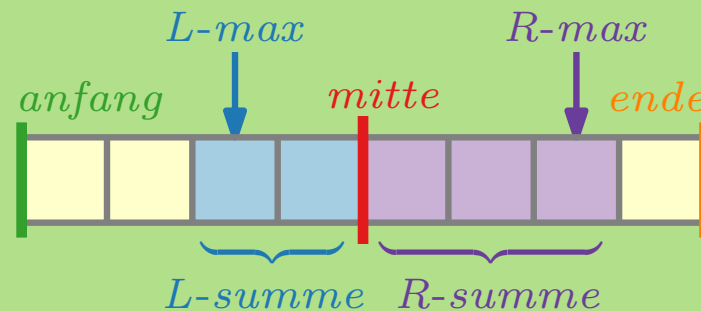
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

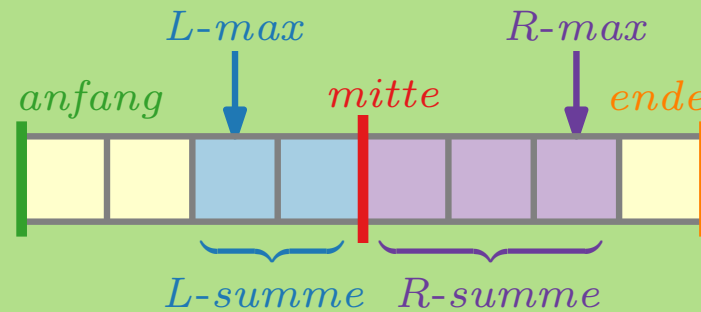
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

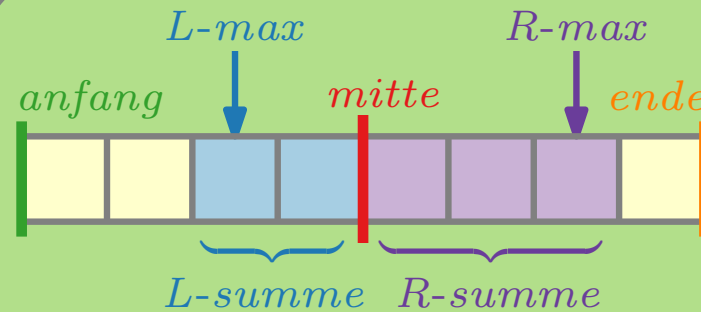
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

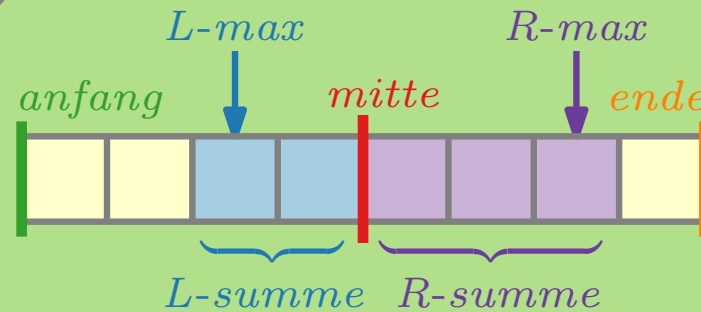
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

$$\text{summe} = S_{i, \text{mitte}}$$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

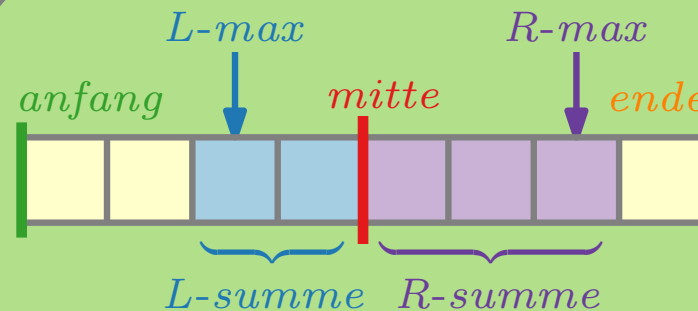
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

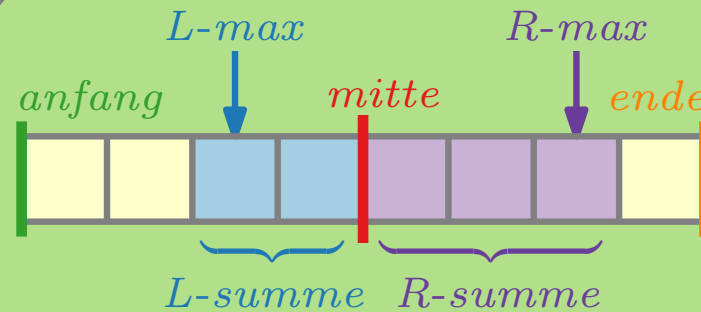
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$summe = 0$

for $i = \textit{mitte}$ **downto** $\textit{anfang}$ **do**

$summe = summe + A[i]$

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

$L\text{-summe} = summe$

$L\text{-max} = i$

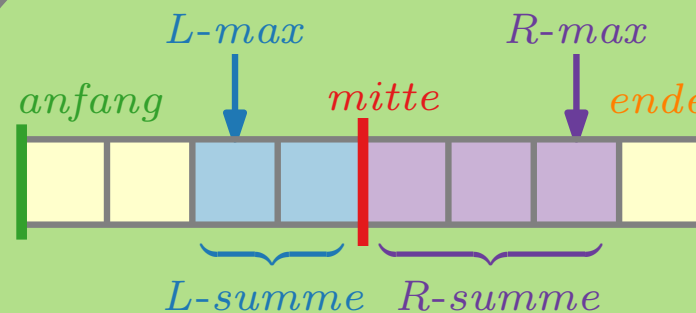
$R\text{-summe} = -\infty$

$summe = 0$

for $i = \textit{mitte} + 1$ **to** $\textit{ende}$ **do**

 // analog zu oben

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



Korrektheit?

Schleifeninvariante:

$summe = S_{i, \textit{mitte}}$ und

$L\text{-summe} =$

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

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

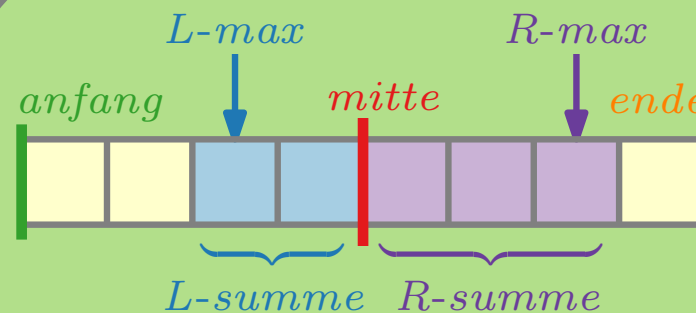
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

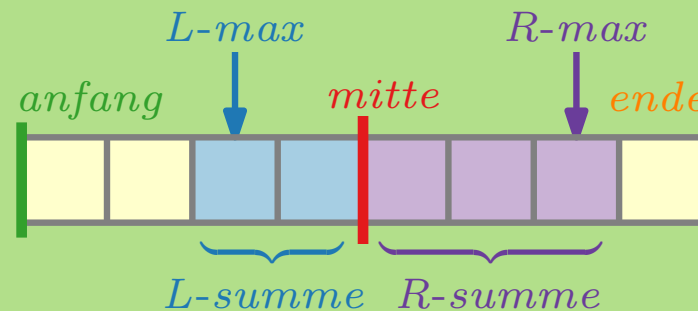
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

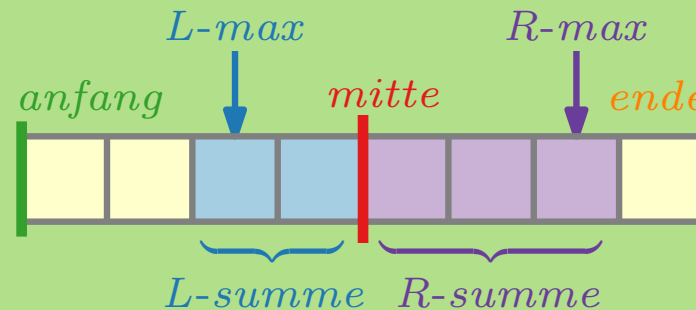
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

:= hier Anz. Additionen

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$summe = 0$

for $i = \textit{mitte}$ **downto** $\textit{anfang}$ **do**

$summe = summe + A[i]$

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

$L\text{-summe} = summe$

$L\text{-max} = i$

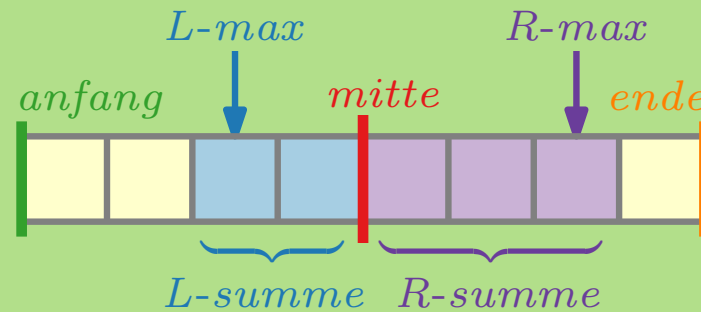
$R\text{-summe} = -\infty$

$summe = 0$

for $i = \textit{mitte} + 1$ **to** $\textit{ende}$ **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i, \textit{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

:=_{hier} Anz. Additionen

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

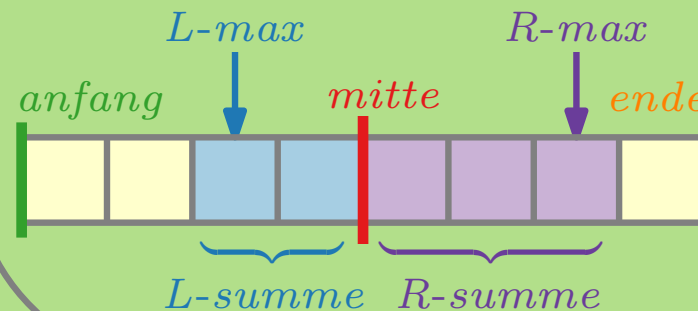
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben +

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

:= hier Anz. Additionen

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

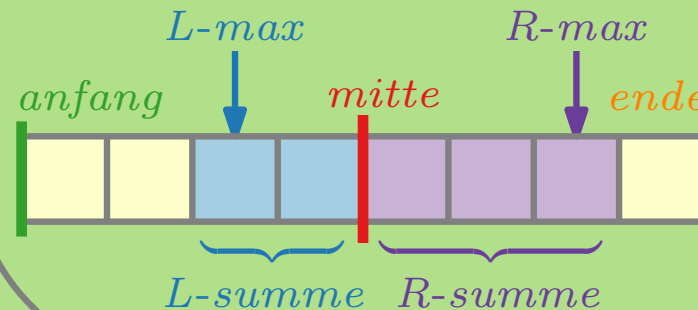
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben +

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

:= hier Anz. Additionen

$= \text{mitte} - \text{anfang} + 1$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

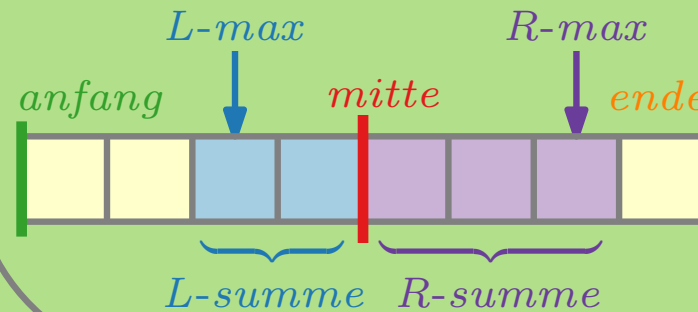
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

:= hier Anz. Additionen

$= \text{mitte} - \text{anfang} + 1$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

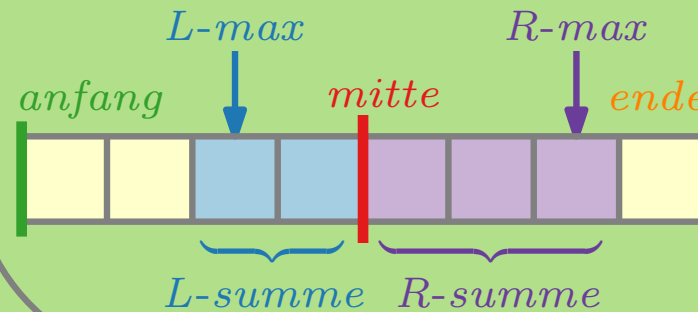
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

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

$= \text{mitte} - \text{anfang} + 1$

$+ \text{ende} - \text{mitte}$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

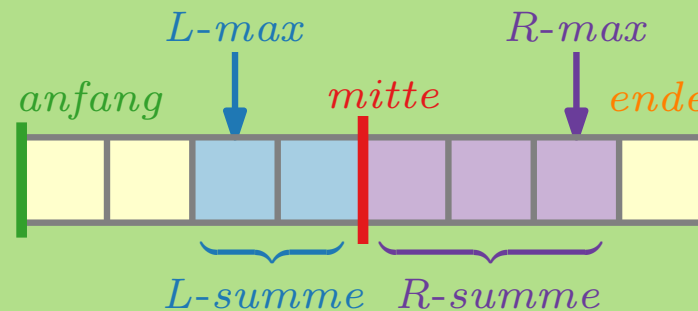
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben +

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

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

$= \text{mitte} - \text{anfang} + 1$

$+ \text{ende} - \text{mitte}$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

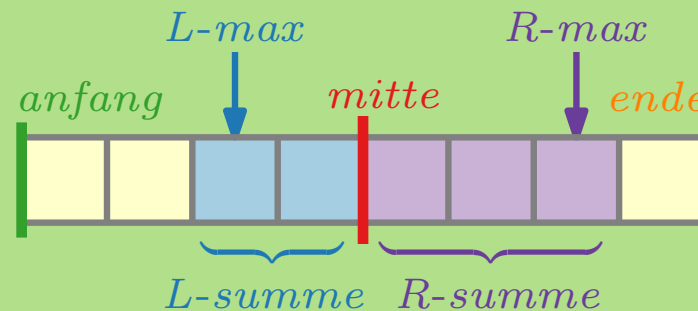
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

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

$= \text{mitte} - \text{anfang} + 1$

$+ \text{ende} - \text{mitte}$

$= \text{ende} - \text{anfang} + 1$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$\text{summe} = 0$

for $i = \text{mitte}$ **downto** anfang **do**

$\text{summe} = \text{summe} + A[i]$

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

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

$L\text{-max} = i$

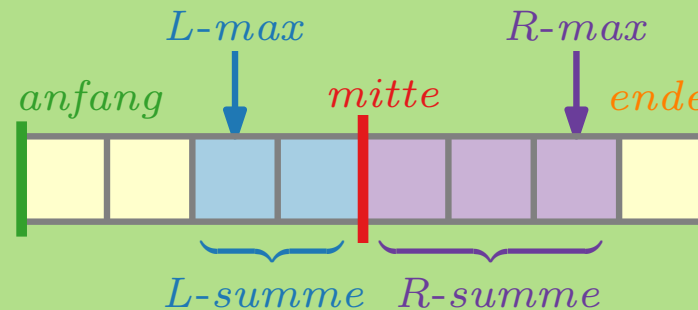
$R\text{-summe} = -\infty$

$\text{summe} = 0$

for $i = \text{mitte} + 1$ **to** ende **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$\text{summe} = S_{i, \text{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit?

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

$= \text{mitte} - \text{anfang} + 1$

$+ \text{ende} - \text{mitte}$

$= \text{ende} - \text{anfang} + 1$

$= n$

Kombiniere

MAXMITTEILFELD(int[] A, int *anfang*, int *mitte*, int *ende*)

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

$summe = 0$

for $i = \textit{mitte}$ **downto** $\textit{anfang}$ **do**

$summe = summe + A[i]$

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

$L\text{-summe} = summe$

$L\text{-max} = i$

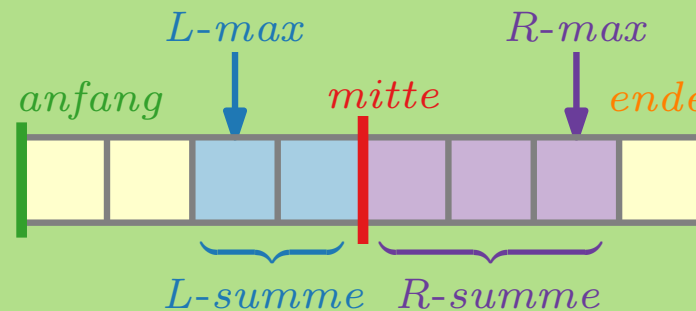
$R\text{-summe} = -\infty$

$summe = 0$

for $i = \textit{mitte} + 1$ **to** $\textit{ende}$ **do**

 // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i, \textit{mitte}}$ und

$L\text{-summe} =$

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

Laufzeit? ✓

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

$= \textit{mitte} - \textit{anfang} + 1$

$+ \textit{ende} - \textit{mitte}$

$= \textit{ende} - \textit{anfang} + 1$

$= n$

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:

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

für $n > 1$:

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

Putting Things Together

Laufzeit von MAXTEILFELD:

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

für $n > 1$:

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

MERGESORT

Putting Things Together

Laufzeit von MAXTEILFELD:

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

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

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

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

MERGESORT

Putting Things Together

Laufzeit von MAXTEILFELD:

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

für $n > 1$:

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

MERGESORT

Putting Things Together

Laufzeit von MAXTEILFELD:

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

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

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

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

MERGESORT

Putting Things Together

Laufzeit von MAXTEILFELD:

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

für $n > 1$: $T_{\text{MT}}(n) \approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{M} \textcolor{red}{M} \text{T}}(n)$

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

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

MERGESORT

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

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

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

MERGESORT

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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



Denkaufgaben:

- Lösen Sie MAXSUM in $\mathcal{O}(n)$ – also in linearer – Zeit!

Putting Things Together

Laufzeit von MAXTEILFELD:

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

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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



Denkaufgaben:

- Lösen Sie MAXSUM in $\mathcal{O}(n)$ – also in linearer – Zeit!
- Und wenn...

Putting Things Together

Laufzeit von MAXTEILFELD:

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

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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



Denkaufgaben:

■ Lösen Sie MAXSUM in $\mathcal{O}(n)$ – also in linearer – Zeit!

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

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

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

MERGESORT

„Runde auf“ zu nächster
Zweierpotenz $n' < 2n$

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



Denkaufgaben:

■ Lösen Sie MAXSUM in $\mathcal{O}(n)$ – also in linearer – Zeit!

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

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

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	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern			
Teile & Herrsche			
Linearer Scan			

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus	Laufzeit		
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche			
Linearer Scan			

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus	Laufzeit		
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan			

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

Algorithmus	Laufzeit		
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan	$\mathcal{O}(n)$		

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

■ Anzahl der Additionen

Algorithmus	Laufzeit		
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan	$\mathcal{O}(n)$		

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

Algorithmus	Laufzeit		
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan	$\mathcal{O}(n)$		

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

Algorithmus	Laufzeit	$n = 1\,000$	
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan	$\mathcal{O}(n)$		

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

Algorithmus	Laufzeit	$n = 1\,000$	
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log n)$		
Linearer Scan	$\mathcal{O}(n)$	10^3	

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

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

Übersicht

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

■ Anzahl der Additionen

Algorithmus	Laufzeit	$n = 1\,000$	
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$		
Teile & Herrsche	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	
Linearer Scan	$\mathcal{O}(n)$	10^3	

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

Algorithmus	Laufzeit	$n = 1\,000$	
Rohe Gewalt	$\mathcal{O}(n^3)$		
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$	10^6	
Teile & Herrsche	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	
Linearer Scan	$\mathcal{O}(n)$	10^3	

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

■ Anzahl der Additionen

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

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

(siehe Buch [CLRS],
Ü-Aufgabe 4.1-5)

Übersicht

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

- Anzahl der Additionen
- geschätzte Rechenzeit

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

Algorithmus	Laufzeit	$n = 1\,000$	$n = 1\,000\,000$
Rohe Gewalt	$\mathcal{O}(n^3)$	10^9 1 s	10^{18} 31,7 y
Reihenfolge der Summen ändern	$\mathcal{O}(n^2)$	10^6 1 ms	10^{12} 1000 s = 17 min
Teile & Herrsche	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$ 3 μ s	$6 \cdot 10^6$ 6 ms
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3 1 μ s	10^6 1 ms