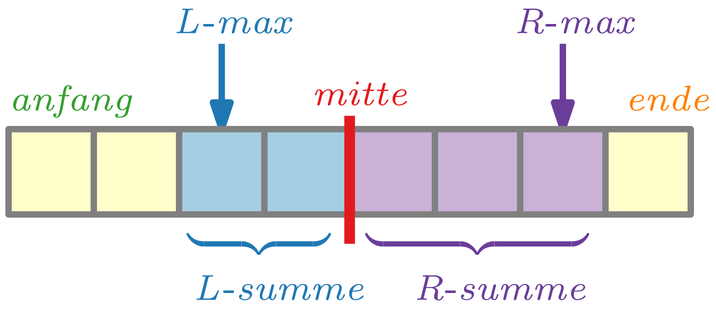
Algorithmen und Datenstrukturen



Vorlesung 4: Maximales Teilfeld

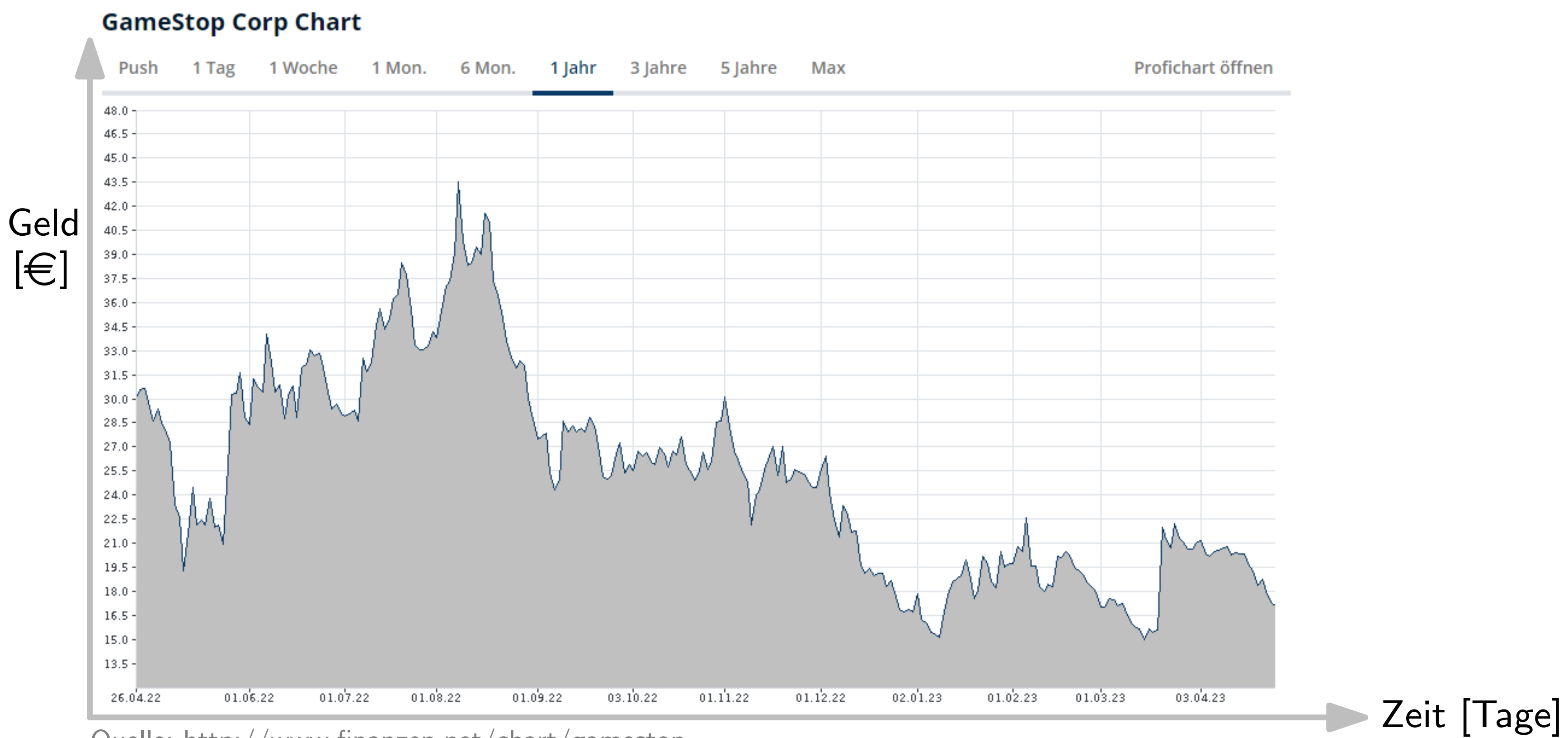
Alexander Wolff

Wintersemester 2025

Analyse von Aktienkursen

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

Analyse von Aktienkursen



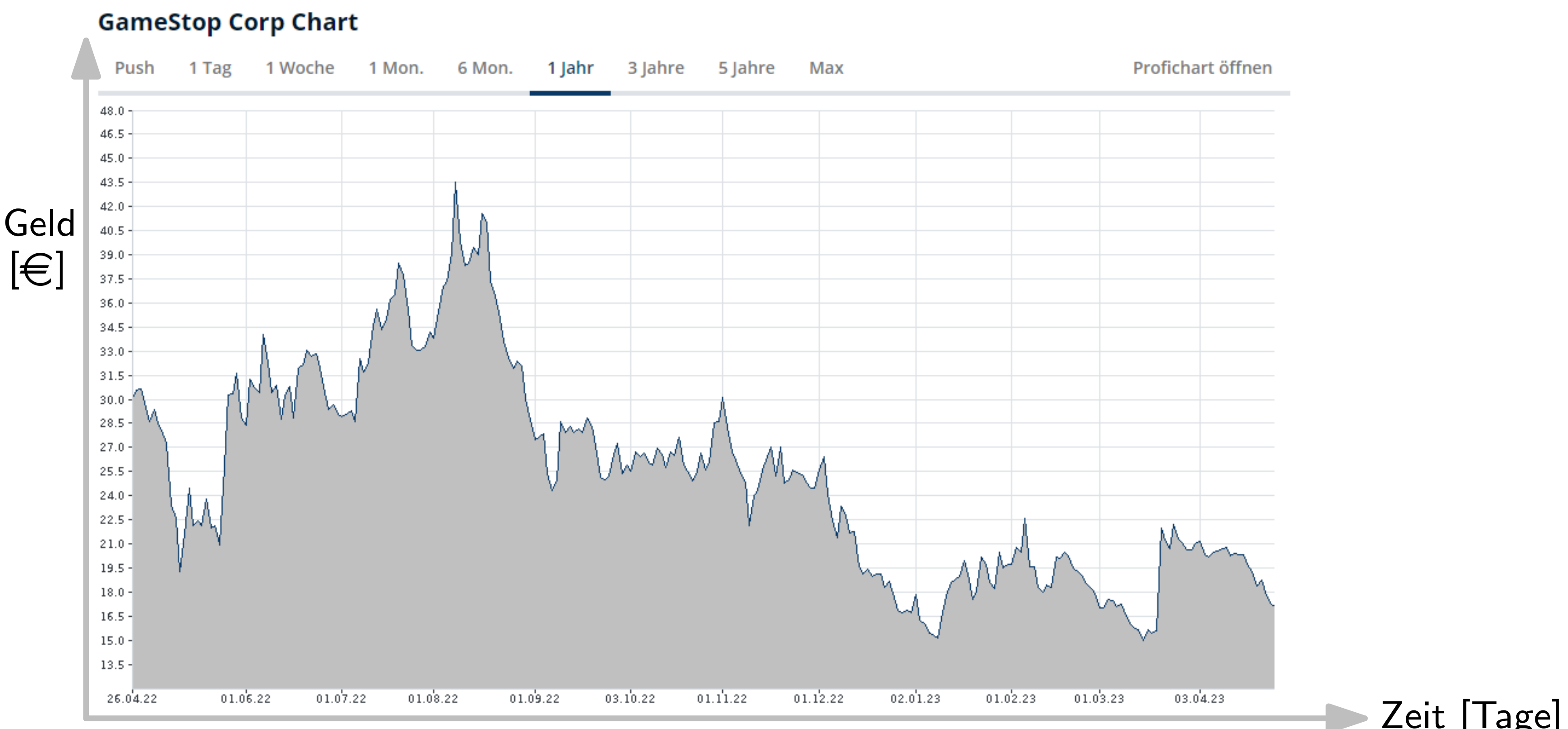
Quelle: http://www.finanzen.net/chart/gamestop

Analyse von Aktienkursen

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

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

Analyse von Aktienkursen

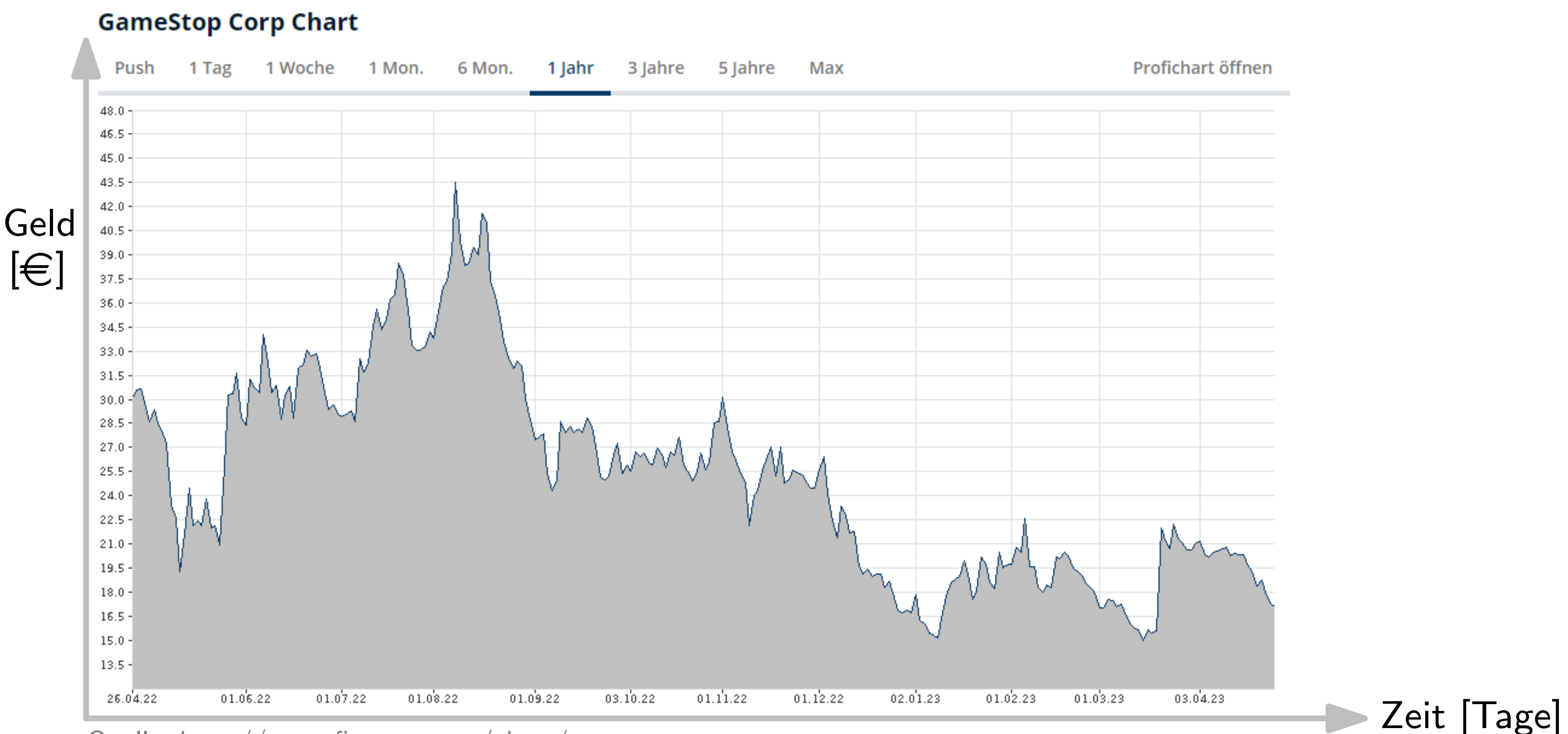


Quelle: http://www.finanzen.net/chart/gamestop

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

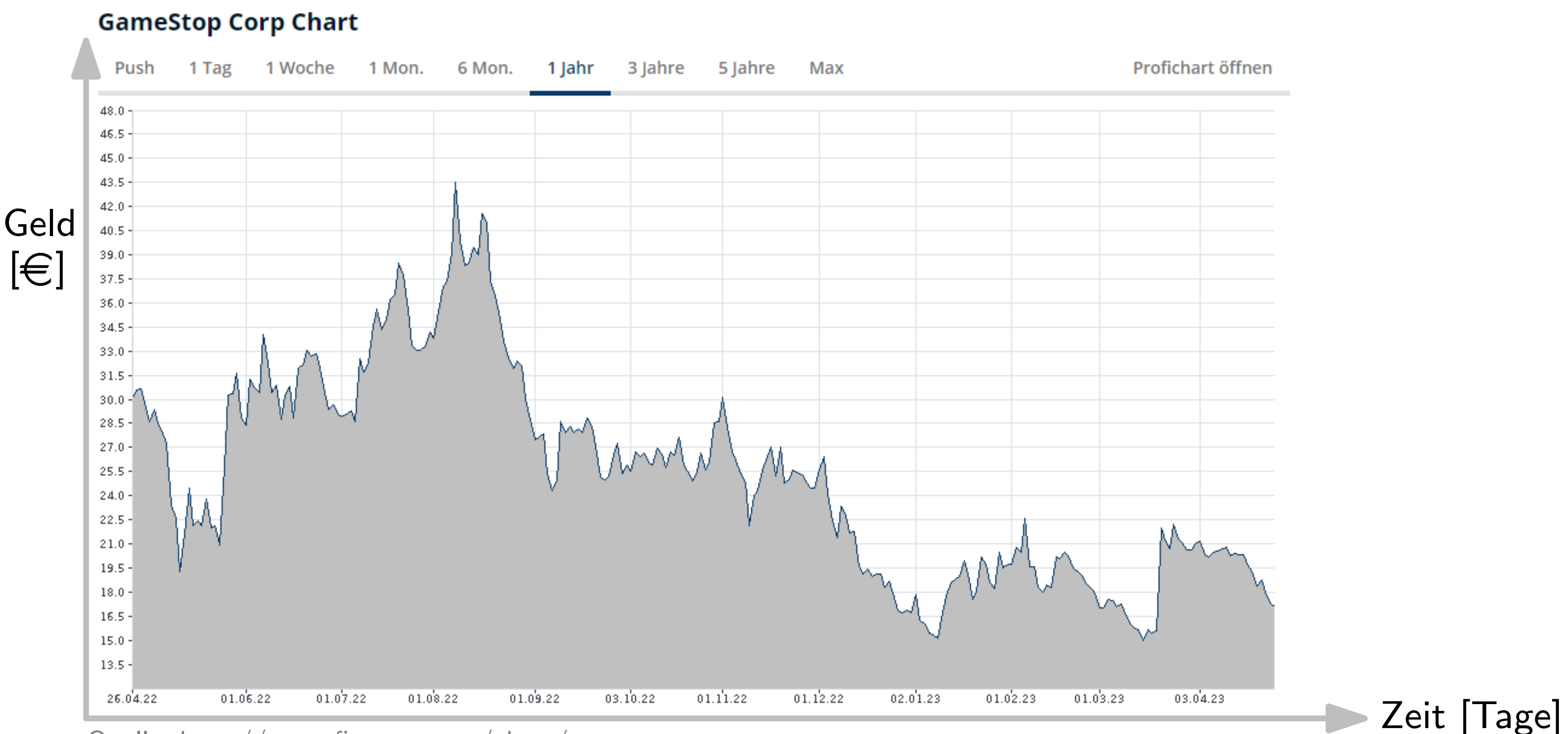


Quelle: http://www.finanzen.net/chart/gamestop

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



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

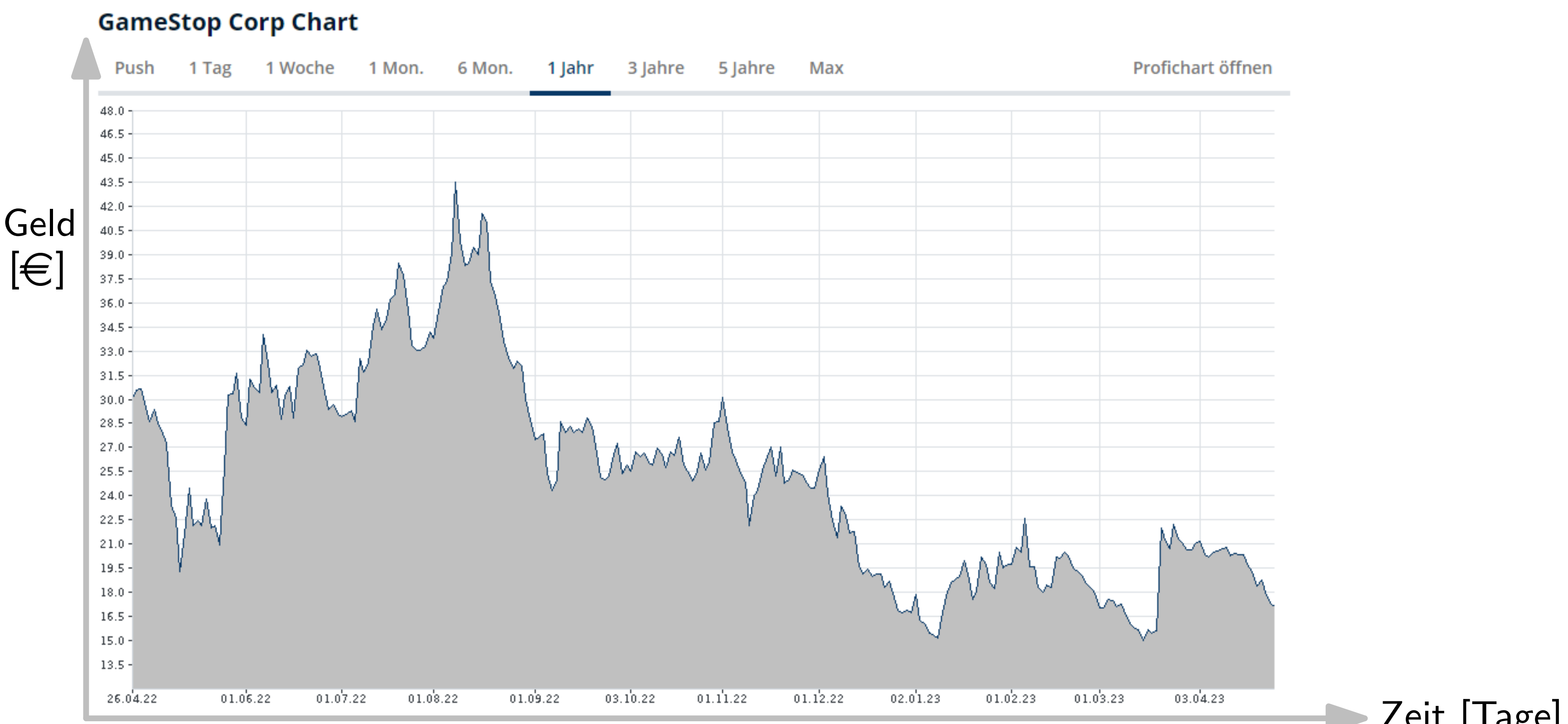
Problem. **Gegeben:** Eine Folge $A[1 \ldots n]$ von Aktienkursen in Euro.

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

Verkaufskurs

Einkaufskurs

Analyse von Aktienkursen



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

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

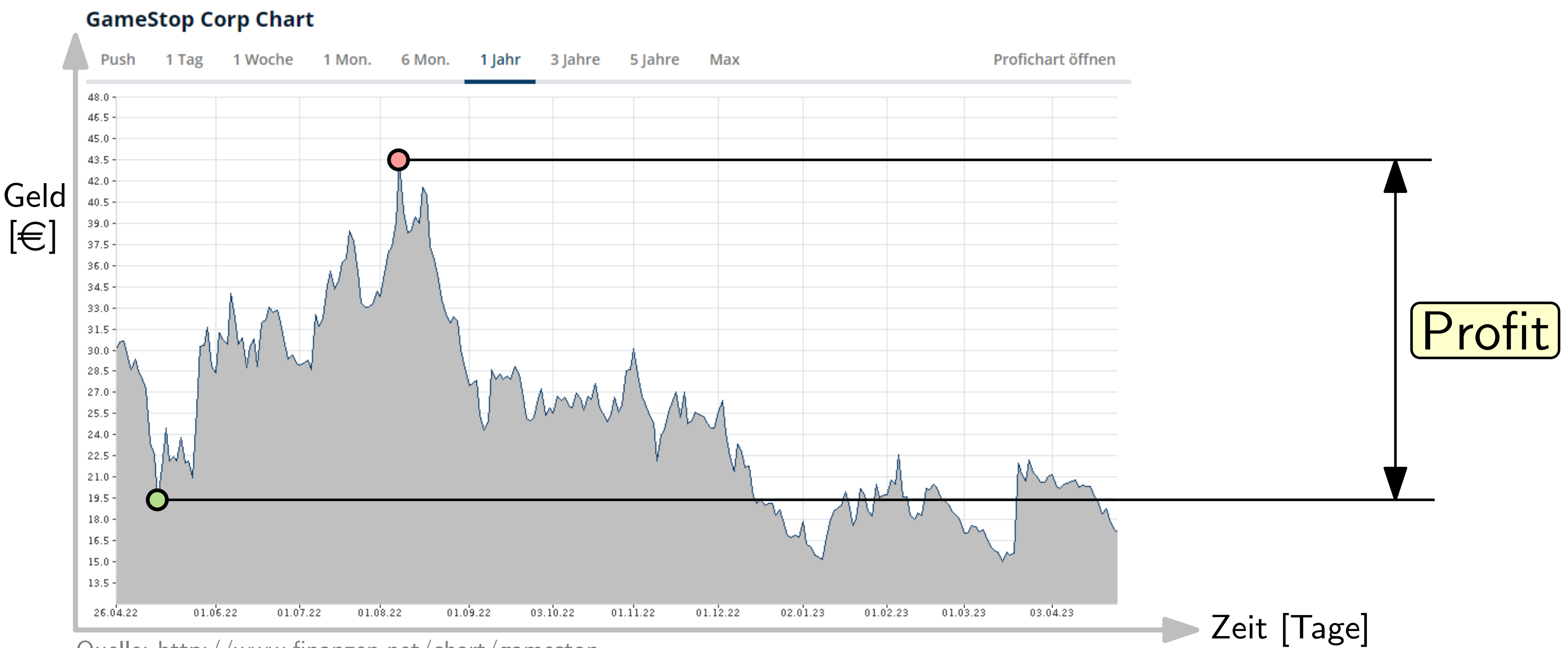
Analyse von Aktienkursen

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

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

Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $\underbrace{\underbrace{A[j]}_{\text{Verkaufskurs}} - \underbrace{A[i]}_{\text{Einkaufskurs}}}_{\text{Profit pro Aktie}}$ **maximal**.

Analyse von Aktienkursen

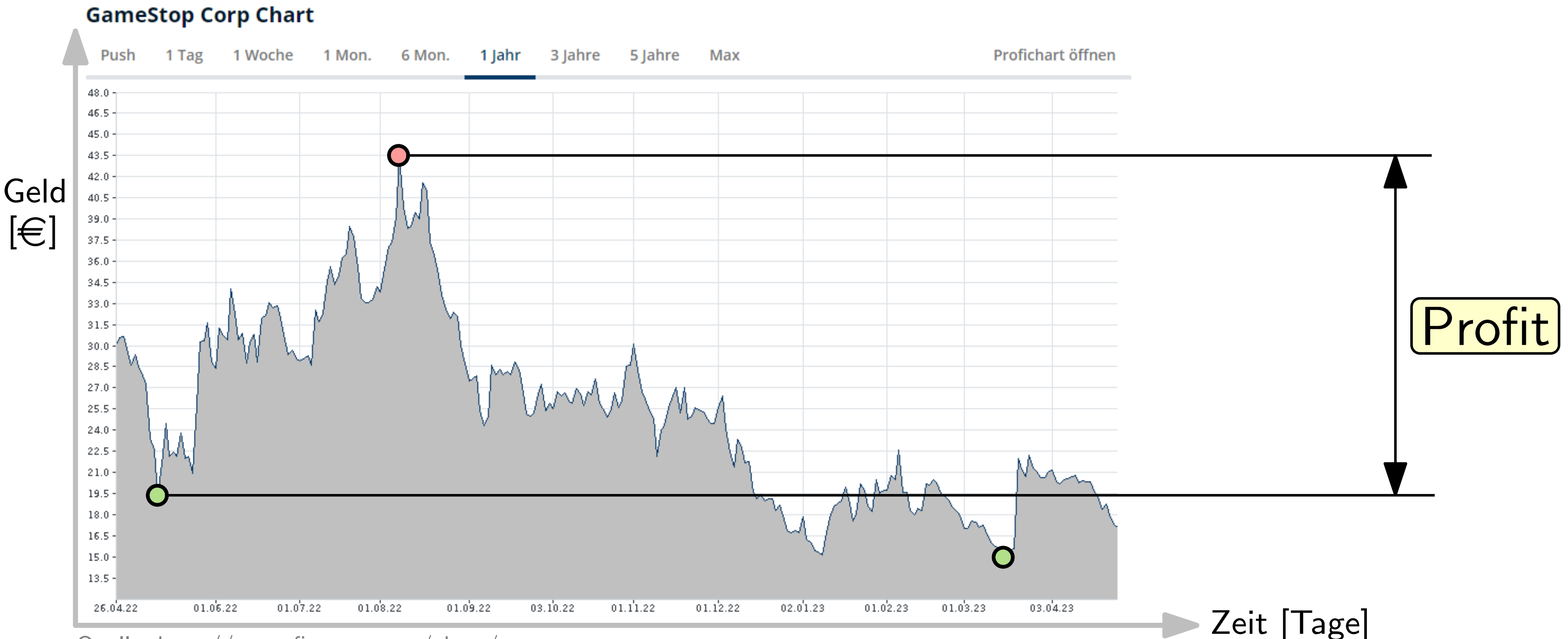


Quelle: http://www.finanzen.net/chart/gamestop

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

Analyse von Aktienkursen



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

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

Analyse von Aktienkursen

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

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

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 $\underbrace{\underset{\text{Verkaufskurs}}{A[j]} - \underset{\text{Einkaufskurs}}{A[i]}}_{\text{Profit pro Aktie}}$ **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. 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 $A[1 \ldots n]$ 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 $A[1 \ldots n]$ 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

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

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

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

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

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

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

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

Ein ähnliches Problem

Problem. MaxSum

Problem. MaxDiff

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

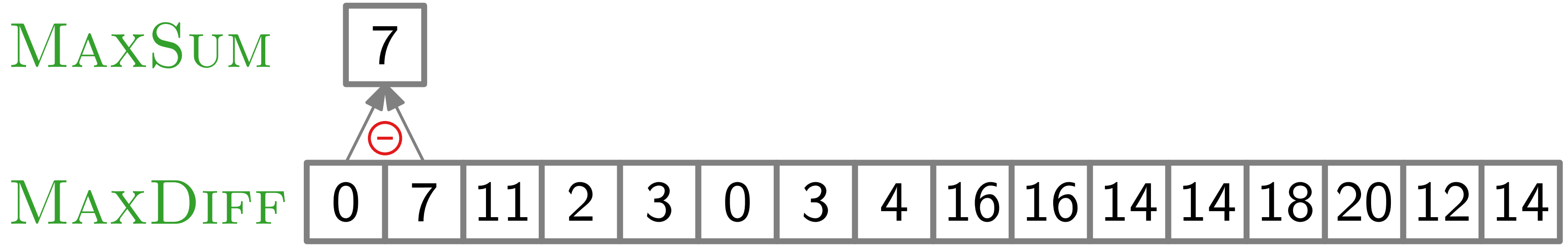
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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.

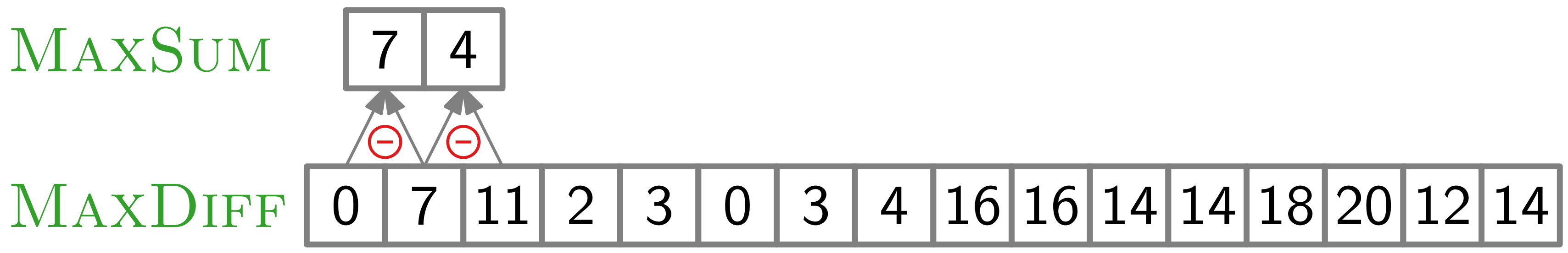
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.



Ein ähnliches Problem

Problem. $\textsc{MaxSum}$
Gegeben: Eine Folge $A[1 \ldots n]$ 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. $\textsc{MaxDiff}$
Gegeben: Eine Folge $A[1 \ldots n]$ 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**.

MaxSum

7	4	-9

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 $A[1 \ldots n]$ 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 $A[1 \ldots n]$ von Aktienkursen in Euro.

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

MaxSum	7	4	-9

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

Ein ähnliches Problem

Problem. MaxSum

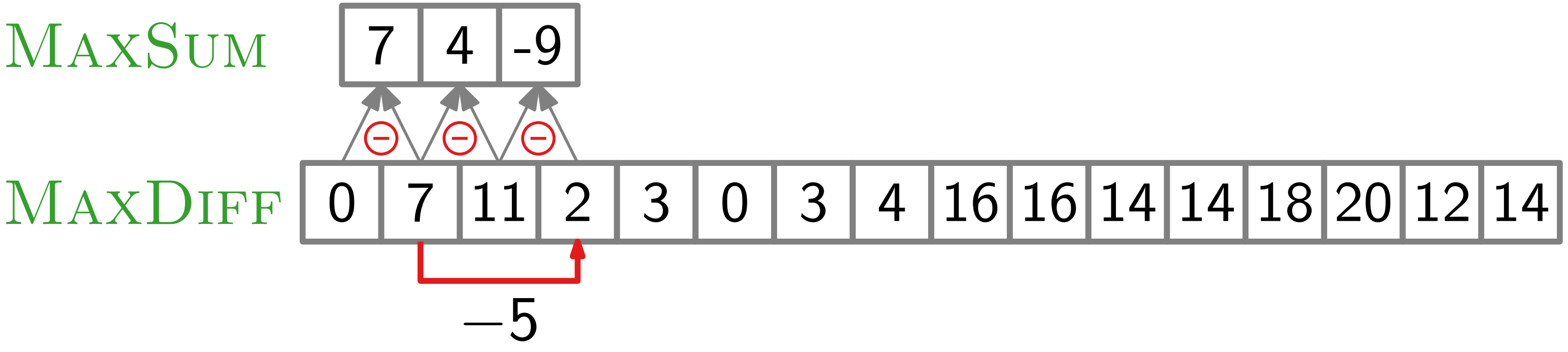
Gegeben: Eine Folge $A[1 \dots n]$ von ganzen Zahlen.

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.

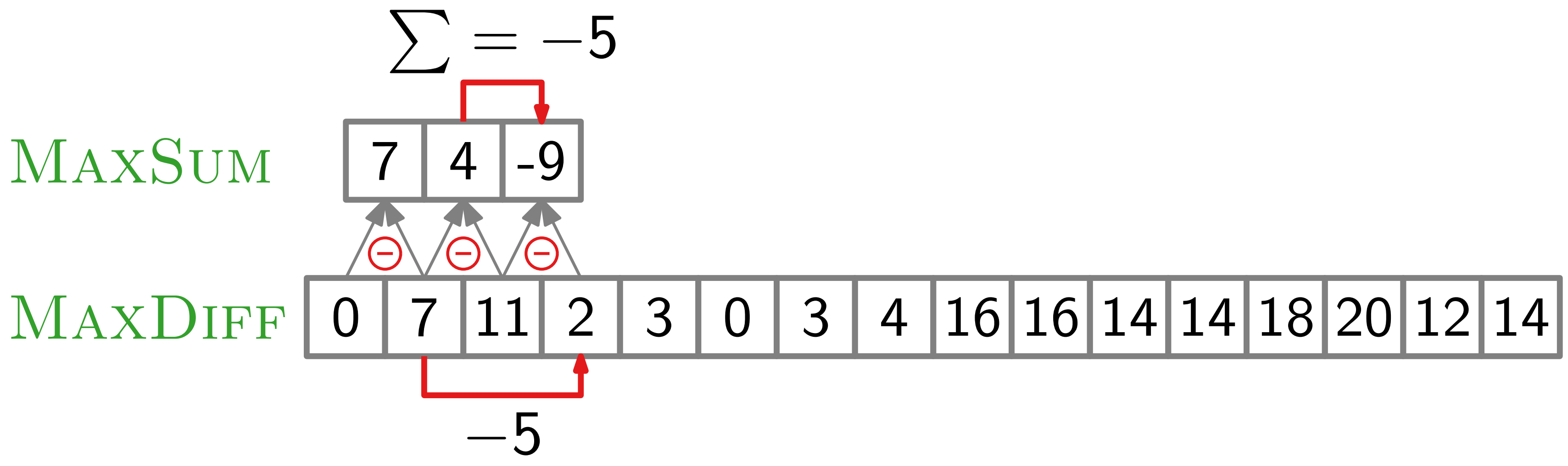
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.



Ein ähnliches Problem

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

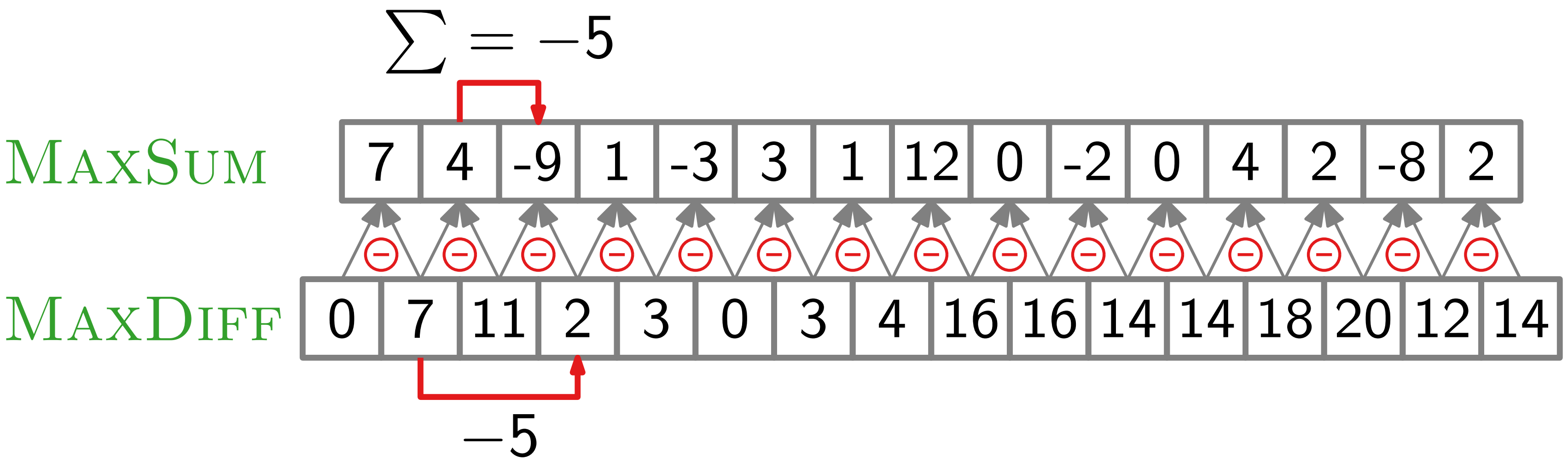
Problem. MaxDiff
Gegeben: Eine Folge $A[1 \ldots n]$ von Aktienkursen in Euro.
Gesucht: Paar (i, j) mit $1 \leq i < j \leq n$, so dass $A[j] - A[i]$ **maximal**.



Ein ähnliches Problem

Problem. MAXSUM
Gegeben: Eine Folge $A[1 \ldots n]$ 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 $A[1 \ldots n]$ 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 $A[1 \ldots n]$ 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 $A[1 \ldots n]$ von ganzen Zahlen.

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

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

$\Rightarrow (6, 13)$ 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. $\textsc{MaxSum}$

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

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

$\Rightarrow (6, 13)$ 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 $A[1 \ldots n]$ 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 $1 \le i \le j \le n$, 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: $\mathcal{O}(n^2) \cdot$

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 $1 \le i \le j \le n$, 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: $\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$
Untere Schranke

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

Untere Schranke (Anz. Paare)

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

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

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

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

Wo ist die Wahrheit?

Genauere Analyse

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

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

Genauere Analyse

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

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

Beobachtung.

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

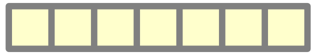
Genauere Analyse

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 

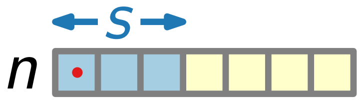
Genauere Analyse

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



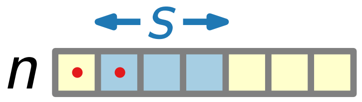
Genauere Analyse

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



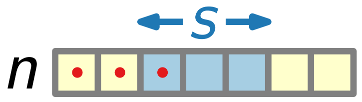
Genauere Analyse

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



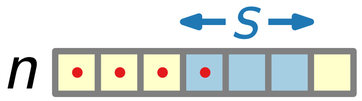
Genauere Analyse

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



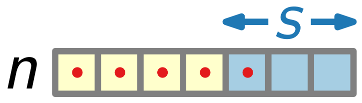
Genauere Analyse

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



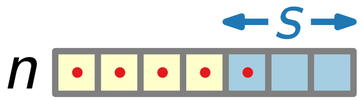
Genauere Analyse

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 $n - s + 1$.



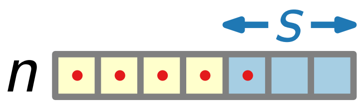
Genauere Analyse

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 **?** Additionen.



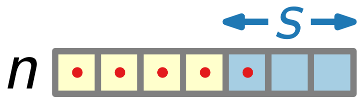
Genauere Analyse

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

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

Beobachtung.

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



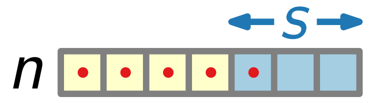
Genauere Analyse

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

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

Beobachtung.

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



$\Rightarrow$ Anz. Add. $=$

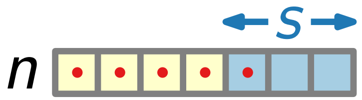
Genauere Analyse

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

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

Beobachtung.

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



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

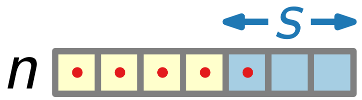
Genauere Analyse

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

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

Beobachtung.

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



- s Summanden benötigen $s - 1$ Additionen.

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

$= n \cdot 0$

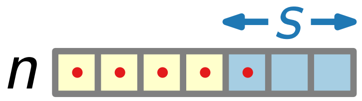
Genauere Analyse

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.



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

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

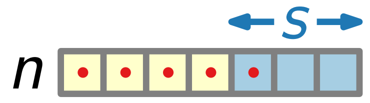
Genauere Analyse

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

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

Beobachtung.

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



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

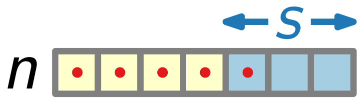
Genauere Analyse

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.



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

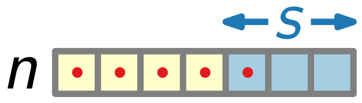
Genauere Analyse

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.



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

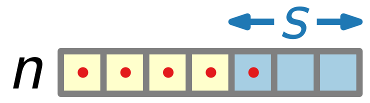
Genauere Analyse

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.



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

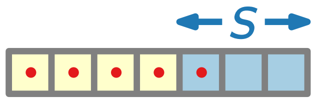
Genauere Analyse

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

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

Beobachtung.

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

n 

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

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

$$\underset{(\text{falls } 4 \mid n)}{=} \qquad \dots + \qquad\qquad + \dots$$

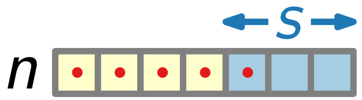
Genauere Analyse

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.



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

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

$$\underset{(\text{falls } 4 \mid n)}{=} \ldots + \overbrace{\ldots + \frac{n}{2} \cdot \frac{n}{2} + \ldots} + \ldots$$

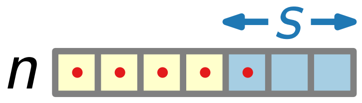
Genauere Analyse

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

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

Beobachtung.

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



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

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

$$\underset{(\text{falls } 4 \mid n)}{=} \ldots + \frac{3n}{4} \cdot \frac{n}{4} + \ldots + \frac{n}{2} \cdot \frac{n}{2} + \ldots + \ldots$$

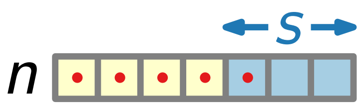
Genauere Analyse

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

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

Beobachtung.

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



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

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

$$\underset{(\text{falls } 4 \mid n)}{=} \ldots + \frac{3n}{4} \cdot \frac{n}{4} + \ldots + \frac{n}{2} \cdot \frac{n}{2} + \ldots + \frac{n}{4} \cdot \frac{3n}{4} + \ldots$$

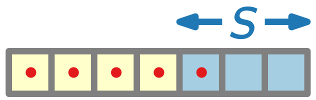
Genauere Analyse

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

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

Beobachtung.

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

n 

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

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

$$\underset{(\text{falls } 4 \mid n)}{=} \ldots + \frac{3n}{4} \cdot \frac{n}{4} + \ldots + \frac{n}{2} \cdot \frac{n}{2} + \ldots + \frac{n}{4} \cdot \frac{3n}{4} + \ldots$$

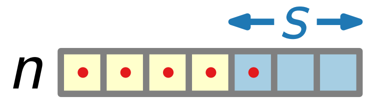
Genauere Analyse

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

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

Beobachtung.

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



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

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

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

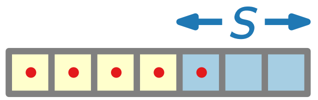
Genauere Analyse

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

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

Beobachtung.

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

n 

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

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

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

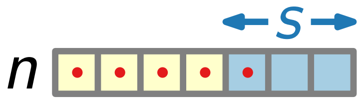
Genauere Analyse

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.



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

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

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

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

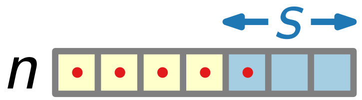
Genauere Analyse

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.



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

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

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

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

Geht das besser?

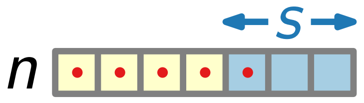
Genauere Analyse

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.



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

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

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

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?

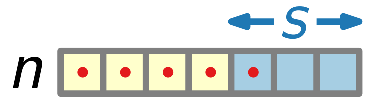
Genauere Analyse

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.



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

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

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

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?

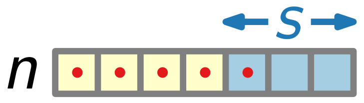
Genauere Analyse

Laufzeit. $\approx$ 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.



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

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

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

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

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

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

Geht das besser?

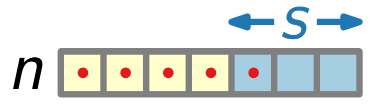
Genauere Analyse

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.



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

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

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

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

Wie berechnen?
$\#\,\text{Add.} = an^3 + bn^2 + cn + d$
Wertetabelle für $n = 1,2,3,4$.

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

Geht das besser?

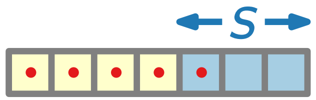
Genauere Analyse

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.

n 

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

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

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

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!

$\Rightarrow$ 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. $\textsc{MaxSum}$

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

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

Idee: Für $i = 1, \dots, n$

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

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

$=$

Wie? $A[i]$

Eine schnellere Lösung

Problem. MAXSUM

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

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

Idee: Für $i = 1, \dots, n$

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

$=$ $+$

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

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}, $S_{i,i+1}$, $S_{i,i+2}$, $S_{i,i+3}$, …, $S_{i,n}$

= + +

Wie? $A[i]$ $A[i+1]$ $A[i+2]$

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

$=$ $+$ $+$ $+$

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

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

$=$ $\quad +$ $\quad +$ $\quad +$ $\quad \dots$ $\quad +$

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

Eine schnellere Lösung

Problem. MaxSum

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

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

Idee: Für $i = 1, \ldots, n$

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

= + + + … +

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

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

$=$ $+$ $+$ $+$ $\dots$ $+$

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

Eine schnellere Lösung

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

Idee: Für $i = 1, \ldots, n$

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

$=$ $\quad + \quad + \quad + \quad \ldots \quad +$

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

Insgesamt

Eine schnellere Lösung

Problem. MaxSum

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

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

Idee: Für $i = 1, \ldots, n$

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

$\ominus \quad \oplus \quad \oplus \quad \oplus \quad \ldots \quad \oplus$

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

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

$=$ $+$ $+$ $+$ $\dots$ $+$

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

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

$=$ $+$ $+$ $+$ $\dots$ $+$

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

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

$= \quad + \quad + \quad + \quad \dots \quad +$

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

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

$=\quad +\quad +\quad +\quad \dots\quad +$

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

Insgesamt $\displaystyle\sum_{i=1}^{n} n - i \quad = \sum$

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

$=$ $+$ $+$ $+$ $\dots$ $+$

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

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? $A[i]$ $\underbrace{A[i+1] \quad A[i+2] \quad A[i+3] \quad \dots \quad A[n]}$

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

$= \quad + \quad + \quad + \quad \dots \quad +$

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

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

Eine schnellere Lösung

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

Idee: Für $i = 1, \ldots, n$

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

$=$ $+$ $+$ $+$ $\ldots$ $+$

Wie? $A[i]$ $\underbrace{A[i+1] \quad A[i+2] \quad A[i+3] \quad \ldots \quad A[n]}_{}$

$n - i$ Additionen

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

1) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ **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 $S_{ii}, \quad S_{i,i+1}, \quad S_{i,i+2}, \quad S_{i,i+3}, \quad \dots, \quad S_{i,n}$

$=$ $+$ $+$ $+$ $\dots$ $+$

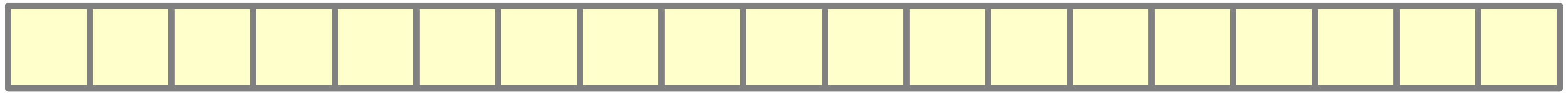
Wie? $A[i] \quad \underbrace{A[i+1] \quad A[i+2] \quad A[i+3] \quad \quad A[n]}_{n-i \text{ Additionen}}$

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**

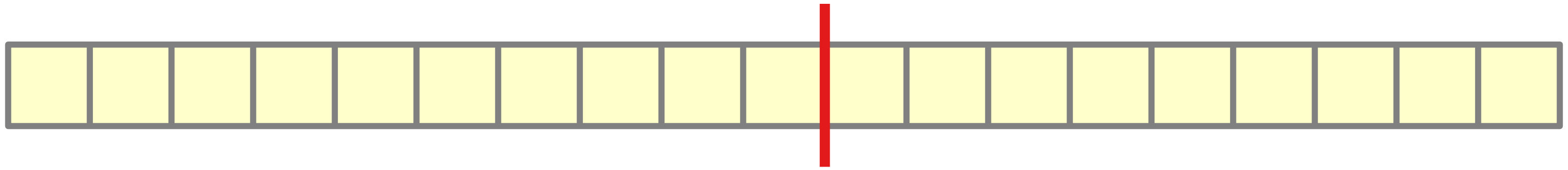
Eine noch schnellere Lösung?

Idee:



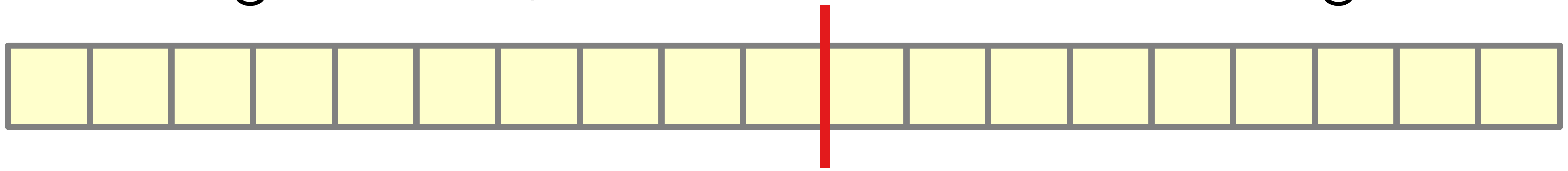
Eine noch schnellere Lösung?

Idee:



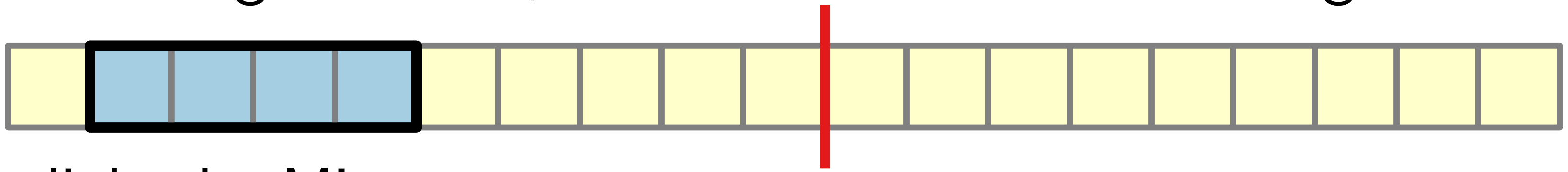
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

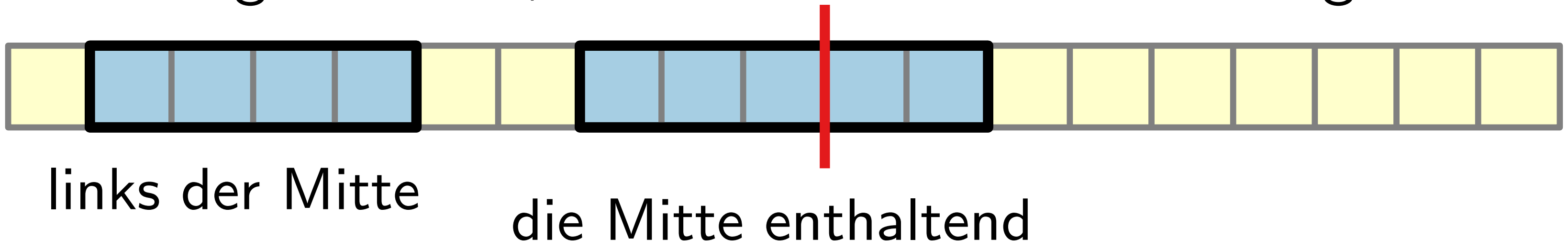
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



links der Mitte

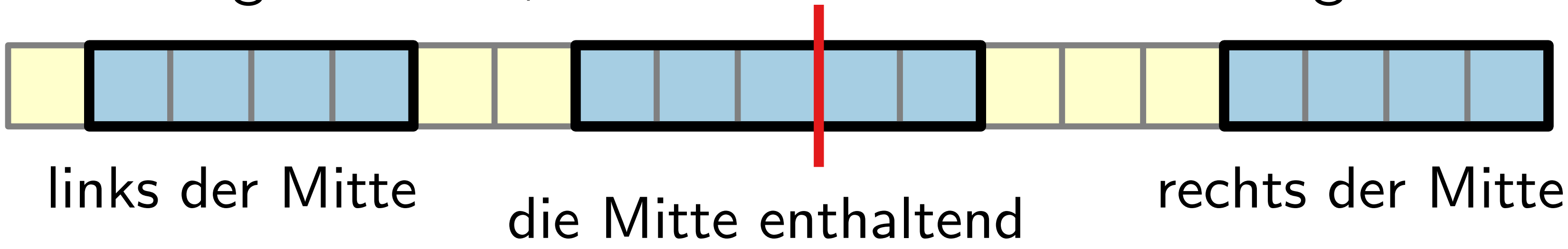
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



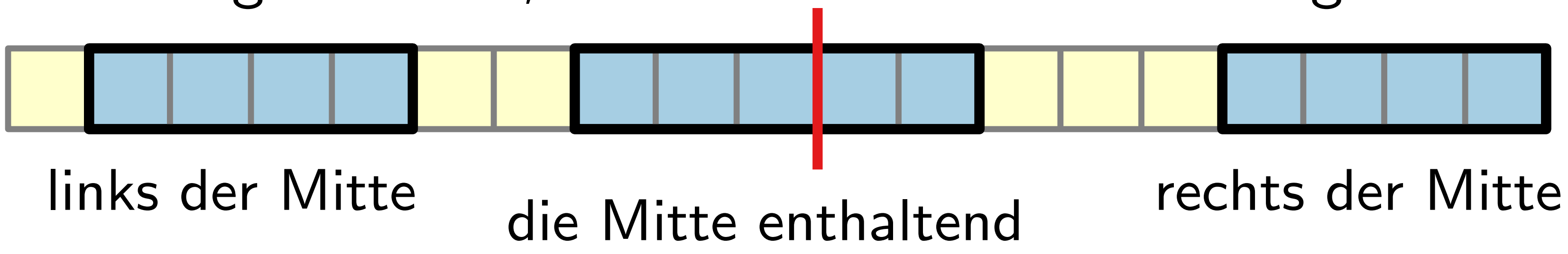
Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Eine noch schnellere Lösung?

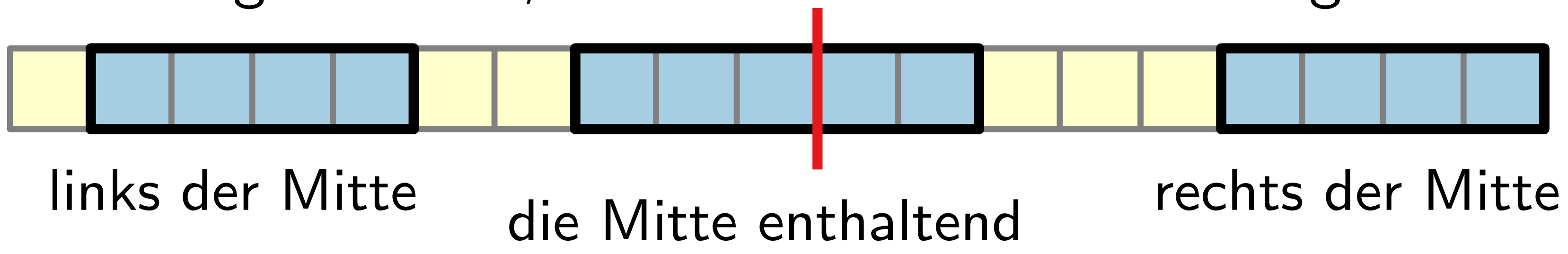
Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:



Nimm Entwurfstechnik **Teile & Herrsche**!

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

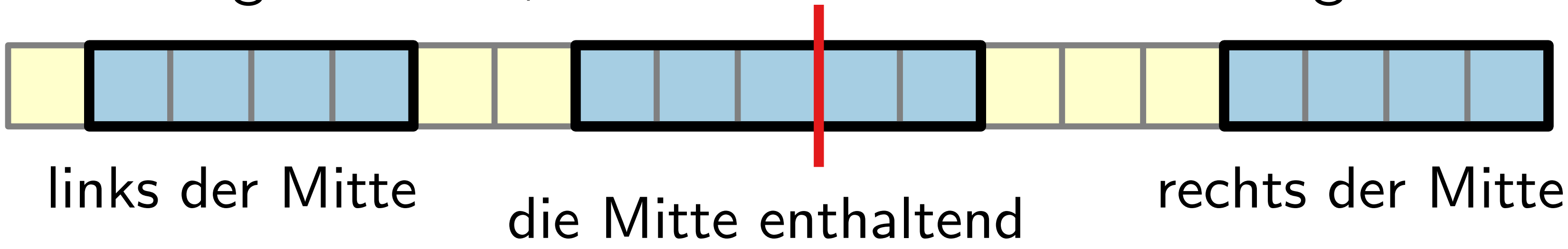


Nimm Entwurfstechnik **Teile & Herrsche**!

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

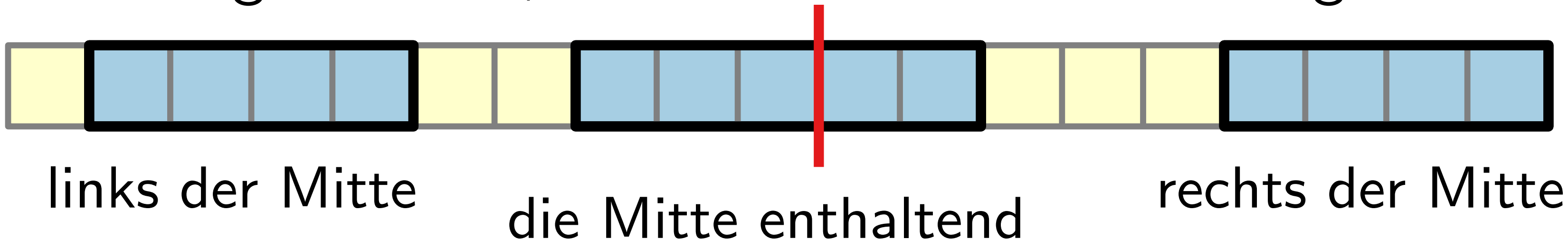


Nimm Entwurfstechnik **Teile & Herrsche**!

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

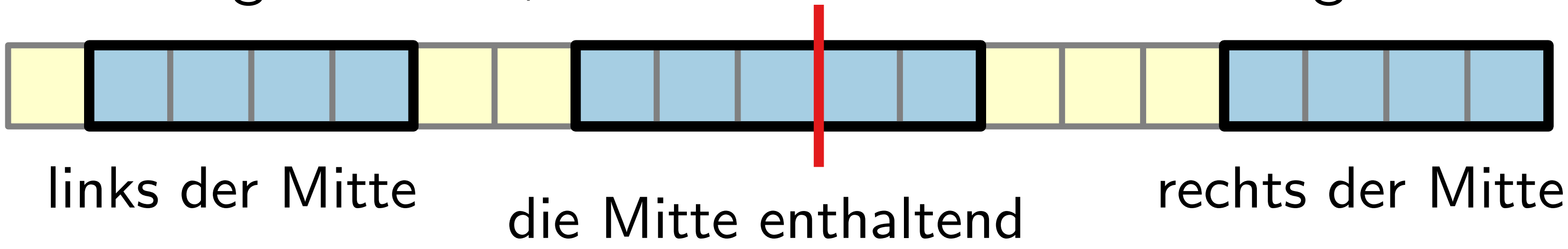


Nimm Entwurfstechnik **Teile & Herrsche**!

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

Eine noch schnellere Lösung?

Idee: Drei Möglichkeiten, wo maximale Teilsumme liegt:

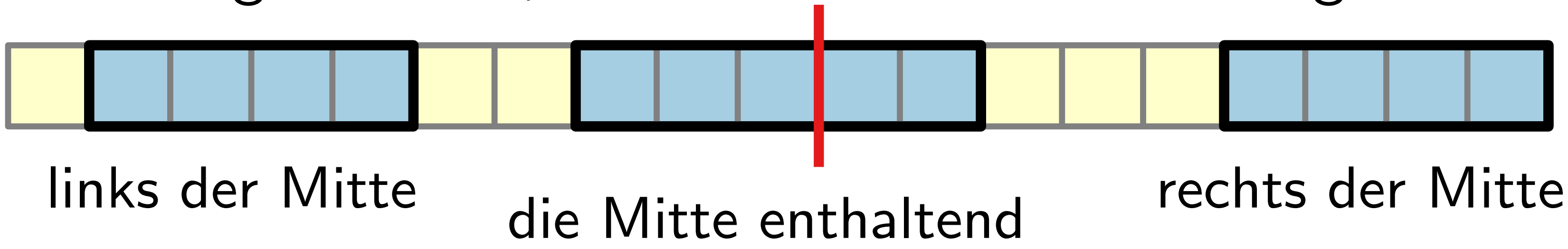


Nimm Entwurfstechnik **Teile & Herrsche**!

- **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:

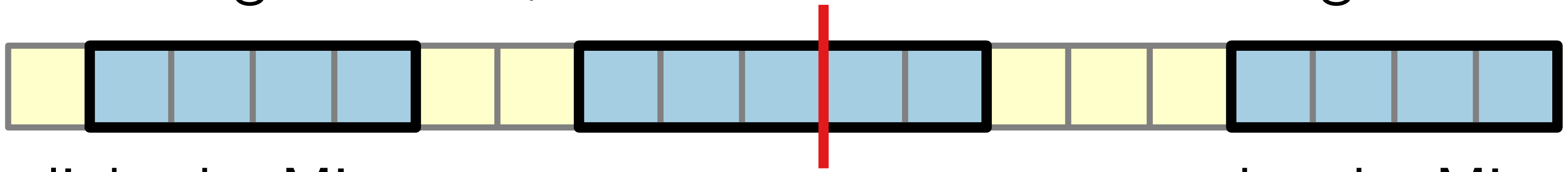


Nimm Entwurfstechnik **Teile & Herrsche**!

- **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:



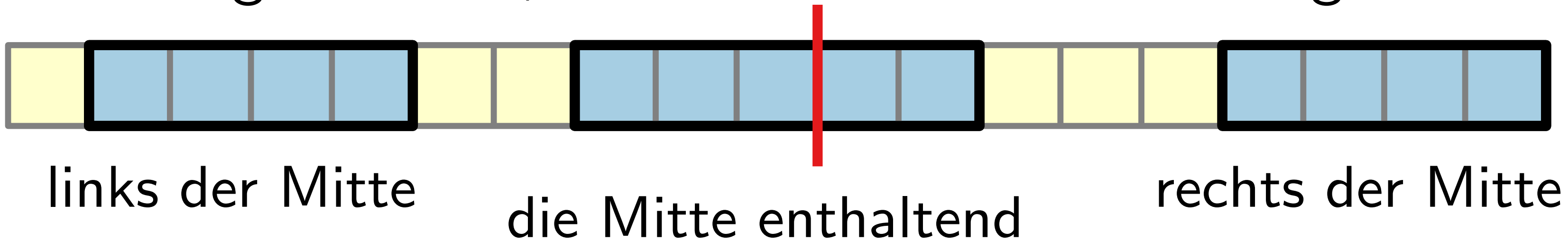
Nimm Entwurfstechnik **Teile & Herrsche**!

- **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:



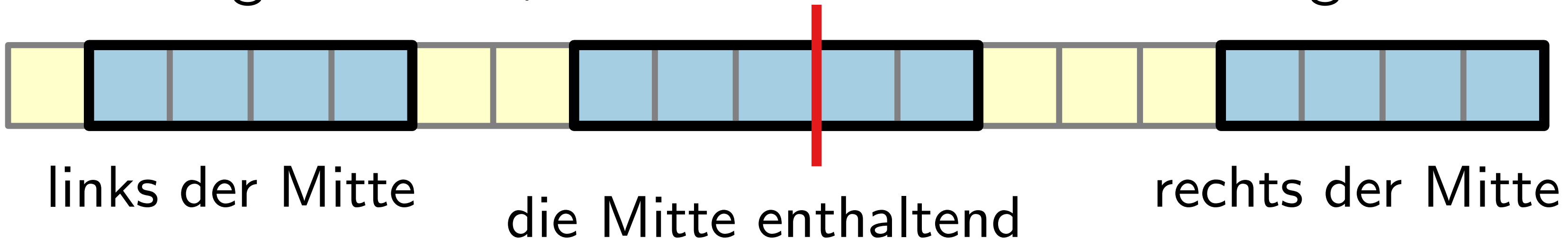
Nimm Entwurfstechnik **Teile & Herrsche**!

- **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:



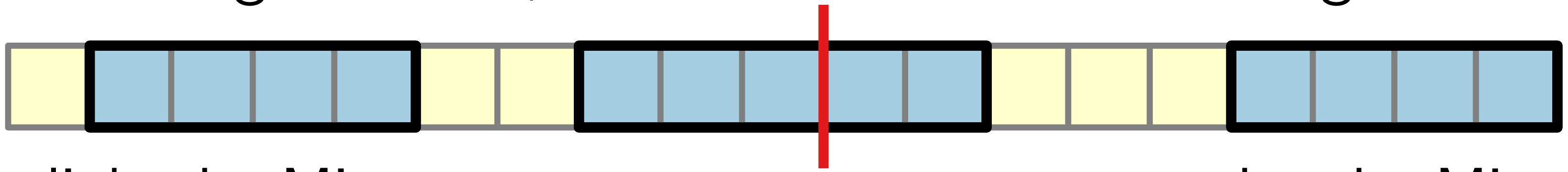
Nimm Entwurfstechnik **Teile & Herrsche**!

- **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:



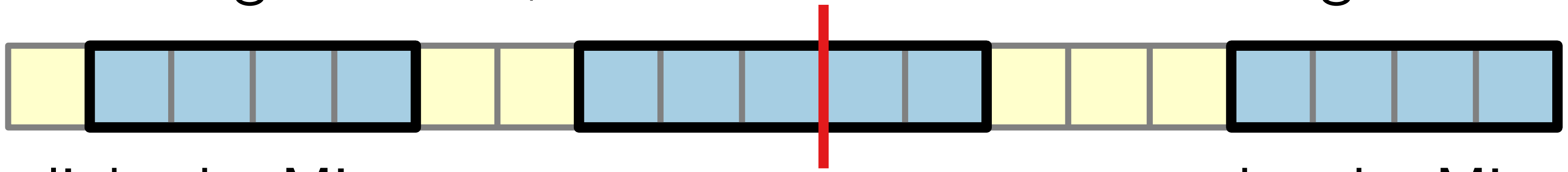
Nimm Entwurfstechnik **Teile & Herrsche**!

- **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**!

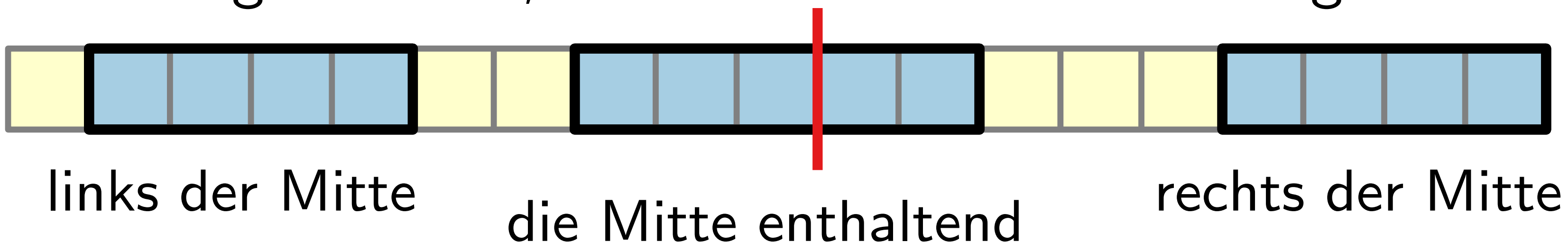
- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere alle Teilsummen, die die Mitte enthalten

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**!

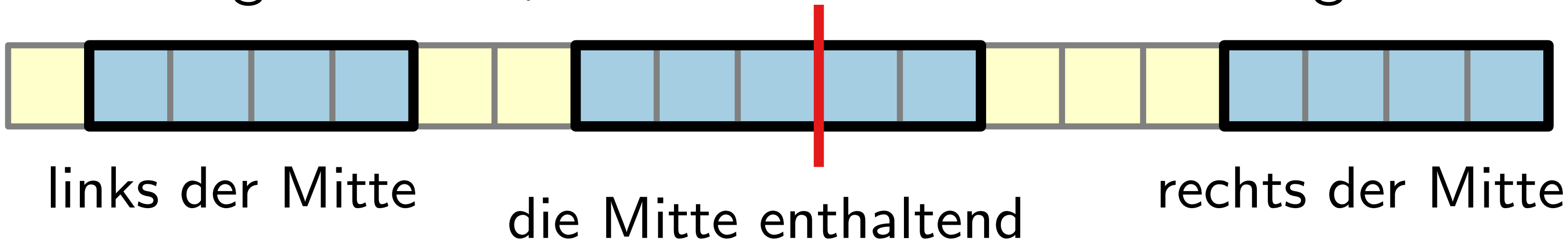
- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere alle Teilsummen, die die Mitte enthalten

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**!

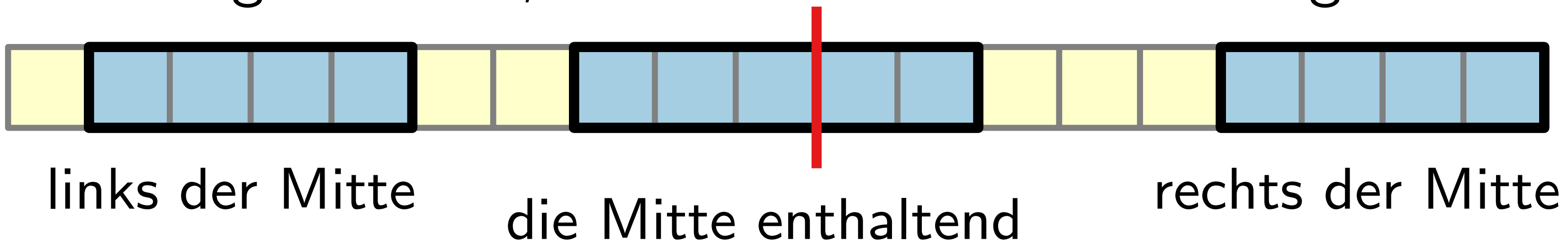
- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere alle Teilsummen, die die Mitte enthalten

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

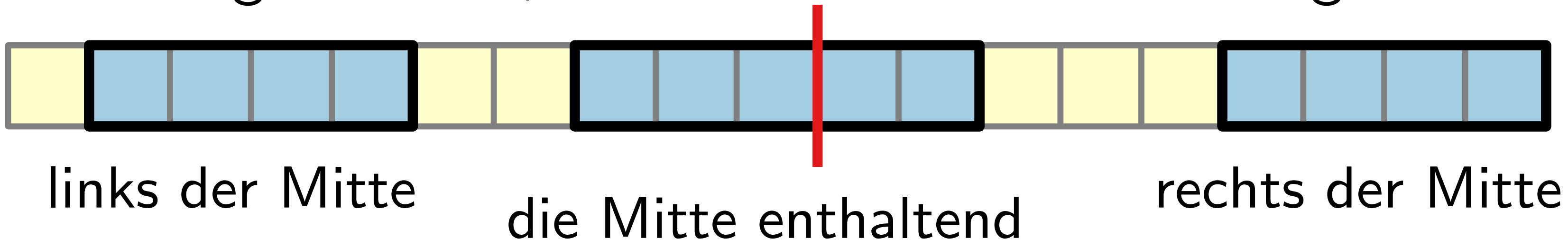
Davon gibt's $\frac{n}{2} \cdot \frac{n}{2} \in \Theta(n^2)$ ☹

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**!

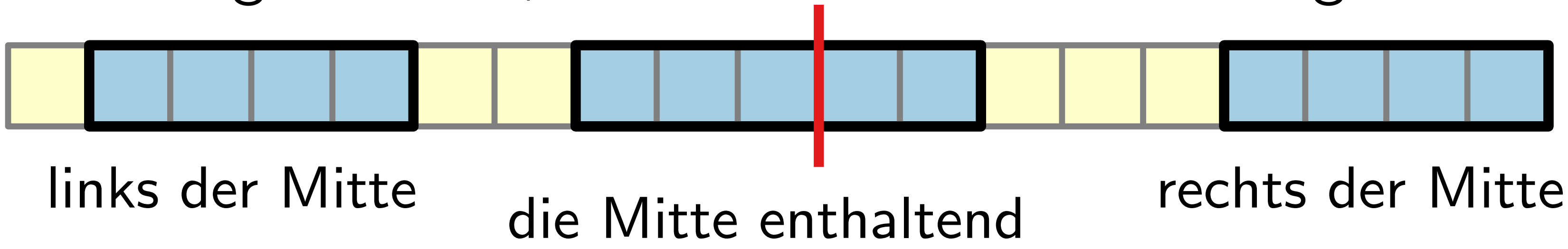
- **teile:** in zwei ungefähr gleichgroße Hälften
- **herrsche:** durch rekursive Aufrufe für linke und rechte Hälfte
- **kombiniere:** kontrolliere alle Teilsummen, die die Mitte enthalten

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.
$\Rightarrow$ Können linken und rechten Teil **unabhängig** voneinander berechnen!

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int[] } A, \text{ int } anfang = 1, \text{ int } ende = A.length)$

)

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ anfang = 1,\ \text{int}\ ende = A.length)$

if $anfang == ende$ **then**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\mathsf{int}[\,]\ A,\ \mathsf{int}\ anfang = 1,\ \mathsf{int}\ ende = A.length)$

if $anfang == ende$ **then**
  return $(anfang, ende, A[anfang])$

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A, \text{ int } \mathit{anfang} = 1, \text{ int } \mathit{ende} = A.length)$

$\left.\begin{array}{l} \textbf{if } \mathit{anfang} == \mathit{ende} \textbf{ then} \\ \quad \mid \quad \textbf{return } (\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}]) \end{array}\right\}$ **herrsche** (in kleinen Teilinstanzen)

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A, \text{int}\ \mathit{anfang} = 1, \text{int}\ \mathit{ende} = A.\mathit{length})$

$$\left.\begin{array}{l}\textbf{if}\ \mathit{anfang} == \mathit{ende}\ \textbf{then}\\ \quad \textbf{return}\ (\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])\end{array}\right\}\ \textbf{herrsche}\ (\text{in }_{\text{kleinen}}\text{ Teilinstanzen})$$

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

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ anfang = 1, \text{int}\ ende = A.length)$

if $anfang == ende$ **then**
  return $(anfang, ende, A[anfang])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $mitte = \lfloor (anfang + ende)/2 \rfloor$

} **teile**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ \mathit{anfang} = 1, \text{int}\ \mathit{ende} = A.length)$

$\left.\begin{array}{l} \textbf{if}\ \mathit{anfang} == \mathit{ende}\ \textbf{then} \\ \quad \textbf{return}\ (\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}]) \end{array}\right\}$ **herrsche** (in kleinen Teilinstanzen)

else

$\quad \mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**

$\quad (L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.\mathit{length})$

$$\left.\begin{array}{l}\textbf{if } \mathit{anfang} == \mathit{ende} \textbf{ then}\\ \quad \textbf{return } (\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])\end{array}\right\}$$ **herrsche** (in kleinen Teilinstanzen)

else

$$\left.\begin{array}{l}\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor\\ (L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})\\ (R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})\end{array}\right\}$$ **teile**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.\mathit{length})$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.\mathit{length})$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ anfang = 1, \text{int}\ ende = A.length)$

if $anfang == ende$ **then**
  return $(anfang, ende, A[anfang])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $mitte = \lfloor (anfang + ende)/2 \rfloor$ } **teile**
  $(L\text{-}anfang, L\text{-}ende, L\text{-}summe) = \textsc{MaxTeilfeld}(A, anfang, mitte)$
  $(R\text{-}anfang, R\text{-}ende, R\text{-}summe) = \textsc{MaxTeilfeld}(A, mitte + 1, ende)$ } **herrsche**
  $(M\text{-}anfang, M\text{-}ende, M\text{-}summe) = \textsc{MaxMittTeilfeld}(A, anfang, mitte, ende)$ } **kombiniere**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ anfang = 1,\ \text{int}\ ende = A.length)$

if $anfang == ende$ **then**
  return $(anfang, ende, A[anfang])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $mitte = \lfloor (anfang + ende)/2 \rfloor$ } **teile**
  $(L\text{-}anfang, L\text{-}ende, L\text{-}summe) = \textsc{MaxTeilfeld}(A, anfang, mitte)$
  $(R\text{-}anfang, R\text{-}ende, R\text{-}summe) = \textsc{MaxTeilfeld}(A, mitte + 1, ende)$ } **herrsche**
  $(M\text{-}anfang, M\text{-}ende, M\text{-}summe) = \textsc{MaxMittTeilfeld}(A, anfang, mitte, ende)$ } **kombiniere**

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int[] } A, \text{ int } \mathit{anfang} = 1, \text{ int } \mathit{ende} = A.length)$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ anfang = 1,\ \text{int}\ ende = A.length)$

$$\left.\begin{array}{l}\textbf{if } anfang == ende \textbf{ then}\\ \quad \textbf{return } (anfang, ende, A[anfang])\end{array}\right\}$$ **herrsche** (in kleinen Teilinstanzen)

else

$$\begin{array}{lll} mitte = \lfloor (anfang + ende)/2 \rfloor & & \} \textbf{ teile} \\ (L\text{-}anfang, L\text{-}ende, L\text{-}summe) & = \textsc{MaxTeilfeld}(A, anfang, mitte) & \multirow{2}{*}{\} \textbf{ herrsche}} \\ (R\text{-}anfang, R\text{-}ende, R\text{-}summe) & = \textsc{MaxTeilfeld}(A, mitte + 1, ende) & \\ (M\text{-}anfang, M\text{-}ende, M\text{-}summe) & = \textsc{MaxMittTeilfeld}(A, anfang, mitte, ende) & \} \textbf{ kombiniere} \end{array}$$

return (Tripel mit größter Summe)

Laufzeit:

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ anfang = 1, \text{int}\ ende = A.length)$

$$\left.\begin{array}{l}\textbf{if } anfang == ende \textbf{ then} \\ \quad \textbf{return } (anfang, ende, A[anfang])\end{array}\right\}$$ **herrsche** (in kleinen Teilinstanzen)

else

$$\left.mitte = \lfloor (anfang + ende)/2 \rfloor\right\}$$ **teile**

$$\left.\begin{array}{l}(L\text{-}anfang, L\text{-}ende, L\text{-}summe) = \textsc{MaxTeilfeld}(A, anfang, mitte) \\ (R\text{-}anfang, R\text{-}ende, R\text{-}summe) = \textsc{MaxTeilfeld}(A, mitte + 1, ende)\end{array}\right\}$$ **herrsche**

$$\left.(M\text{-}anfang, M\text{-}ende, M\text{-}summe) = \textsc{MaxMittTeilfeld}(A, anfang, mitte, ende)\right\}$$ **kombiniere**

return (Tripel mit größter Summe)

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

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

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

für $n > 1$: $T_{\mathrm{MT}}(n) =$

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ \mathit{anfang} = 1, \text{int}\ \mathit{ende} = A.length)$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

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

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

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.\mathit{length})$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor (\mathit{anfang} + \mathit{ende})/2 \rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

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

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

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.length)$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$ } **herrsche**
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

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

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

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[]\ A, \text{int}\ \mathit{anfang} = 1, \text{int}\ \mathit{ende} = A.\mathit{length})$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

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

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

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

Teile & Herrsche

$\textsc{MaxTeilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang} = 1,\ \text{int}\ \mathit{ende} = A.\mathit{length})$

if $\mathit{anfang} == \mathit{ende}$ **then**
  return $(\mathit{anfang}, \mathit{ende}, A[\mathit{anfang}])$ } **herrsche** (in kleinen Teilinstanzen)
else
  $\mathit{mitte} = \lfloor(\mathit{anfang} + \mathit{ende})/2\rfloor$ } **teile**
  $(L\text{-}\mathit{anfang}, L\text{-}\mathit{ende}, L\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{anfang}, \mathit{mitte})$
  $(R\text{-}\mathit{anfang}, R\text{-}\mathit{ende}, R\text{-}\mathit{summe}) = \textsc{MaxTeilfeld}(A, \mathit{mitte}+1, \mathit{ende})$ } **herrsche**
  $(M\text{-}\mathit{anfang}, M\text{-}\mathit{ende}, M\text{-}\mathit{summe}) = \textsc{MaxMittTeilfeld}(A, \mathit{anfang}, \mathit{mitte}, \mathit{ende})$ } **kombiniere**
  return (Tripel mit größter Summe)

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

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

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

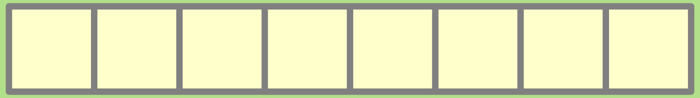
$T_{\mathrm{MMT}}(n) = ?$

Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

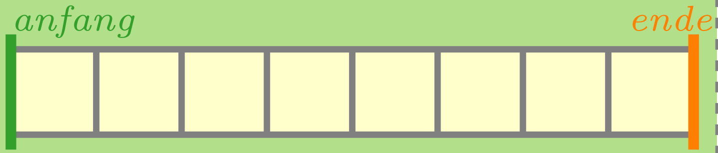
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)



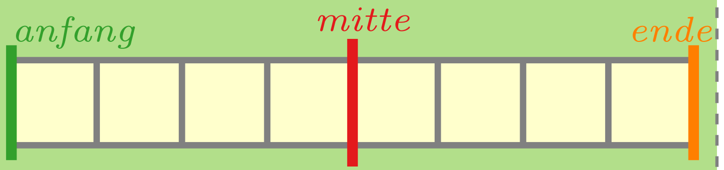
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)



Kombiniere

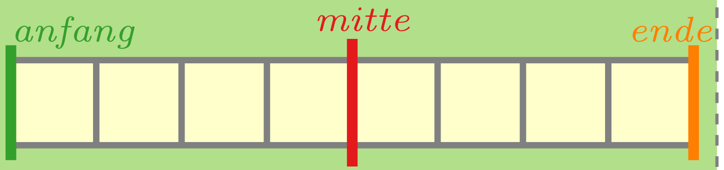
$\textsc{MaxMittTeilfeld}(\text{int[] } A, \text{ int } anfang, \text{ int } mitte, \text{ int } ende)$



Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

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

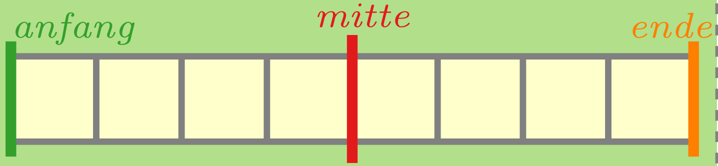


Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$



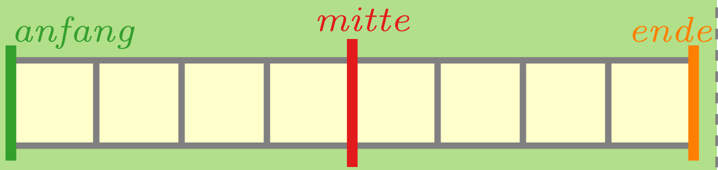
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



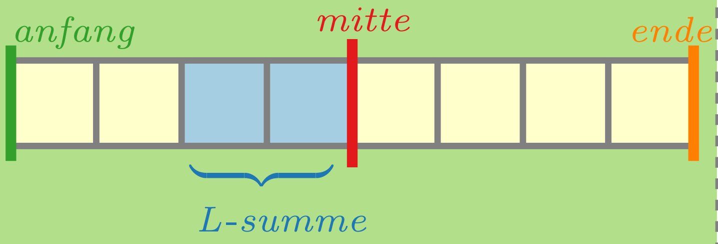
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



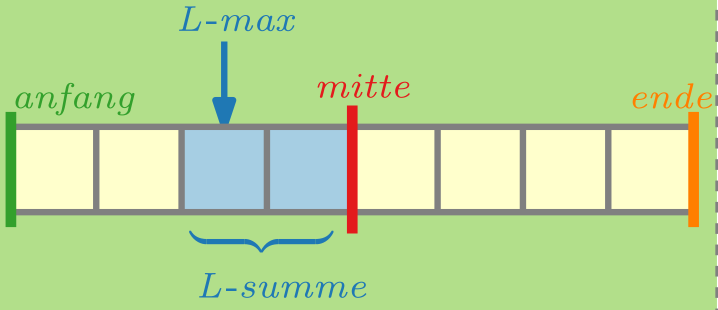
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



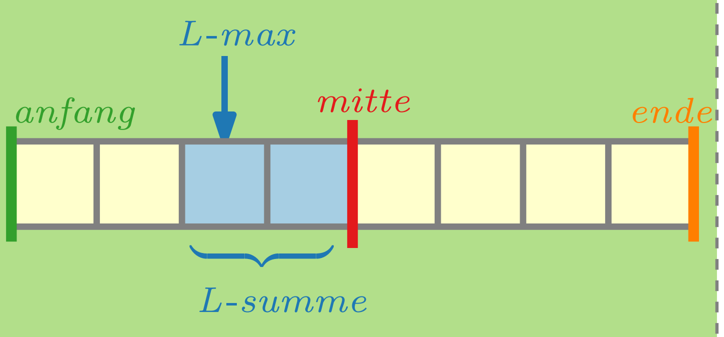
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



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

$summe = 0$

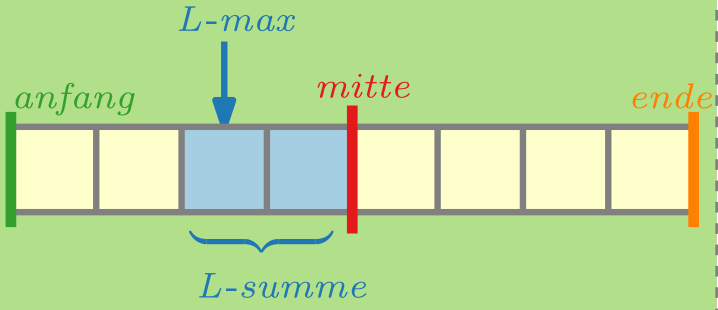
Kombiniere

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

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

$summe = 0$

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



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

$summe = 0$

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

// analog zu oben

Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

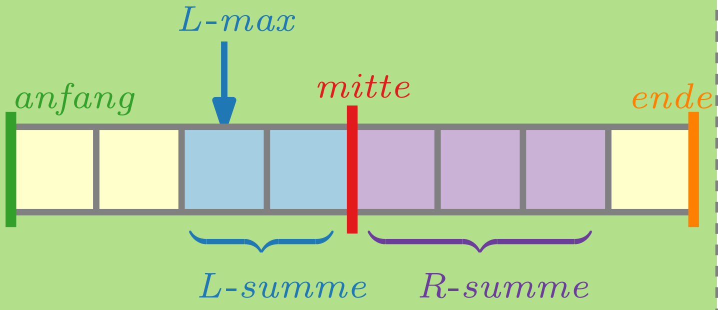
for $i = mitte$ **downto** $anfang$ **do**

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

$summe = 0$

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

// analog zu oben



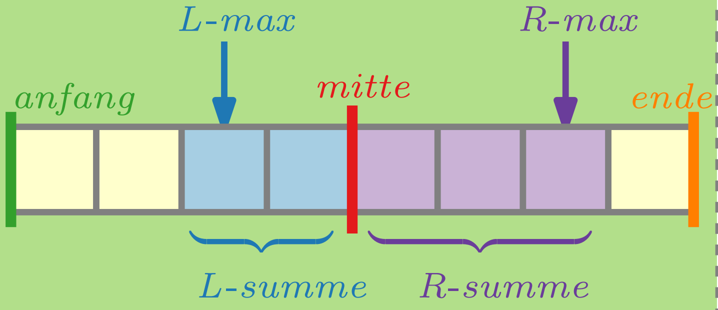
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



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

$summe = 0$

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

// analog zu oben

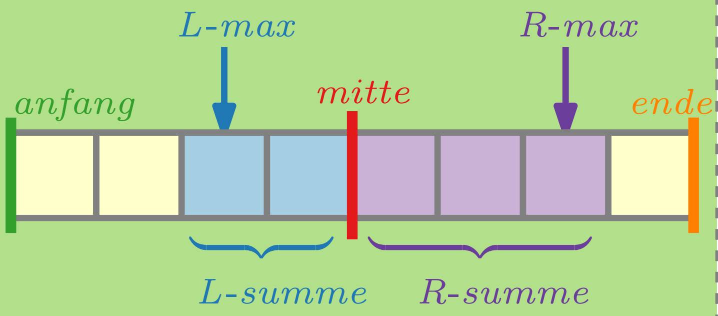
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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



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

$summe = 0$

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

// analog zu oben

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

Kombiniere

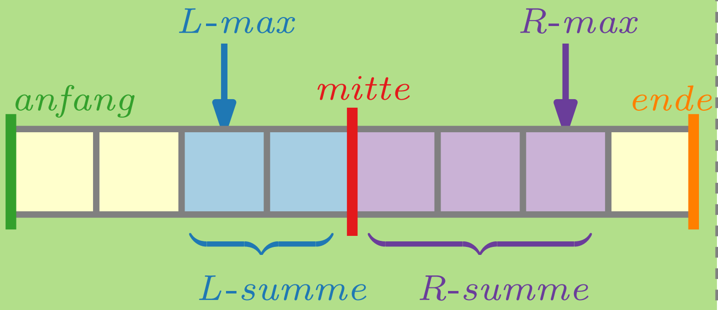
$\textsc{Max}\textsc{Mitt}\textsc{Teilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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

Vervollständigen Sie den Algorithmus!



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

$summe = 0$

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

// analog zu oben

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

Kombiniere

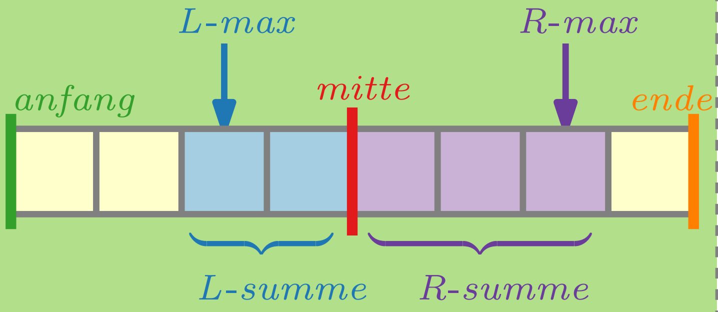
MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

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

  $summe = summe + A[i]$



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

$summe = 0$

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

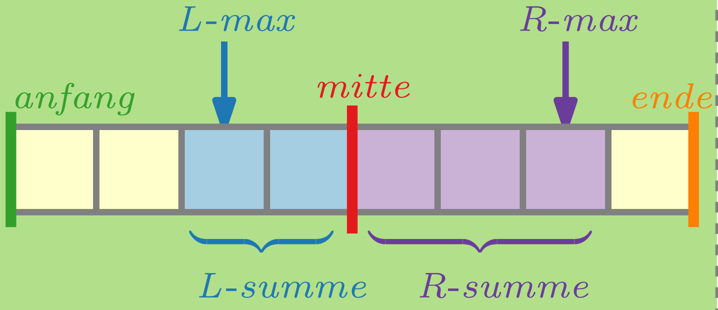
  // analog zu oben

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

Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **do**
  $summe = summe + A[i]$
  if $summe > L\text{-}summe$ **then**



$R\text{-}summe = -\infty$
$summe = 0$
for $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben

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

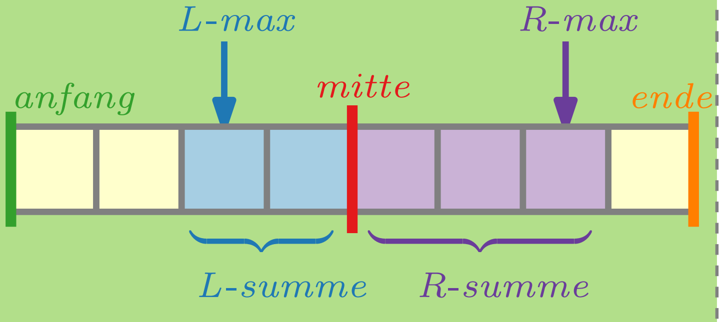
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben

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



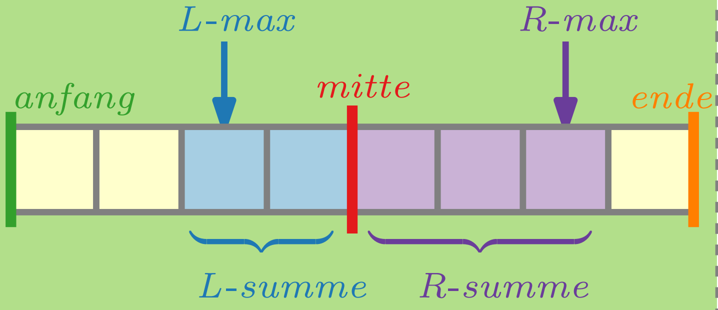
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben

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



Korrektheit?

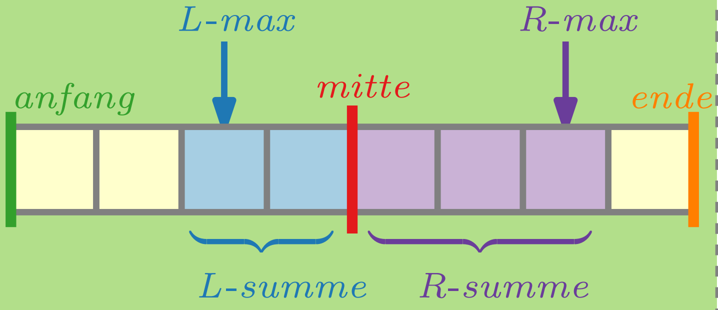
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
    // analog zu oben

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



Korrektheit?

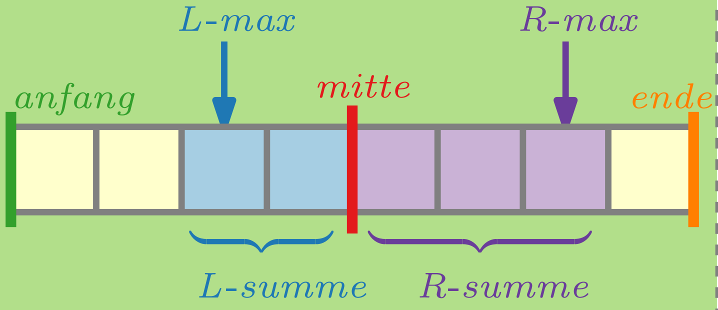
Schleifeninvariante:

Kombiniere

```
MaxMittTeilfeld(int[] A, int anfang, int mitte, int ende)
    L-summe = −∞
    summe = 0
    for i = mitte downto anfang do
        summe = summe + A[i]
        if summe > L-summe then
            L-summe = summe
            L-max = i

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

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



Korrektheit?

Schleifeninvariante:

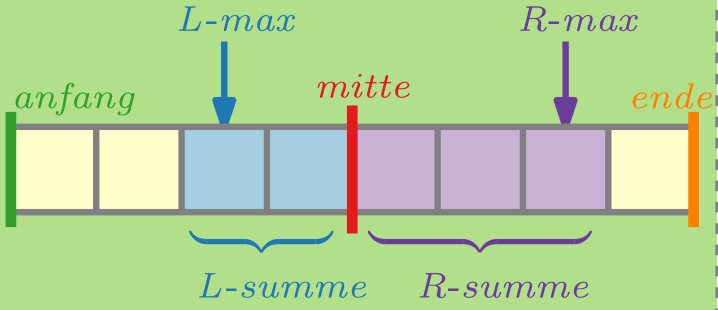
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben

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



Korrektheit?

Schleifeninvariante:

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

Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

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

$summe = 0$

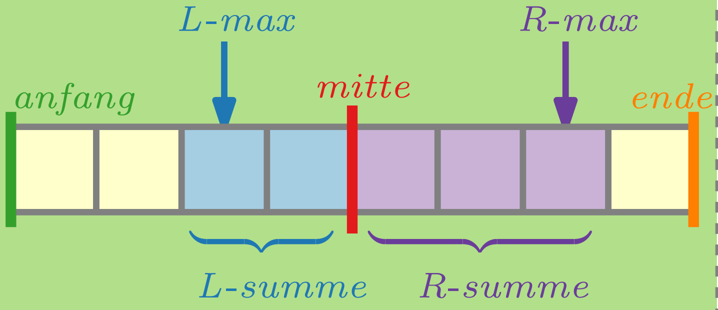
for $i = mitte$ **downto** $anfang$ **do**

  $summe = summe + A[i]$

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

    $L\text{-}summe = summe$

    $L\text{-}max = i$



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

$summe = 0$

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

  // analog zu oben

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

Korrektheit?

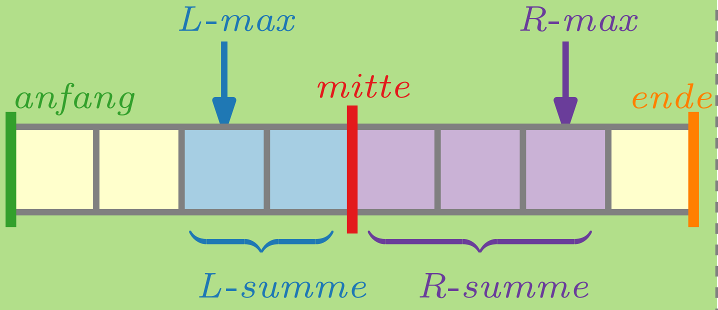
Schleifeninvariante:

$summe = S_{i,mitte}$ und

Kombiniere

$\textsc{Max}\textsc{Mitt}\textsc{Teilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben

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

Korrektheit?

Schleifeninvariante:

$summe = S_{i,mitte}$ und

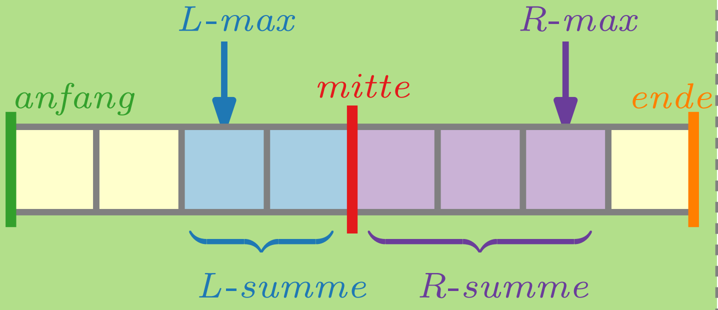
$L\text{-}summe =$

Kombiniere

```
MaxMittTeilfeld(int[] A, int anfang, int mitte, int ende)
    L-summe = −∞
    summe = 0
    for i = mitte downto anfang do
        summe = summe + A[i]
        if summe > L-summe then
            L-summe = summe
            L-max = i

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

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



Korrektheit?

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

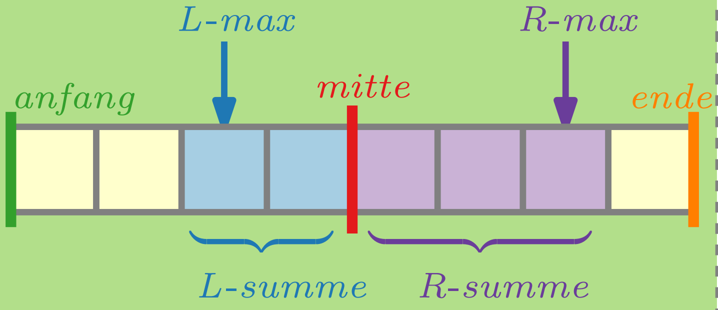
Kombiniere

$\textsc{MaxMittTeilfeld}(\text{int}[]\ A, \text{int}\ anfang, \text{int}\ mitte, \text{int}\ ende)$

```
L-summe = −∞
summe = 0
for i = mitte downto anfang do
    summe = summe + A[i]
    if summe > L-summe then
        L-summe = summe
        L-max = i

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

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

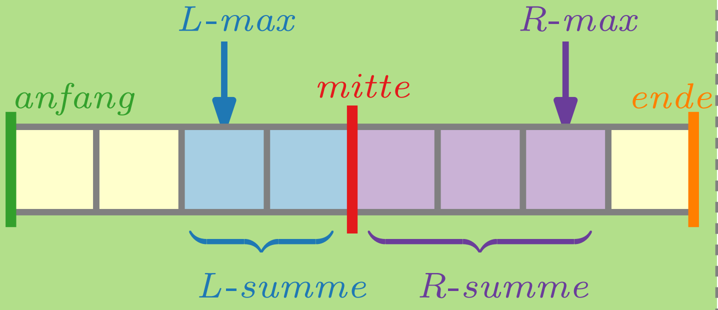
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

```
L-summe = −∞
summe = 0
for i = mitte downto anfang do
    summe = summe + A[i]
    if summe > L-summe then
        L-summe = summe
        L-max = i

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

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

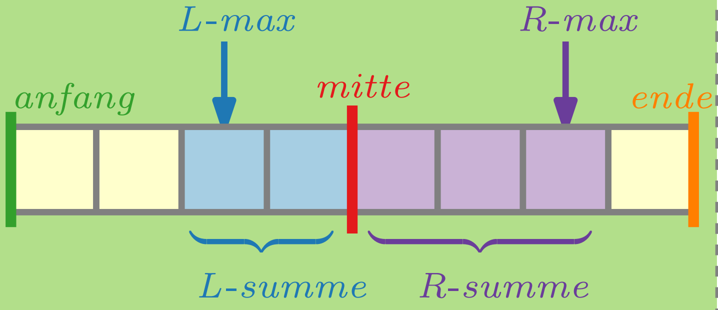
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

```
L-summe = −∞
summe = 0
for i = mitte downto anfang do
    summe = summe + A[i]
    if summe > L-summe then
        L-summe = summe
        L-max = i

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

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

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

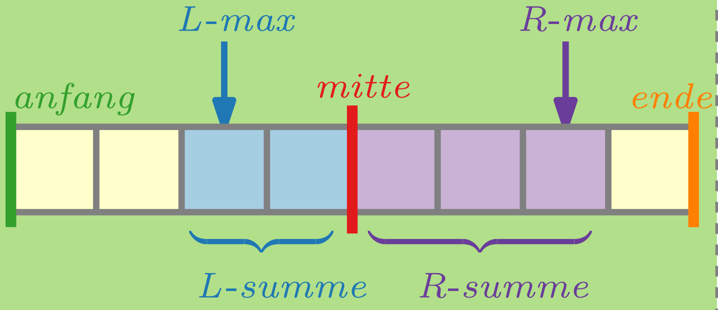
Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben +

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

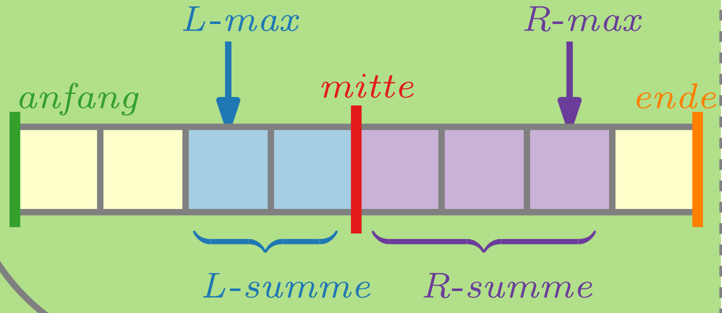
Laufzeit?

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

Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

- $L\text{-}summe = -\infty$
- $summe = 0$
- **for** $i = mitte$ **downto** $anfang$ **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** $i = mitte + 1$ **to** $ende$ **do**
  - // analog zu oben $+$
- **return** $(L\text{-}max, R\text{-}max, L\text{-}summe + R\text{-}summe)$



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

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

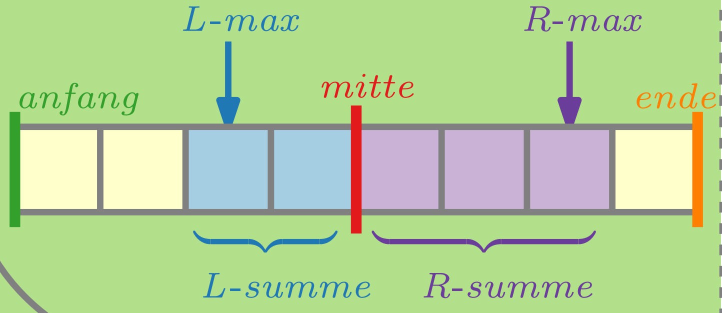
Kombiniere

$\textsc{Max}\textsc{Mitt}\textsc{Teilfeld}$(int[] A, int anfang, int mitte, int ende)

    $\mathit{L\text{-}summe} = -\infty$
    $\mathit{summe} = 0$
    for $i = \mathit{mitte}$ **downto** anfang **do**
        $\mathit{summe} = \mathit{summe} + A[i]$
        if $\mathit{summe} > \mathit{L\text{-}summe}$ **then**
            $\mathit{L\text{-}summe} = \mathit{summe}$
            $\mathit{L\text{-}max} = i$

    $\mathit{R\text{-}summe} = -\infty$
    $\mathit{summe} = 0$
    for $i = \mathit{mitte} + 1$ **to** ende **do**
        // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

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

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

Laufzeit?

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

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

Kombiniere

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

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

$\quad summe = 0$

$\quad$ **for** $i =$ *mitte* **downto** *anfang* **do**

$\qquad summe = summe + A[i]$

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

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

$\qquad\quad L\text{-}max = i$

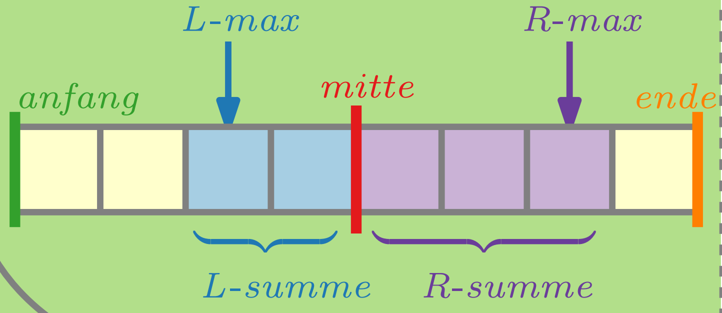
$\quad R\text{-}summe = -\infty$

$\quad summe = 0$

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

$\qquad$ // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

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

$= mitte - anfang + 1$

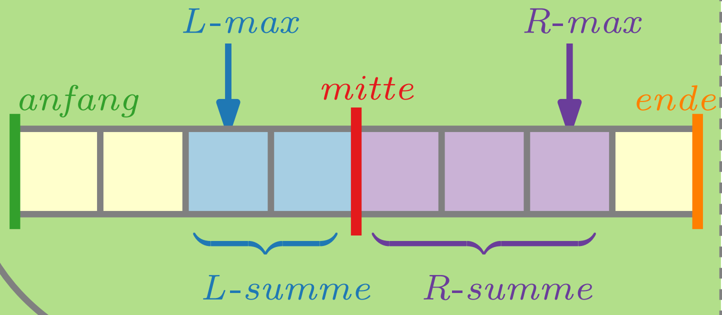
Kombiniere

$\textsc{Max}\textsc{Mitt}\textsc{Teilfeld}(\text{int}[\,]\ A,\ \text{int}\ \mathit{anfang},\ \text{int}\ \mathit{mitte},\ \text{int}\ \mathit{ende})$

```
    L-summe = −∞
    summe = 0
    for i = mitte downto anfang do
        summe = summe + A[i]
        if summe > L-summe then
            L-summe = summe
            L-max = i

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

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



Korrektheit? ✓

Schleifeninvariante:

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

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

Laufzeit?

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

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

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

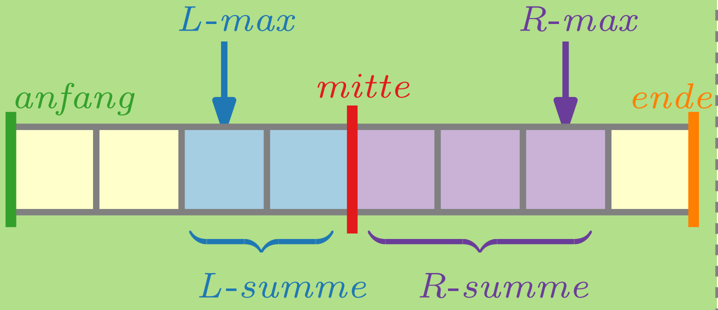
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

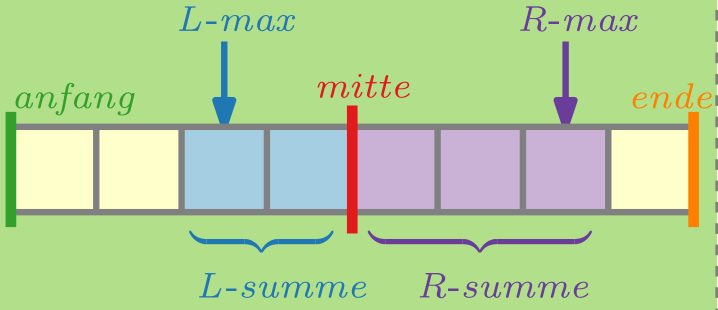
$:=_{\text{hier}}$ Anz. Additionen

$= mitte - anfang + 1$
$+ ende - mitte$

Kombiniere

MaxMittTeilfeld(int[] A, int $anfang$, int $mitte$, int $ende$)

- $L\text{-}summe = -\infty$
- $summe = 0$
- **for** $i = mitte$ **downto** $anfang$ **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** $i = mitte + 1$ **to** $ende$ **do**
  - // analog zu oben $+$
- **return** $(L\text{-}max, R\text{-}max, L\text{-}summe + R\text{-}summe)$



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit?

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

$= mitte - anfang + 1$

$+ ende - mitte$

$= ende - anfang + 1$

Kombiniere

$\textsc{MaxMittTeilfeld}(\text{int}[]\ A, \text{int}\ \mathit{anfang}, \text{int}\ \mathit{mitte}, \text{int}\ \mathit{ende})$

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

$\mathit{summe} = 0$

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

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

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

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

    $\mathit{L\text{-}max} = i$

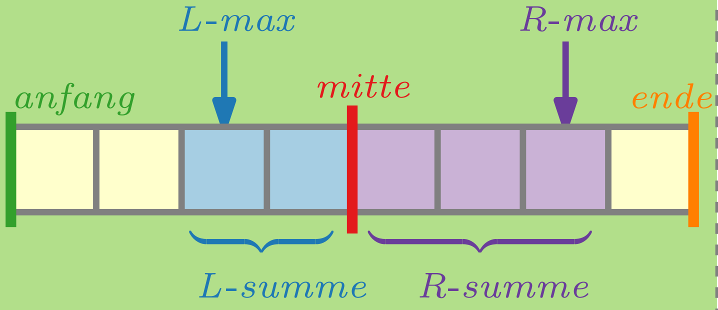
$\mathit{R\text{-}summe} = -\infty$

$\mathit{summe} = 0$

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

  // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

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

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

Laufzeit?

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

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

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

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

$= n$

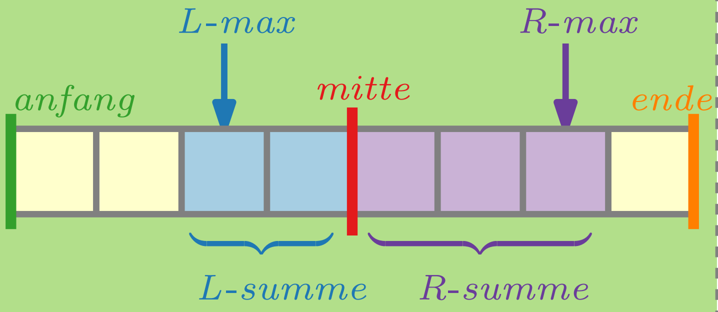
Kombiniere

$\textsc{MaxMittTeilfeld}$(int[] A, int $anfang$, int $mitte$, int $ende$)

$L\text{-}summe = -\infty$
$summe = 0$
for $i = mitte$ **downto** $anfang$ **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 $i = mitte + 1$ **to** $ende$ **do**
  // analog zu oben $+$

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



Korrektheit? ✓

Schleifeninvariante:

$summe = S_{i,mitte}$ und

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

Laufzeit? ✓

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

$= mitte - anfang + 1$

$\quad + ende - mitte$

$= ende - anfang + 1$

$= n$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

Putting Things Together

Laufzeit von $\textsc{MaxTeilfeld}$:

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

für $n > 1$:

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

$\textsc{MergeSort}$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

für $n > 1$:

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

[für $n =$ Zweierpotenz]

MergeSort

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

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

MergeSort

Putting Things Together

Laufzeit von $\textsc{MaxTeilfeld}$**:**

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

für $n > 1$: $$T_{\mathrm{MT}}(n) \approx 2 \cdot T_{\mathrm{MT}}(n/2) + T_{\mathrm{MMT}}(n)$$
$$= 2 \cdot T_{\mathrm{MT}}(n/2) + n$$
$$= V_{\mathrm{MS}}(n) \quad = \mathcal{O}(n \log_{\cancel{2}} n)$$ [für n = Zweierpotenz]

$\textsc{MergeSort}$

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

$$= V_{\mathrm{MS}}(n) = \mathcal{O}(n \log_{\cancel{2}} n)$$ [für $n =$ Zweierpotenz]

MergeSort

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

$$= V_{\mathrm{MS}}(n) \quad = \mathcal{O}(n \log_{\cancel{2}} n)$$

~~[für n = Zweierpotenz]~~

MergeSort

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

~~[für n = Zweierpotenz]~~

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

MergeSort

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

~~[für n = Zweierpotenz]~~

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

MergeSort

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

Putting Things Together

Laufzeit von MaxTeilfeld:

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

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

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

$$= V_{\mathrm{MS}}(n) \quad = \mathcal{O}(n \log_{\cancel{2}} n)$$

~~[für n = Zweierpotenz]~~

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

MergeSort

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:

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

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

~~[für n = Zweierpotenz]~~

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

MergeSort

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:

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

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

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

$$= V_{\mathrm{MS}}(n) \quad = \mathcal{O}(n \log_{\cancel{2}} n)$$

~~[für n = Zweierpotenz]~~

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

MergeSort

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...** $\quad T(n) = 2 \cdot T(n/2) + 4n \quad$ (und $T(1) = \Theta(1)$)

Putting Things Together

Laufzeit von MaxTeilfeld:

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

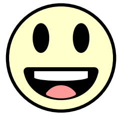
für $n > 1$: $$T_{\text{MT}}(n) \approx 2 \cdot T_{\text{MT}}(n/2) + T_{\text{MMT}}(n)$$
$$= 2 \cdot T_{\text{MT}}(n/2) + n$$
$$= V_{\text{MS}}(n) \quad = \mathcal{O}(n \log_{\cancel{2}} n)$$

~~[für n = Zweierpotenz]~~

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

MergeSort

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$		

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$		

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$		

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$		

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 (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	$\mathcal{O}(n \log_{10} n)$		
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 (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	$\mathcal{O}(n^2)$	10^6	
Teile & Herrsche	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	
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	$\mathcal{O}(n^3)$	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 (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**	$n = 1\,000$	$n = 1\,000\,000$
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 (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**	$n = 1\,000$	$n = 1\,000\,000$
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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	10^6

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	10^6

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	10^6

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3 $1\,\mu s$	10^6

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 (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3 $1\,\mu s$	10^6

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	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	$3\,\mu s$	$6 \cdot 10^6$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	$1\,\mu s$	10^6

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	$\mathcal{O}(n\ \log_{10} n)$	$3 \cdot 10^3$	$3\ \mu s$	$6 \cdot 10^6$
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	$1\ \mu s$	10^6

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	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	$3\,\mu s$	$6 \cdot 10^6$	
Linearer Scan	$\mathcal{O}(n)$	10^3	$1\,\mu s$	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	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	$3\,\mu s$	$6 \cdot 10^6$	6 ms
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	$1\,\mu s$	10^6	1 ms

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	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	$3\,\mu$s	$6 \cdot 10^6$	6 ms
Linearer Scan	$\mathcal{O}(n)$	10^3	$1\,\mu$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	$\mathcal{O}(n \log_{10} n)$	$3 \cdot 10^3$	$3\,\mu$s	$6 \cdot 10^6$	6 ms
Linearer Scan	$\mathcal{O}(n)$	10^3	$1\,\mu$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 μs	$6 \cdot 10^6$	6 ms
Linearer Scan	$\mathcal{O}(n)$	10^3	1 μs	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 μs	$6 \cdot 10^6$	6 ms
Linearer Scan (siehe Buch [CLRS], Ü-Aufgabe 4.1-5)	$\mathcal{O}(n)$	10^3	1 μs	10^6	1 ms