

Einführung in die Programmierung

Algorithmen, Pseudocode, Sortieren

Weitere Sortieralgorithmen

- Sortieren ist ein sehr intensiv untersuchtes Problem
- Es gibt eine große Zahl von Algorithmen mit jeweils verschiedenen Varianten
- Generell ordnet man Sortierverfahren in zwei Gruppen:
 - 1) Vergleichende Sortierverfahren
 - A. Einfache Sortierverfahren
 - B. Fortgeschrittene Sortierverfahren
 - 2) Nicht vergleichende Sortierverfahren

- Einfache vergleichende Sortierverfahren
 - Sortieren durch Einfügen (insertion sort)
 - Sortieren durch Auswählen (selection sort)
 - Sortieren durch Vertauschen (bubble sort)
- Fortgeschrittene vergleichende Sortierverfahren
 - Sortieren durch Gruppieren (quick sort)
 - Sortieren durch Mischen (merge sort)
- Nicht vergleichende Sortierverfahren
 - Sortieren durch Zählen (count sort)
 - Sortieren durch Fachverteilen (radix sort)

Problem: Sortieren

- Eingabe: Folge von n Zahlen (a_1, \dots, a_n)
- Ausgabe: Permutation (a'_1, \dots, a'_n) von (a_1, \dots, a_n) , so dass $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Beispiel:

- Eingabe: 15, 7, 3, 18, 8, 4
- Ausgabe: 3, 4, 7, 8, 15, 18

Wiederholung: Insertion Sort

InsertionSort(Array A)

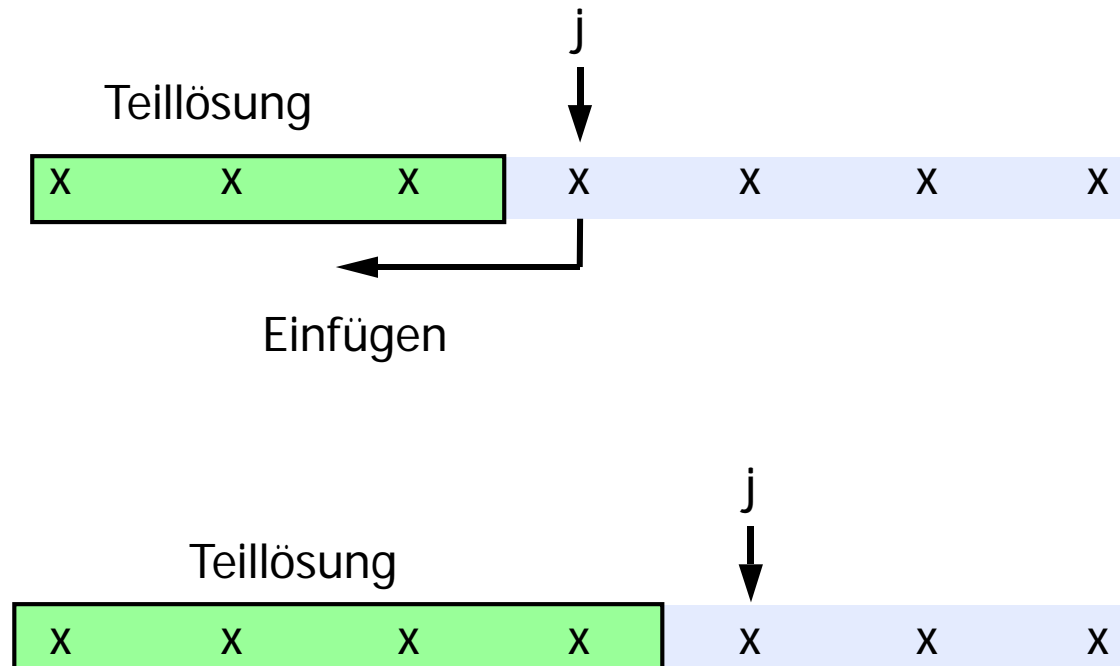
1. **for** $j \leftarrow 2$ **to** $\text{length}(A)$ **do**
2. $\text{key} \leftarrow A[j]$
3. $i \leftarrow j-1$
4. **while** $i > 0$ and $A[i] > \text{key}$ **do**
5. $A[i+1] \leftarrow A[i]$
6. $i \leftarrow i-1$
7. $A[i+1] \leftarrow \text{key}$

Idee InsertionSort

- *Die ersten $j-1$ Elemente sind sortiert (zu Beginn $j=2$)*
- *Innerhalb eines Schleifendurchlaufs wird das j -te Element in die sortierte Folge eingefügt*
- *Am Ende ist die gesamte Folge sortiert*

Sortieren durch Einfügen (insertion sort)

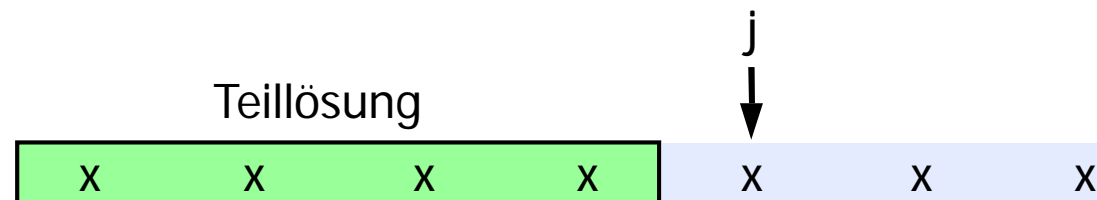
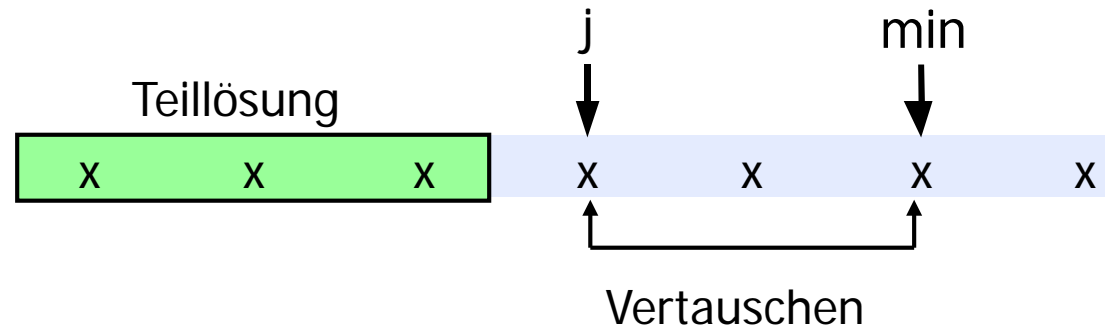
- Arbeitsweise



Selection Sort

Sortieren durch Auswählen (selection sort)

- Minimum der verbleibenden unsortierten Restfolge wird direkt ausgewählt und mit dem aktuellen Element vertauscht



SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $h \leftarrow A[\text{min}]$
6. $A[\text{min}] \leftarrow A[j]$
7. $A[j] \leftarrow h$

Idee SelectionSort

- *Die ersten $j-1$ Elemente sind sortiert (zu Beginn $j=1$)*
- *Innerhalb eines Schleifendurchlaufs wird das j -kleinste Element (entspricht dem kleinsten aus dem Rest) an die sortierte Folge „angehängt“*
- *Am Ende ist die gesamte Folge sortiert*

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

Idee SelectionSort

- *Die ersten $j-1$ Elemente sind sortiert (zu Beginn $j=1$)*
- *Innerhalb eines Schleifendurchlaufs wird das j -kleinste Element (entspricht dem kleinste aus dem Rest) an die sortierte Folge „angehängt“*
- *Am Ende ist die gesamte Folge sortiert*

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

8	15	3	14	7	6	18	19
---	----	---	----	---	---	----	----

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

8	15	3	14	7	6	18	19
---	----	---	----	---	---	----	----

j

n

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

$\text{min} = 1$



j

n

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

$\text{min} = 1$



j

i

n

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

$\text{min} = 1$



j

i

n

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel

$\text{min} = 1$



j

i

n

Selection Sort – mit swap

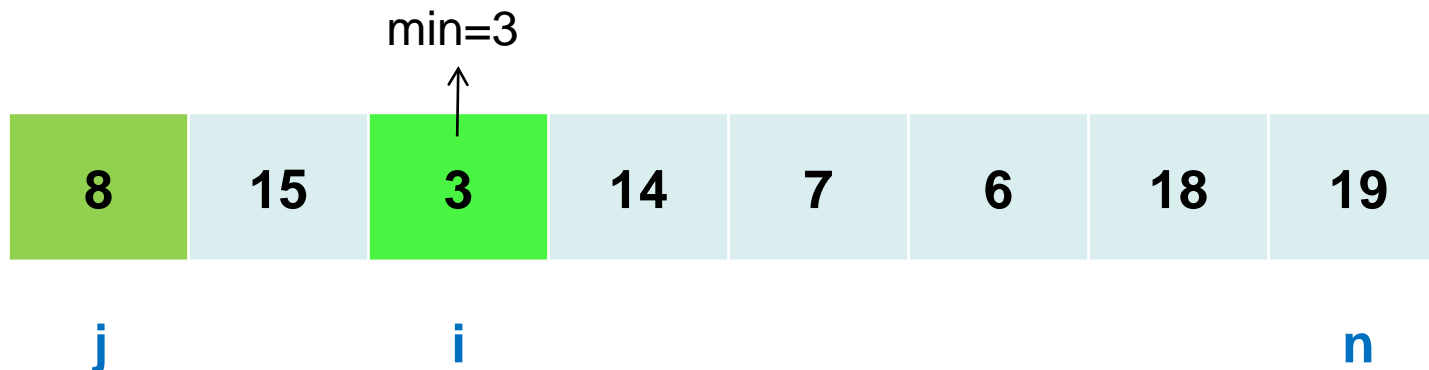
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

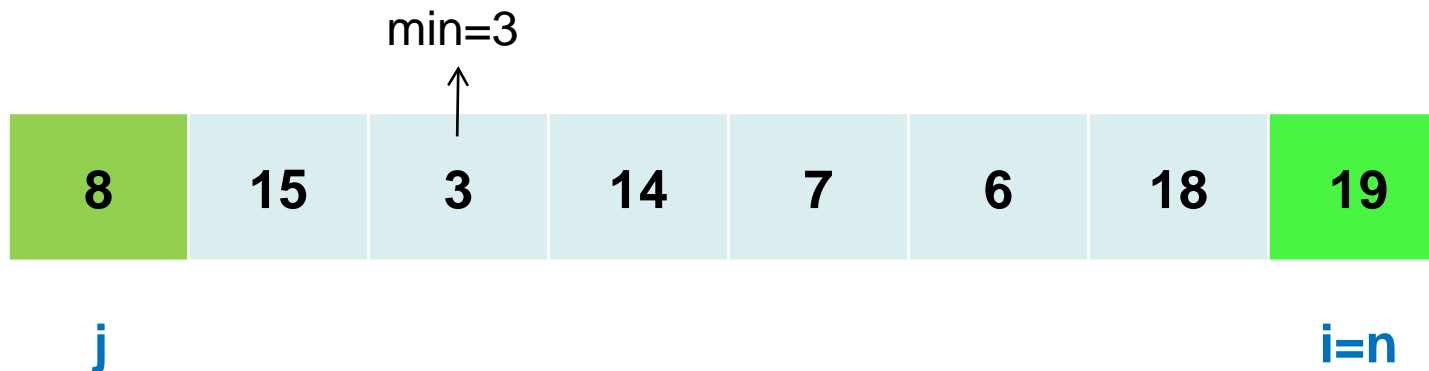
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

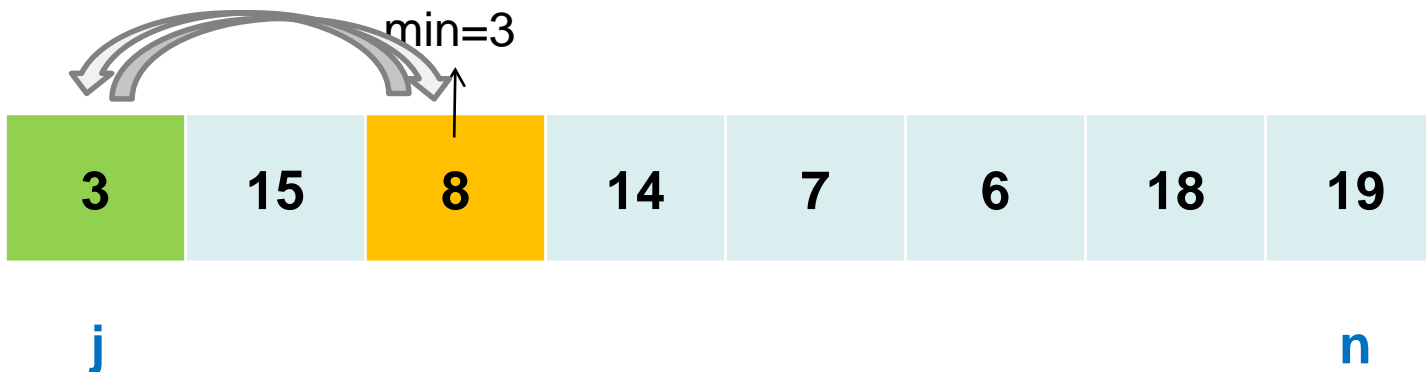
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

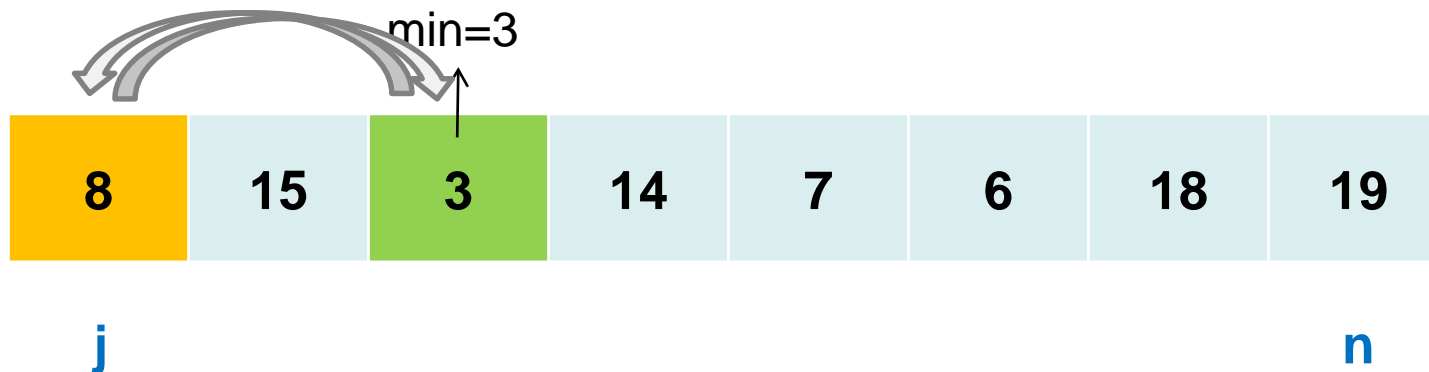
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

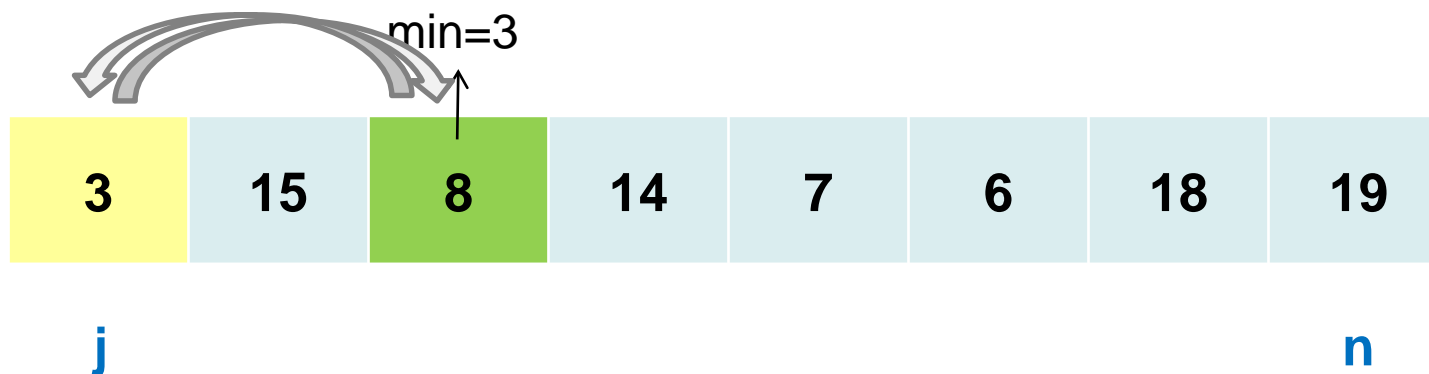
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

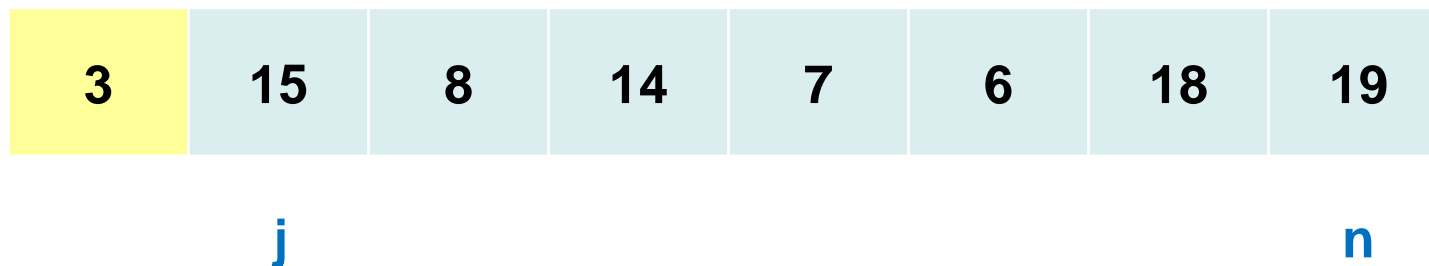
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



➤ Eingabegröße n

- $\text{length}(A) = n$

5. swap(A, min, j)

min=2



Selection Sort – mit swap

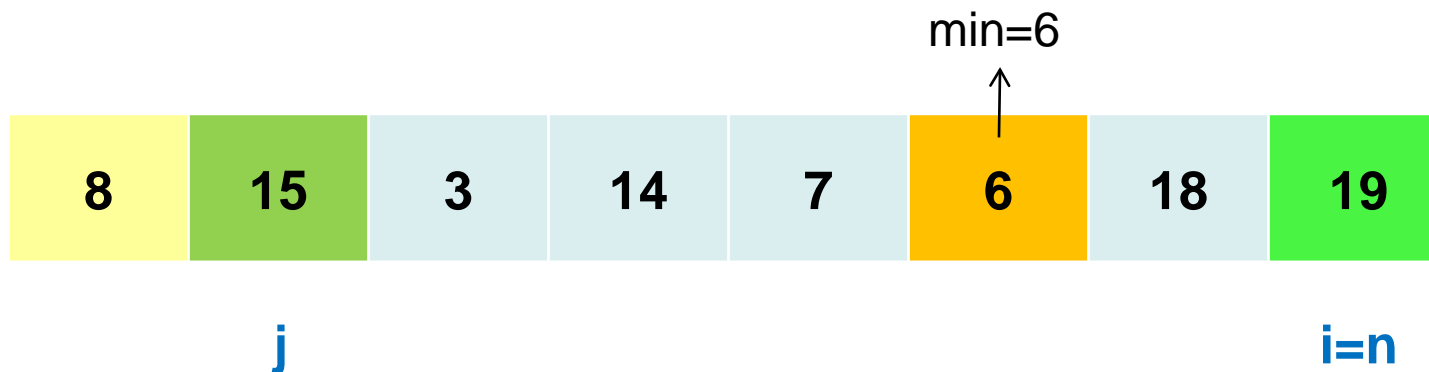
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



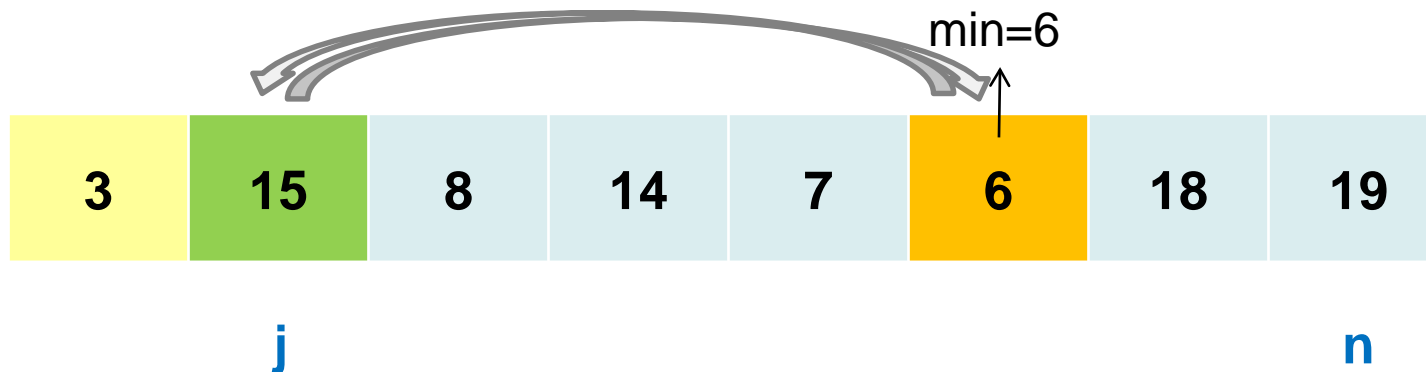
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

- Eingabegröße n
- $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

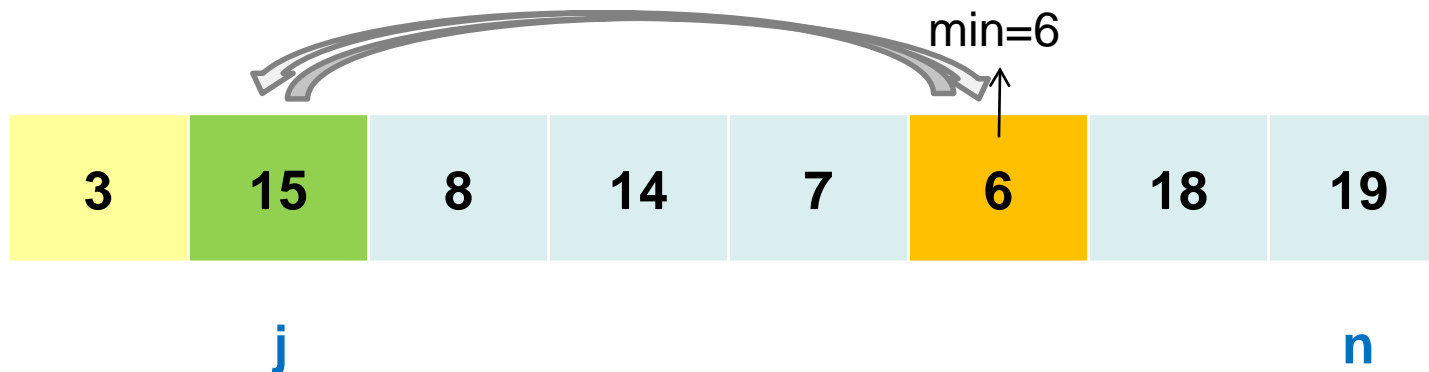
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

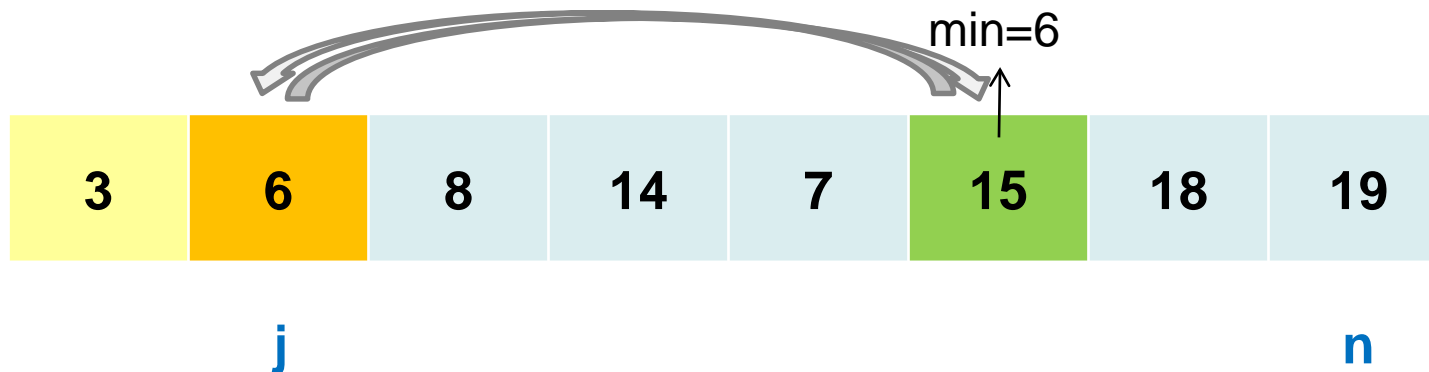
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

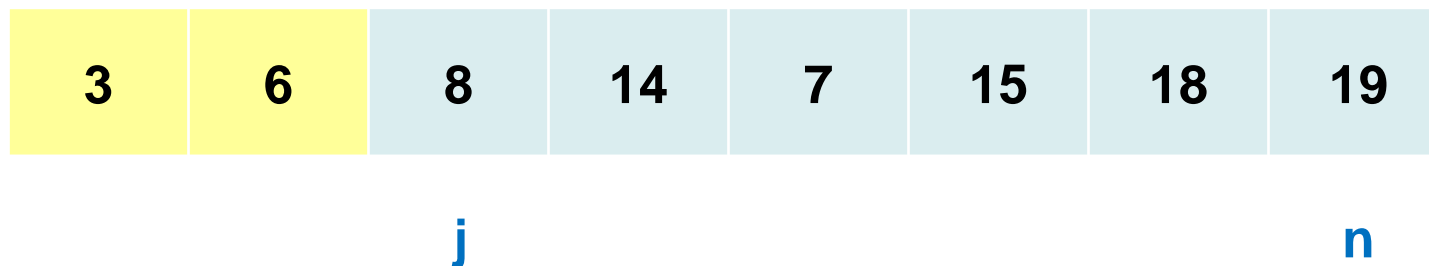
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



➤ Eingabegrößen

- $\text{length}(A) = n$

5. swap(A, min, j)

min=3



Selection Sort – mit swap

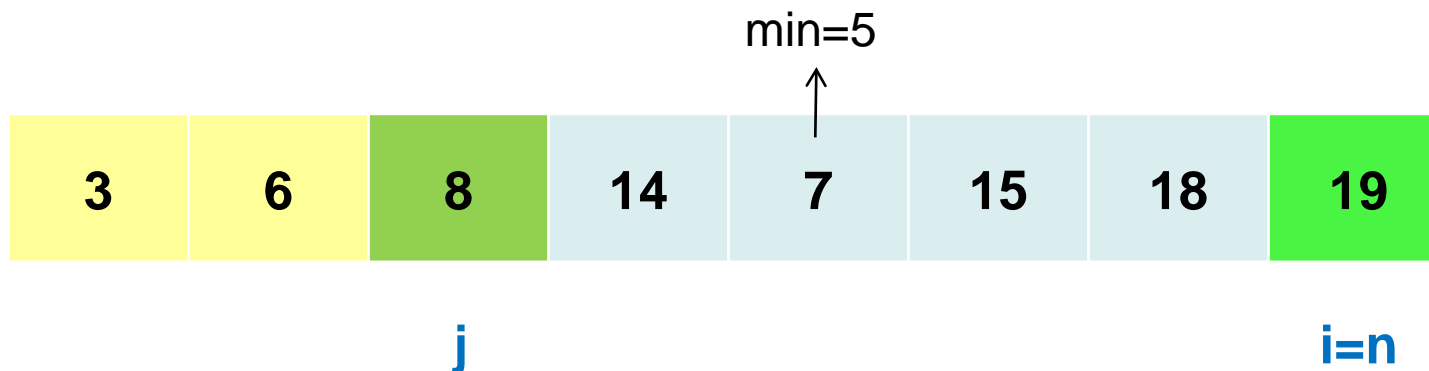
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

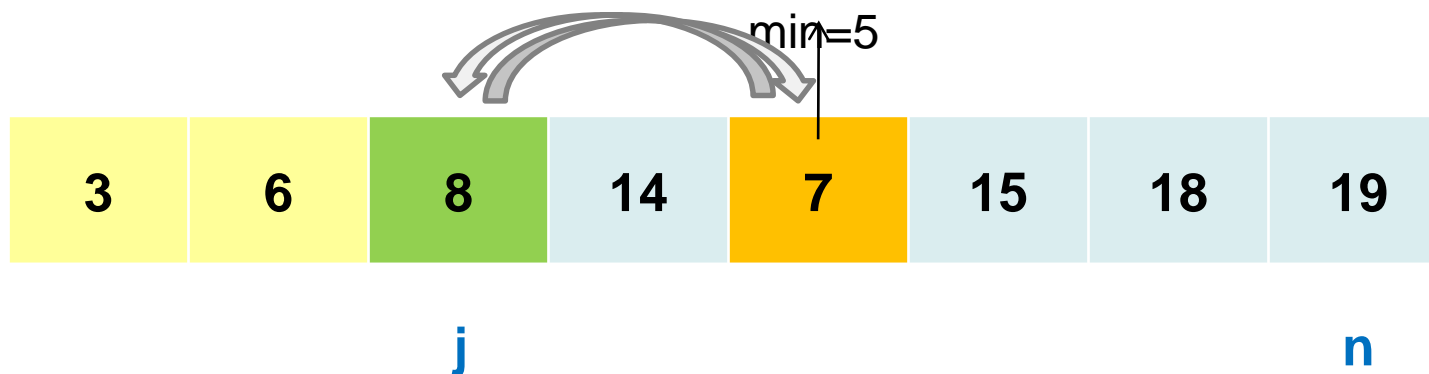
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

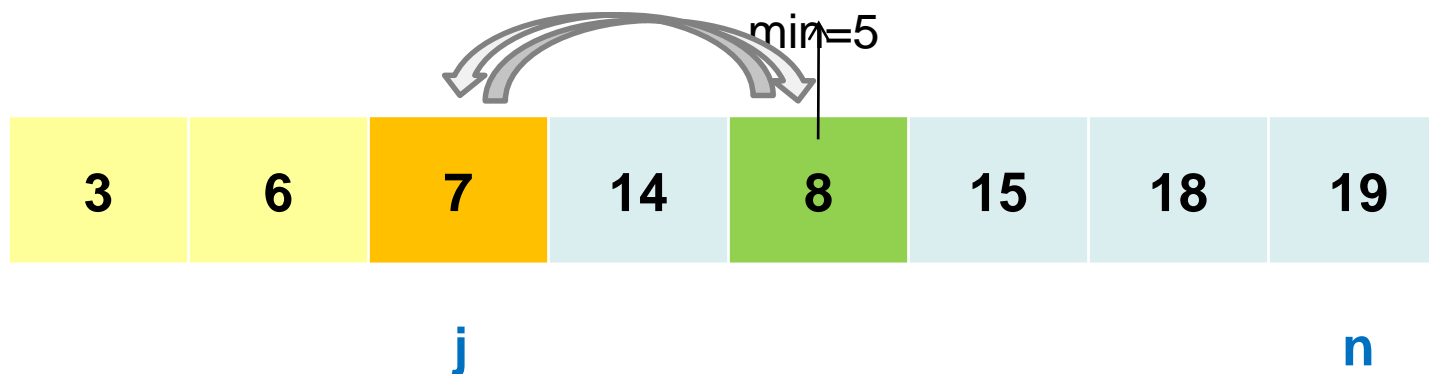
SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

➤ Eingabegröße n

➤ $\text{length}(A) = n$

Beispiel



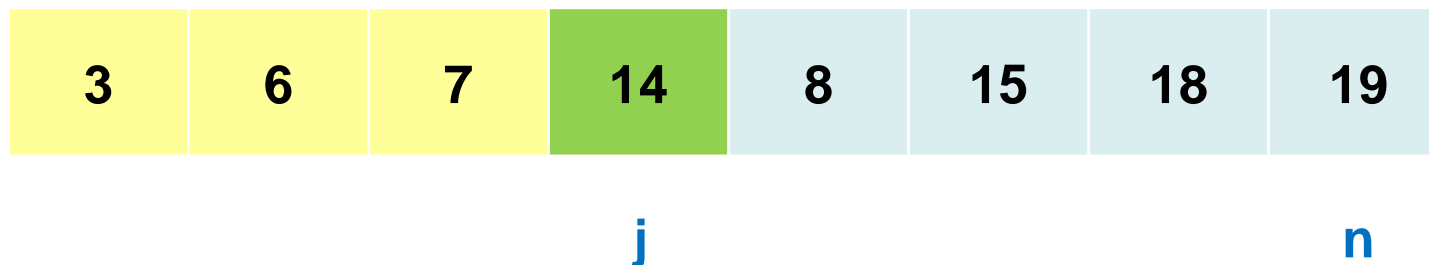
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



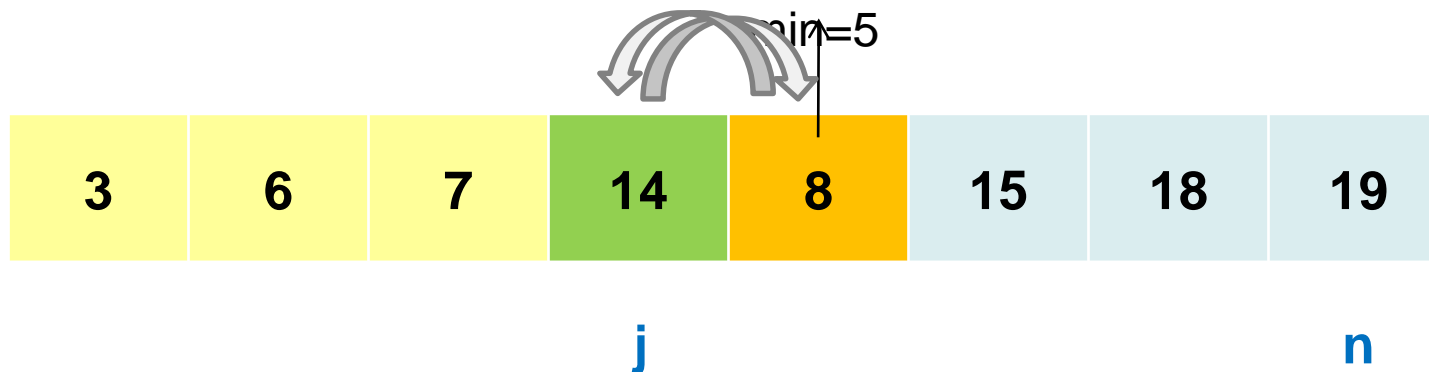
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



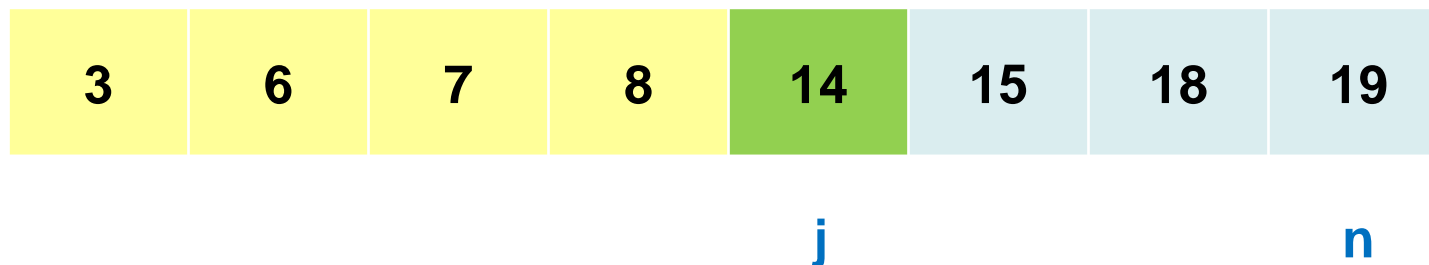
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



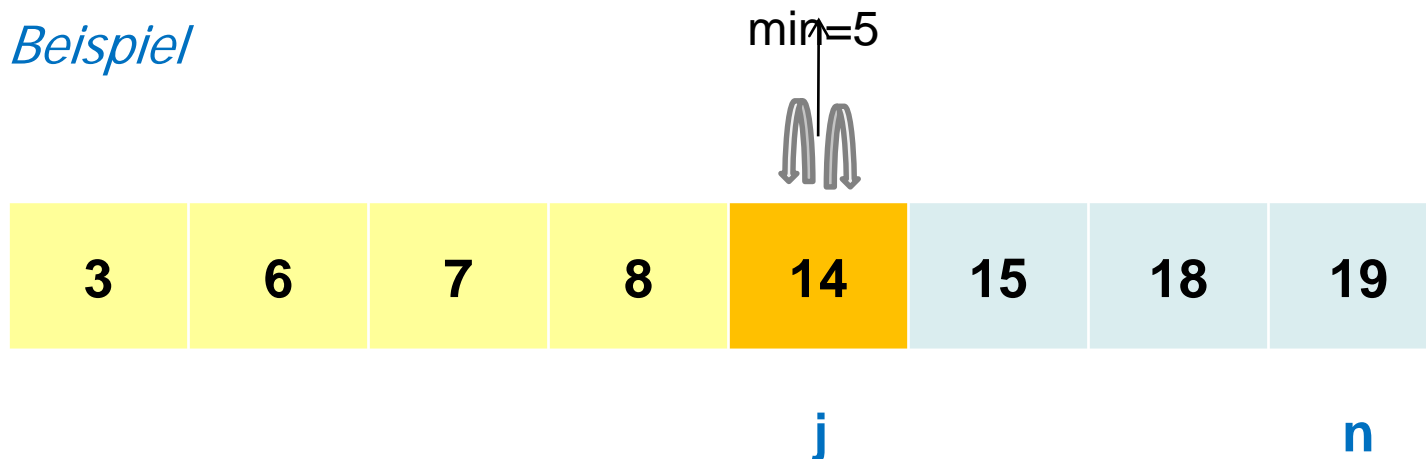
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. $\text{swap}(A, \text{min}, j)$

- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel

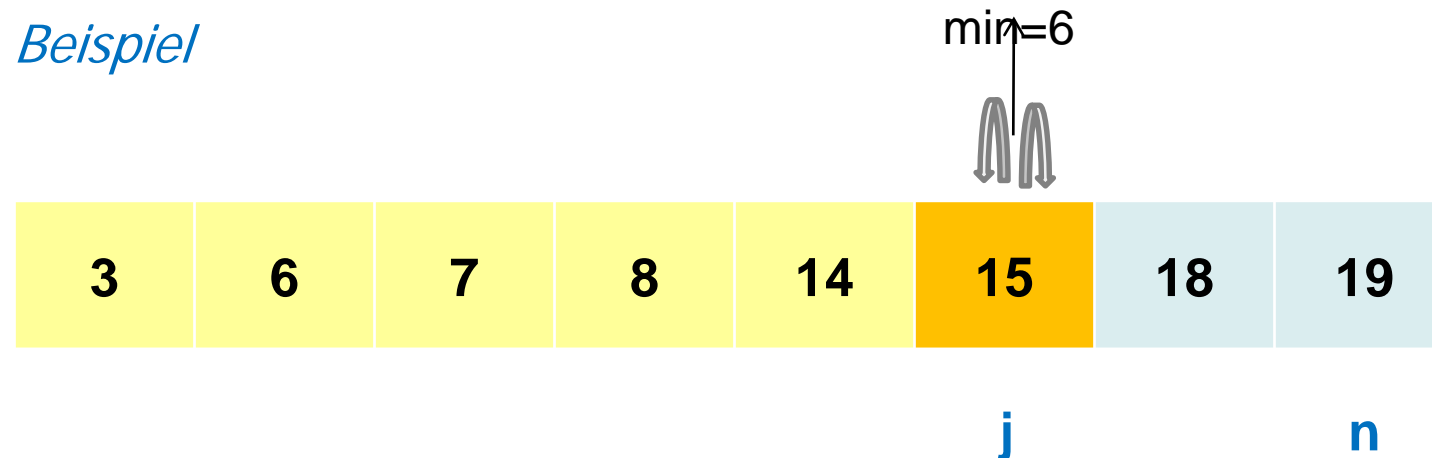


Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

Beispiel



- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Selection Sort – mit swap

SelectionSort(Array A)

1. **for** $j \leftarrow 1$ **to** $\text{length}(A) - 1$ **do**
2. $\text{min} \leftarrow j$
3. **for** $i \leftarrow j + 1$ **to** $\text{length}(A)$ **do**
4. **if** $A[i] < A[\text{min}]$ **then** $\text{min} \leftarrow i$
5. swap(A, min, j)

- Eingabegröße n
- $\text{length}(A) = n$
- Finde das kleinste Element
- In $A[j+1 \dots n]$
- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel

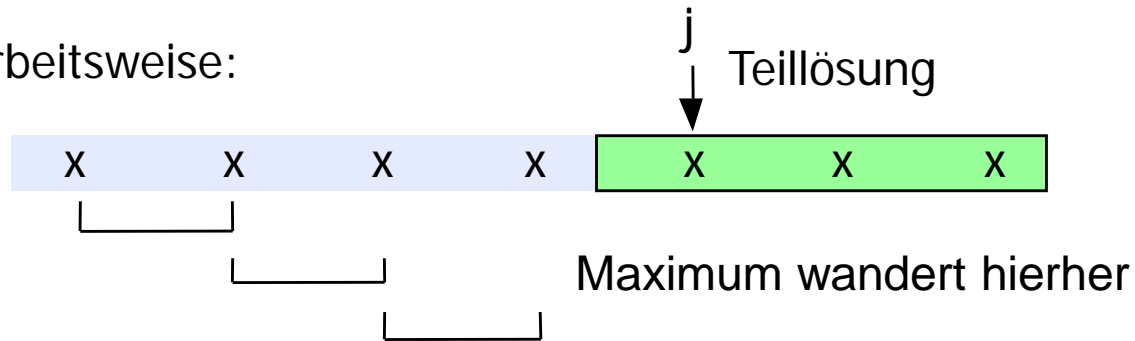
3	6	7	8	14	15	18	19
---	---	---	---	----	----	----	----

$j=n$

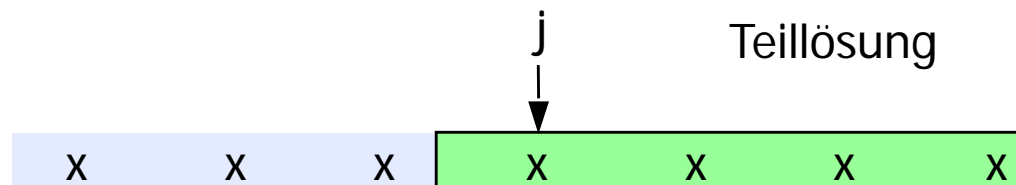
Bubble Sort

Bubble Sort

- Arbeitsweise:

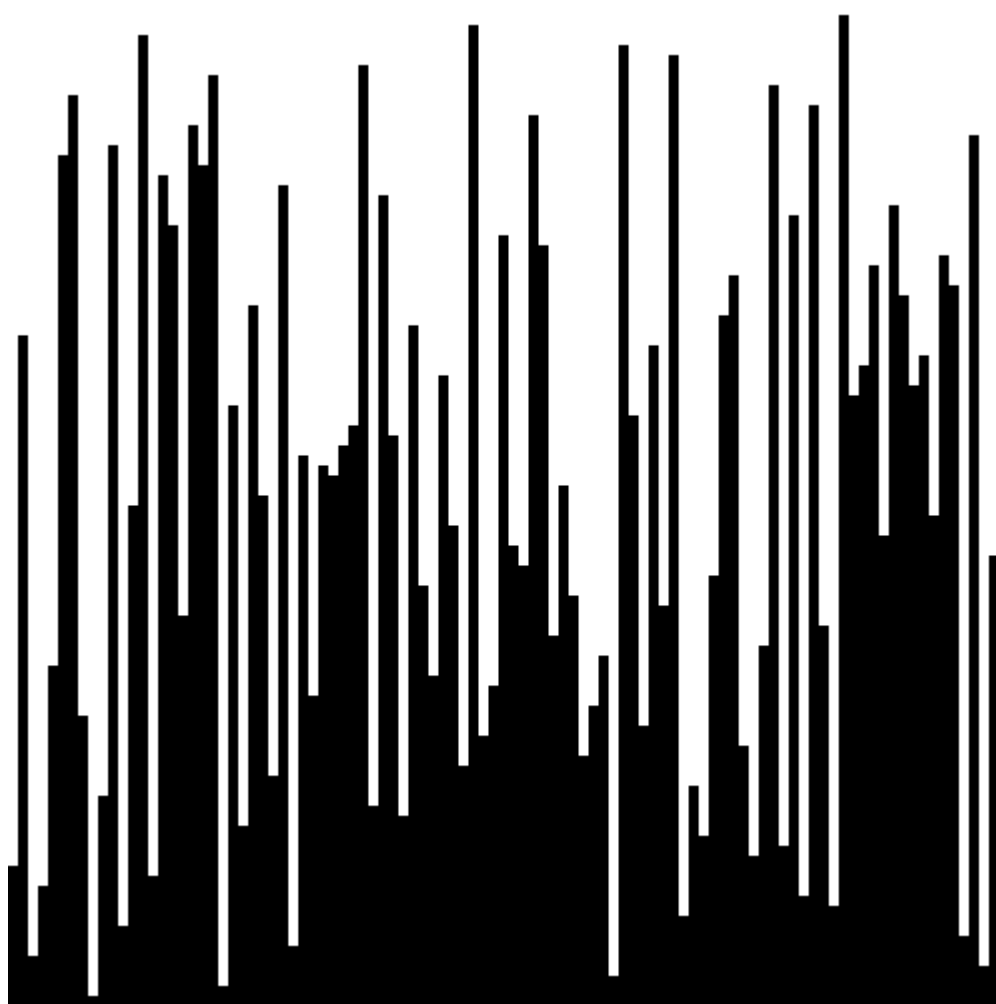


Sequenz von Vergleichen
und evtl. Vertauschungen



Kann auch umgekehrt arbeiten, so dass die Minima nach links wandern.

Bubble Sort



BubbleSort(Array A)

1. **for** $j \leftarrow \text{length}(A) - 1$ **downto** 1 **do**
2. **for** $i \leftarrow 1$ **to** j **do**
3. **if** $A[i] > A[i+1]$ **then** swap($A, i, i+1$)

Idee BubbleSort

- *Die letzten Elemente von j bis n sind sortiert (zu Beginn $j = n-1$)*
- Die größten Elemente steigen auf (bubblen), wie Luftblasen, die zu ihrer richtigen Position aufsteigen
- *Am Ende ist die gesamte Folge sortiert*

- Alle drei Verfahren finden die Lösung durch schrittweises Sortieren mittels Vergleichen.
- Dabei verkleinern sie in jedem Schritt das Restproblem um eins.
- D.h., der Teil des Arrays der unsortiert ist verkleinert sich mit jedem Durchlauf der äußeren Schleife um 1.

- Einfache vergleichende Sortierverfahren
 - Sortieren durch Einfügen (insertion sort)
 - Sortieren durch Auswählen (selection sort)
 - Sortieren durch Vertauschen (bubble sort)
- Fortgeschrittene vergleichende Sortierverfahren
 - Sortieren durch Gruppieren (quick sort)
 - Sortieren durch Mischen (merge sort)
- **Nicht vergleichende Sortierverfahren**
 - **Sortieren durch Zählen (count sort)**
 - Sortieren durch Fachverteilen (radix sort)

Schnelle, digitale Sortierverfahren

- Unter gewissen Einschränkungen des Wertebereichs können die Werte dazu verwendet werden, den endgültigen Platz direkt anzusteuern.
 - Sortieren durch Zählen (count sort)
 - Sortieren durch Fachverteilen (radix sort)
- Diese Verfahren sind jedoch nicht immer sinnvoll einsetzbar, z.B. wenn
 - Das Sortieren stabil sein soll, d.h. positionstreu
 - Der Wertebereich zu groß ist

Sortieren durch Zählen (count sort)

- Annahme:
 - Die Werte stammen aus einem kleinen Wertebereich, d.h. sie liegen so dicht, dass sie zum Indizieren eines Arrays verwendet werden können.
 - Es ist wahrscheinlich, dass Werte mehrfach auftreten.
- Idee
 - Die Häufigkeit jedes Elements wird ermittelt und daraus wird die endgültige Lage im Zielbehälter berechnet (streuendes Umspeichern).
 - Zum Schluss kann die Folge in den ursprünglichen Behälter zurückkopiert werden.

Count Sort – Beispiel

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7

Count Sort – Beispiel

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
Ausgabe							
Index	1	2	3	4	5	6	7

Count Sort – Beispiel

Zwischenspeicher

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
Ausgabe							
Index	1	2	3	4	5	6	7

Wert	Anzahl

Count Sort – Beispiel

Zwischenspeicher

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
Ausgabe							
Index	1	2	3	4	5	6	7

Wert	Anzahl
1	
2	
3	
4	
5	

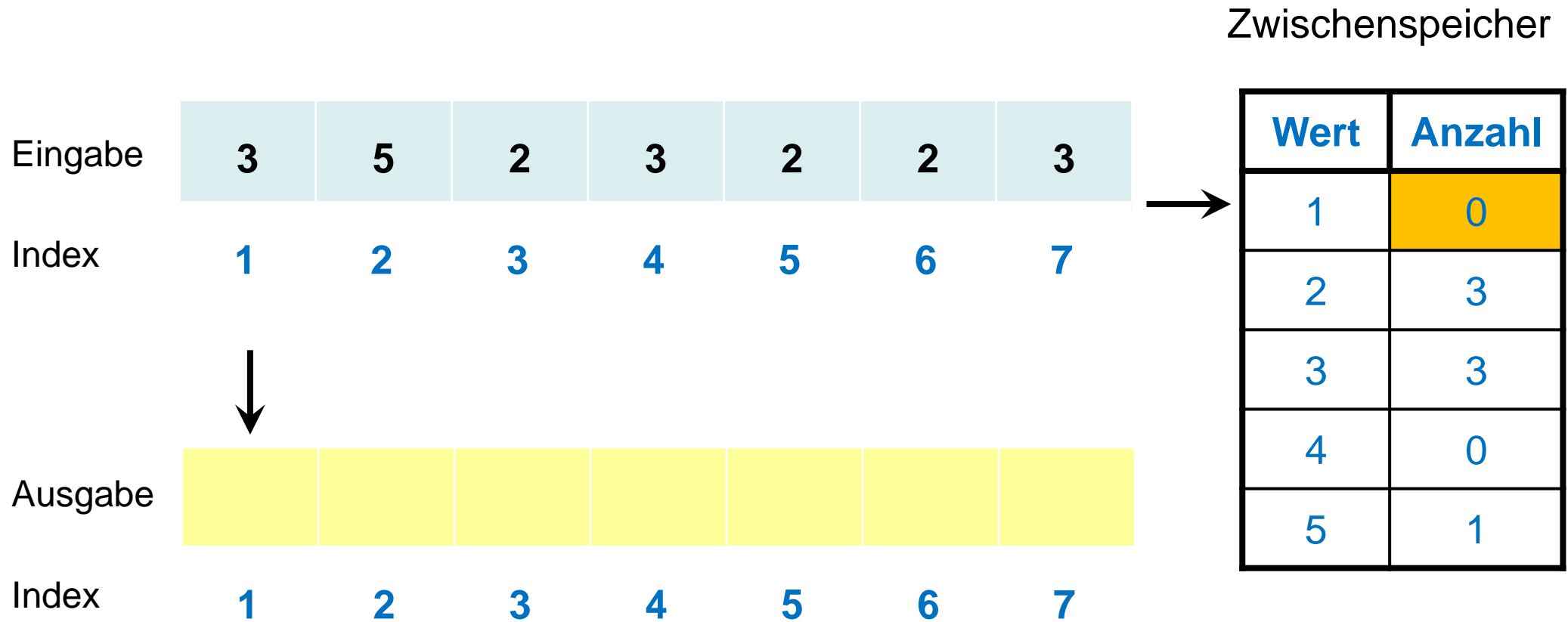
Count Sort – Beispiel

Zwischenspeicher

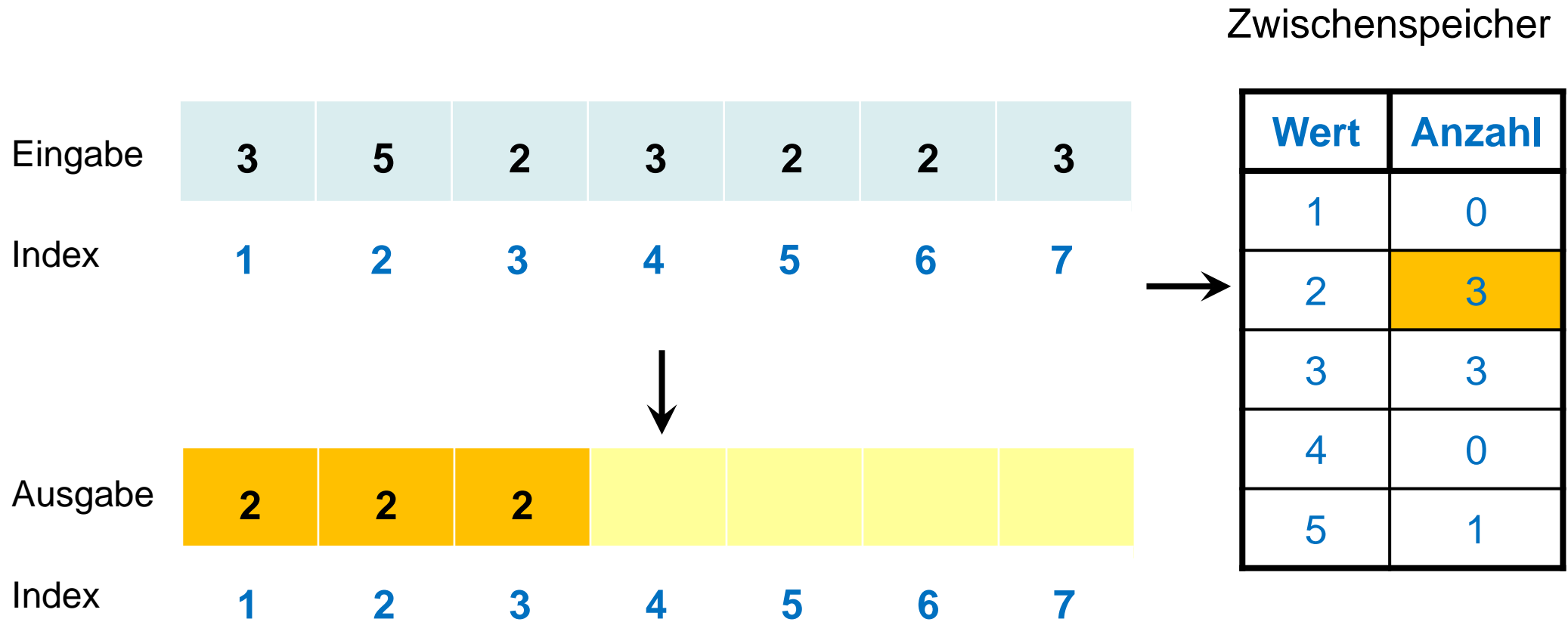
Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
Ausgabe							
Index	1	2	3	4	5	6	7

Wert	Anzahl
1	0
2	3
3	3
4	0
5	1

Count Sort – Beispiel



Count Sort – Beispiel



Count Sort – Beispiel

Zwischenspeicher

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
							↓
Ausgabe	2	2	2	3	3	3	
Index	1	2	3	4	5	6	7

Wert	Anzahl
1	0
2	3
3	3
4	0
5	1

Count Sort – Beispiel

Zwischenspeicher

Eingabe	3	5	2	3	2	2	3
Index	1	2	3	4	5	6	7
Ausgabe	2	2	2	3	3	3	
Index	1	2	3	4	5	6	7

Wert	Anzahl
1	0
2	3
3	3
4	0
5	1

Count Sort – Beispiel

Zwischenspeicher

Eingabe	3	5	2	3	2	2	3
---------	---	---	---	---	---	---	---

Index	1	2	3	4	5	6	7
-------	---	---	---	---	---	---	---

Ausgabe	2	2	2	3	3	3	5
---------	---	---	---	---	---	---	---

Index	1	2	3	4	5	6	7
-------	---	---	---	---	---	---	---

Wert	Anzahl
1	0
2	3
3	3
4	0
5	1



Count Sort – Beispiel

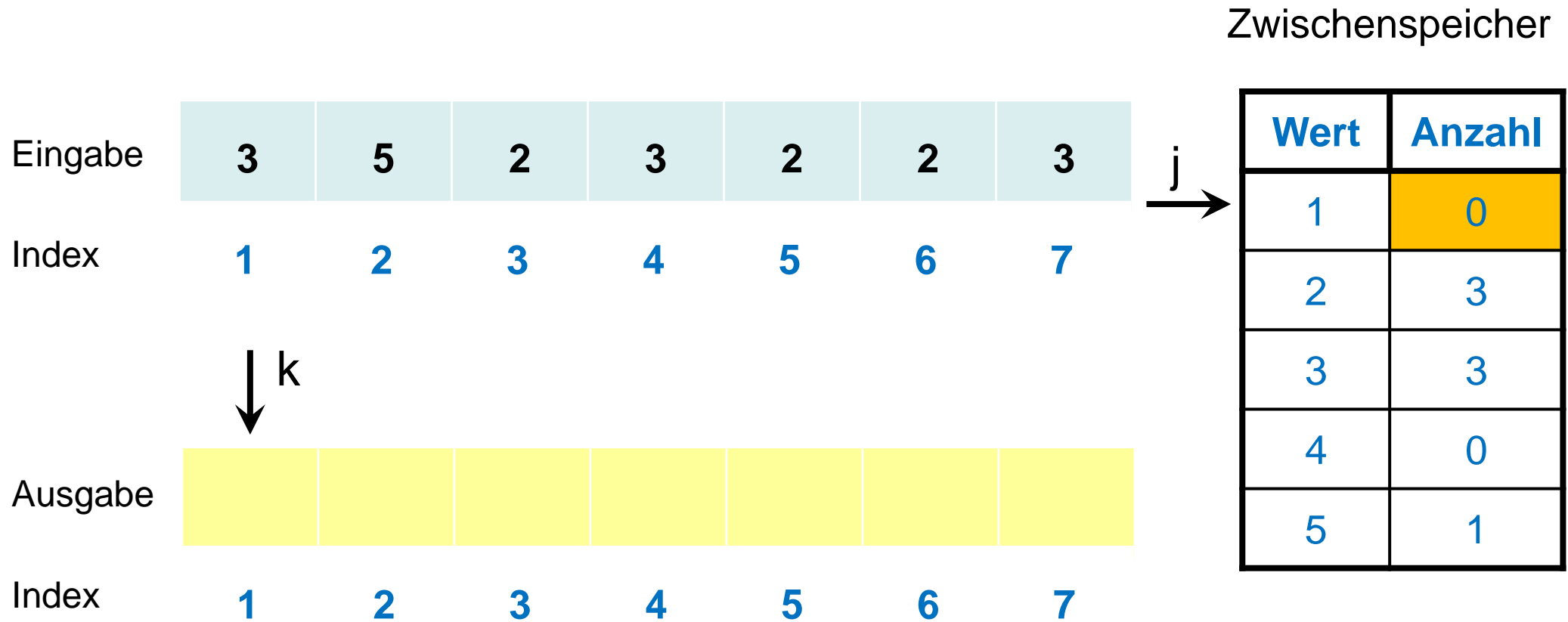
Ausgabe	2	2	2	3	3	3	5
Index	1	2	3	4	5	6	7

CountSort(Array A_in, Array A_out)

1. C ist Hilfsarray mit 0 initialisiert
2. **for** j \leftarrow 1 **to** length(A) **do**
3. C[A_in[j]] \leftarrow C[A_in[j]] + 1
4. k \leftarrow 1
5. **for** j \leftarrow 1 **to** length(C) **do**
6. **for** i \leftarrow 1 **to** C[j] **do**
7. A_out[k] \leftarrow j
8. k \leftarrow k + 1

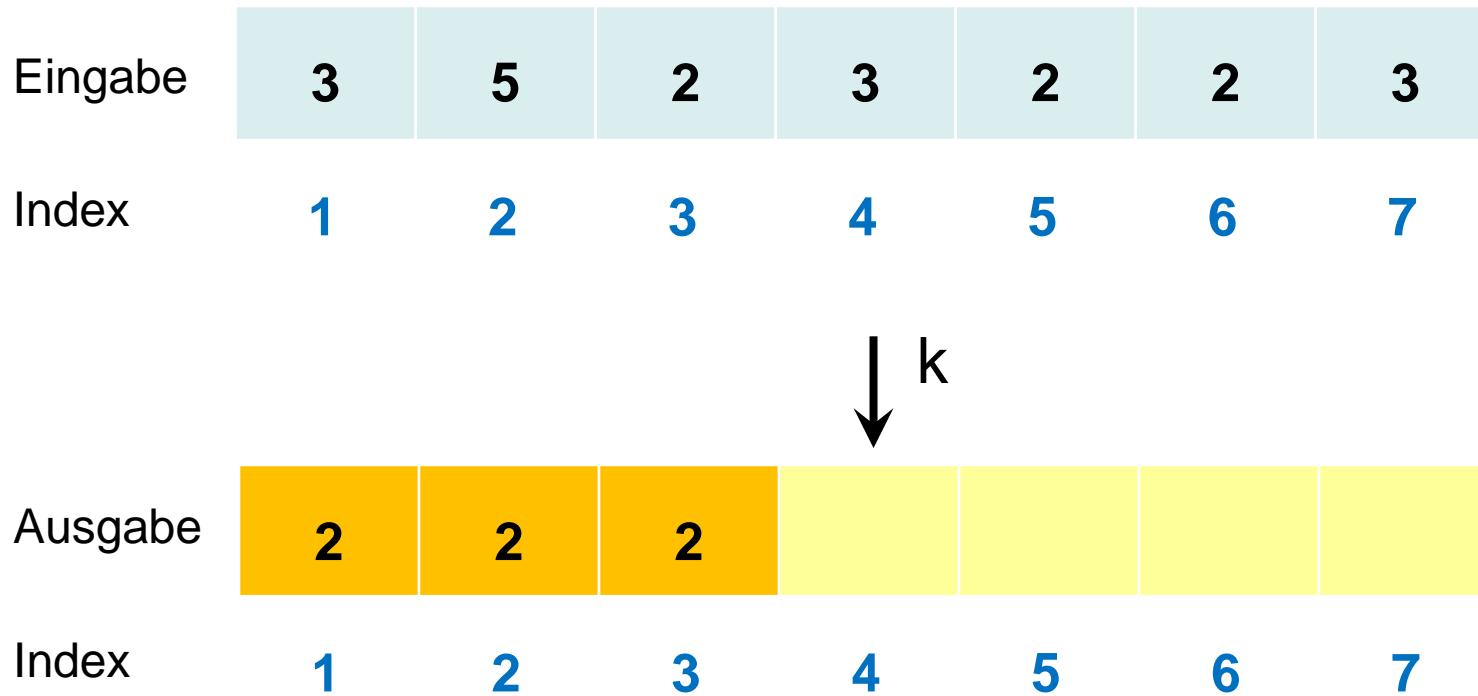
- Annahmen:
- Eingabegröße n
- length(A_in) = length(A_out) = n
- **Wertebereich von A_in: 1 – m**
- **length(C) = m**
- Zähle, wie häufig jedes Element vorkommt
- Füge jedes Element der Reihe nach entsprechend seiner Häufigkeit in das Array hinein.

Count Sort – Beispiel



Count Sort – Beispiel

Zwischenspeicher



j →

Wert	Anzahl
1	0
2	3
3	3
4	0
5	1

2tes Beispiel: Count Sort

$j:$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$A_{in}:$	3	1	3	3	5	7	2	8	9	7	6	3	8	1	8	8

$j:$	1	2	3	4	5	6	7	8	9	10						
$C:$	2	1	4	0	1	1	2	4	1	0						
	1	3	4	8	8	9	10	12	16	17						
	↓	↘	↘	↘	↘	↘	↘	↘	↘	↘						
$A_{out}:$	1	1	2	3	3	3	3	5	6	7	7	8	8	8	8	9
$k:$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Werte
Zähler
Position

CountSort(Array A_in, A_out)

1. C ist Hilfsarray mit 0 initialisiert

Initialisierung des
Hilfsarrays

2. **for** j \leftarrow 1 **to** length(A_in) **do**
3. C[A_in[j]] \leftarrow C[A_in[j]] + 1

Berechnung der
Häufigkeiten

4. k \leftarrow 1
5. **for** j \leftarrow 1 **to** length(C) **do**
6. **for** i \leftarrow 1 **to** C[j] **do**
7. A_out[k] \leftarrow j
8. k \leftarrow k + 1

Schreiben des
sortierten Arrays