Einführung in die Programmierung

Algorithmen, Pseudocode, Sortieren

Weitere Sortieralgorithmen

Überblick

- 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

Überblick

- 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)

Sortieren

Problem: Sortieren

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

Beispiel:

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

Wiederholung: Insertion Sort

Insertion Sort

InsertionSort(Array A)

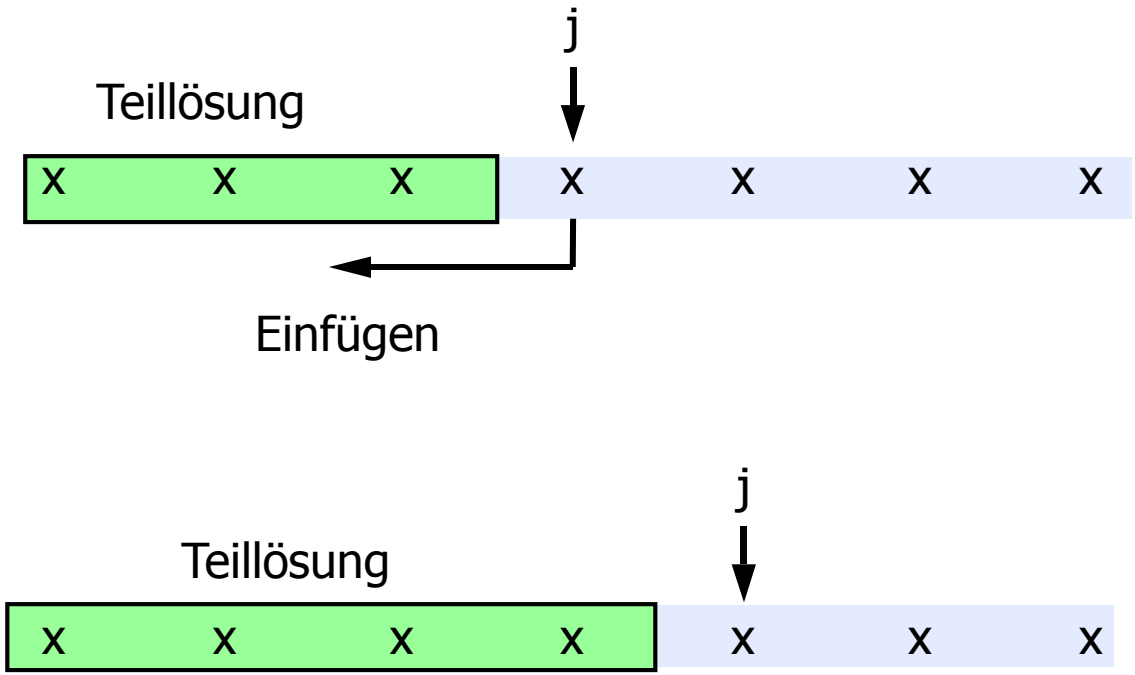
1. **for** j ← 2 **to** length(A) **do**
2. key ← A[j]
3. i ← j-1
4. **while** i>0 and A[i]>key **do**
5. A[i+1] ← A[i]
6. i ← i-1
7. A[i+1] ← key

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)

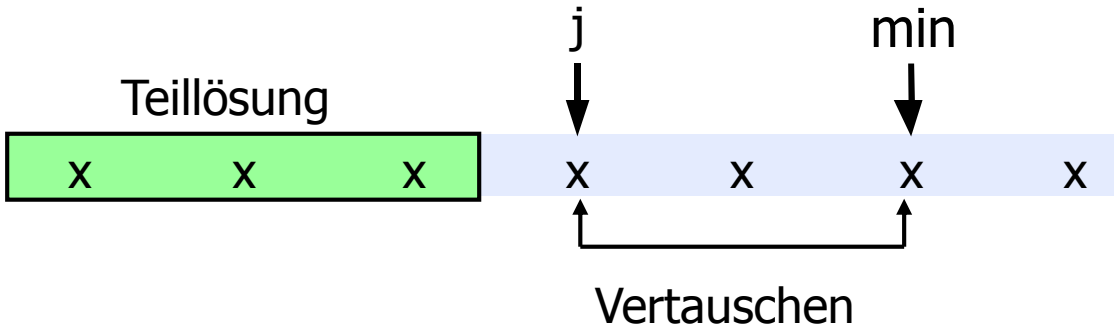
- Arbeitsweise

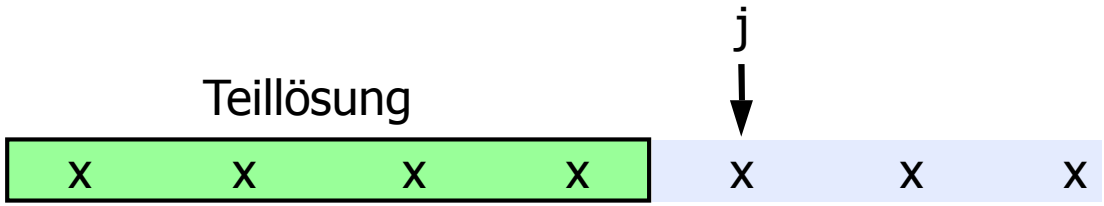


Selection Sort

Sortieren durch Auswählen (selection sort)

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





Selection Sort

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. h ← A[min]
6. A[min] ← A[j]
7. A[j] ← h

Idee SelectionSort

- *Die ersten j-1 Elemente sind sortiert (zu Beginn j=1)*
- *Innerhalb eines Schleifendurchlaufs wird das j-kleineste 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 ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

Idee SelectionSort

- *Die ersten j-1 Elemente sind sortiert (zu Beginn j=1)*
- *Innerhalb eines Schleifendurchlaufs wird das j-kleineste 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 ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel

8	15	3	14	7	6	18	19

Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.            if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

8	15	3	14	7	6	18	19
j							n

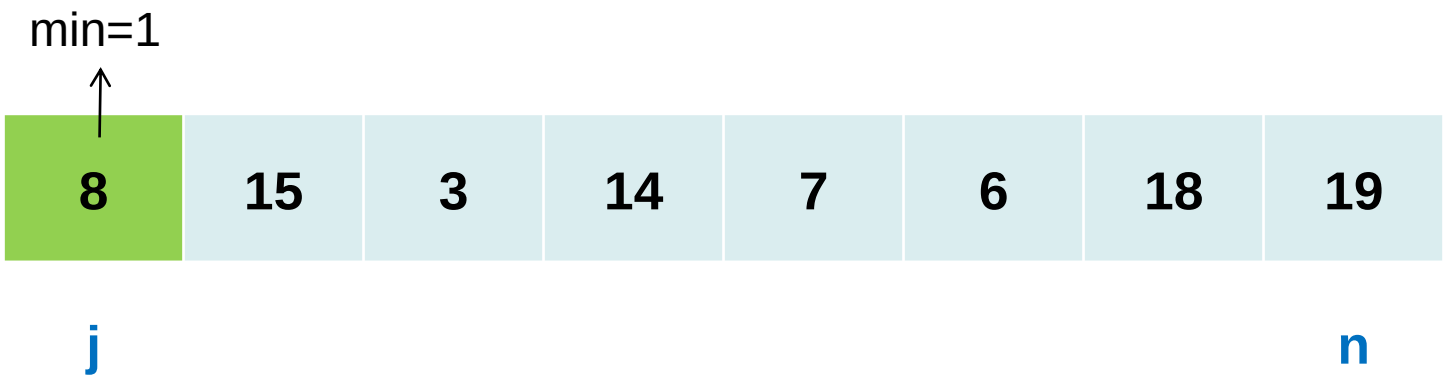
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel

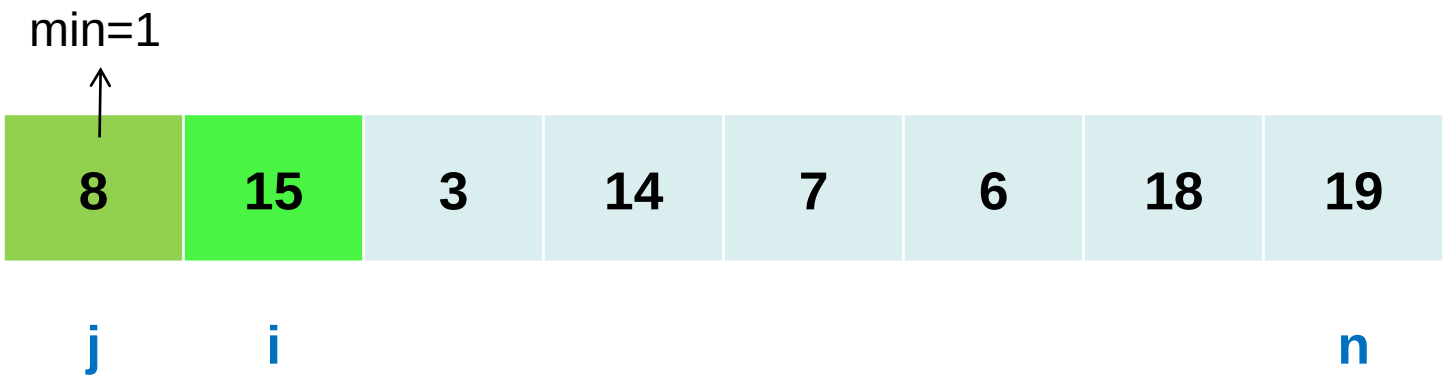


Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.           if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

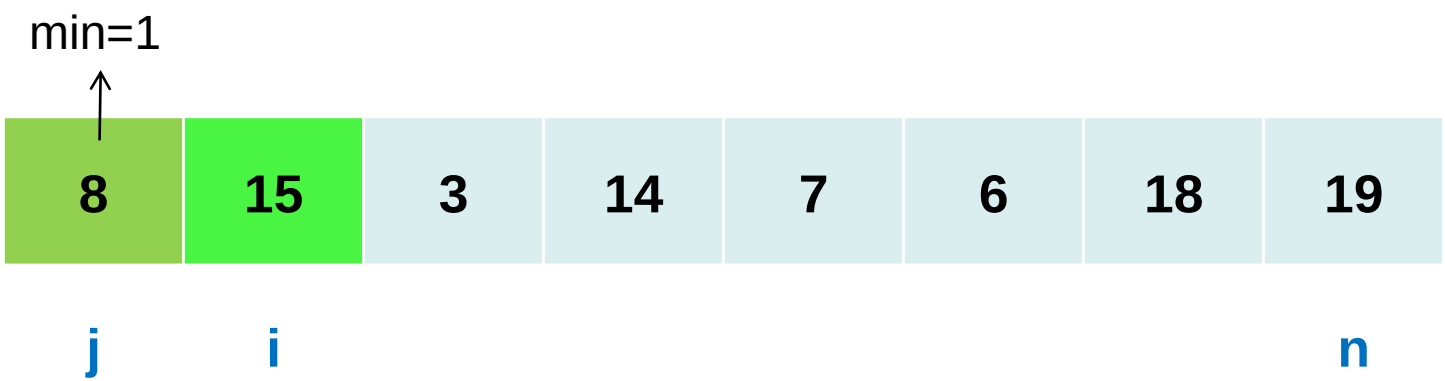


Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.         min ← j
3.         for i ← j + 1 to length(A) do
4.             if A[i] < A[min] then min ← i
5.         swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

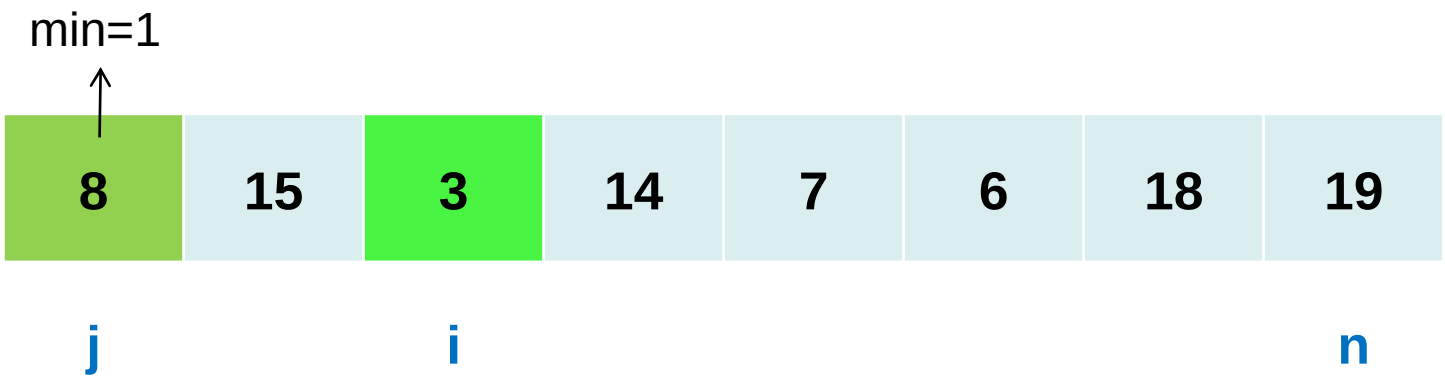


Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.           if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel



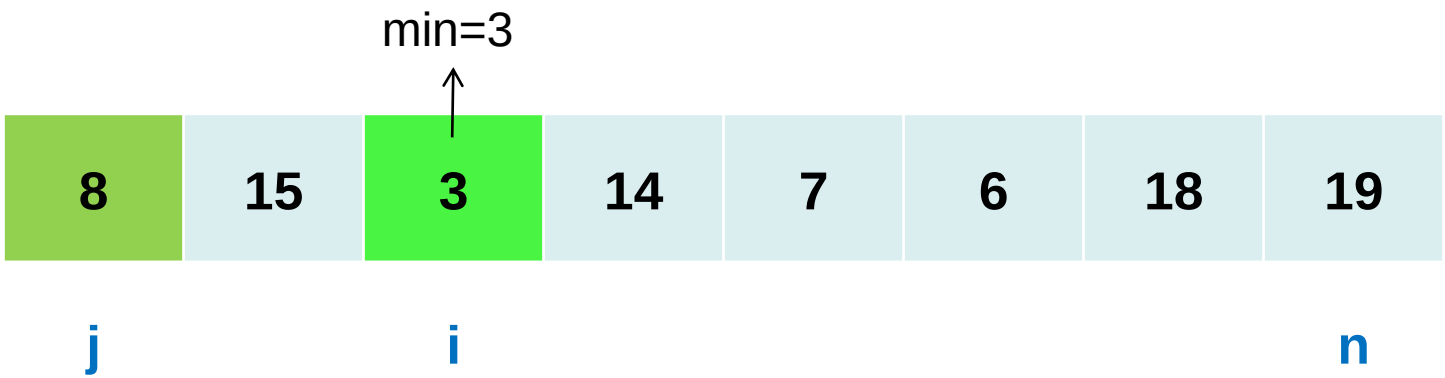
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel

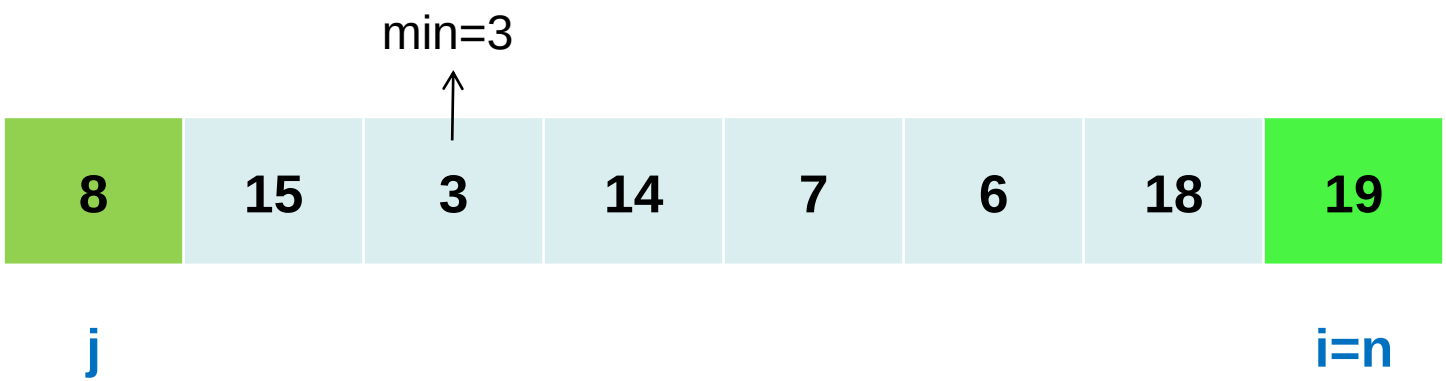


Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.     min ← j
3.     for i ← j + 1 to length(A) do
4.         if A[i] < A[min] then min ← i
5.     swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

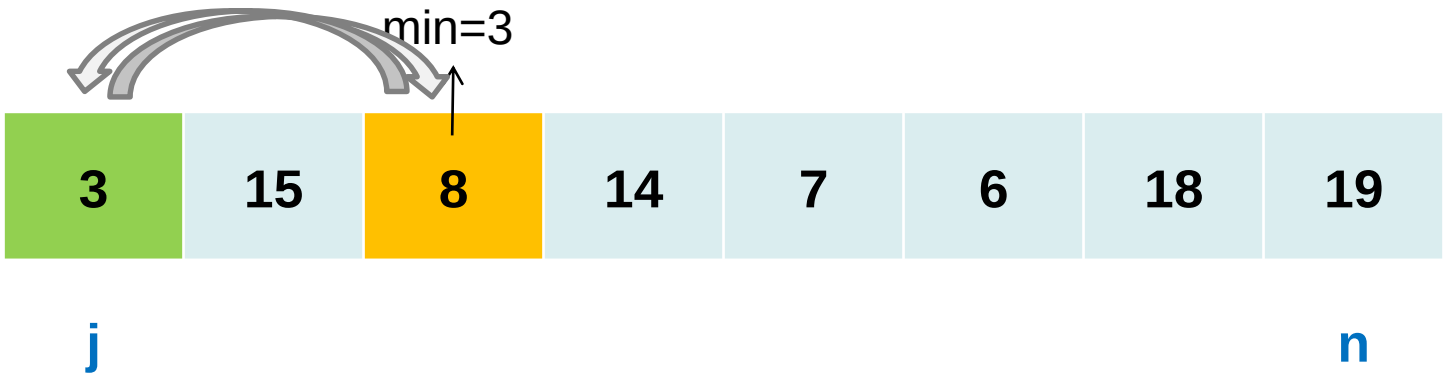


Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.           if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel



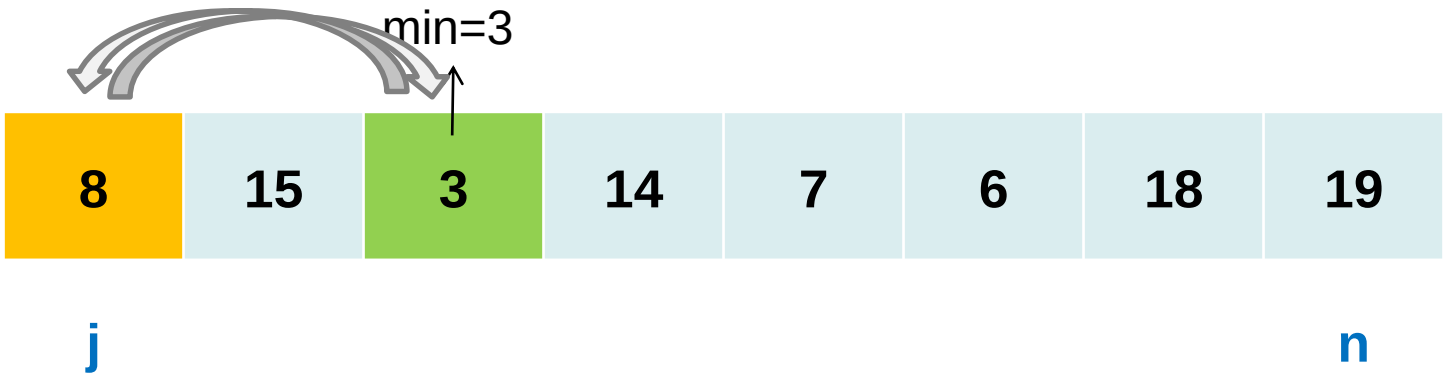
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



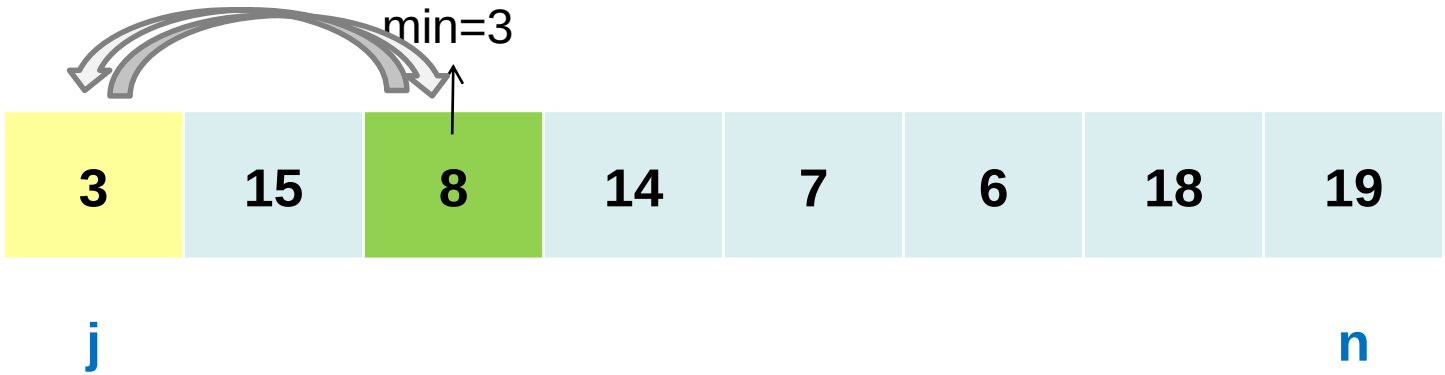
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.            if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

3	15	8	14	7	6	18	19
	j						n

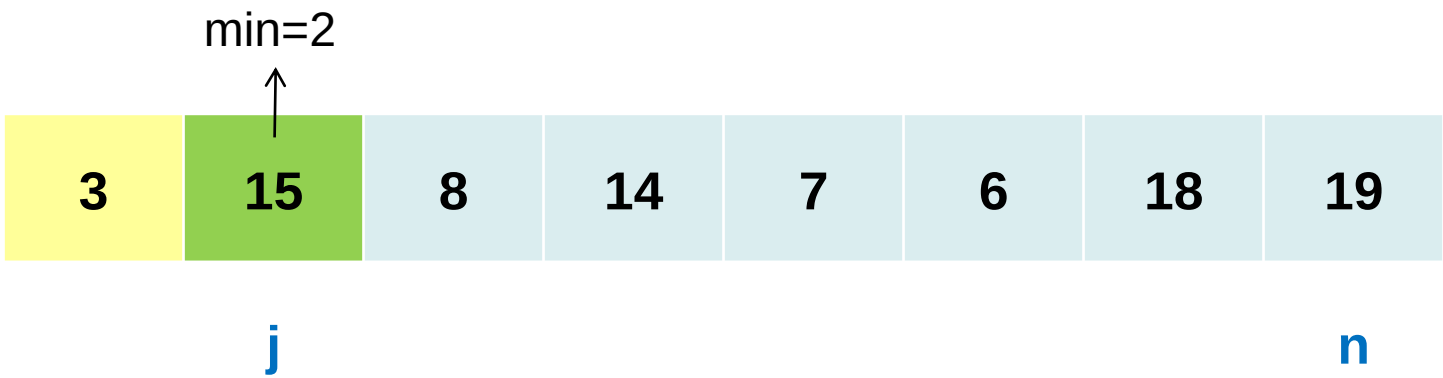
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



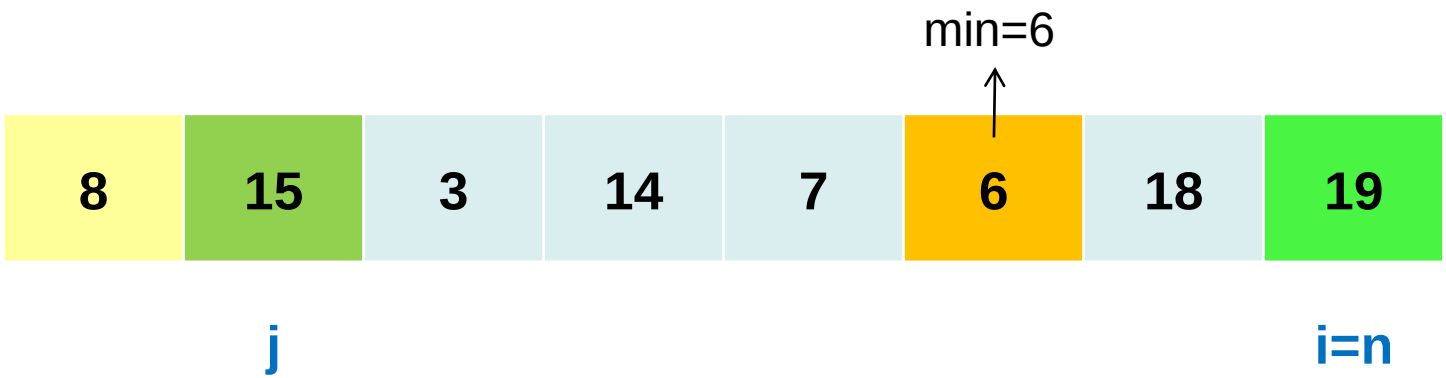
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



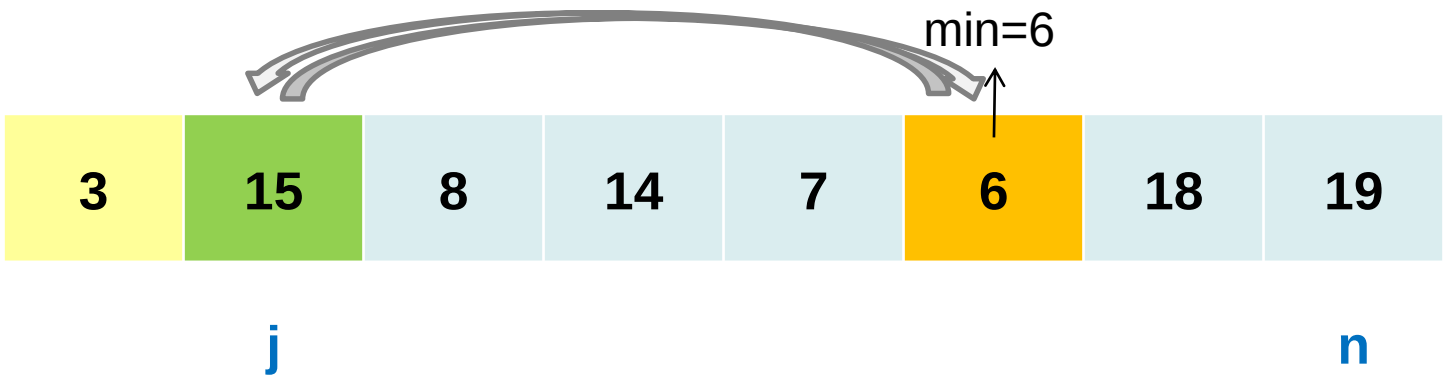
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



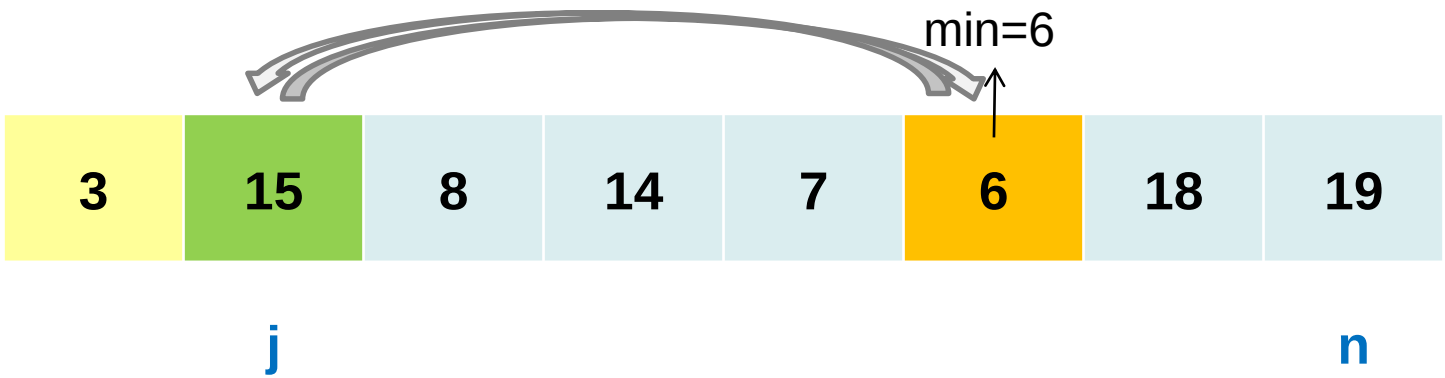
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



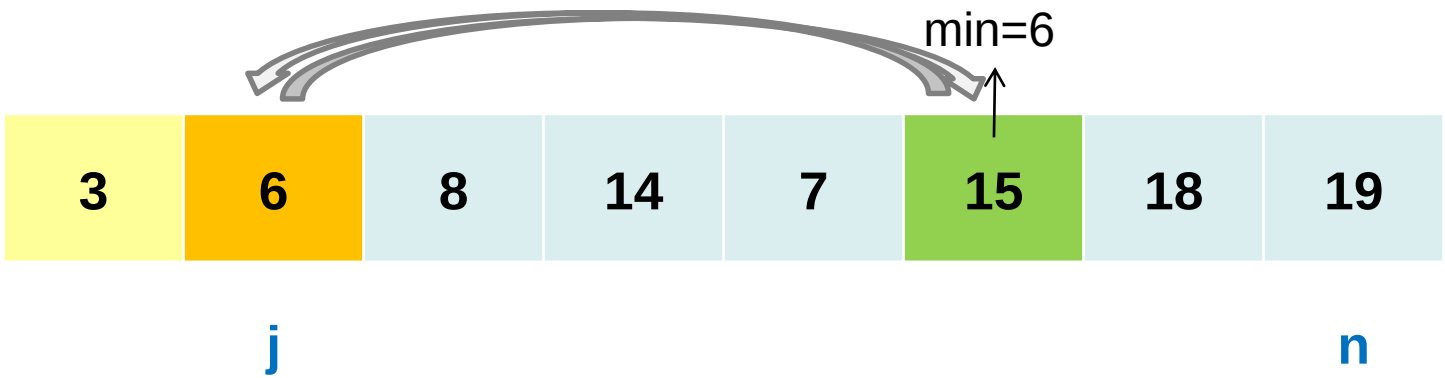
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.            if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

Beispiel

3	6	8	14	7	15	18	19
		j					n

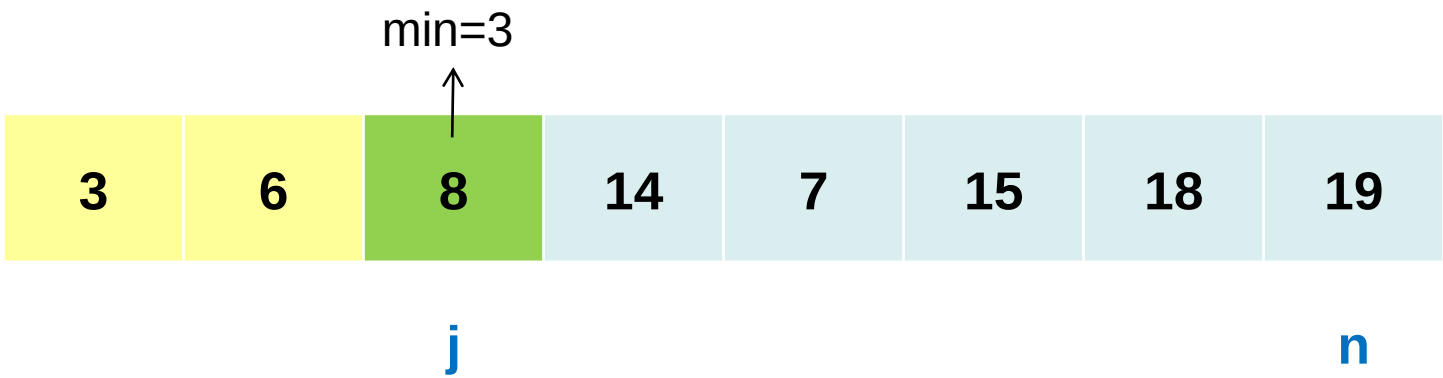
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



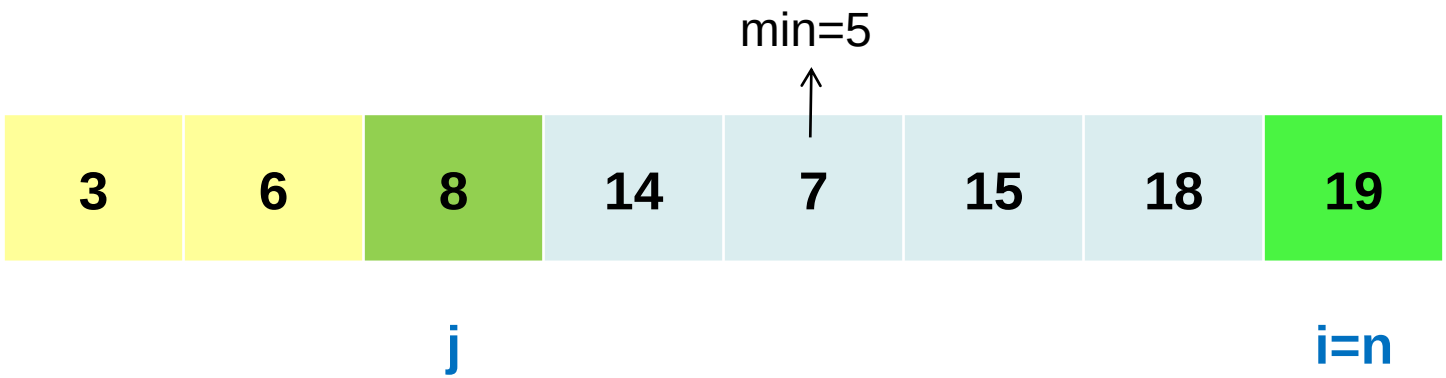
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



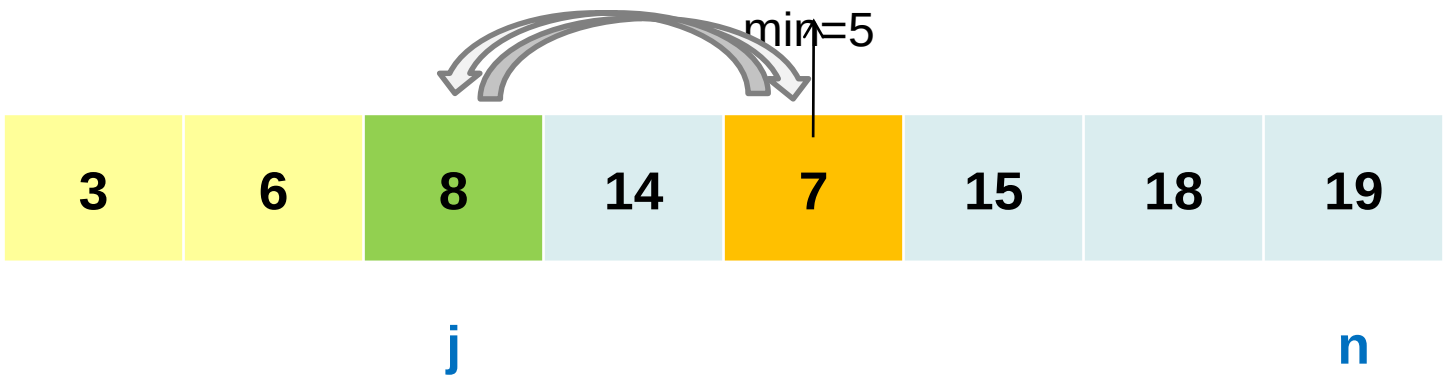
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



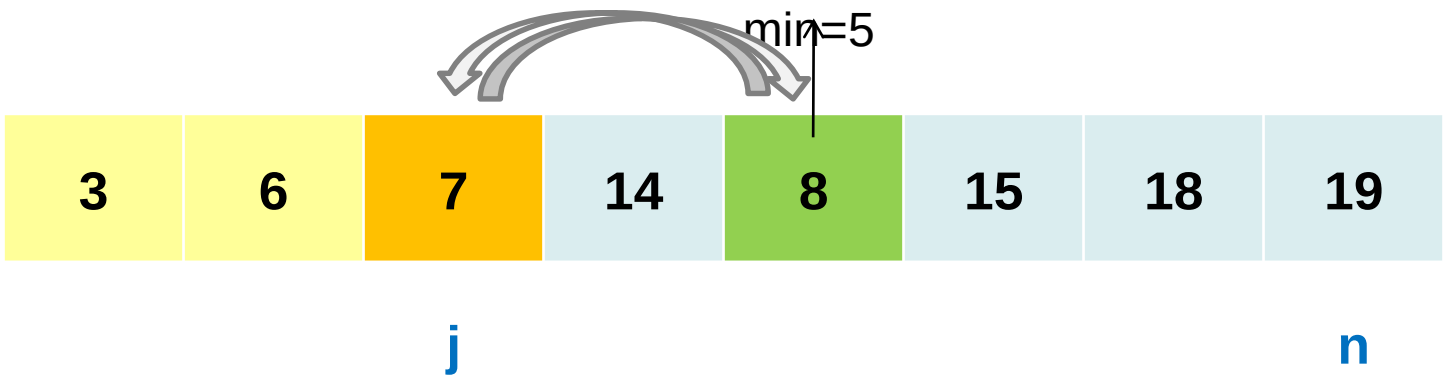
Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

Beispiel

3	6	7	14	8	15	18	19
			j				n

Selection Sort – mit swap

SelectionSort(Array A)

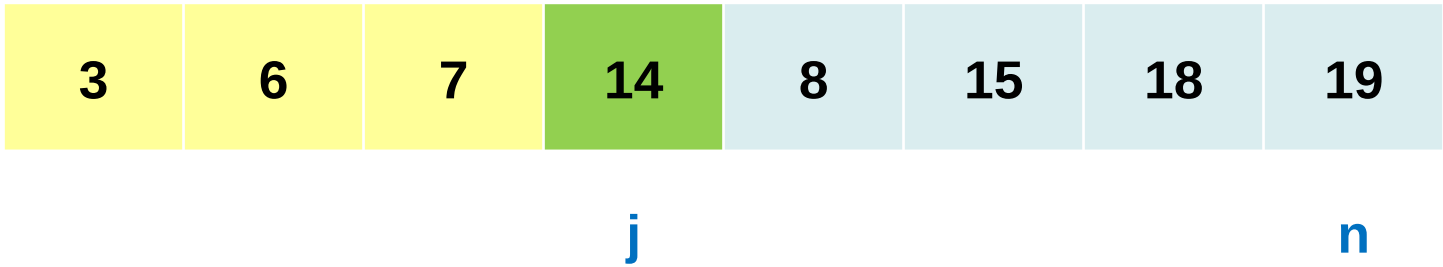
1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

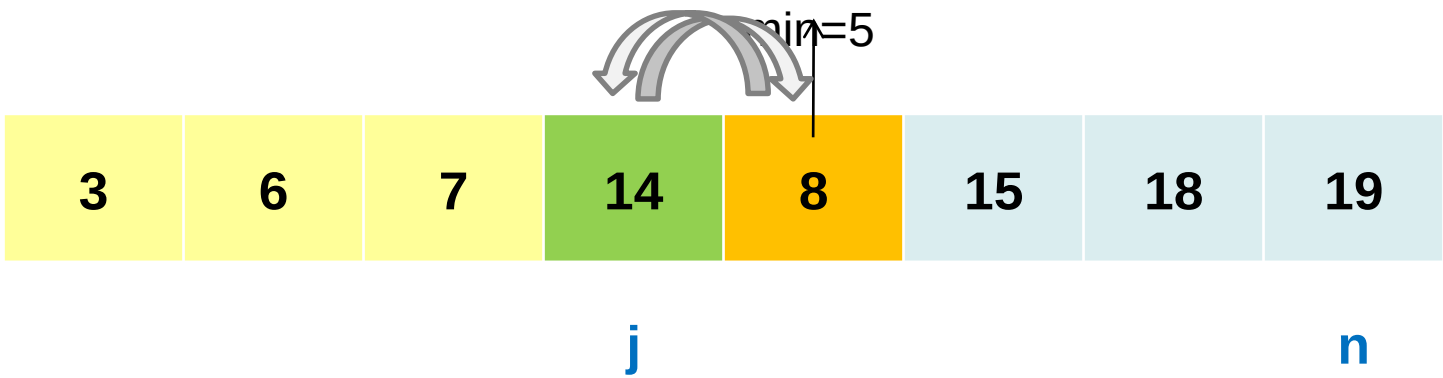
1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

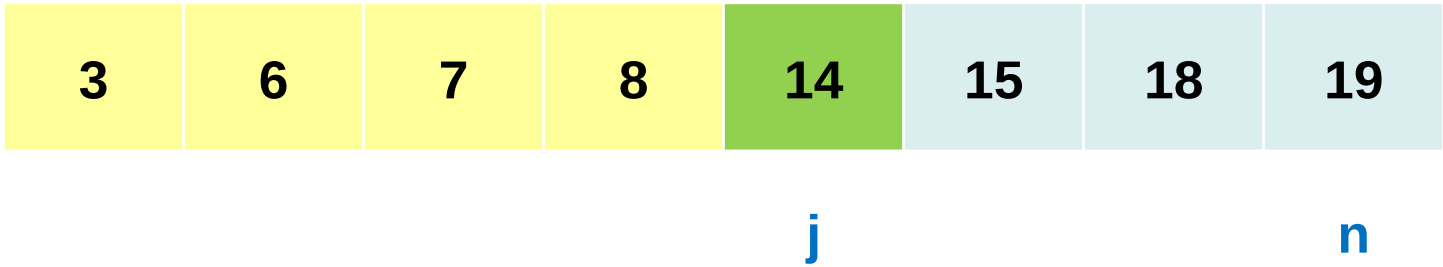
1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

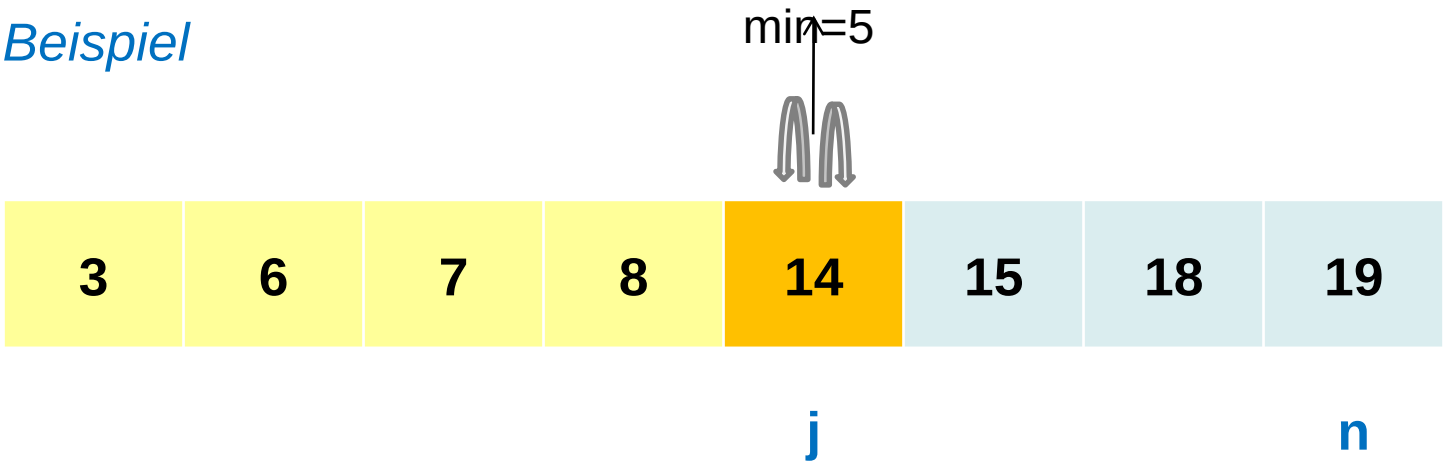
1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



Selection Sort – mit swap

SelectionSort(Array A)

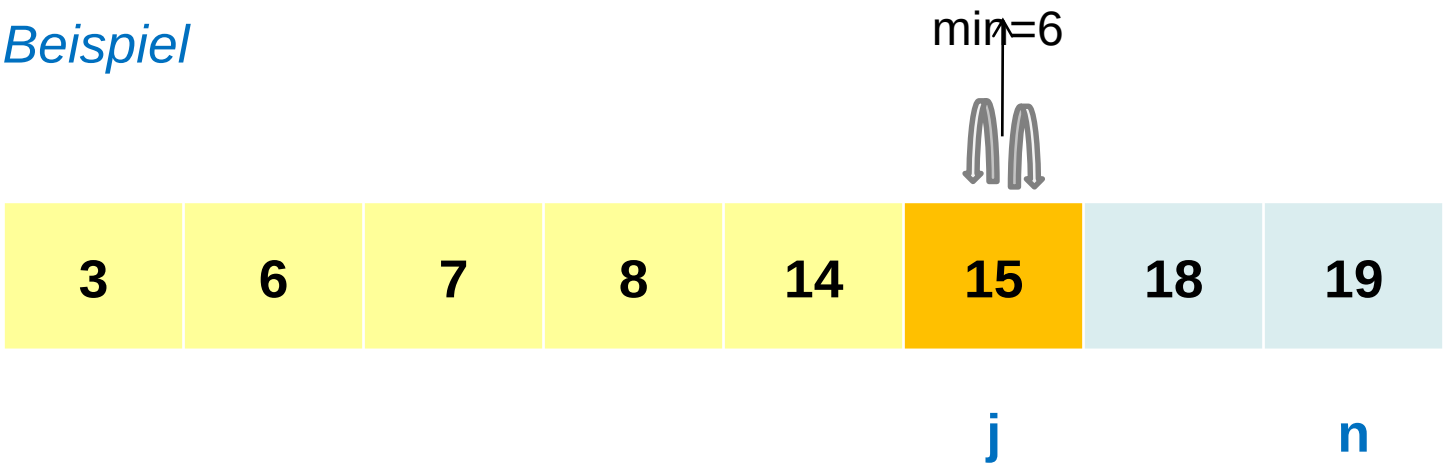
1. **for** j ←1 **to** length(A) - 1 **do**
2. min ← j
3. **for** i ← j + 1 **to** length(A) **do**
4. **if** A[i] < A[min] **then** min ← i
5. swap(A, min, j)

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel



Selection Sort – mit swap

```
SelectionSort(Array A)
1. for j ←1 to length(A) - 1 do
2.       min ← j
3.       for i ← j + 1 to length(A) do
4.           if A[i] < A[min] then min ← i
5.       swap(A, min, j)
```

- Eingabegröße n
- length(A) = n

- Finde das kleinste Element
- In A[j+1…n]

- Vertausche das aktuelle Element mit „dem kleinsten“

Beispiel

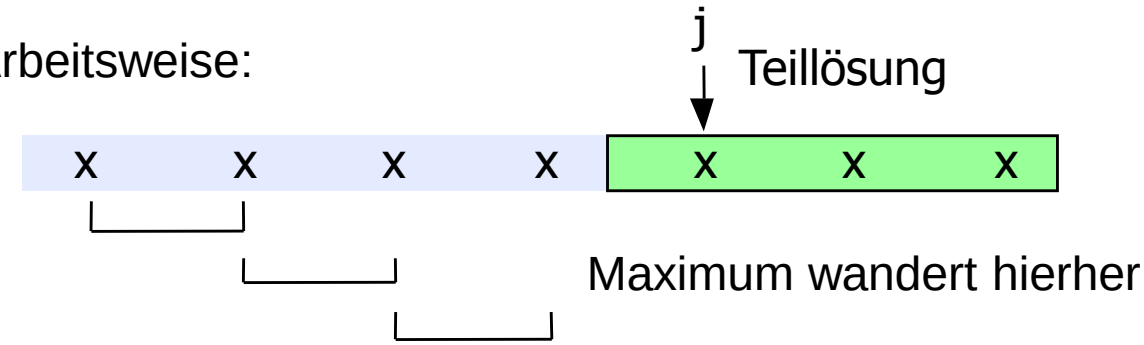
3	6	7	8	14	15	18	19

j=n

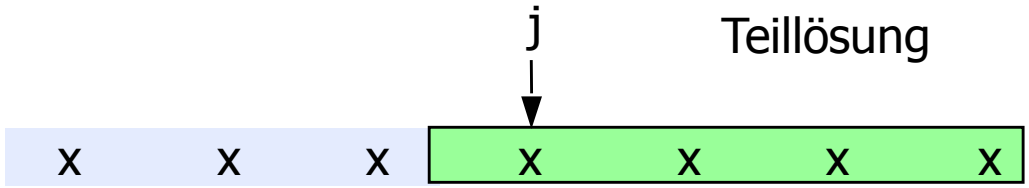
Bubble Sort

Bubble Sort

- Arbeitsweise:

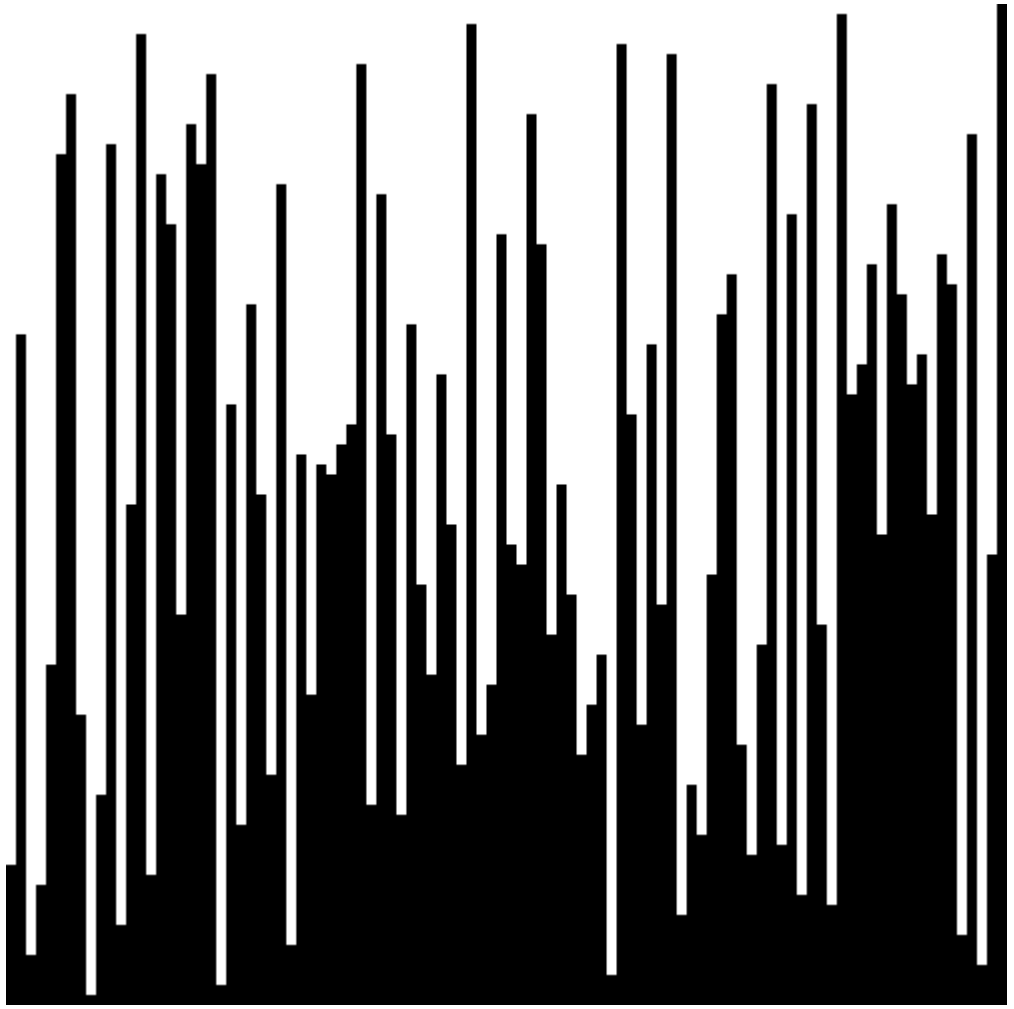


Sequenz von Vergleichen
und evtl. Vertauschungen



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

Bubble Sort



Bubble Sort

BubbleSort(Array A)

1. **for** j ← length(A) -1 **downto** 1 **do**
2. **for** i ←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*

Anmerkungen

- 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.

Überblick

- 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

Wert	Anzahl

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

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

Zwischenspeicher

Wert	Anzahl
1	
2	
3	
4	
5	

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**

Zwischenspeicher

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

Count Sort – Beispiel

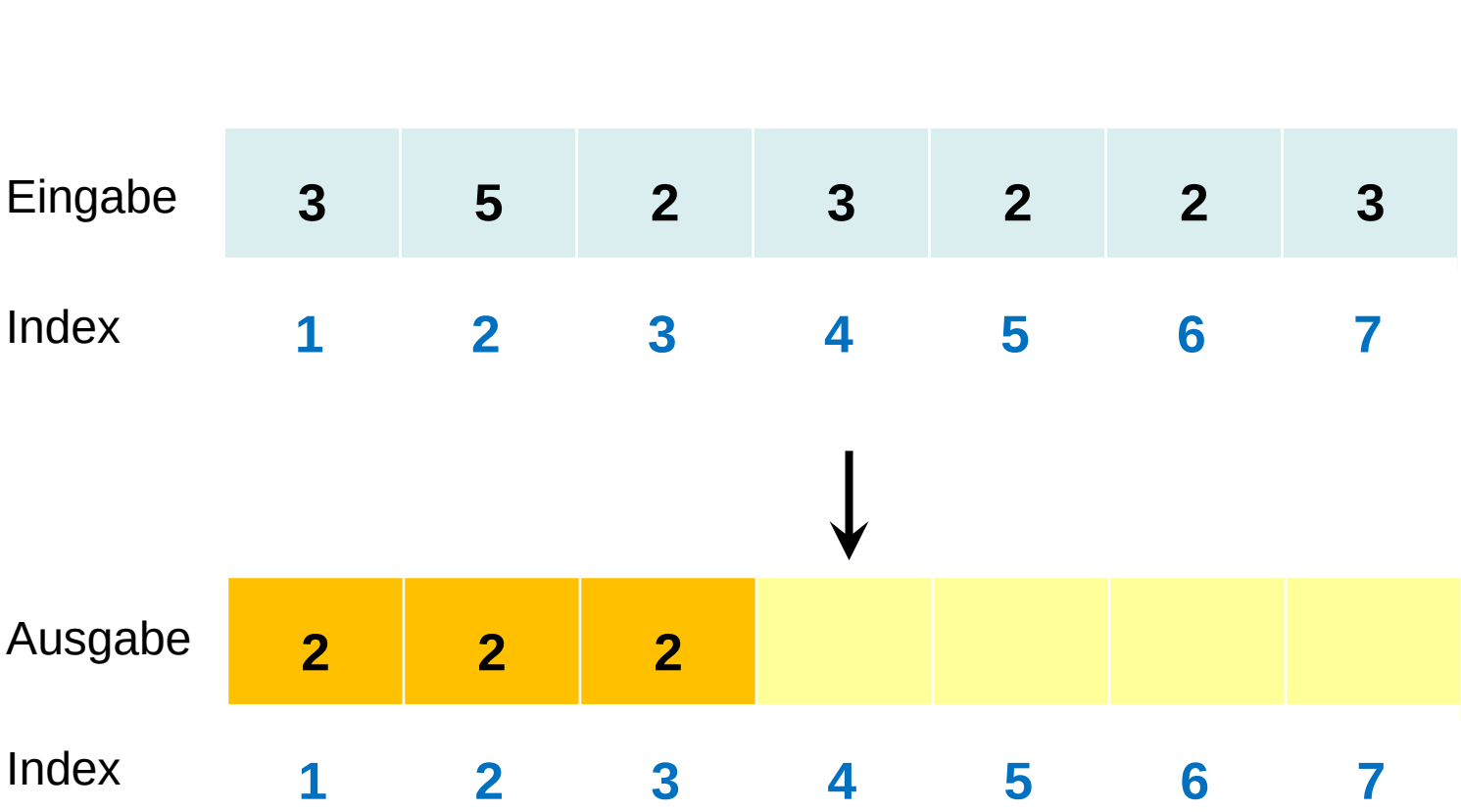
Zwischenspeicher

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

Ausgabe							
Index	1	2	3	4	5	6	7

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

Count Sort – Beispiel



Zwischenspeicher

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	
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**

Count Sort

CountSort(Array A_in, Array A_out)

1. C ist Hilfsarray mit 0 initialisiert

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

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

Zwischenspeicher

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

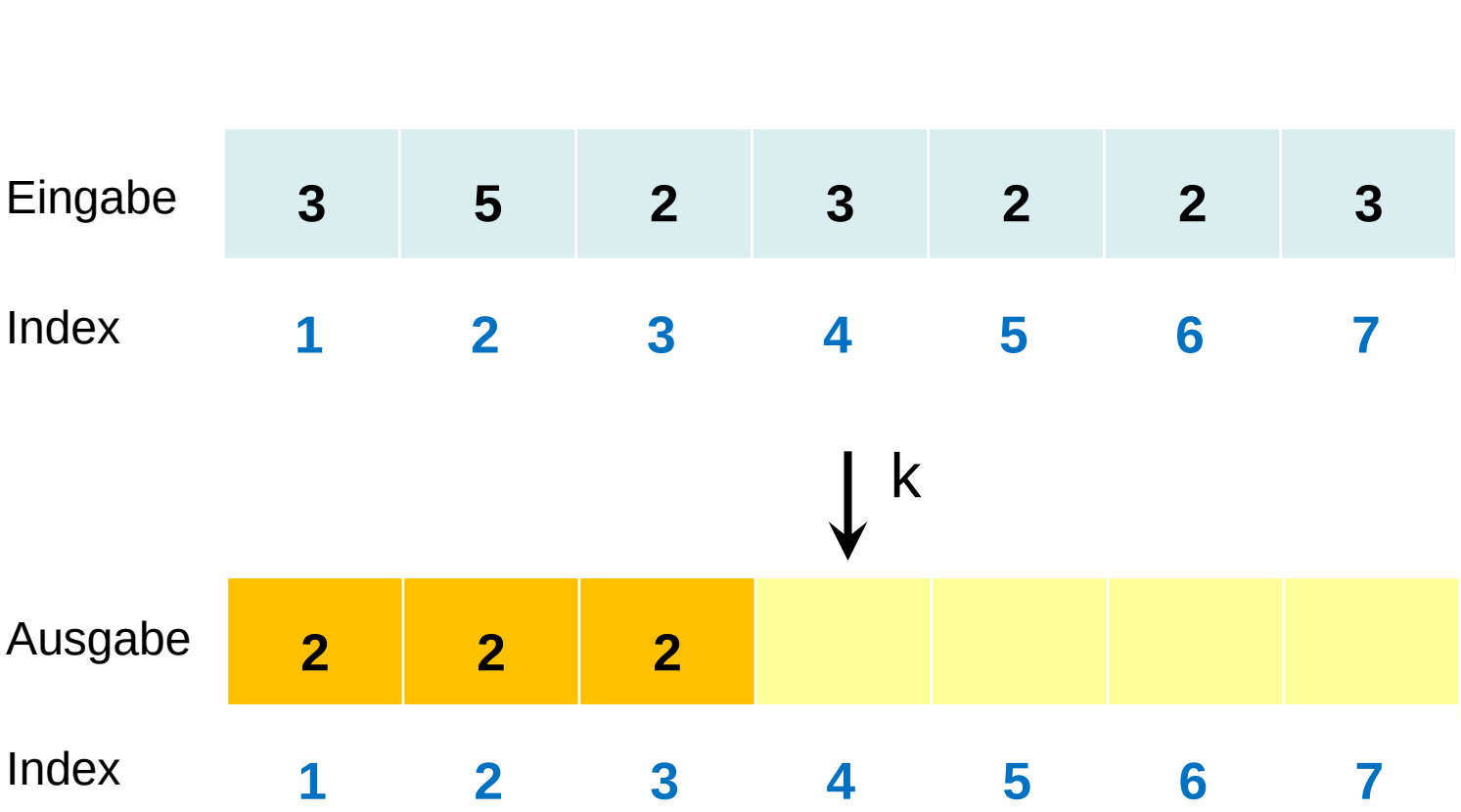
j →

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

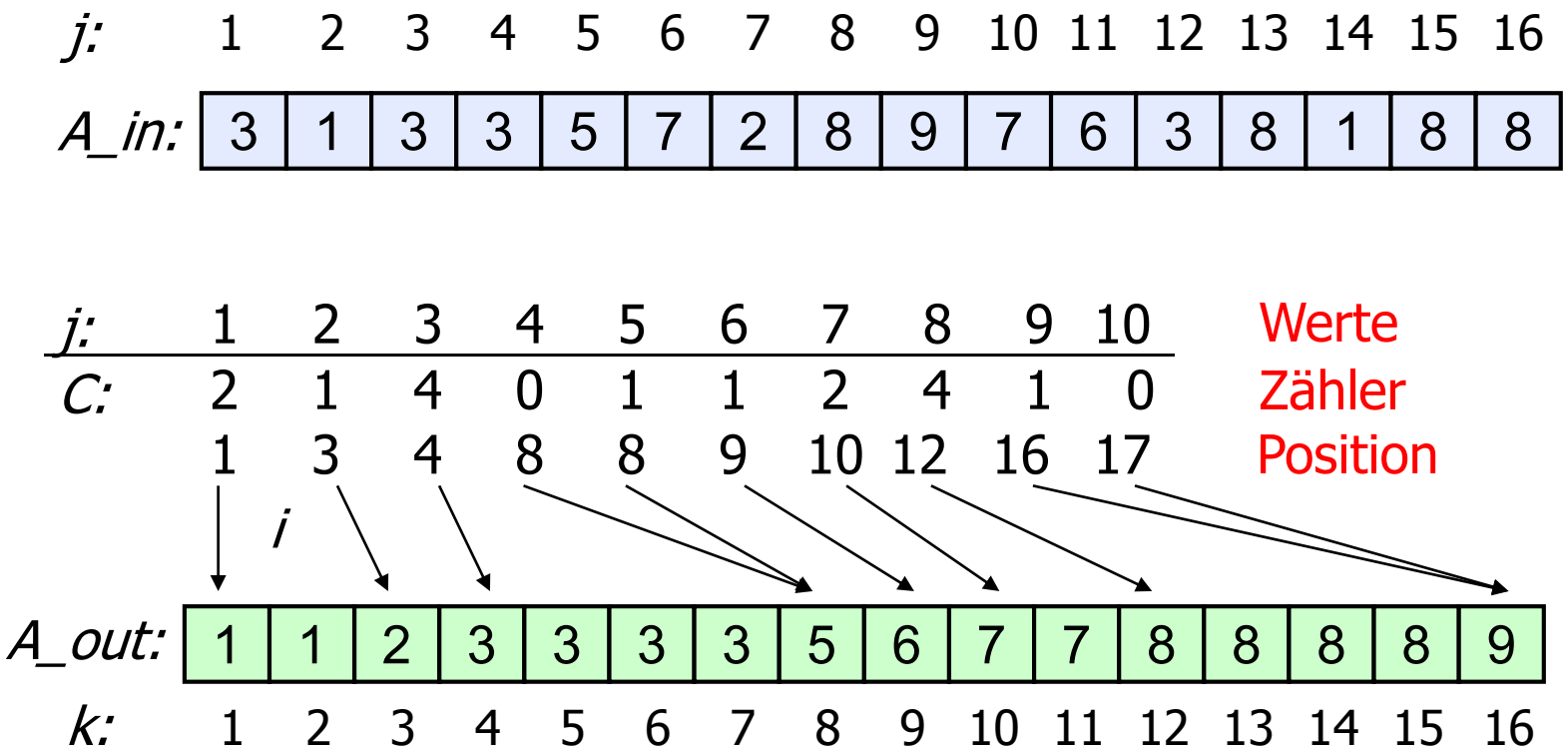
k ↓

Ausgabe							
Index	1	2	3	4	5	6	7

Count Sort – Beispiel



2tes Beispiel: Count Sort



Count Sort

CountSort(Array A_in, A_out)

1. C ist Hilfsarray mit 0 initialisiert

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

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

Initialisierung des Hilfsarrays

Berechnung der Häufigkeiten

Schreiben des sortierten Arrays