

# 3. Inkrementelle Algorithmen

---

*Definition 3.1:* Bei einem **inkrementellen Algorithmus** wird sukzessive die Teillösung für die ersten  $i$  Objekte aus der bereits bekannten Teillösung für die ersten  $i-1$  Objekte berechnet,  $i=1, \dots, n$ .

*Beispiel Min-Search:*

- Objekte sind Einträge des Eingabearrays.
- Teilproblem bestehend aus ersten  $i$  Objekten bedeutet Minimum im Teilarray  $A[1..i]$  bestimmen.

# Sortieren und inkrementelle Algorithmen

---

Eingabe beim Sortieren: Folge von  $n$  Zahlen  $(a_1, a_2, \dots, a_n)$ .

Ausgabe beim Sortieren: Umordnung  $(b_1, b_2, \dots, b_n)$  der Eingabefolge, so dass  $b_1 \leq b_2 \leq \dots \leq b_n$ .

Sortieralgorithmus: Verfahren, das zu jeder Folge  $(a_1, a_2, \dots, a_n)$  sortierte Umordnung  $(b_1, b_2, \dots, b_n)$  berechnet.

Eingabe: (31,41,59,26,51,48)

Ausgabe: (26,31,41,48,51,59)

# Insertion-Sort

---

*Idee:* Sukzessive wird eine Sortierung der Teilarrays  $A[1..i]$ ,  $1 \leq i \leq \text{length}(A)$  berechnet.

Insertion - Sort( $A$ )

```
1 for  $j \leftarrow 2$  to  $\text{length}(A)$ 
2   do  $\text{key} \leftarrow A[j]$ 
3     ▷ Füge  $A[j]$  in die sortierte Folge  $A[1..j-1]$  ein.
4      $i \leftarrow j-1$ 
5     while  $i > 0$  and  $A[i] > \text{key}$ 
6       do  $A[i+1] \leftarrow A[i]$ 
7          $i \leftarrow i-1$ 
8    $A[i+1] \leftarrow \text{key}$ 
```

# 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.      $A[i+1] \leftarrow \text{key}$

**Beispiel:**

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

# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

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

# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

	1						n
8	15	3	14	7	6	18	19

# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

	1	j					n	
	8	15	3	14	7	6	18	19

# Insertion Sort

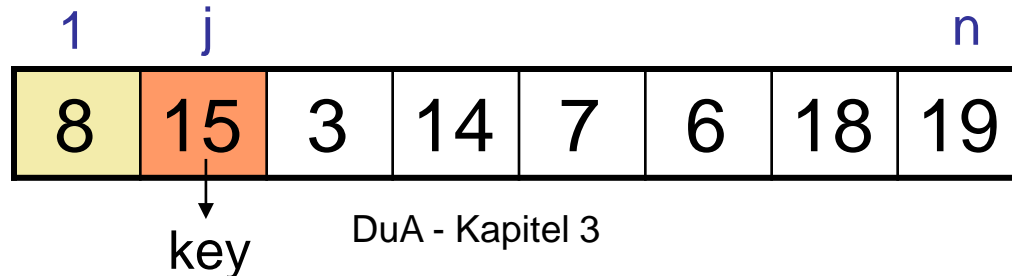
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**





# Insertion Sort

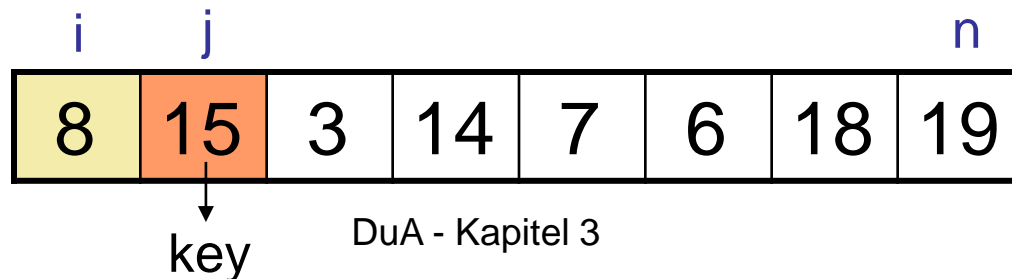
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# Insertion Sort

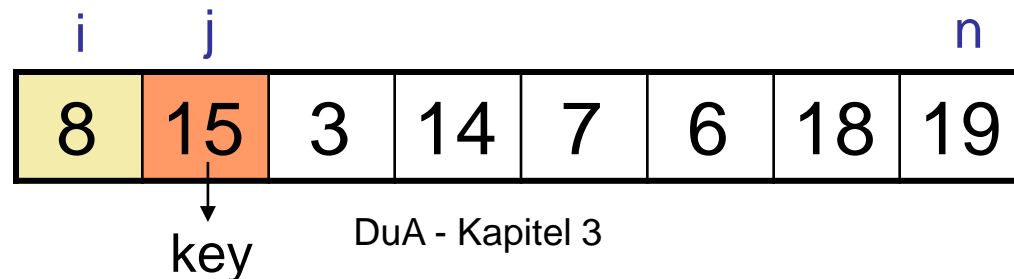
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# Insertion Sort

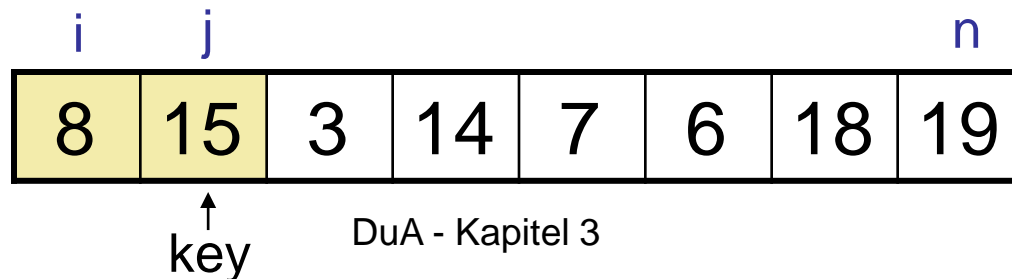
InsertionSort(Array A)

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.          $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

	1		j				n	
	8	15	3	14	7	6	18	19

# Insertion Sort

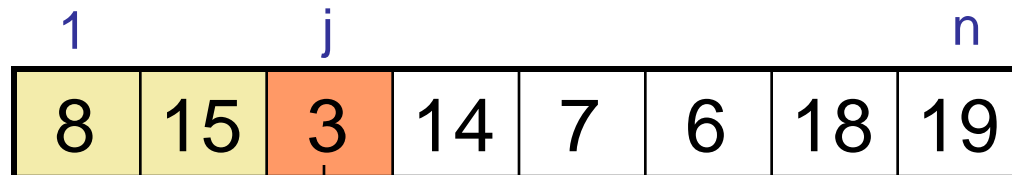
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

	1	i	j				n	
	8	15	3	14	7	6	18	19

# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

1	i	j					n
8	15	3	14	7	6	18	19

# Insertion Sort

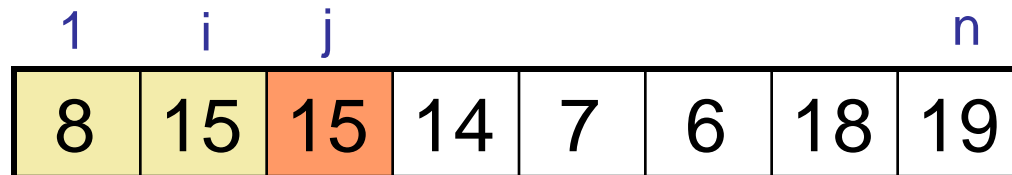
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**





# Insertion Sort

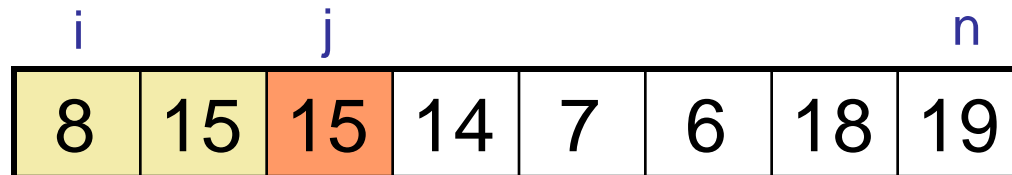
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# 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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**

	$i$		$j$				$n$
8	15	15	14	7	6	18	19

# Insertion Sort

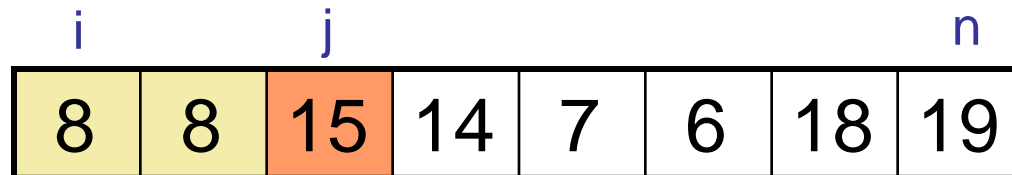
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



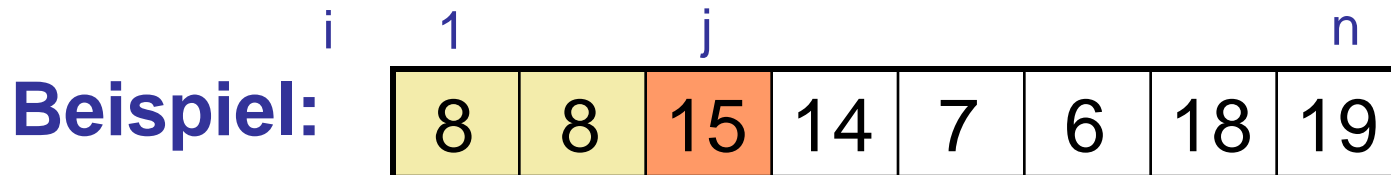
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$



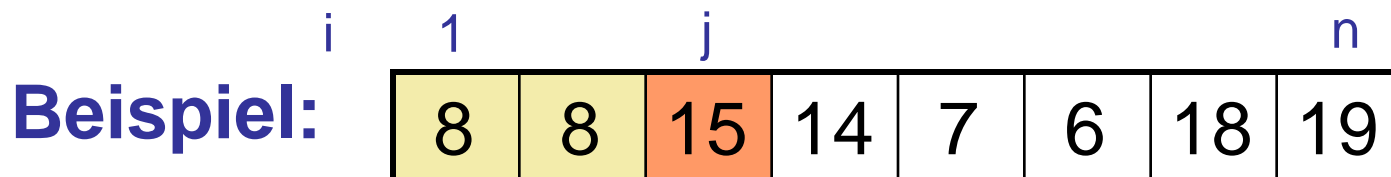
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$



# Insertion Sort

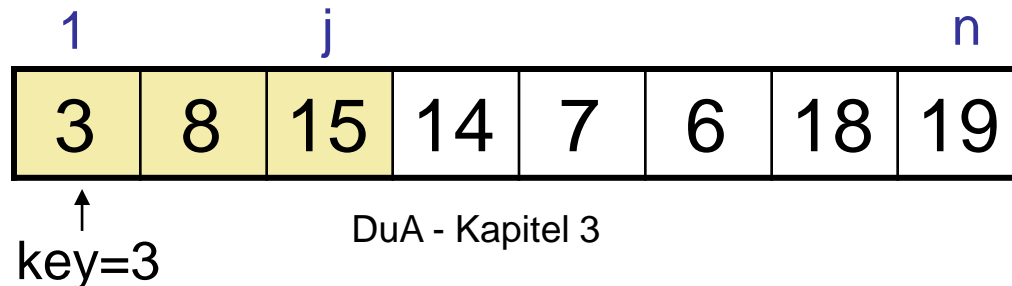
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# Insertion Sort

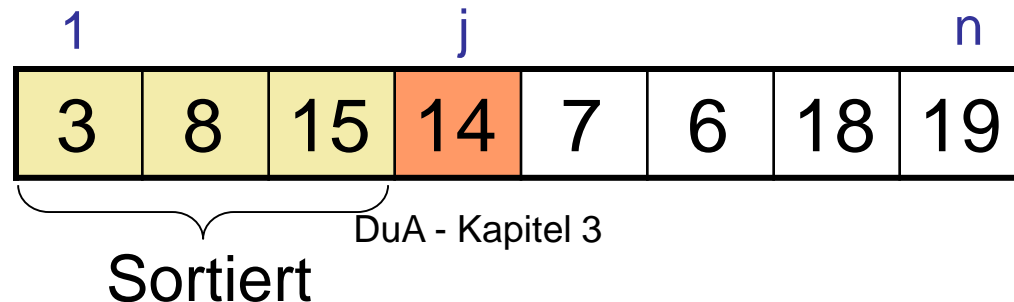
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



# Insertion Sort

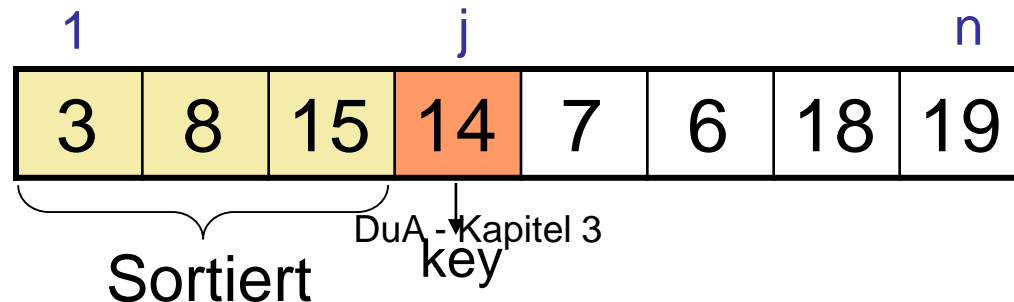
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**





# Insertion Sort

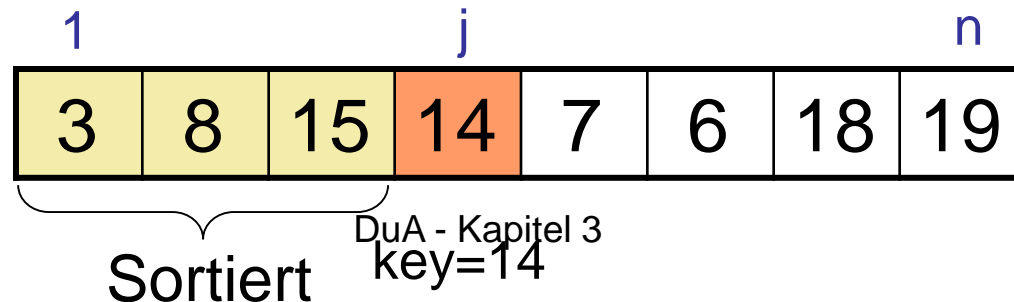
InsertionSort(Array A)

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.      $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤  $(\text{length}(A)=n)$

**Beispiel:**



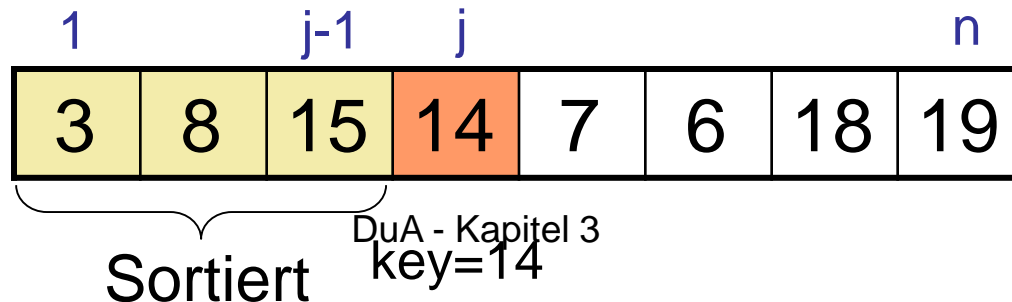
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts

**Beispiel:**



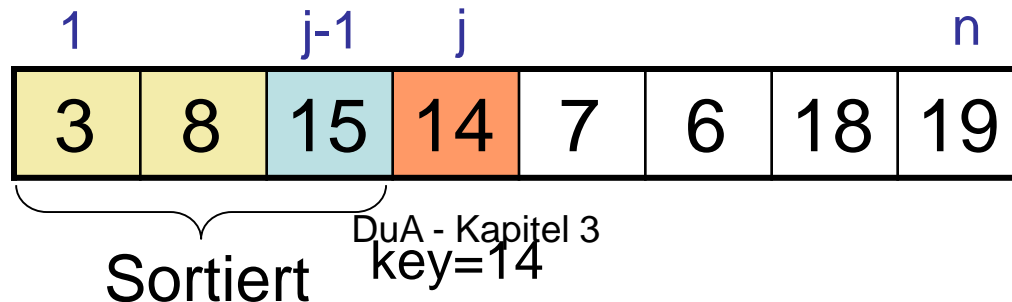
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts

**Beispiel:**



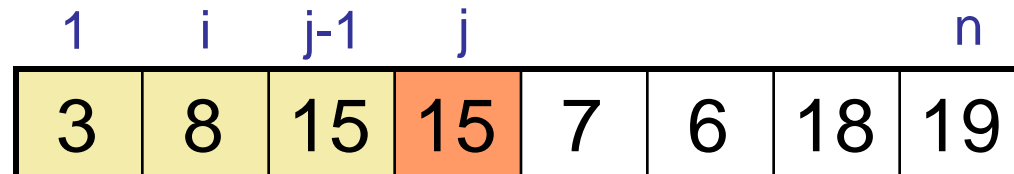
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts

**Beispiel:**



# 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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts

**Beispiel:**

	1	i	i+1	j			n	
	3	8	15	15	7	6	18	19

# 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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**

	1	i	i+1	j			n	
	3	8	15	15	7	6	18	19

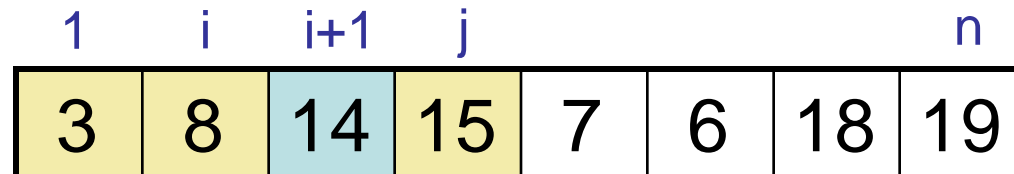
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



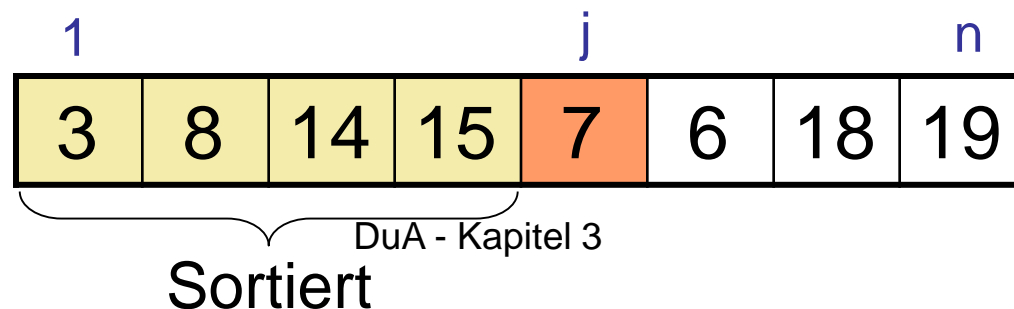
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**





# Insertion Sort

InsertionSort(Array A)

1. **for**  $j \leftarrow 2$  **to** length(A) **do**

2. **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.  $A[i+1] \leftarrow \text{key}$

➤ Eingabegröße  $n$

➤ (length(A)= $n$ )

➤ verschiebe alle

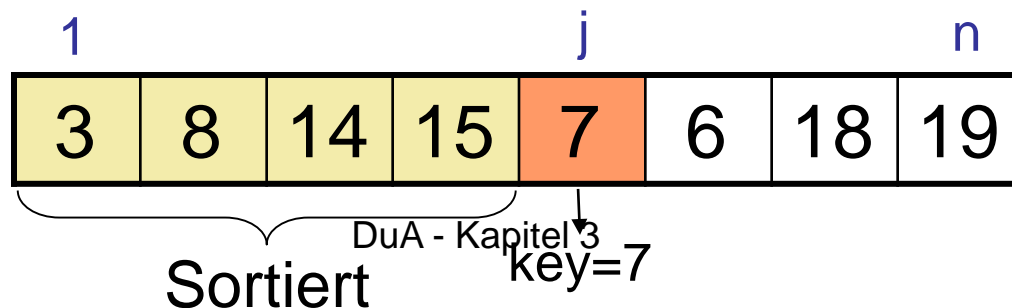
➤  $A[1, \dots, j-1]$ , die größer als

➤ key sind eine Stelle

➤ nach rechts

➤ Speichere key in „Lücke“

**Beispiel:**



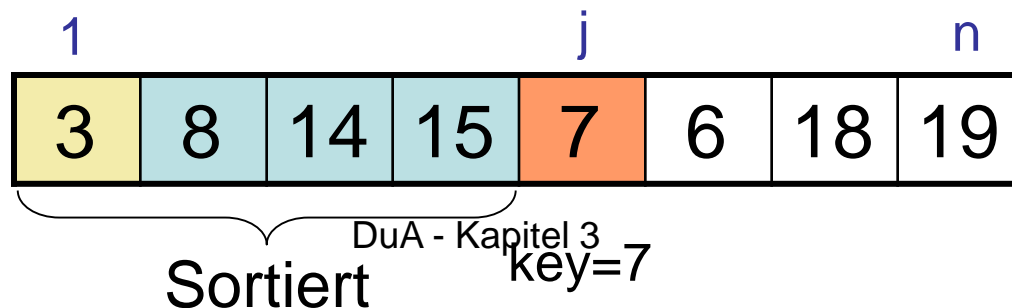
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



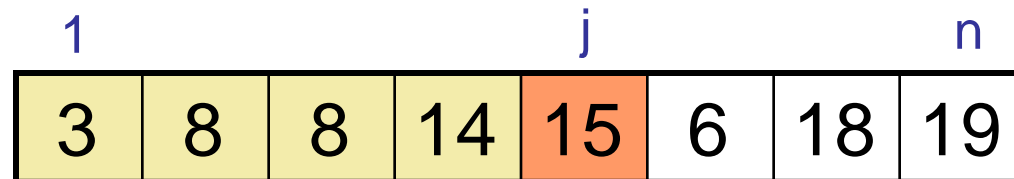
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



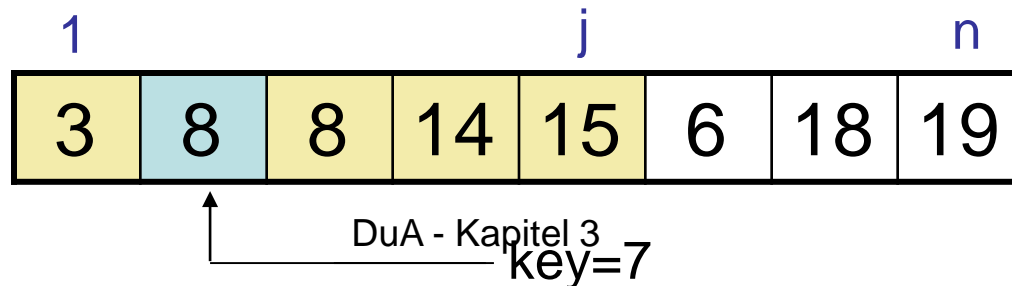
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



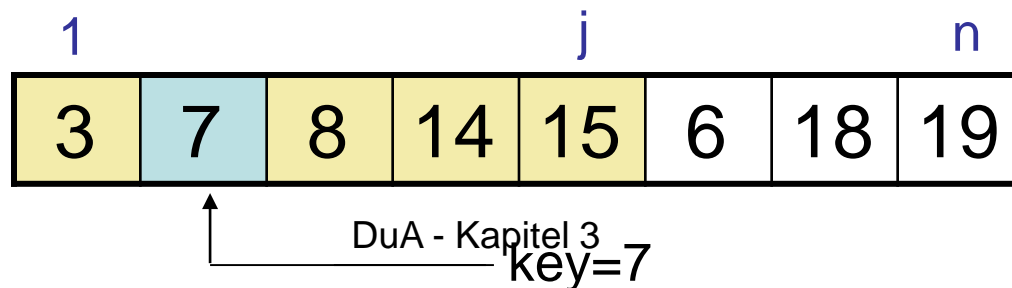
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



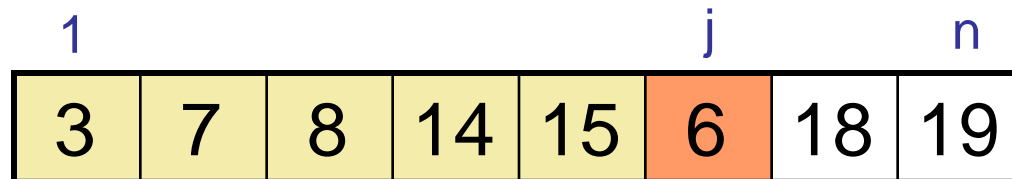
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



Sortiert

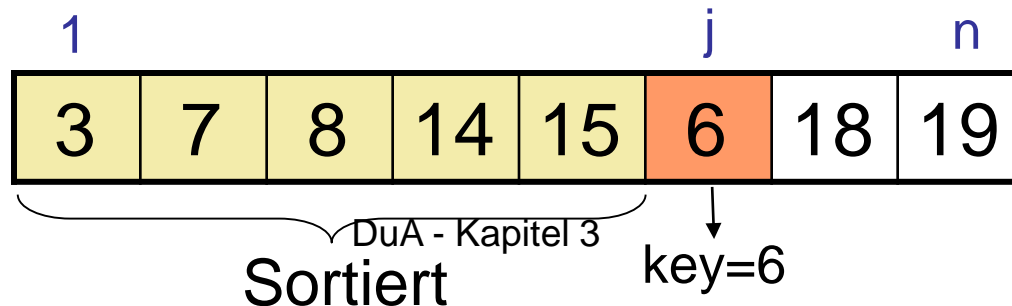
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



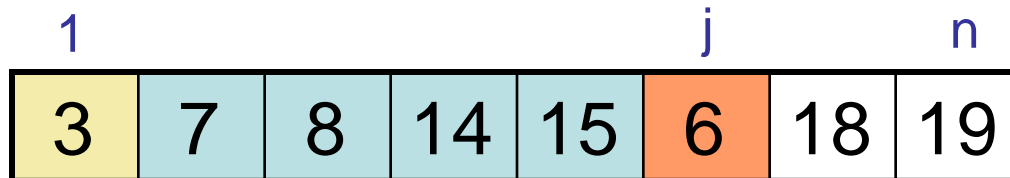
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**





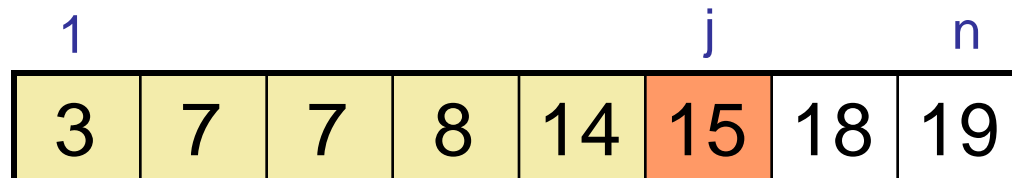
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



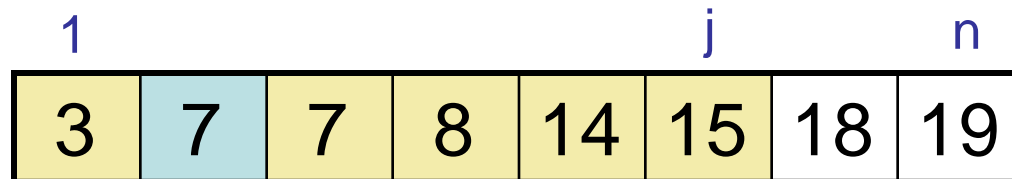
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



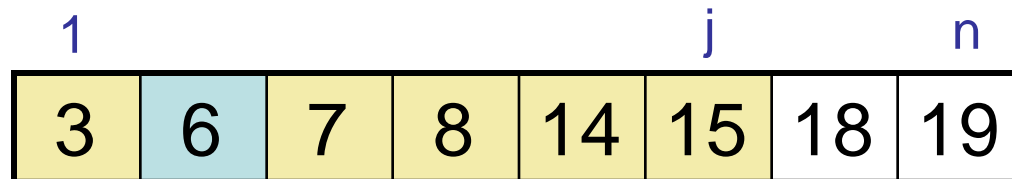
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



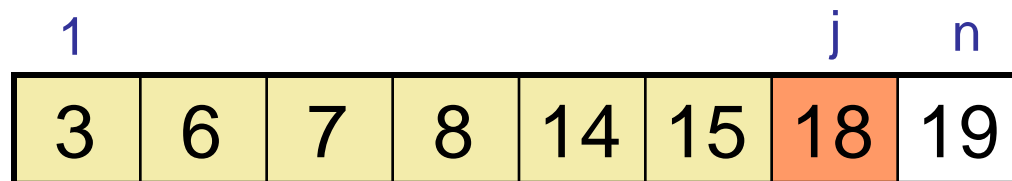
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



DuA - Kapitel 3

Sortiert

# 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
```

➤ Eingabegröße n

➤ (length(A)=n)

➤ verschiebe alle

➤ A[1,...,j-1], die größer als

➤ key sind eine Stelle

➤ nach rechts

➤ Speichere key in „Lücke“

**Beispiel:**

	1					j	n
3	6	7	8	14	15	18	19

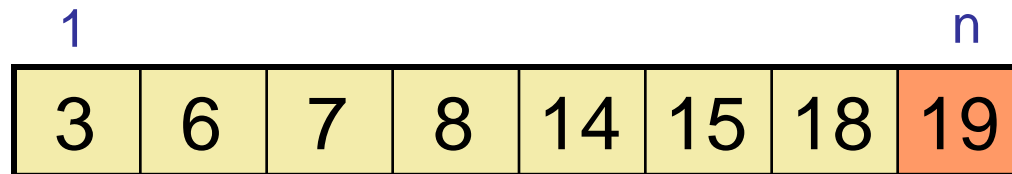
# Insertion Sort

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.      $A[i+1] \leftarrow \text{key}$

- Eingabegröße  $n$
- $(\text{length}(A)=n)$
- verschiebe alle
- $A[1, \dots, j-1]$ , die größer als
- $\text{key}$  sind eine Stelle
- nach rechts
- Speichere  $\text{key}$  in „Lücke“

**Beispiel:**



# 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
```

➤ Eingabegröße n

➤ (length(A)=n)

➤ verschiebe alle

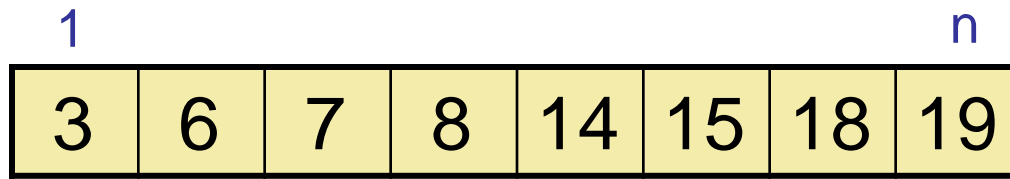
➤ A[1,..,j-1], die größer als

➤ key sind eine Stelle

➤ nach rechts

➤ Speichere key in „Lücke“

**Beispiel:**



DuA - Kapitel 3

Sortiert

# Invariante bei Insertion-Sort

---

- $A_0[i]$  : anfänglicher  $i$ -ter Wert von Array  $A$
- $A[i]$  : aktueller  $i$ -ter Wert von Array  $A$

**Invariante  $I(j)$ :**  $A[1..j-1]$  enthält die Zahlen in  $A_0[1..j-1]$  in sortierter Reihenfolge.

**Initialisierung:** anfangs gilt offensichtlich  $I(2)$  (nur das erste Element ist sortiert) und damit auch  $I(j)$  für  $j=2$ .

**Erhaltung:** wenn am Anfang der for-Schleife  $I(j)$  gilt, dann gilt am Ende der for-Schleife  $I(j+1)$ .

**Terminierung:** Am Ende gilt  $I(\text{length}(A)+1)$  d.h.  $A[1.. \text{length}(A)]$  enthält die sortierten Zahlen in  $A_0[1..\text{length}(A)]$ , woraus die Korrektheit folgt.



# Invariante bei Insertion-Sort

---

## Satz 3.2

- Insertion-Sort sortiert eine Folge von  $n$  Zahlen aufsteigend.

## Beweis:

- Wir zeigen, dass die Schleifeninvariante  $I(j)$  alle Anforderungen erfüllt.
- Die Initialisierungs- und Terminierungsanforderungen der Schleifeninvariante  $I(j)$  gelten offensichtlich (sofern die Erhaltung gilt).
- Es bleibt, die Erhaltung der Schleifeninvariante zu beweisen.

# Invariante bei Insertion-Sort

---

Erhaltung ( $j \rightarrow j+1$ ):

- Angenommen  $I(j)$  gelte am Anfang der Schleife für ein  $j$
- Insertionsort merkt sich  $A[j]$  in Variable  $key$
- Sei  $1 \leq k \leq j-1$  der kleinste Index mit  $A[k] > key$  oder  $k=j$ , falls ein solcher nicht existiert
- Der Algorithmus verschiebt  $A[k, \dots, j-1]$  nach  $A[k+1, \dots, j]$
- Dann wird  $A[k]$  auf den Wert  $key$  gesetzt
- Danach gilt:
  - (1)  $A[1] \leq A[2] \leq \dots \leq A[k-1]$  wegen Annahme  $I(j)$  oben
  - (2)  $A[k-1] \leq A[k] \leq A[k+1]$  nach Ablauf der Schleife
  - (3)  $A[k+1] \leq A[k+2] \leq \dots \leq A[j]$  wegen Annahme  $I(j)$  oben
- Aus (1)-(3) folgt  $I(j+1)$  und damit die Erhaltung.

Formale Analyse (nächste Seite):

# Invariante bei Insertion-Sort

Insertion-Sort(A)

```

1  ▷ I(2)
   for j ← 2 to length(A) do
2  ▷ I(j)
   key ← A[j]
   ▷ I(j) ∧ key = A0[j]
3  i ← j - 1
   ▷ ( I(j,i) : A[1..i,i+2..j] enthält sortierte Zahlen in A0[1..j-1] ) ∧ key = A0[j]
4  while i > 0 and A[i] > key do
   ▷ I(j,i) ∧ key = A0[j] ∧ key < A[i] ∧ i > 0
5     A[i+1] ← A[i]
   ▷ I(j,i-1) ∧ key = A0[j] ∧ key < A[i+1] ∧ i > 0
6     i ← i - 1
   ▷ I(j,i) ∧ key = A0[j] ∧ key < A[i+2] ∧ i ≥ 0
   ▷ Fall (a): i = 0 ⇒ I(j,0) ∧ key = A0[j] ∧ key < A[2]
   ▷ Fall (b): A[i] ≤ key ⇒ I(j,i) ∧ key = A0[j] ∧ A[i] ≤ key < A[i+2]
7  A[i+1] ← key
   ▷ I(j+1)
   ▷ I(length[A]+1)
   ⇒ A[1..length[A]] enthält die sortierten Zahlen von A0[1..length[A]]

```

Initialisierung

j=2

Erhaltung

Terminierung

d.h. Algo korrekt

# Insertion-Sort – Analyse (1)

---

InsertionSort(Array A)

Laufzeit

1. **for**  $j \leftarrow 2$  **to**  $\text{length}(A)$  **do**

$\sum_{j=2}^n T(I)$

2.      $\text{key} \leftarrow A[j]$

$O(1)$

3.      $i \leftarrow j-1$

$O(1)$

4.     **while**  $i > 0$  and  $A[i] > \text{key}$  **do**

$\leq \sum_{i=1}^{j-1} T(I)$

5.          $A[i+1] \leftarrow A[i]$

$O(1)$

6.          $i \leftarrow i-1$

$O(1)$

7.      $A[i+1] \leftarrow \text{key}$

$O(1)$

verwende  
 $\Phi(i)=i$

Laufzeit:  $O(\sum_{j=2}^n ((\sum_{i=1}^{j-1} 1) + 1)) = O(n^2)$

# Insertion-Sort – Analyse (2)

---

**Satz 3.3:** Insertion-Sort hat eine worst-case Laufzeit von  $\Theta(n^2)$ .

**Beweis:**

- $O(n^2)$ : schon gezeigt
- $\Omega(n^2)$ : worst-case Folge ist absteigend sortierte Folge, da in diesem Fall die while-Schleife  $j-1$ -mal für jedes  $j$  durchlaufen wird.