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,…,*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 $(a_1, a_2, \ldots, a_n)$.

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

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

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

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

Insertion-Sort

Idee: Sukzessive wird eine Sortierung der Teilarrays $A[1..i]$, $1 \le i \le length(A)$ berechnet.

Insertion-Sort(A)

```
1 for j ← 2 to length(A)
2     do key ← A[j]
3        ▷ Füge A[j] in die sortierte Folge A[1..j – 1] ein.
4        i ← j – 1
5        while i > 0 and A[i] > key
6              do A[i + 1] ← A[i]
7                 i ← i – 1
8     A[i + 1] ← key
```

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

Beispiel:

8	15	3	14	7	6	18	19

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)

Beispiel:

8	15	3	14	7	6	18	19

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)

Beispiel:

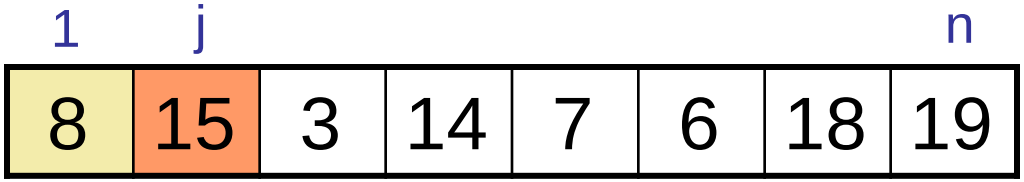
1							n
8	15	3	14	7	6	18	19

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)

Beispiel:

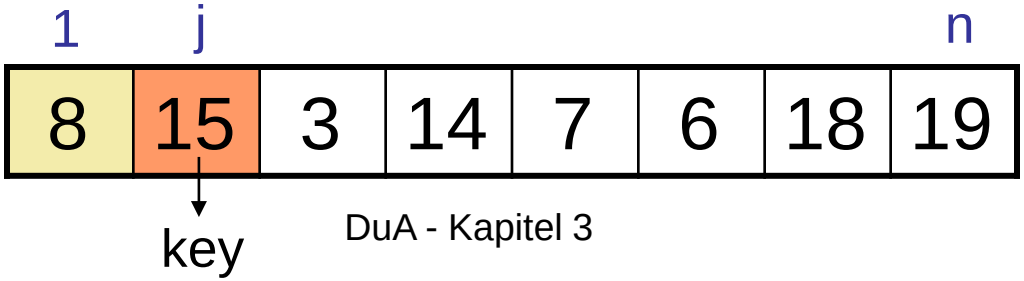


Insertion Sort

```
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)

Beispiel:

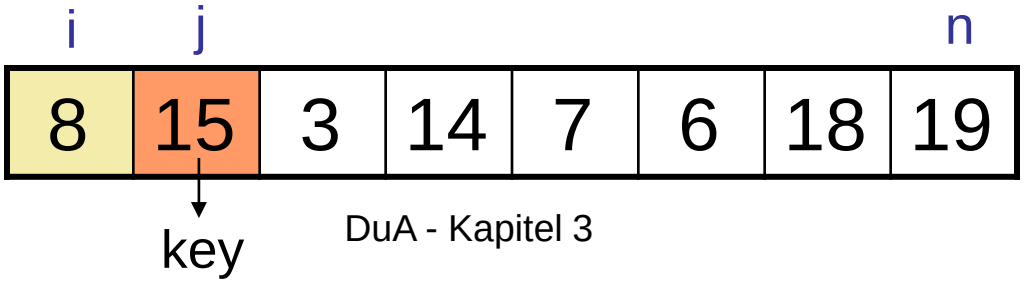


Insertion Sort

```
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)

Beispiel:

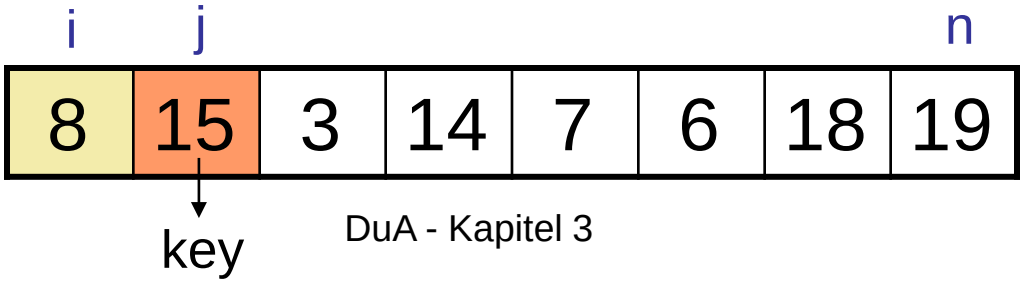


Insertion Sort

```
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)

Beispiel:

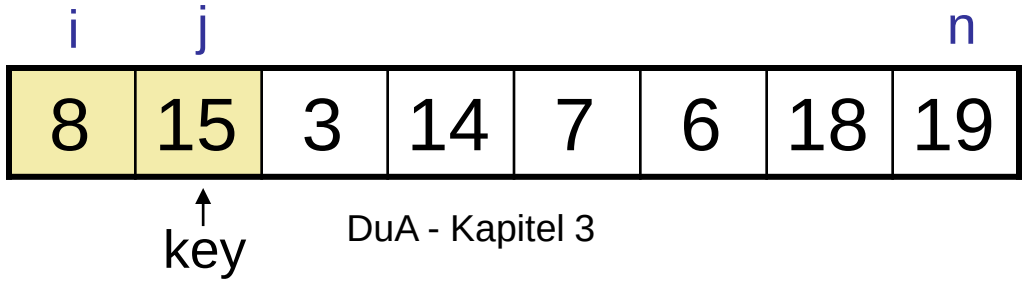


Insertion Sort

```
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)

Beispiel:

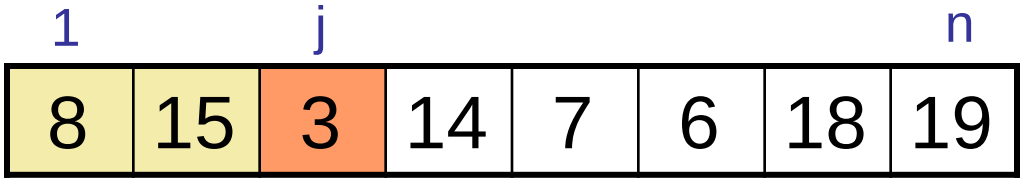


Insertion Sort

```
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)

Beispiel:

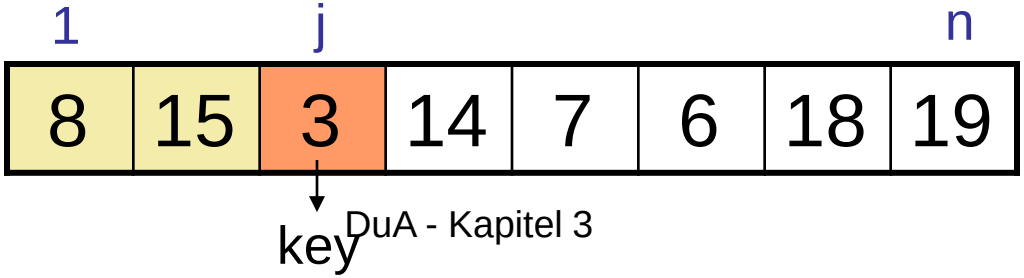


Insertion Sort

```
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)

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**

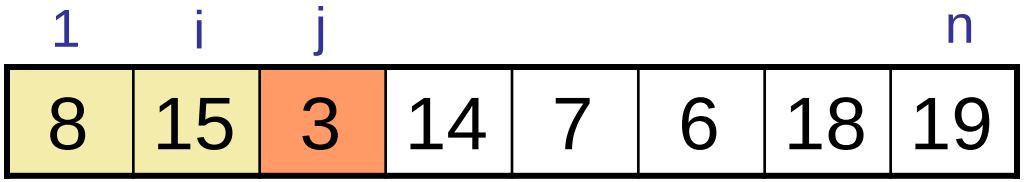
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

Beispiel:



key=3

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**

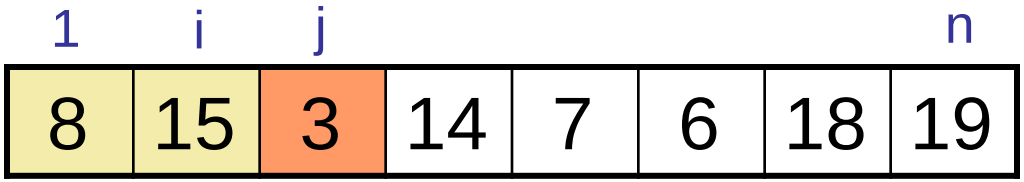
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

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**

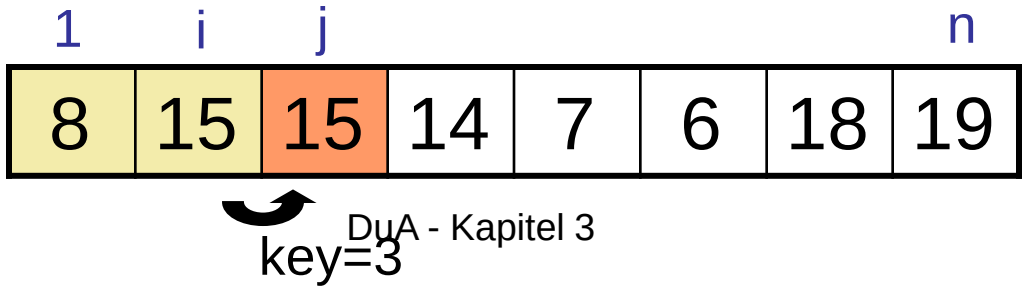
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

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**

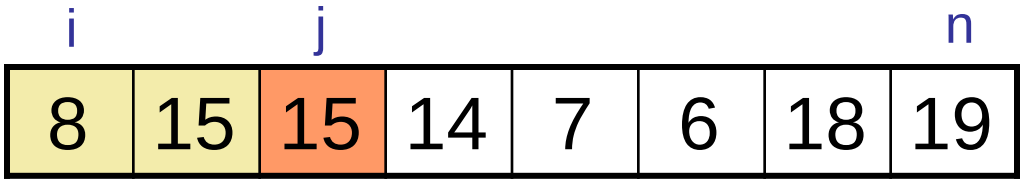
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

Beispiel:



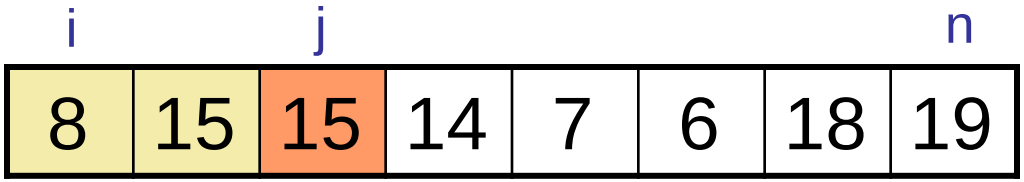
key=3

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)

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**

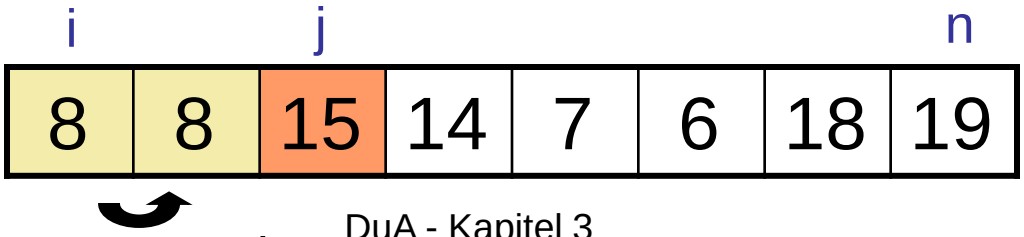
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

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**

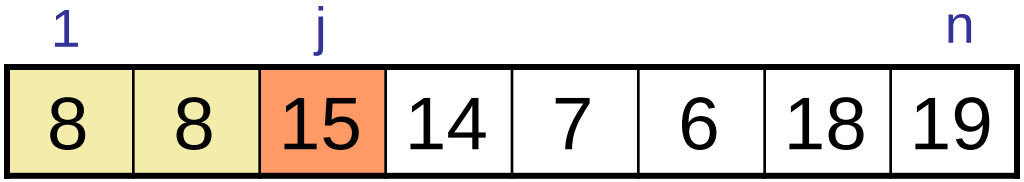
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

Beispiel:



key=3

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**

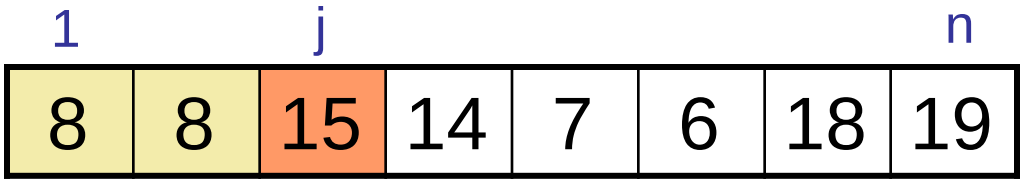
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

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**

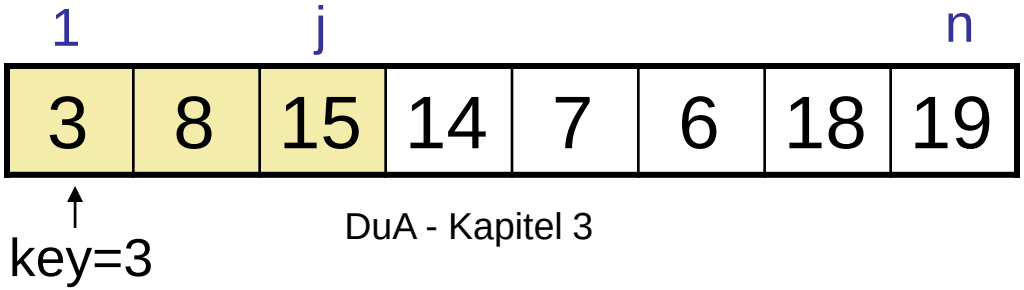
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

Beispiel:

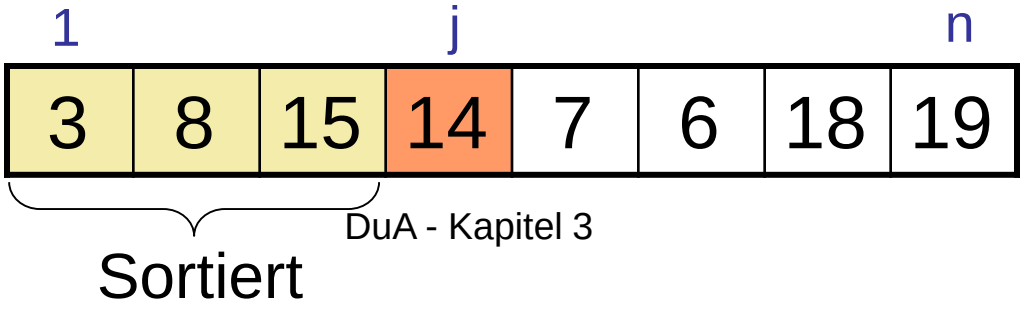


Insertion Sort

```
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)

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**

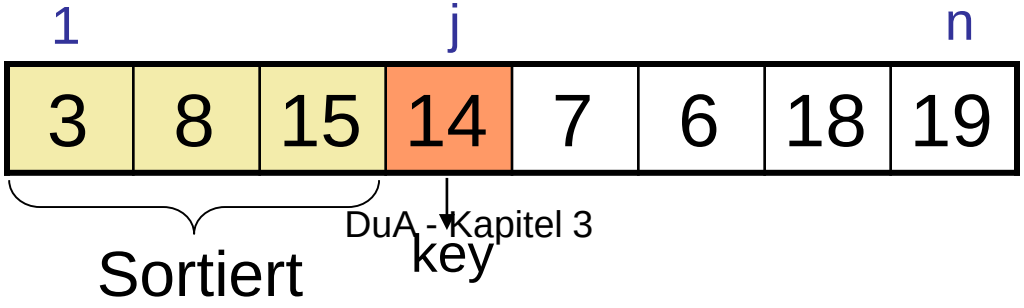
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

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**

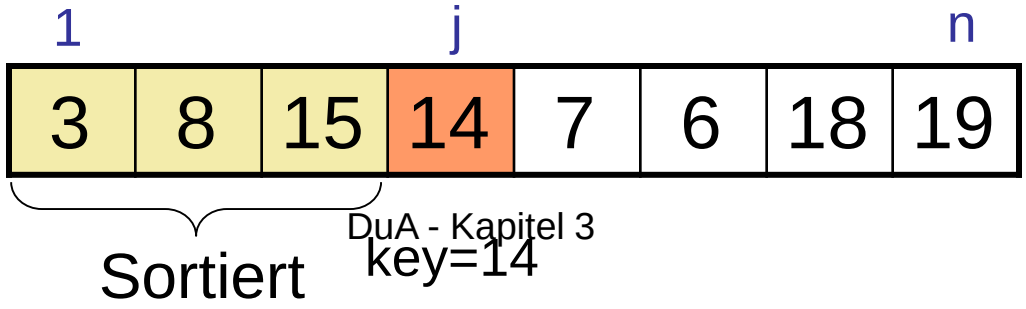
5. A[i+1] ← A[i]

6. i ← i-1

7. A[i+1] ← key

- Eingabegröße n
- (length(A)=n)

Beispiel:

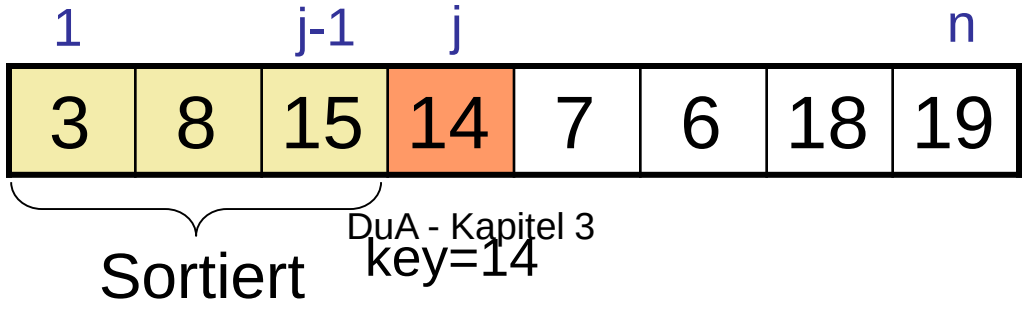


Insertion Sort

```
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

Beispiel:

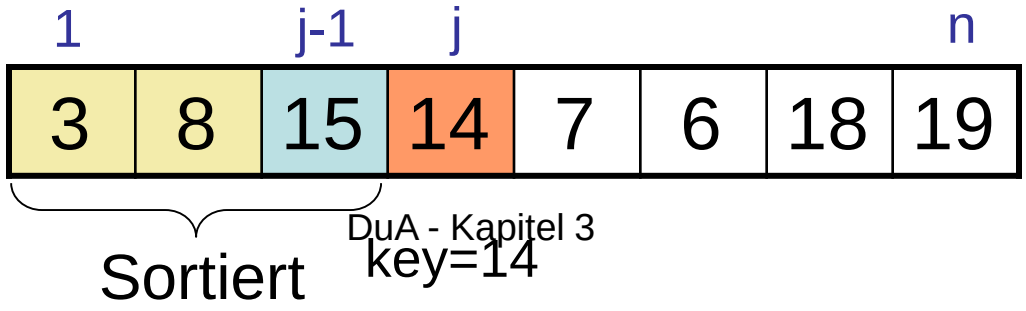


Insertion Sort

```
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

Beispiel:

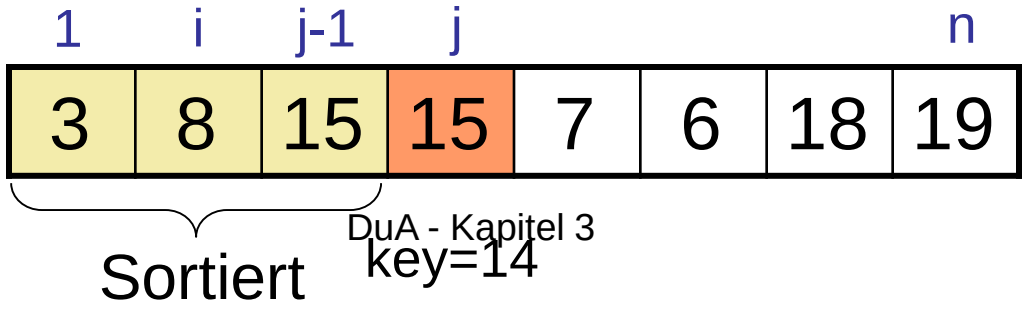


Insertion Sort

```
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

Beispiel:

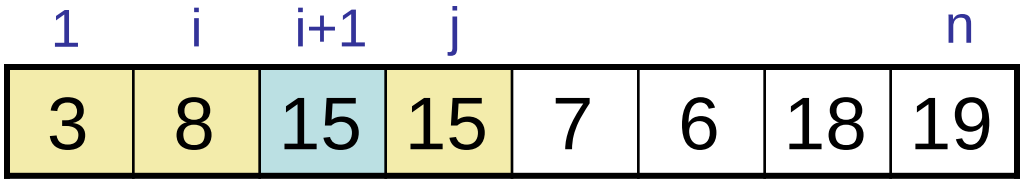


Insertion Sort

```
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

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**

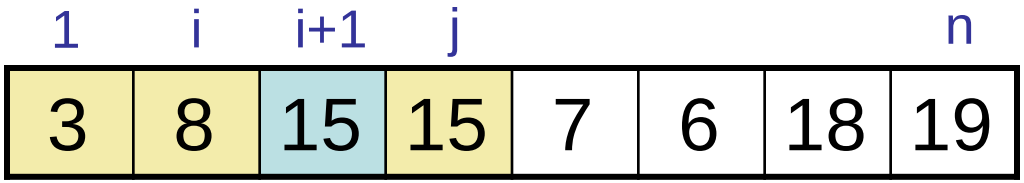
5. A[i+1] ← A[i]

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:

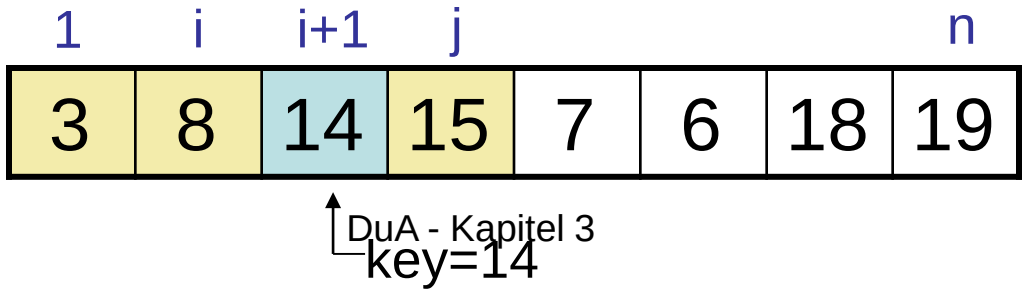


Insertion Sort

```
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:

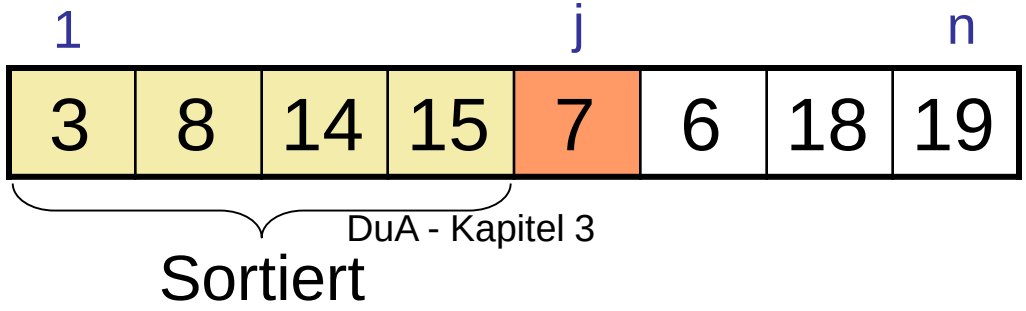


Insertion Sort

```
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:

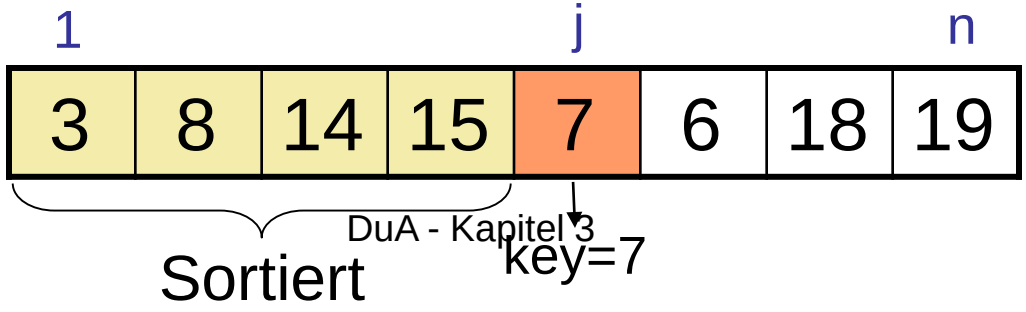


Insertion Sort

```
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:

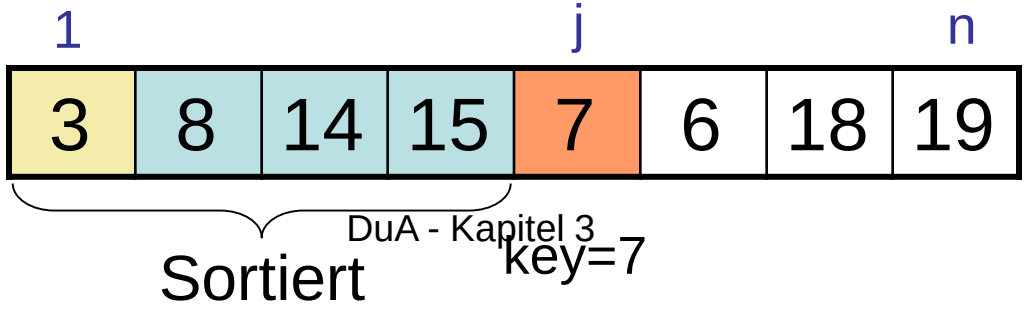


Insertion Sort

```
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:

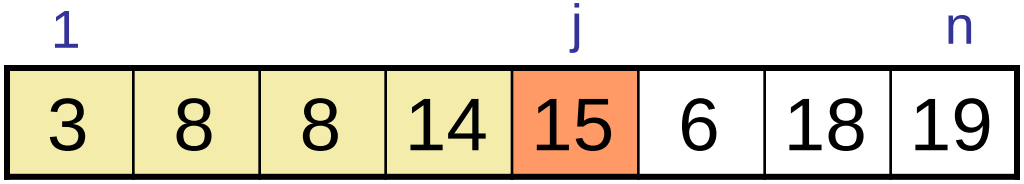


Insertion Sort

```
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:

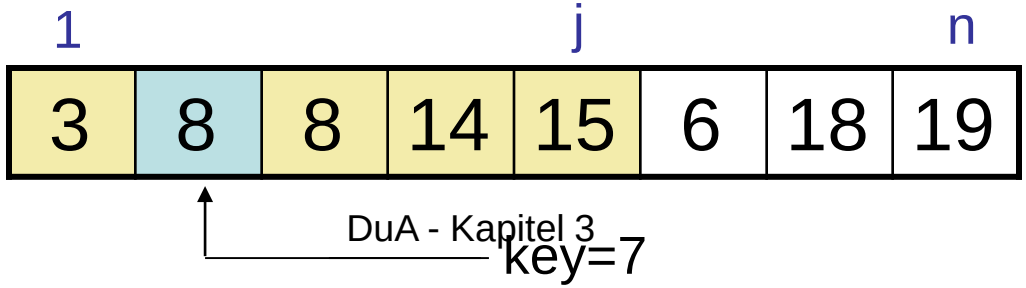


Insertion Sort

```
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:

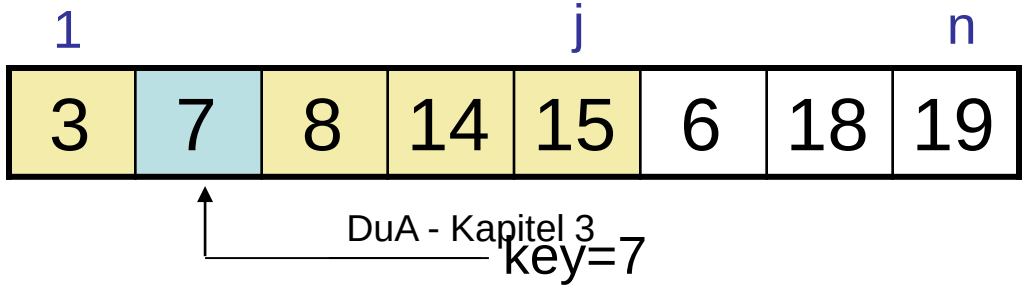


Insertion Sort

```
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:

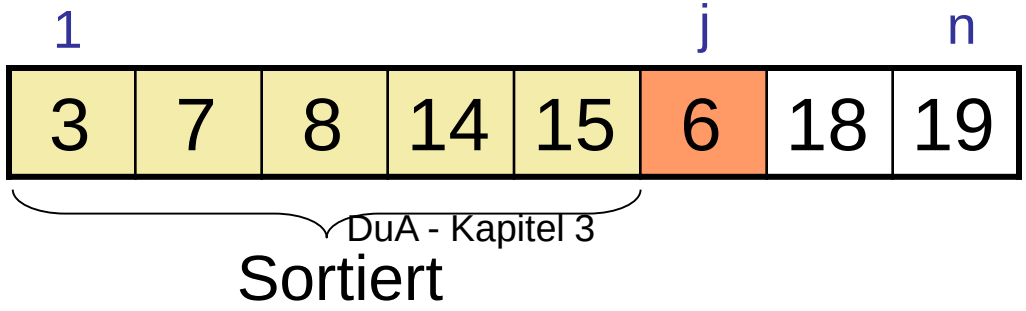


Insertion Sort

```
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:

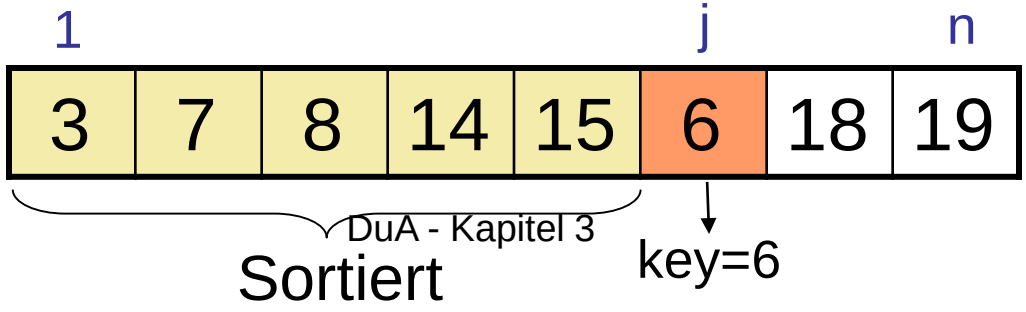


Insertion Sort

```
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:

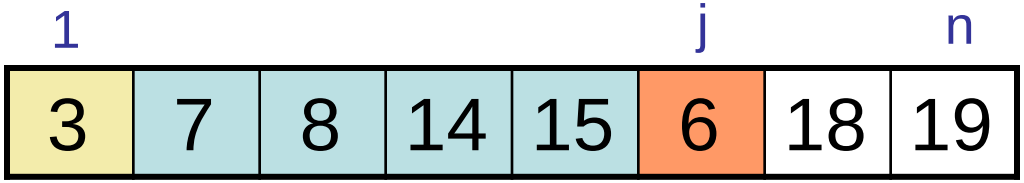


Insertion Sort

```
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:

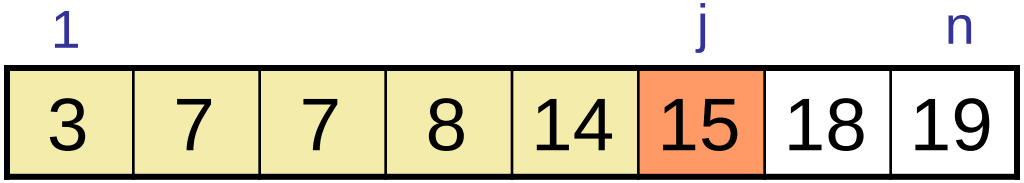


Insertion Sort

```
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:

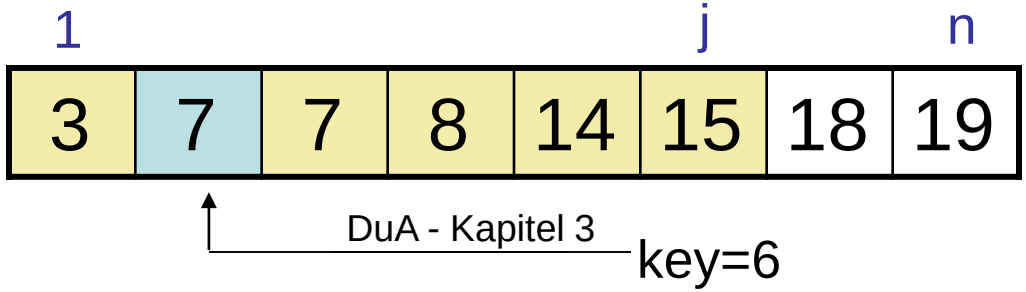


Insertion Sort

```
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:

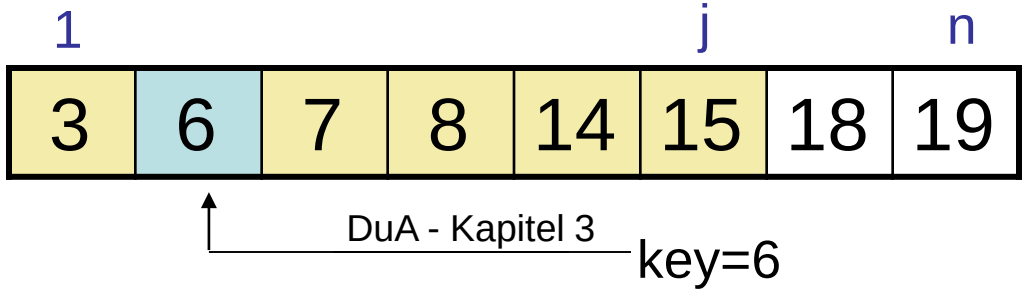


Insertion Sort

```
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:



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**

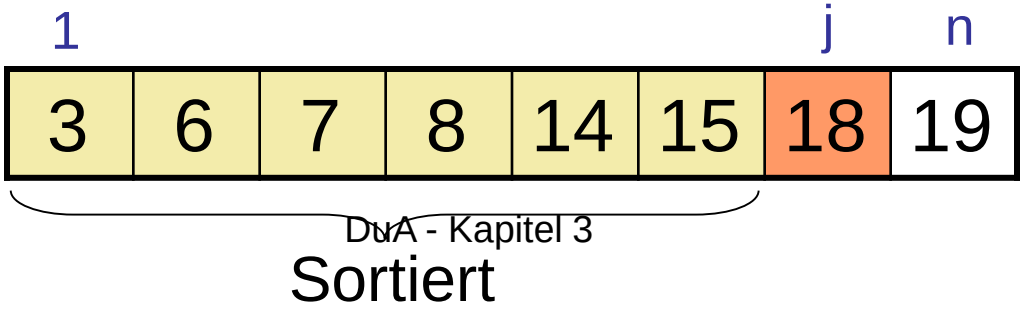
5. A[i+1] ← A[i]

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:

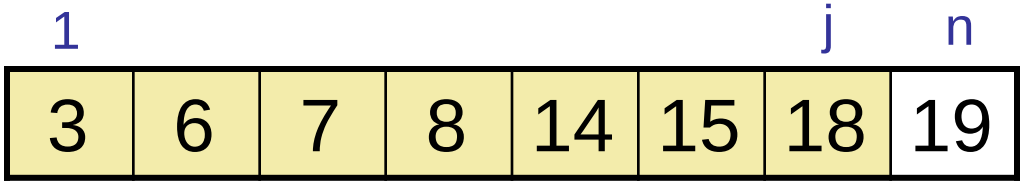


Insertion Sort

```
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:

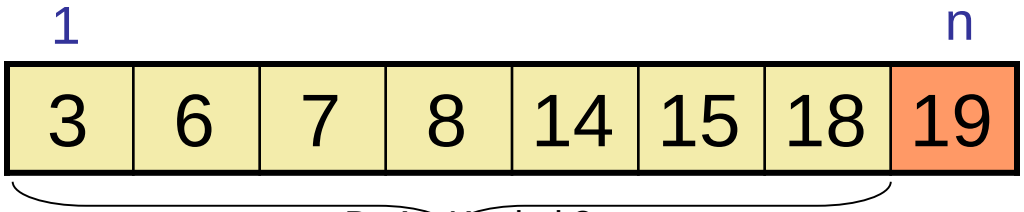


Insertion Sort

```
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:

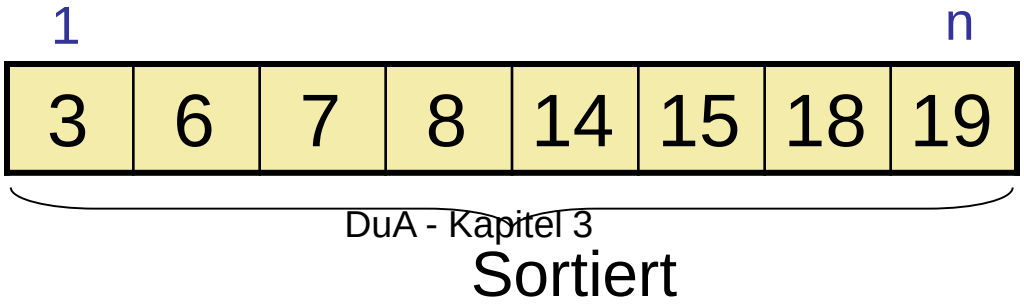


Insertion Sort

```
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:



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): $A[1..j\text{-}1]$ enthält die Zahlen in $A_0[1..j\text{-}1]$ 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(*length*(*A*)+1) d.h. $A[1. . length(A)]$ enthält die sortierten Zahlen in A_0[1..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.

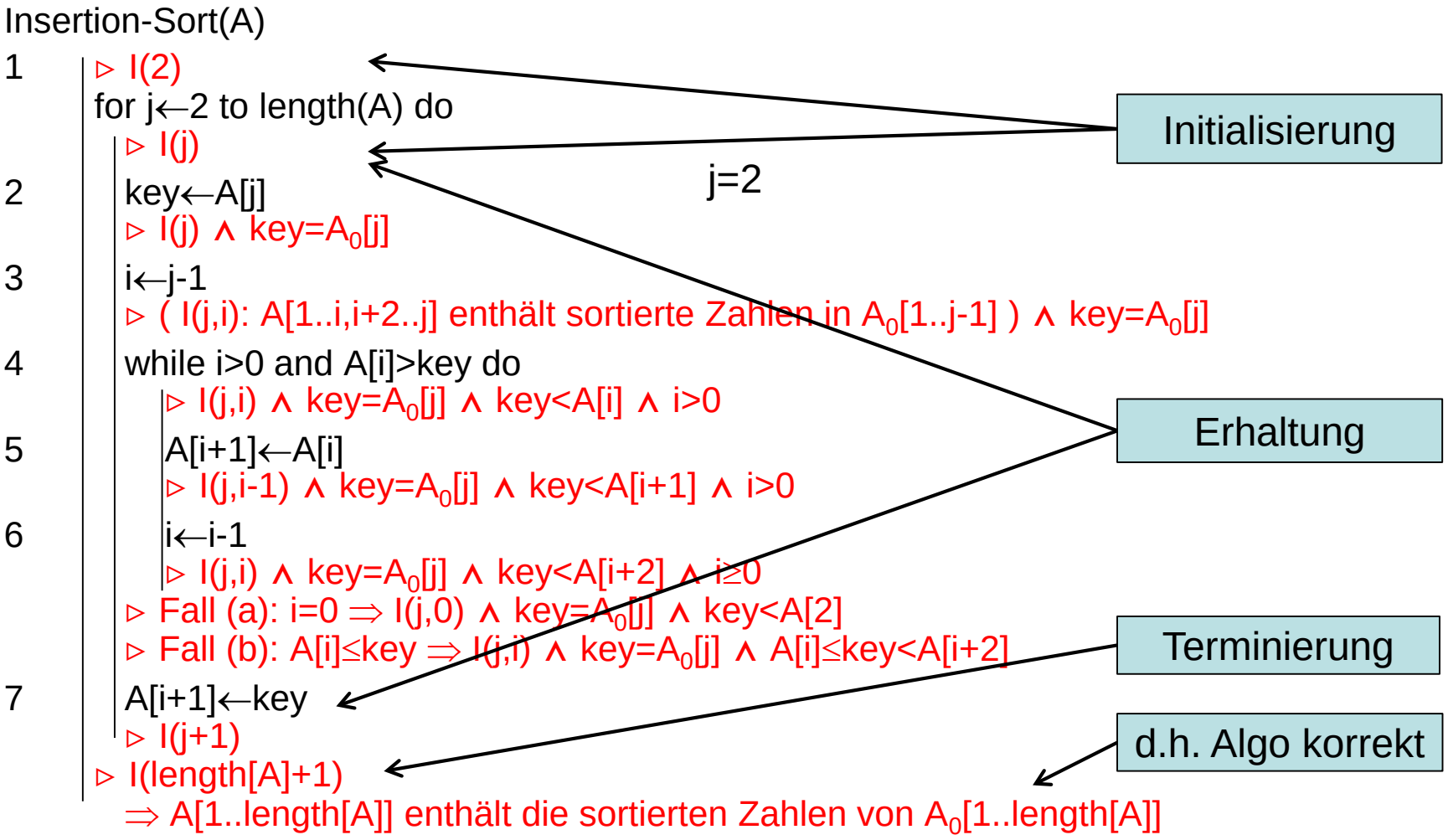
Invariante bei Insertion-Sort

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 $A[k,..,j-1]$ nach $A[k+1,..,j]$
- Dann wird $A[k]$ auf den Wert key gesetzt
- Danach gilt:
  (1) $A[1] \leq A[2] \leq \ldots \leq A[k-1]$ wegen Annahme $I(j)$ oben
  (2) $A[k-1] \leq A[k] \leq A[k+1]$ nach Ablauf der Schleife
  (3) $A[k+1] \leq A[k+2] \leq \ldots \leq A[j]$ 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 – Analyse (1)

Insertion-Sort(A)	cost	times
1 for j $\leftarrow$ 2 to length(A)	c_1	n
2 do key $\leftarrow A[j]$	c_2	n-1
3 ▷ Einfügen von $A[j]$.		
4 $i \leftarrow j-1$	c_4	n-1
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] \leftarrow key$	c_8	n-1

Hierbei ist $t_j \leq j$ die Anzahl der Durchläufe der Abfrage in Zeile 5.

Insertion-Sort – Analyse (2)

Satz 3.3. Insertion-Sort besitzt worst case Laufzeit $\Theta(n^2)$.

Für den Beweis ist zu zeigen:

1. Es gibt ein c_2, so dass die Laufzeit von Insertion-Sort bei allen Eingaben der Größe n immer höchstens $c_2 n^2$ ist.

2. Es gibt ein c_1, so dass für alle n eine Eingabe I_n der Größe n existiert bei der Insertion-Sort mindestens Laufzeit $c_1 n^2$ besitzt.