UNIVERSITÄT BONN

Algorithmen und Berechnungskomplexität I

Prof. Dr. Anne Driemel
Institut für Informatik
Abteilung V

(Folien Prof. Dr. Heiko Röglin)

1.1 Grundbegriffe

Algorithmus = Handlungsvorschrift zur Lösung eines Problems

Muss so präzise beschrieben sein, dass sie vom Computer ausgeführt werden kann.

Dabei wird eine Eingabe in eine Ausgabe transformiert.

1.1 Grundbegriffe

Algorithmus = Handlungsvorschrift zur Lösung eines Problems

Muss so präzise beschrieben sein, dass sie vom Computer ausgeführt werden kann.

Dabei wird eine Eingabe in eine Ausgabe transformiert.

Sortieren:	$\{7, 3, 5, 9, 12, 2\}$	$\longrightarrow$	$(2, 3, 5, 7, 9, 12)$
Primzahltest:	60	$\longrightarrow$	nein
Finden eines Primfaktors:	60	$\longrightarrow$	2, 3 oder 5

1.1 Grundbegriffe

Algorithmus = Handlungsvorschrift zur Lösung eines Problems

Muss so präzise beschrieben sein, dass sie vom Computer ausgeführt werden kann.

Dabei wird eine Eingabe in eine Ausgabe transformiert.

Sortieren:	$\{7, 3, 5, 9, 12, 2\}$	$\longrightarrow$	$(2, 3, 5, 7, 9, 12)$
Primzahltest:	60	$\longrightarrow$	nein
Finden eines Primfaktors:	60	$\longrightarrow$	2, 3 oder 5

Algorithmus zum Primzahltest

Bei Eingabe n: Teste alle Zahlen $2, 3, 4, \ldots, n-1$ ob sie Teiler von n sind.

Ausgabe: **Nein** falls Teiler gefunden, sonst **Ja**.

1.1 Grundbegriffe

Navigationsgerät

Was ist der kürzeste Weg von A nach B?

Online Banking, Einkaufen im Internet

Wie werden die Daten ver- und entschlüsselt?

Google

Suchmaschinen

Wie sucht man in einer großen Datenmenge?

Paketdienste, Logistikdienstleister

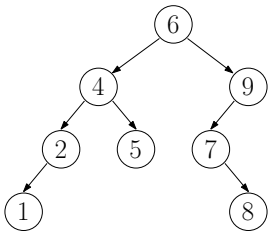
In welcher Reihenfolge werden Kunden beliefert?

1.1 Grundbegriffe

Ziel: **Korrekte** und **schnelle Algorithmen**
Wichtigstes Hilfsmittel: **Datenstrukturen**

Wie sollen Daten gespeichert werden, um schnelle Ausführung wichtiger Operationen zu ermöglichen?

Beispiel: **Suchbäume**



Lernziele

- Kenntnis grundlegender Datenstrukturen, Methoden und Algorithmen
- Fähigkeit, ein gegebenes Problem zu analysieren:
  - Einordnung der Schwierigkeit des Problems
  - Auswahl geeigneter Datenstrukturen und Algorithmen
- Entwurf von Algorithmen
- Analyse von Algorithmen

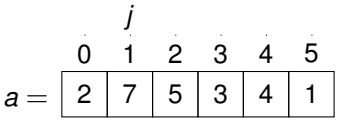
1.2 Ein erstes Beispiel: Insertionsort

INSERTIONSORT(**int**[ ] *a*)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```

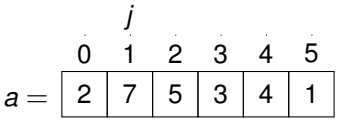
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



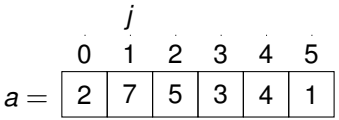
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

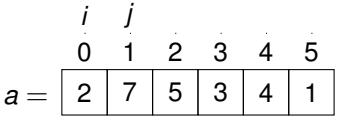
```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

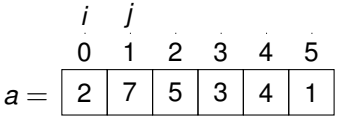
INSERTIONSORT(**int**[ ] *a*)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

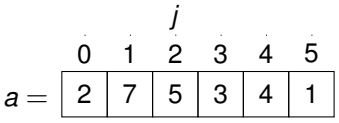
```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



$x = 7$

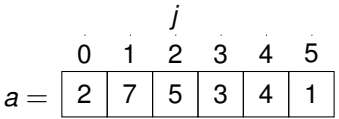
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



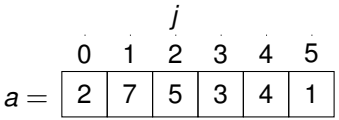
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



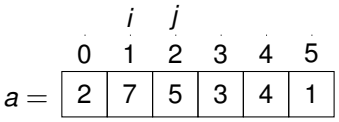
$x = 5$

1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```

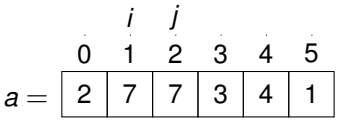
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



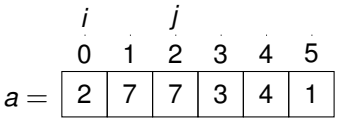
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



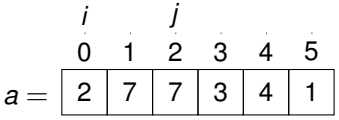
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



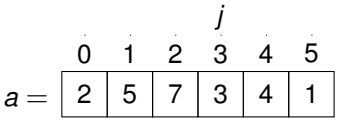
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



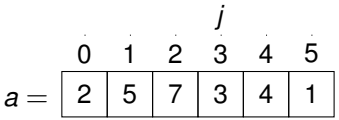
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



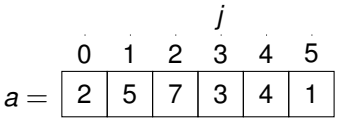
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```

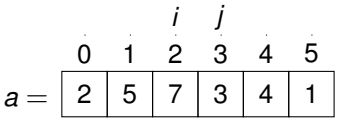


$x = 3$

1.2 Ein erstes Beispiel: Insertionsort

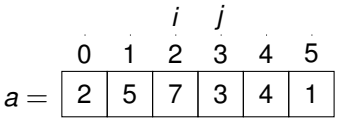
INSERTIONSORT(**int**[ ] *a*)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



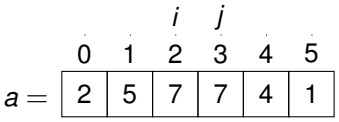
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



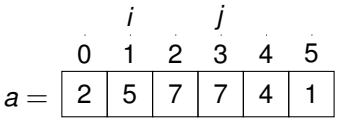
1.2 Ein erstes Beispiel: Insertionsort

```
1   INSERTIONSORT(int[] a)
2   for (int j = 1; j < a.length; j++) {
3       // Füge a[j] in die bereits sortierte
4           Sequenz a[0 ... j − 1] ein.
5       int x = a[j];
6       int i = j − 1;
7       while ((i >= 0) && (a[i] > x)) {
8           a[i + 1] = a[i];
9           i−−;
10      }
11      a[i + 1] = x;
    }
    return a;
```



1.2 Ein erstes Beispiel: Insertionsort

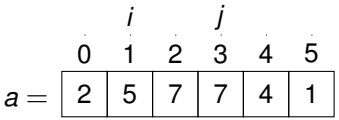
```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

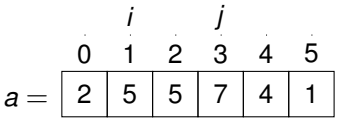
INSERTIONSORT(**int**[ ] a)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



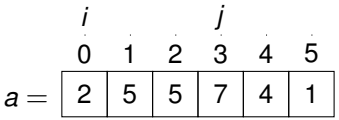
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



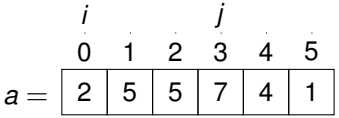
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

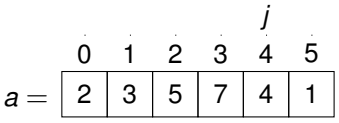
```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



1.2 Ein erstes Beispiel: Insertionsort

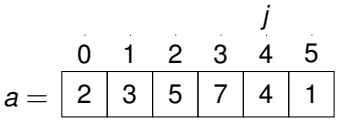
INSERTIONSORT(**int**[ ] *a*)

```
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 . . . j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



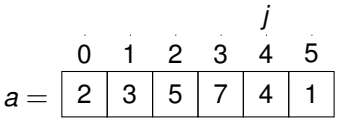
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

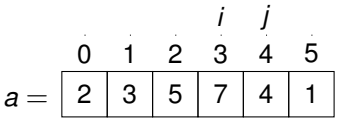
```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



$x = 4$

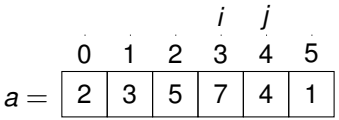
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



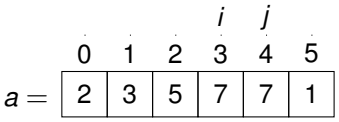
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



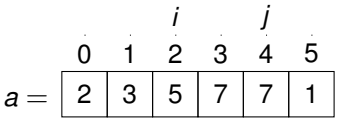
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



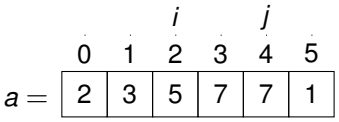
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



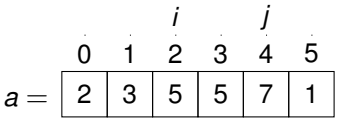
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



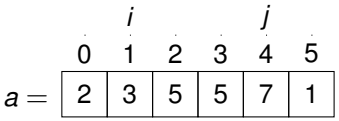
1.2 Ein erstes Beispiel: Insertionsort

```
1   INSERTIONSORT(int[] a)
2   for (int j = 1; j < a.length; j++) {
3       // Füge a[j] in die bereits sortierte
4           Sequenz a[0 ... j − 1] ein.
5       int x = a[j];
6       int i = j − 1;
7       while ((i >= 0) && (a[i] > x)) {
8           a[i + 1] = a[i];
9           i−−;
10      }
11      a[i + 1] = x;
    }
    return a;
```



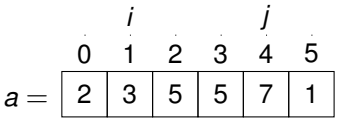
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



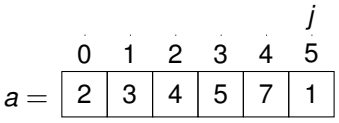
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



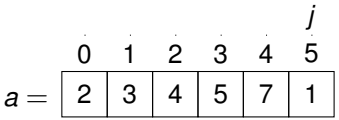
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



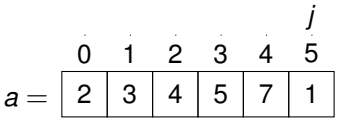
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

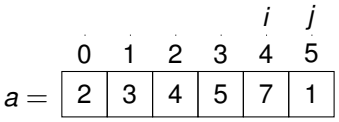
```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



$x = 1$

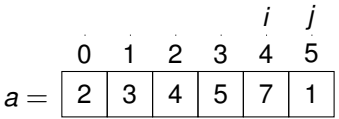
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



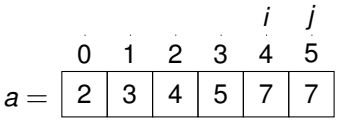
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



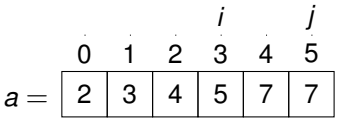
INSERTIONSORT(**int**[ ] *a*)

```
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 . . . j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



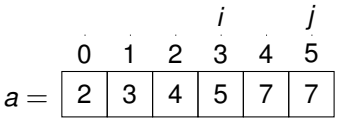
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

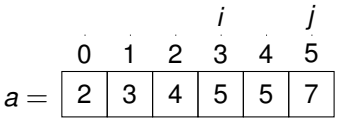
```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

INSERTIONSORT(**int**[ ] *a*)

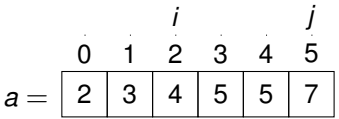
```
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 . . . j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



1.2 Ein erstes Beispiel: Insertionsort

INSERTIONSORT(**int**[ ] *a*)

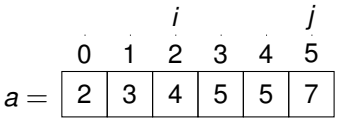
```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



$x = 1$

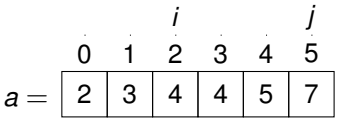
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



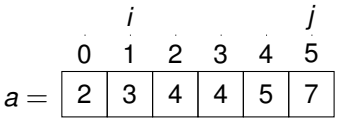
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



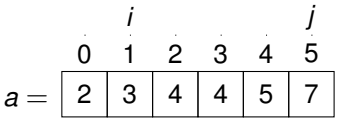
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



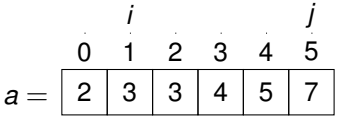
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



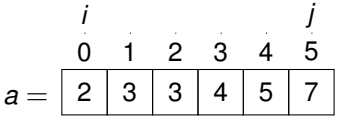
1.2 Ein erstes Beispiel: Insertionsort

```
1   INSERTIONSORT(int[] a)
2   for (int j = 1; j < a.length; j++) {
3       // Füge a[j] in die bereits sortierte
4           Sequenz a[0 ... j − 1] ein.
5       int x = a[j];
6       int i = j − 1;
7       while ((i >= 0) && (a[i] > x)) {
8           a[i + 1] = a[i];
9           i−−;
10      }
11      a[i + 1] = x;
    }
    return a;
```



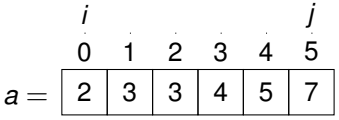
1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 ... j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
9       a[i + 1] = x;
10  }
11  return a;
```



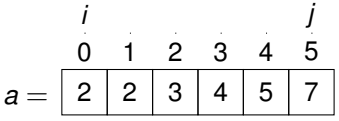
1.2 Ein erstes Beispiel: Insertionsort

```
1   INSERTIONSORT(int[] a)
2   for (int j = 1; j < a.length; j++) {
3       // Füge a[j] in die bereits sortierte
4           Sequenz a[0 ... j − 1] ein.
5       int x = a[j];
6       int i = j − 1;
7       while ((i >= 0) && (a[i] > x)) {
8           a[i + 1] = a[i];
9           i−−;
10      }
11      a[i + 1] = x;
    }
    return a;
```



1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```

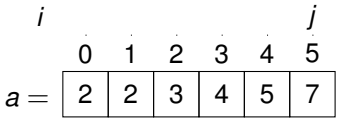


$x = 1$

1.2 Ein erstes Beispiel: Insertionsort

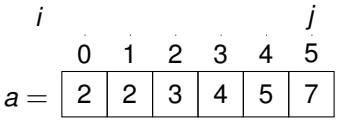
INSERTIONSORT(**int**[ ] *a*)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



1.2 Ein erstes Beispiel: Insertionsort

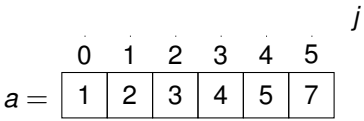
```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```



$x = 1$

INSERTIONSORT(**int**[ ] *a*)

```
1   for (int j = 1; j < a.length; j++) {
2       // Füge a[j] in die bereits sortierte
            Sequenz a[0 . . . j − 1] ein.
3       int x = a[j];
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8        }
9       a[i + 1] = x;
10   }
11  return a;
```



```
INSERTIONSORT(int[] a)
1    for (int j = 1; j < a.length; j++) {
2        // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
3        int x = a[j];
4        int i = j − 1;
5        while ((i >= 0) && (a[i] > x)) {
6            a[i + 1] = a[i];
7            i−−;
8        }
9        a[i + 1] = x;
10   }
11   return a;
```

$a =$

0	1	2	3	4	5
1	2	3	4	5	7

$x =$

Theorem 1.1

Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe.

Theorem 1.1

Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe.

Beweis: Sei $b[0 \ldots n-1]$ der initiale Inhalt von $a[0 \ldots n-1]$, wobei $n = a.\text{length}$.

Theorem 1.1

Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe.

Beweis: Sei $b[0 \ldots n-1]$ der initiale Inhalt von $a[0 \ldots n-1]$, wobei $n = a.\text{length}$.

```
INSERTIONSORT(int[] a)
 1    for (int j = 1; j < a.length; j++) {
 2        // Füge a[j] in die bereits sortierte
              Sequenz a[0 ... j − 1] ein.
 3        int x = a[j];
 4        int i = j − 1;
 5        while ((i >= 0) && (a[i] > x)) {
 6            a[i + 1] = a[i];
 7            i−−;
 8        }
 9        a[i + 1] = x;
10    }
11    return a;
```

Definition:

Für Zahlenfolge c sei $\text{sort}(c)$ aufsteigende Permutation von c.

1.2 Ein erstes Beispiel: Insertionsort

Theorem 1.1

Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe.

Beweis: Sei $b[0 \ldots n-1]$ der initiale Inhalt von $a[0 \ldots n-1]$, wobei $n = a.\text{length}$.

```
INSERTIONSORT(int[] a)
 1    for (int j = 1; j < a.length; j++) {
 2        // Füge a[j] in die bereits sortierte
              Sequenz a[0 ... j − 1] ein.
 3        int x = a[j];
 4        int i = j − 1;
 5        while ((i >= 0) && (a[i] > x)) {
 6            a[i + 1] = a[i];
 7            i−−;
 8        }
 9        a[i + 1] = x;
10    }
11    return a;
```

Definition:

Für Zahlenfolge c sei $\text{sort}(c)$ aufsteigende Permutation von c.

Invariante in Zeile 1:

$a[0 \ldots j-1] = \text{sort}(b[0 \ldots j-1])$,
$a[j \ldots n-1] = b[j \ldots n-1]$

Theorem 1.1

Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe.

Beweis: Sei $b[0 \ldots n-1]$ der initiale Inhalt von $a[0 \ldots n-1]$, wobei $n = a.\text{length}$.

```
INSERTIONSORT(int[] a)
 1   for (int j = 1; j < a.length; j++) {
 2       // Füge a[j] in die bereits sortierte
             Sequenz a[0 ... j − 1] ein.
 3       int x = a[j];
 4       int i = j − 1;
 5       while ((i >= 0) && (a[i] > x)) {
 6           a[i + 1] = a[i];
 7           i−−;
 8       }
 9       a[i + 1] = x;
10   }
11   return a;
```

Definition:

Für Zahlenfolge c sei $\text{sort}(c)$ aufsteigende Permutation von c.

Invariante in Zeile 1:

$a[0 \ldots j-1] = \text{sort}(b[0 \ldots j-1])$,
$a[j \ldots n-1] = b[j \ldots n-1]$

Induktionsanfang $j = 1$:

$a[0] = b[0] = \text{sort}(b[0])$
$a[1 \ldots n-1] = b[1 \ldots n-1]$

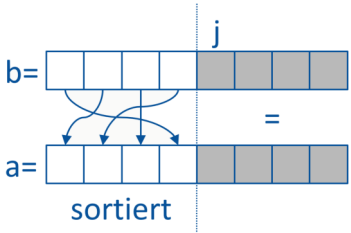
Invariante in Zeile 1:

$a[0 \ldots j-1] = \mathrm{sort}(b[0 \ldots j-1])$,

$a[j \ldots n-1] = b[j \ldots n-1]$

Induktionsschritt

Angenommen die Invariante gilt für ein $j \geq 1$:



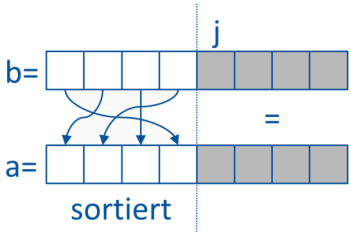
Invariante in Zeile 1:

$a[0 \ldots j-1] = \mathrm{sort}(b[0 \ldots j-1])$,

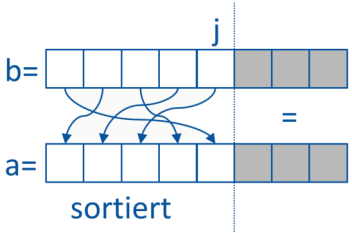
$a[j \ldots n-1] = b[j \ldots n-1]$

Induktionsschritt

Angenommen die Invariante gilt für ein $j \geq 1$:



Im betrachteten Durchlauf der for-Schleife wird $b[j]$ an der richtigen Stelle einsortiert.



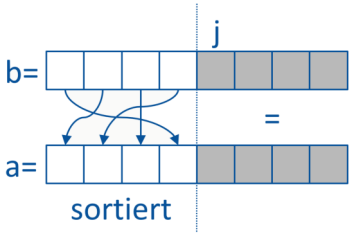
Invariante in Zeile 1:

$a[0 \ldots j-1] = \mathrm{sort}(b[0 \ldots j-1])$,

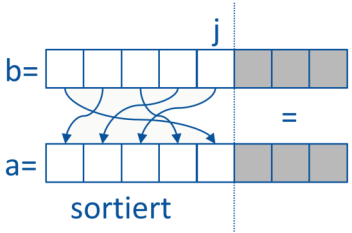
$a[j \ldots n-1] = b[j \ldots n-1]$

Induktionsschritt

Angenommen die Invariante gilt für ein $j \geq 1$:



Im betrachteten Durchlauf der for-Schleife wird $b[j]$ an der richtigen Stelle einsortiert.



Danach gilt

$a[0 \ldots j] = \mathrm{sort}(b[0 \ldots j])$,

$a[j+1 \ldots n-1] = b[j+1 \ldots n-1]$.

$\Rightarrow$ Die Invariante gilt auch für $j+1$.

Invariante in Zeile 1:

$a[0 \ldots j-1] = \mathrm{sort}(b[0 \ldots j-1]),$

$a[j \ldots n-1] = b[j \ldots n-1]$

Für $j = n$ ergibt sich:

$a[0 \ldots n-1] = \mathrm{sort}(b[0 \ldots n-1]).$

Invariante in Zeile 1:

$a[0 \ldots j-1] = \mathrm{sort}(b[0 \ldots j-1])$,

$a[j \ldots n-1] = b[j \ldots n-1]$

Für $j = n$ ergibt sich:

$a[0 \ldots n-1] = \mathrm{sort}(b[0 \ldots n-1])$.

$\Rightarrow$ Die Ausgabe von INSERTIONSORT ist eine aufsteigend sortierte Permutation der Eingabe. $\square$

```
INSERTIONSORT(int[] a)
 1  for (int j = 1; j < a.length; j++) {
 3      int x = a[j];
 4      int i = j − 1;
 5      while ((i >= 0) && (a[i] > x)) {
 6          a[i + 1] = a[i];
 7          i−−;
 8      }
 9      a[i + 1] = x;
10  }
11  return a;
```

1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)                                  # Ausführungen
 1   for (int j = 1; j < a.length; j++) {                  n
 3       int x = a[j];
 4       int i = j − 1;
 5       while ((i >= 0) && (a[i] > x)) {
 6           a[i + 1] = a[i];
 7           i−−;
 8       }
 9       a[i + 1] = x;
10   }
11   return a;
```

1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)                          # Ausführungen
 1  for (int j = 1; j < a.length; j++) {        n
 3      int x = a[j];                            n − 1
 4      int i = j − 1;
 5      while ((i >= 0) && (a[i] > x)) {
 6          a[i + 1] = a[i];
 7          i−−;
 8      }
 9      a[i + 1] = x;
10  }
11  return a;
```

1.2 Ein erstes Beispiel: Insertionsort

```
INSERTIONSORT(int[] a)                          # Ausführungen
 1  for (int j = 1; j < a.length; j++) {        n
 3      int x = a[j];                            n − 1
 4      int i = j − 1;                           n − 1
 5      while ((i >= 0) && (a[i] > x)) {
 6          a[i + 1] = a[i];
 7          i−−;
 8      }
 9      a[i + 1] = x;
10  }
11  return a;
```

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) {	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j-1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	
7	i−−;	
8	}	
9	$a[i+1] = x$;	
10	}	
11	**return** a;	

$t_j =$ **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) {	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j-1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1}(t_j - 1)$
7	i––;	
8	}	
9	$a[i+1] = x$;	
10	}	
11	**return** a;	

$t_j =$ **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) {	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j-1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1}(t_j - 1)$
7	i−−;	$\sum_{j=1}^{n-1}(t_j - 1)$
8	}	
9	$a[i+1] = x$;	
10	}	
11	**return** a;	

$t_j =$ **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) {	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j-1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1}(t_j - 1)$
7	i−−;	$\sum_{j=1}^{n-1}(t_j - 1)$
8	}	
9	$a[i+1] = x$;	$n-1$
10	}	
11	**return** a;	

$t_j =$ **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) $\{$	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j - 1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) $\{$	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1} (t_j - 1)$
7	i−−;	$\sum_{j=1}^{n-1} (t_j - 1)$
8	$\}$	
9	$a[i+1] = x$;	$n-1$
10	$\}$	
11	**return** a;	1

$t_j =$ **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	**# Ausführungen**
1	**for** (**int** $j = 1; j < a$.length; j++) {	n
3	**int** $x = a[j]$;	$n-1$
4	**int** $i = j-1$;	$n-1$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1}(t_j - 1)$
7	i−−;	$\sum_{j=1}^{n-1}(t_j - 1)$
8	}	
9	$a[i+1] = x$;	$n-1$
10	}	
11	**return** a;	1

t_j = **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

c_i = **Anzahl Prozessorbefehle, um Zeile i auszuführen**

1.2 Ein erstes Beispiel: Insertionsort

	INSERTIONSORT(**int**[ ] a)	# Ausführungen	# Prozessorbefehle
1	**for** (**int** $j = 1; j < a$.length; j++) {	n	$c_1 n$
3	**int** $x = a[j]$;	$n-1$	$c_3(n-1)$
4	**int** $i = j-1$;	$n-1$	$c_4(n-1)$
5	**while** (($i >= 0$) && ($a[i] > x$)) {	$\sum_{j=1}^{n-1} t_j$	$c_5 \sum_{j=1}^{n-1} t_j$
6	$a[i+1] = a[i]$;	$\sum_{j=1}^{n-1}(t_j - 1)$	$c_6 \sum_{j=1}^{n-1}(t_j - 1)$
7	i--;	$\sum_{j=1}^{n-1}(t_j - 1)$	$c_7 \sum_{j=1}^{n-1}(t_j - 1)$
8	}		
9	$a[i+1] = x$;	$n-1$	$c_9(n-1)$
10	}		
11	**return** a;	1	c_{11}

t_j = **Anzahl Ausführungen von Zeile 5** in j-ter Iteration der for-Schleife

c_i = **Anzahl Prozessorbefehle, um Zeile i auszuführen**

Gesamtzahl der Prozessorbefehle:

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} t_j \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} (t_j - 1) \right) + c_{11}$$

Gesamtzahl der Prozessorbefehle:

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} t_j \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} (t_j - 1) \right) + c_{11}$$

Best Case $t_j = 1$:

$$T = c_1 n + (c_3 + c_4 + c_5 + c_9)(n-1) + c_{11}$$

Gesamtzahl der Prozessorbefehle:

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} t_j \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} (t_j - 1) \right) + c_{11}$$

Best Case $t_j = 1$:

$$T = c_1 n + (c_3 + c_4 + c_5 + c_9)(n-1) + c_{11} \leq an + b$$

für geeignete Konstanten $a, b \in \mathbb{R}$

Gesamtzahl der Prozessorbefehle:

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} t_j \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} (t_j - 1) \right) + c_{11}$$

Best Case $t_j = 1$:

$$T = c_1 n + (c_3 + c_4 + c_5 + c_9)(n-1) + c_{11} \leq an + b$$

für geeignete Konstanten $a, b \in \mathbb{R}$

$\Rightarrow$ Im „Best Case“ wächst die Laufzeit von INSERTIONSORT **linear mit** n.

Gesamtzahl der Prozessorbefehle:

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} t_j \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} (t_j - 1) \right) + c_{11}$$

Best Case $t_j = 1$:

$$T = c_1 n + (c_3 + c_4 + c_5 + c_9)(n-1) + c_{11} \leq an + b$$

für geeignete Konstanten $a, b \in \mathbb{R}$

$\Rightarrow$ Im „Best Case“ wächst die Laufzeit von INSERTIONSORT **linear mit** n.

Aber: Der Best Case tritt nur ein, wenn die **Eingabe bereits sortiert ist**.

INSERTIONSORT(**int**[ ] a)

```
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
```

Worst Case: $t_j = j + 1$.

INSERTIONSORT(**int**[ ] *a*)

```
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
```

Worst Case: $t_j = j + 1$.

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5 \left(\sum_{j=1}^{n-1} (j+1) \right) + (c_6 + c_7) \left(\sum_{j=1}^{n-1} j \right) + c_{11}$$

INSERTIONSORT(**int**[ ] *a*)

```
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
```

Worst Case: $t_j = j + 1$.

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5\left(\sum_{j=1}^{n-1}(j+1)\right) + (c_6 + c_7)\left(\sum_{j=1}^{n-1} j\right) + c_{11}$$

$$\leq c_5 \cdot \frac{n(n+1)}{2} + (c_6 + c_7) \cdot \frac{(n-1)n}{2} + (c_1 + c_3 + c_4 + c_9)n + c_{11}$$

INSERTIONSORT(**int**[ ] *a*)

```
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
```

Worst Case: $t_j = j + 1$.

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5\left(\sum_{j=1}^{n-1}(j+1)\right) + (c_6 + c_7)\left(\sum_{j=1}^{n-1} j\right) + c_{11}$$

$$\leq c_5 \cdot \frac{n(n+1)}{2} + (c_6 + c_7) \cdot \frac{(n-1)n}{2} + (c_1 + c_3 + c_4 + c_9)n + c_{11}$$

$$\leq an^2 + bn + c \quad \text{für geeignete Konstanten } a, b, c \in \mathbb{R}$$

INSERTIONSORT(**int**[ ] *a*)

```
4       int i = j − 1;
5       while ((i >= 0) && (a[i] > x)) {
6           a[i + 1] = a[i];
7           i−−;
8       }
```

Worst Case: $t_j = j + 1$.

$$T = c_1 n + (c_3 + c_4 + c_9)(n-1) + c_5\left(\sum_{j=1}^{n-1}(j+1)\right) + (c_6 + c_7)\left(\sum_{j=1}^{n-1} j\right) + c_{11}$$

$$\leq c_5 \cdot \frac{n(n+1)}{2} + (c_6 + c_7) \cdot \frac{(n-1)n}{2} + (c_1 + c_3 + c_4 + c_9)n + c_{11}$$

$$\leq an^2 + bn + c \quad \text{für geeignete Konstanten } a, b, c \in \mathbb{R}$$

$\Rightarrow$ Im „Worst Case“ wächst die Laufzeit von INSERTIONSORT **quadratisch mit** n.

1.3 Registermaschinen

Wir benötigen als Grundlage ein **formales Rechnermodell**.

1.3 Registermaschinen

Wir benötigen als Grundlage ein **formales Rechnermodell**.

Registermaschinen

Speicher: Register $c(0), c(1), c(2), \ldots \in \mathbb{N}$

Operationen: Arithmetische Operationen, Sprungbefehle

entspricht **rudimentärer Assemblersprache**

Laufzeit: eine Zeiteinheit pro Operation

1.3 Registermaschinen

Wir benötigen als Grundlage ein **formales Rechnermodell**.

Registermaschinen

Speicher: Register $c(0), c(1), c(2), \ldots \in \mathbb{N}$

Operationen: Arithmetische Operationen, Sprungbefehle

entspricht **rudimentärer Assemblersprache**

Laufzeit: eine Zeiteinheit pro Operation

Abstraktes Modell eines realen Rechners

Oft ein gutes Modell, aber nicht immer

Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(a) Wir sagen f **wächst nicht schneller** *als* g und schreiben $f = O(g)$, wenn

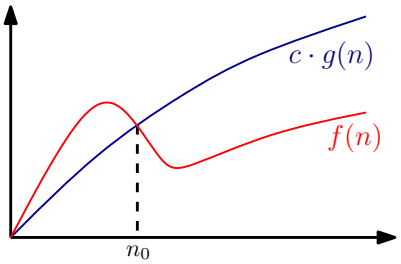
$$\exists c \in \mathbb{R} : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \colon f(n) \leq c \cdot g(n).$$

Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(a) Wir sagen f **wächst nicht schneller** *als* g und schreiben $f = O(g)$, wenn

$$\exists c \in \mathbb{R} : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \colon f(n) \leq c \cdot g(n).$$

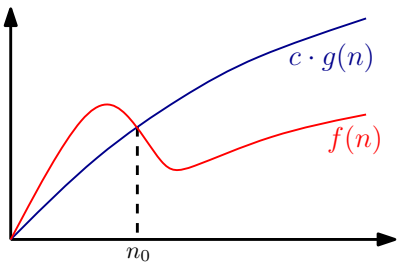


Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(a) Wir sagen f **wächst nicht schneller** *als* g und schreiben $f = O(g)$, wenn

$$\exists c \in \mathbb{R} : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 \colon f(n) \leq c \cdot g(n).$$

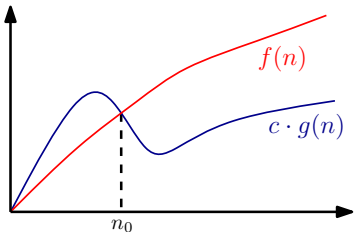


Worst-Case-Laufzeit von INSERTIONSORT: $an^2 + bn + c \leq (a + b + c)n^2 = O(n^2)$

Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(b) Wir sagen f **wächst mindestens so schnell** *wie* g und schreiben $f = \Omega(g)$, wenn $g = O(f)$ gilt.

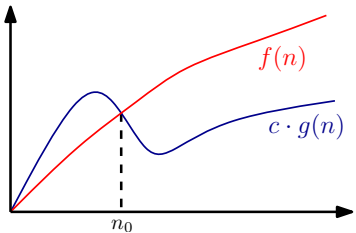


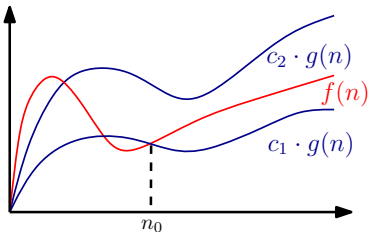
Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(b) Wir sagen f **wächst mindestens so schnell** *wie* g und schreiben $f = \Omega(g)$, wenn $g = O(f)$ gilt.

(c) Wir sagen f *und* g **sind in der gleichen Größenordnung** und schreiben $f = \Theta(g)$, wenn $f = O(g)$ und $f = \Omega(g)$ gilt.



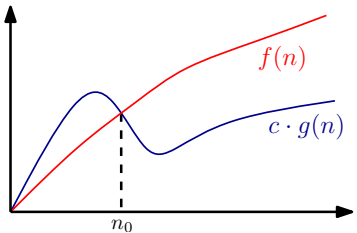


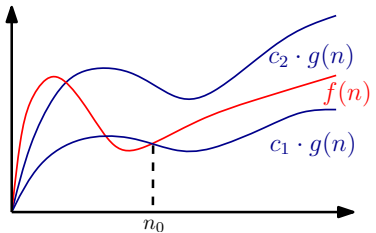
Definition 1.2

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(b) Wir sagen f **wächst mindestens so schnell** *wie* g und schreiben $f = \Omega(g)$, wenn $g = O(f)$ gilt.

(c) Wir sagen f *und* g **sind in der gleichen Größenordnung** und schreiben $f = \Theta(g)$, wenn $f = O(g)$ und $f = \Omega(g)$ gilt.





Worst-Case-Laufzeit von INSERTIONSORT: $\Theta(n^2)$

O-Notation hilft uns, von **unwesentlichen Konstanten zu abstrahieren**.

Vorsicht: O-Notation ist **nicht immer das richtige Abstraktionsniveau**.

- Eine Laufzeit von $10^{100}n = \Theta(n)$ ist praktisch nicht besser als $2n^2 = \Theta(n^2)$.
- O-Notation verliert an Aussagekraft, wenn n_0 sehr groß gewählt werden muss.
- In Anwendungen versucht man oft, auch die konstanten Faktoren zu optimieren.

Beispiele

- Für jedes **Polynom** $p(n) = \sum_{i=0}^{d} c_i n^i$ mit $c_d > 0$ gilt $p = \Theta(n^d)$.

Beispiele

- Für jedes **Polynom** $p(n) = \sum_{i=0}^{d} c_i n^i$ mit $c_d > 0$ gilt $p = \Theta(n^d)$.
- Für $a > 1$ und $b > 1$ gilt $\log_a(n) = \Theta(\log_b(n))$ wegen $\log_a(n) = \log_b(n) / \log_b(a)$.

Beispiele

- Für jedes **Polynom** $p(n) = \sum_{i=0}^{d} c_i n^i$ mit $c_d > 0$ gilt $p = \Theta(n^d)$.
- Für $a > 1$ und $b > 1$ gilt $\log_a(n) = \Theta(\log_b(n))$ wegen $\log_a(n) = \log_b(n) / \log_b(a)$.
- Für $a > 0$ und $b > 1$ gilt $\log_b(n^a) = \Theta(\log n)$ wegen $\log_b(n^a) = a \log_b n$.

Beispiele

- Für jedes **Polynom** $p(n) = \sum_{i=0}^{d} c_i n^i$ mit $c_d > 0$ gilt $p = \Theta(n^d)$.
- Für $a > 1$ und $b > 1$ gilt $\log_a(n) = \Theta(\log_b(n))$ wegen $\log_a(n) = \log_b(n) / \log_b(a)$.
- Für $a > 0$ und $b > 1$ gilt $\log_b(n^a) = \Theta(\log n)$ wegen $\log_b(n^a) = a \log_b n$.
- Gilt $f = O(1)$, so ist f für alle n durch **eine Konstante nach oben beschränkt**.

Beispiele

- Für jedes **Polynom** $p(n) = \sum_{i=0}^{d} c_i n^i$ mit $c_d > 0$ gilt $p = \Theta(n^d)$.
- Für $a > 1$ und $b > 1$ gilt $\log_a(n) = \Theta(\log_b(n))$ wegen $\log_a(n) = \log_b(n) / \log_b(a)$.
- Für $a > 0$ und $b > 1$ gilt $\log_b(n^a) = \Theta(\log n)$ wegen $\log_b(n^a) = a \log_b n$.
- Gilt $f = O(1)$, so ist f für alle n durch **eine Konstante nach oben beschränkt**.

$f : \mathbb{N} \to \mathbb{R}$ heißt **asymptotisch positiv**, wenn

$$\exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) > 0.$$

Wir erweitern Definition 1.2 auf asymptotisch positive Funktionen.

Definition 1.3

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(a) Wir sagen f **wächst langsamer** *als* g und schreiben $f = o(g)$, wenn

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Definition 1.3

Es seien $f, g\colon \mathbb{N} \to \mathbb{R}_{>0}$ zwei Funktionen.

(a) Wir sagen f **wächst langsamer** *als* g und schreiben $f = o(g)$, wenn

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

(b) Wir sagen f **wächst schneller** *als* g und schreiben $f = \omega(g)$, wenn $g = o(f)$ gilt.

Beispiele

- Es seien $p(n) = \sum_{i=0}^{d_1} c_i n^i$ und $q(n) = \sum_{i=0}^{d_2} c_i n^i$ mit $c_{d_1}, c_{d_2} > 0$ **Polynome**.
  - Für $d_1 < d_2$ gilt $p = o(q)$.
  - Für $d_1 > d_2$ gilt $p = \omega(q)$.
  - Für $d_1 = d_2$ gilt $p = \Theta(q)$.

Beispiele

- Es seien $p(n) = \sum_{i=0}^{d_1} c_i n^i$ und $q(n) = \sum_{i=0}^{d_2} c_i n^i$ mit $c_{d_1}, c_{d_2} > 0$ **Polynome**.
  - Für $d_1 < d_2$ gilt $p = o(q)$.
  - Für $d_1 > d_2$ gilt $p = \omega(q)$.
  - Für $d_1 = d_2$ gilt $p = \Theta(q)$.
- Für $k > 0$ und $\varepsilon > 0$ gilt $(\log_2(n))^k = o(n^\varepsilon)$.

Beispiele

- Es seien $p(n) = \sum_{i=0}^{d_1} c_i n^i$ und $q(n) = \sum_{i=0}^{d_2} c_i n^i$ mit $c_{d_1}, c_{d_2} > 0$ **Polynome**.
  - Für $d_1 < d_2$ gilt $p = o(q)$.
  - Für $d_1 > d_2$ gilt $p = \omega(q)$.
  - Für $d_1 = d_2$ gilt $p = \Theta(q)$.
- Für $k > 0$ und $\varepsilon > 0$ gilt $(\log_2(n))^k = o(n^\varepsilon)$.
- Für $k > 0$ und $a > 1$ gilt $n^k = o(a^n)$.

Beispiele

- Es seien $p(n) = \sum_{i=0}^{d_1} c_i n^i$ und $q(n) = \sum_{i=0}^{d_2} c_i n^i$ mit $c_{d_1}, c_{d_2} > 0$ **Polynome**.
  - Für $d_1 < d_2$ gilt $p = o(q)$.
  - Für $d_1 > d_2$ gilt $p = \omega(q)$.
  - Für $d_1 = d_2$ gilt $p = \Theta(q)$.
- Für $k > 0$ und $\varepsilon > 0$ gilt $(\log_2(n))^k = o(n^\varepsilon)$.
- Für $k > 0$ und $a > 1$ gilt $n^k = o(a^n)$.
- Für $a > 1$ und $b > a$ gilt $a^n = o(b^n)$.