

# Algorithmen und Datenstrukturen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Prof. Stefan Roth, SS 2025

---

01

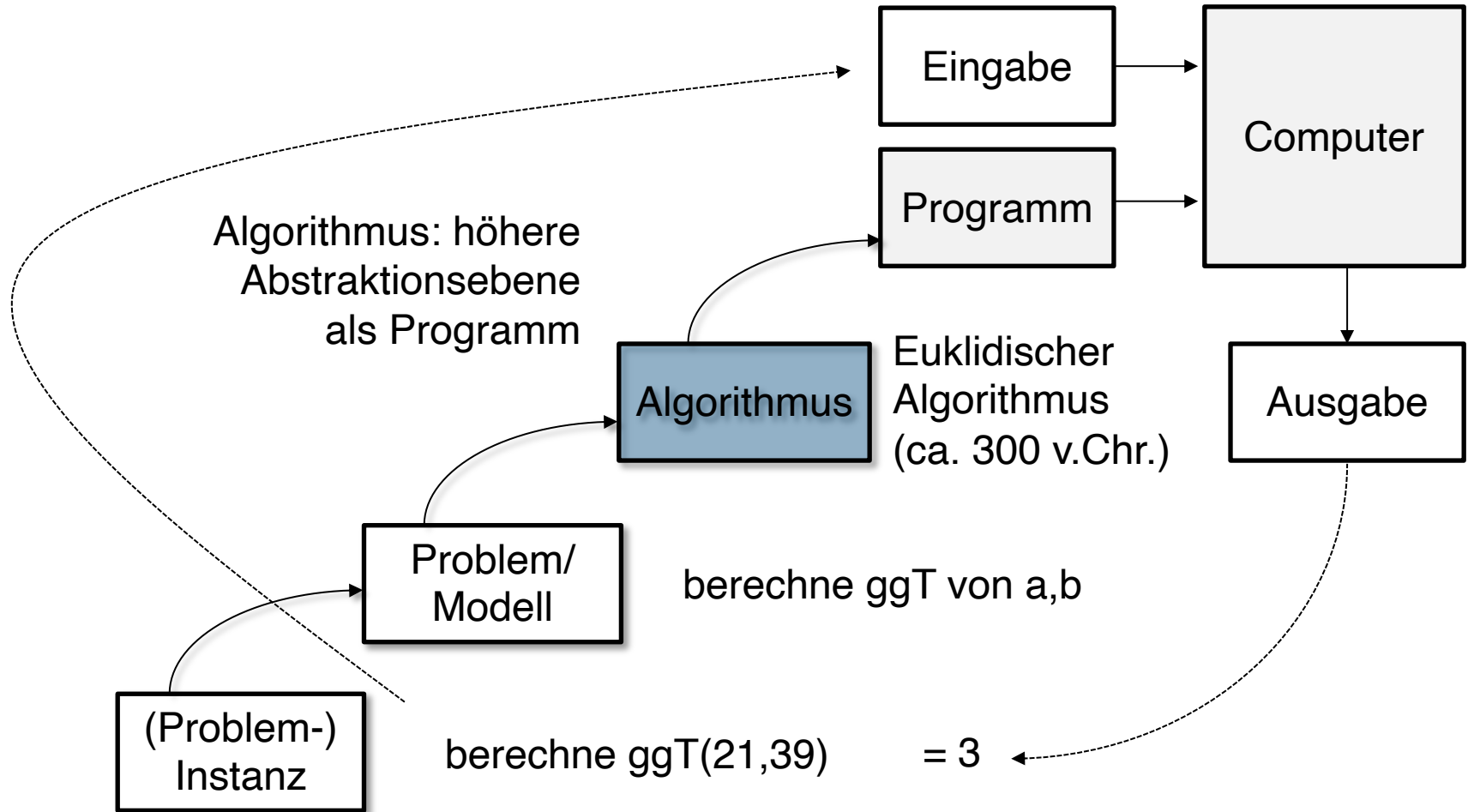
Einleitung

Folien beruhen auf der Veranstaltung von Prof. Marc Fischlin und Christian Janson aus dem SS 2024

---

# Algorithmen

# Verortung



# Algorithmus

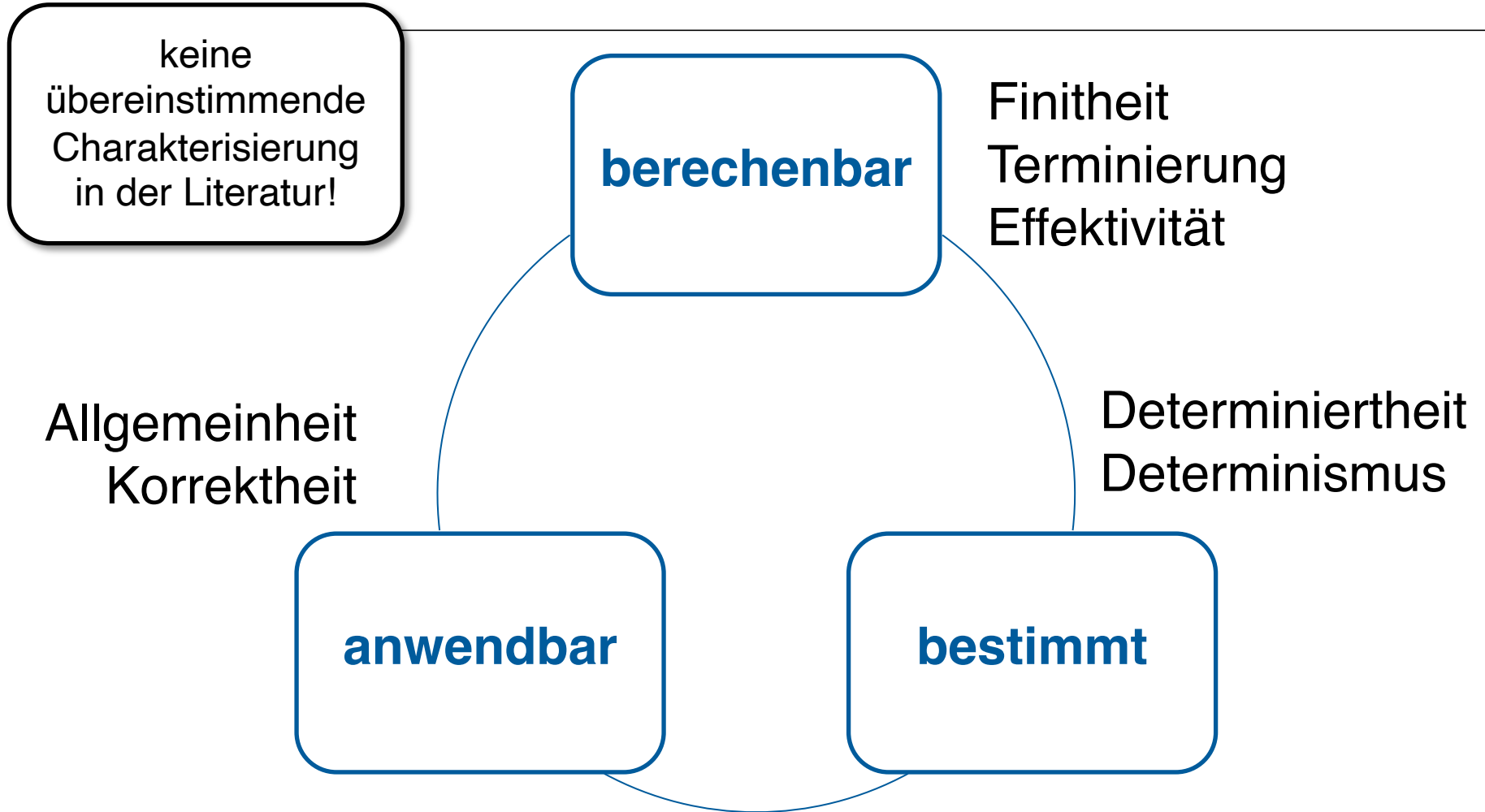
## Algorithmus:

Eine aus endlich vielen Schritten bestehende, ausführbare Handlungsvorschrift zur eindeutigen Umwandlung von Eingabe- in Ausgabedaten

nach Cormen et al., Introduction to Algorithms

Namensgeber: Ibn Musa Al-Chwarismis (ca. 825):  
Buch: Regeln der Wiedereinsetzung und Reduktion,  
ins Lateinische als „Al-Chwarismis Buch“ übersetzt

# Allgemeine Charakteristika Algorithmen (I)



# Allgemeine Charakteristika Algorithmen (II)

**berechenbar**

Finitheit  
Terminierung  
Effektivität

**Finitheit:** Algorithmus hat endliche Beschreibung  
**Terminierung:** Algorithmus stoppt in endlicher Zeit  
**Effektivität:** Schritte sind auf Maschine ausführbar

# Allgemeine Charakteristika Algorithmen (III)

## **Determiniertheit:**

Algorithmus liefert bei gleicher Eingabe  
gleiche Ausgabe

## **Determinismus:**

Algorithmus durchläuft für gleiche Eingabe  
immer die gleichen Schritte/Zustände

anwendbar

bestimmt

# Allgemeine Charakteristika Algorithmen (IV)

**Allgemeinheit:**  
Algorithmus für  
ganze Problemklasse anwendbar

**Korrektheit:**  
Falls Algorithmus terminiert, ist die  
Ausgabe richtig

Allgemeinheit  
Korrektheit

**anwendbar**

**bestimmt**



# Beispiel: Euklidischer Algorithmus (I)

```
gcd(a,b) // a,b ≥ 0 integers

1  IF b==0 THEN
2    return a
3  ELSE //a mod b Divisionsrest
4    return gcd(b,a mod b)
```

**Finitheit:** Algorithmus hat endliche Beschreibung ✓

**Terminierung:** Algorithmus stoppt in endlicher Zeit

**Effektivität:** Schritte sind auf Maschine ausführbar ✓

# Beispiel: Euklidischer Algorithmus (II)

```
gcd(a,b) // a,b ≥ 0 integers

1  IF b==0 THEN
2    return a
3  ELSE //a mod b Divisionsrest
4    return gcd(b,a mod b)
```

**Terminierung:** Algorithmus stoppt in endlicher Zeit ✓

Divisionsrest ist stets zwischen 0 und  $b-1$ , so dass zweites Argument in jeder Iteration um mindestens 1 kleiner wird und schließlich Basisfall  $b=0$  erreicht wird

# Beispiel: Euklidischer Algorithmus (III)

```
gcd(a,b) // a,b ≥ 0 integers

1  IF b==0 THEN
2    return a
3  ELSE //a mod b Divisionsrest
4    return gcd(b,a mod b)
```

**Determiniertheit:** gleiche Eingabe, gleiche Ausgabe ✓

**Determinismus:** gleiche Eingabe, gleiche Schritte ✓

# Beispiel: Euklidischer Algorithmus (IV)

```
gcd(a,b) // a,b ≥ 0 integers

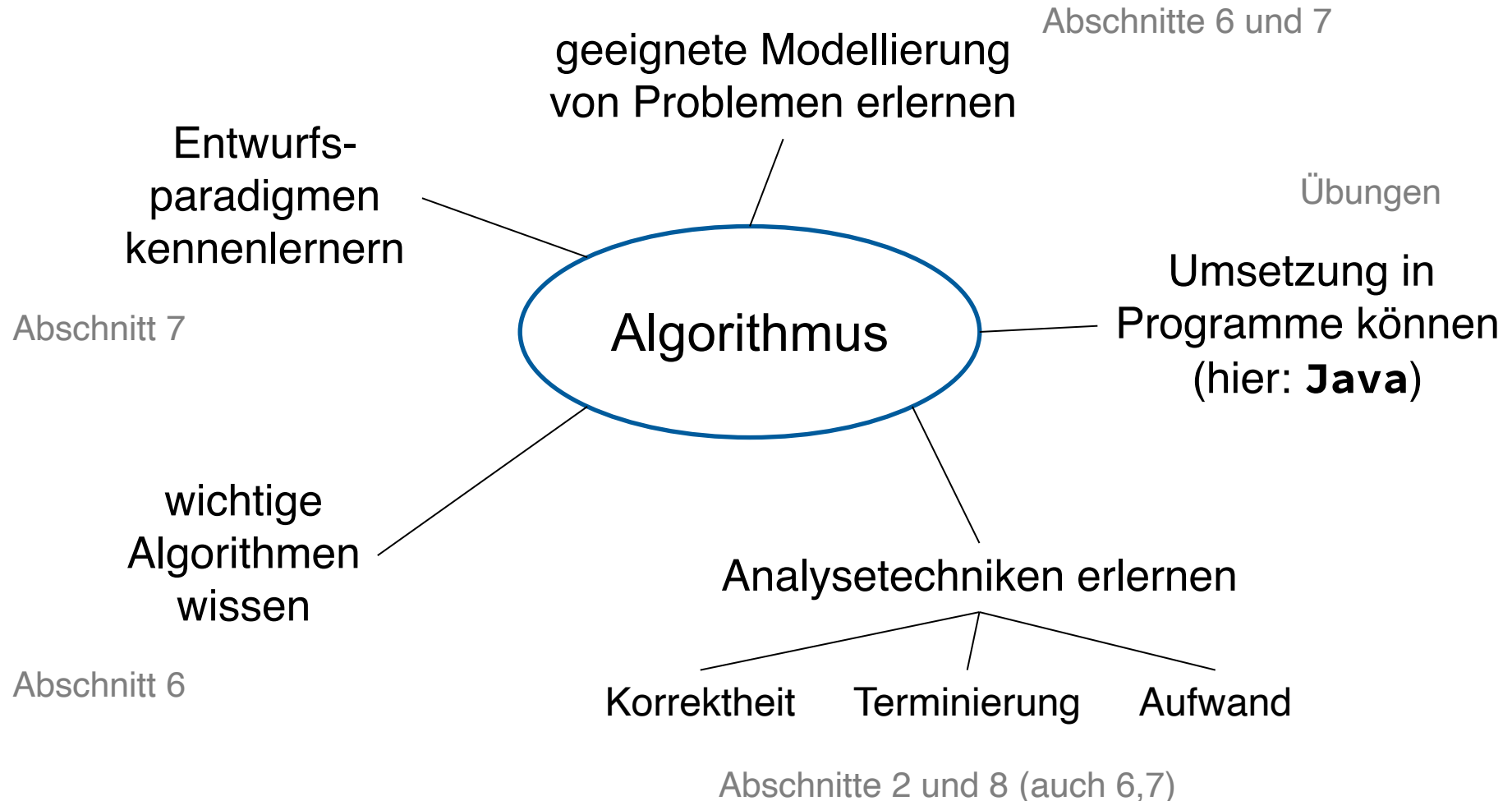
1  IF b==0 THEN
2    return a
3  ELSE //a mod b Divisionsrest
4    return gcd(b,a mod b)
```

**Allgemeinheit:** für Problemklasse anwendbar ✓

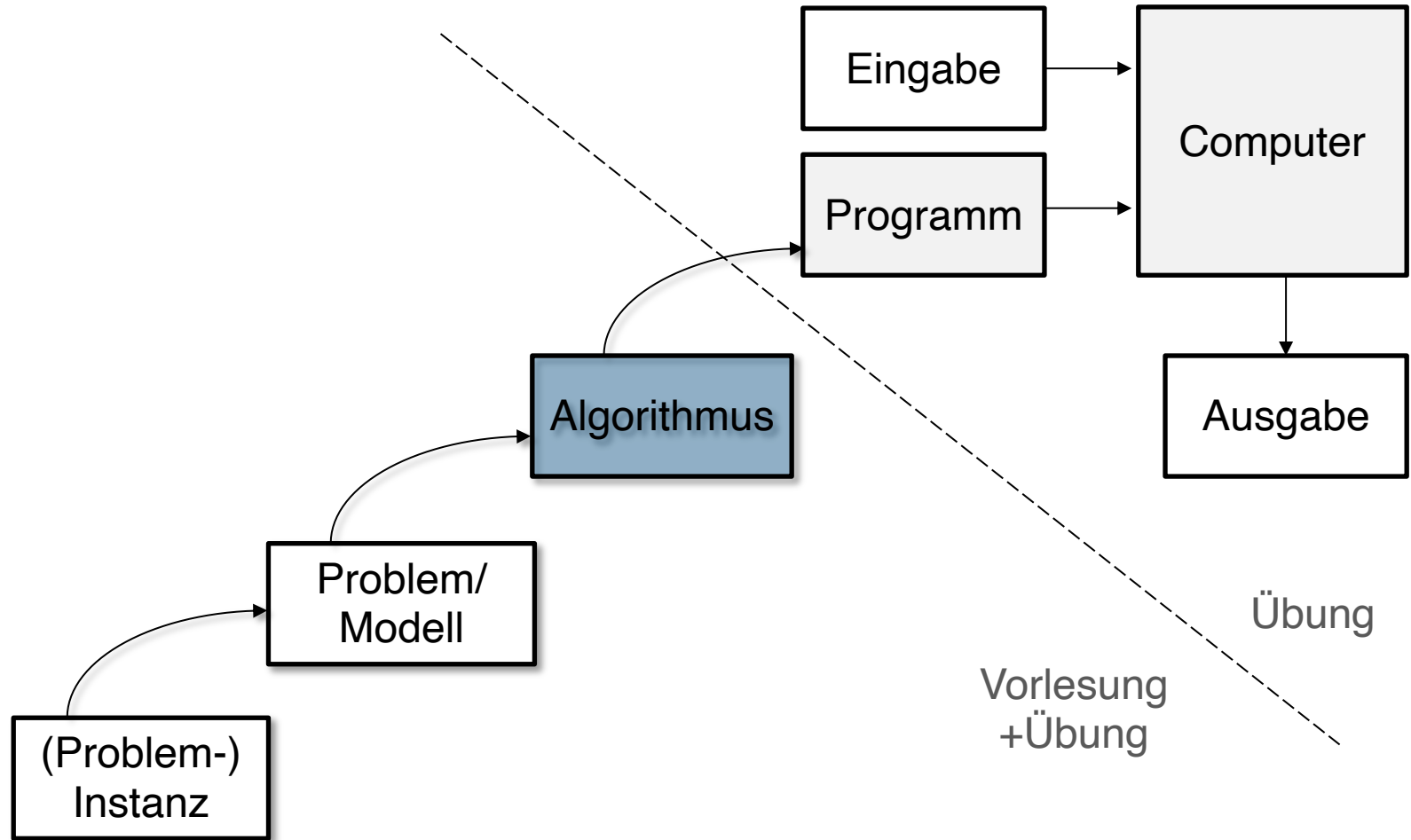
**Korrektheit:** falls terminiert, Ausgabe richtig ✓

Folgt aus  $\gcd(a, b) = \gcd(b, a \bmod b)$  (ohne Beweis)  
sowie  $\gcd(a, 0) = a$

# Lernziele Algorithmen



# Vorlesung und Übung (I)



# Vorlesung und Übung (II)

Code generiert mittels ChatGPT auf die Frage  
„Wie könnte bfs in java aussehen“, 21.März 2023

lauffähiger  
Java-Code

```
...  
1 public void bfs(int s) {  
2     boolean[] visited  
        = new boolean[V];  
3     LinkedList<Integer> queue  
        = new LinkedList<>();  
4  
5     visited[s] = true;  
6     queue.add(s);  
...
```

Pseudocode:  
einfacher Zugang

**BFS(G,s)**

```
1 s.color=GRAY; ...  
2 newQueue(Q);  
3 enqueue(Q,s);  
...
```

Übung

Vorlesung  
+Übung



Wie verhalten sich  
Determiniertheit (gleiche Eingabe, gleiche Ausgabe) und  
Determinismus (gleiche Eingabe, gleiche Schritte/Zustände)  
zueinander?



Was halten Sie von folgender Idee, die mod-Funktion  
(z.B. für den Euklidischen Algorithmus) umzusetzen?

```
MOD(a,b) //a,b≥0 integers  
1  WHILE a≥b DO  
2      a=a-b  
3  END WHILE  
4  return a
```



# Datenstrukturen

# Datenstrukturen

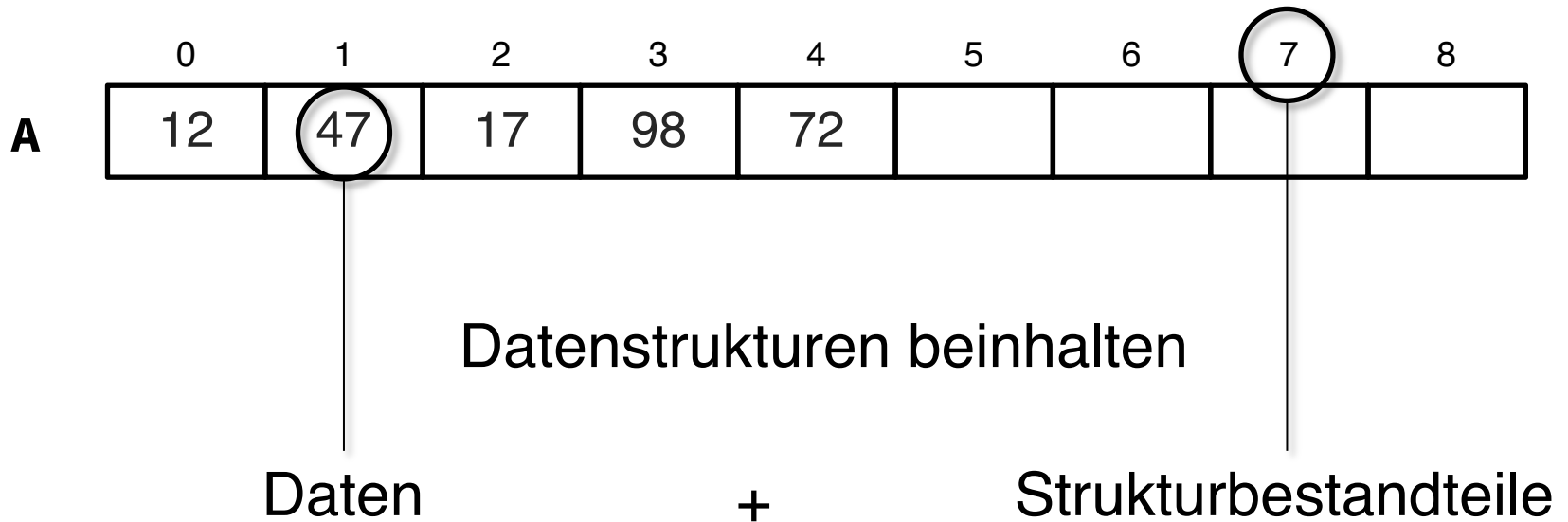
---

## Datenstrukturen:

Eine Datenstruktur ist eine Methode,  
Daten für den Zugriff und die Modifikation zu organisieren

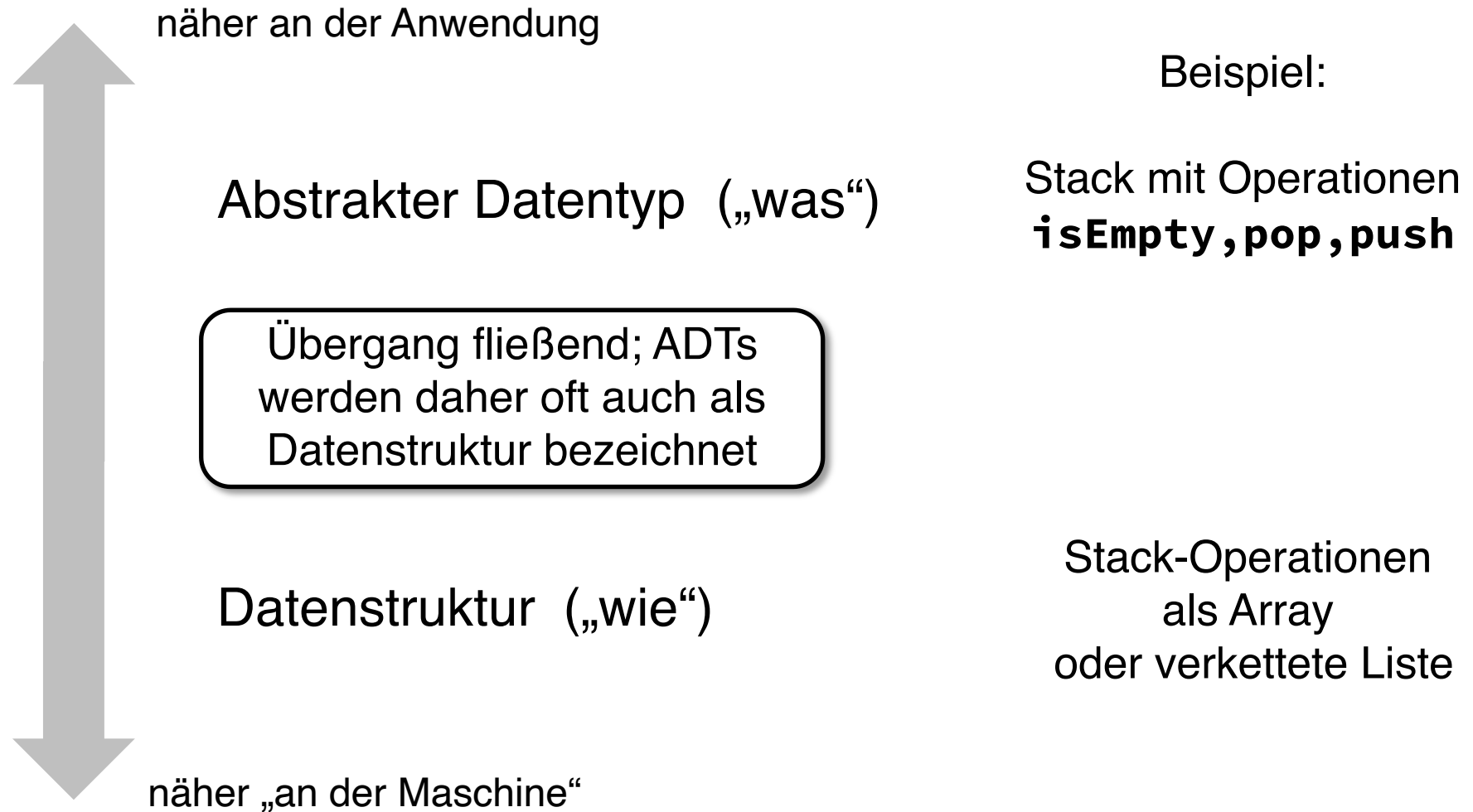
nach Cormen et al., Introduction to Algorithms

# Beispiel: Array

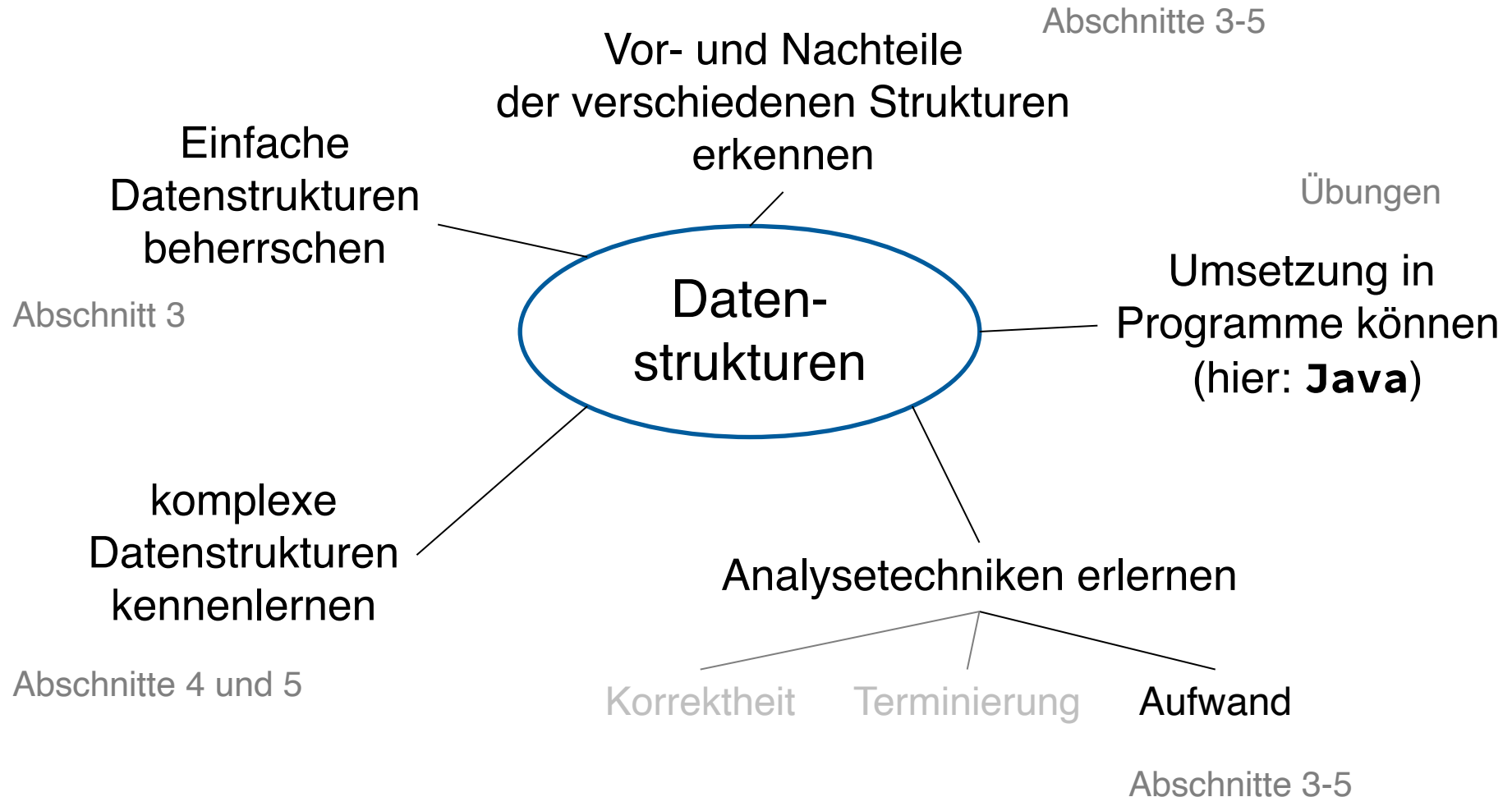


z.B. **A[7]** ist achter  
Eintrag im Speicher

# Abstrakte Datentypen (ADTs) und Datenstrukturen



# Lernziele Datenstrukturen



---

# Algorithmen *und* Datenstrukturen

# Datenstrukturen in Algorithmen



„Suche  
kürzesten  
Weg“

Abschnitt 6

**Dijkstra(...)**

```
1  ...
2  WHILE...
3    „gib mir den
   kleinsten Wert“
4  ...
```

**gesucht:  
Datenstruktur,  
mit der man leicht  
kleinste Einträge  
finden kann**

wirkt sich auf Effizienz  
des Algorithmus aus

# Algorithmen für Datenstrukturen



„Konstruiere eine Datenstruktur, mit der man schnell kleinste Werte finden kann“

komplexere  
Datenstruktur  
(z.B. Heap)

Abschnitt 4 (auch 3 und 5)

einfache  
Datenstruktur  
(z.B. Array)