

Algorithmen und Datenstrukturen



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Cryptoplexity

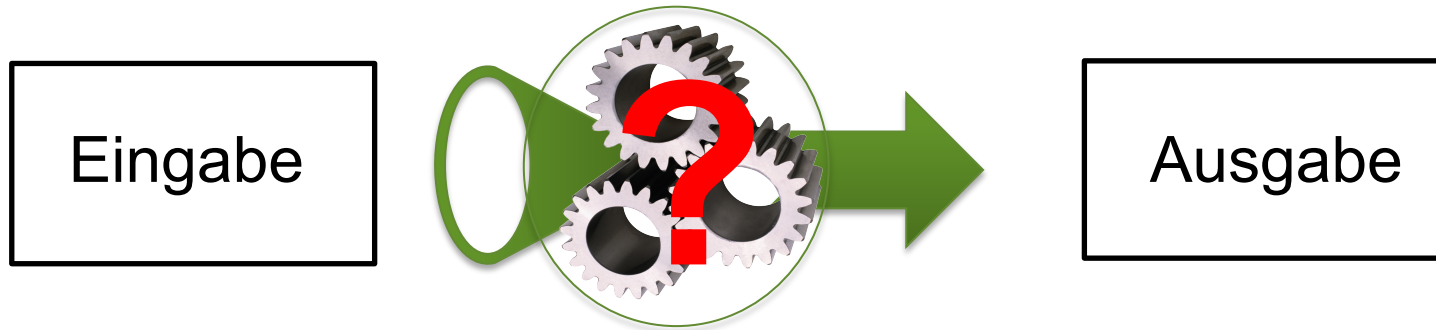
Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

Prof. Marc Fischlin, Dr. Christian Janson, SS 2020

01
Einleitung

Probleme in der Informatik

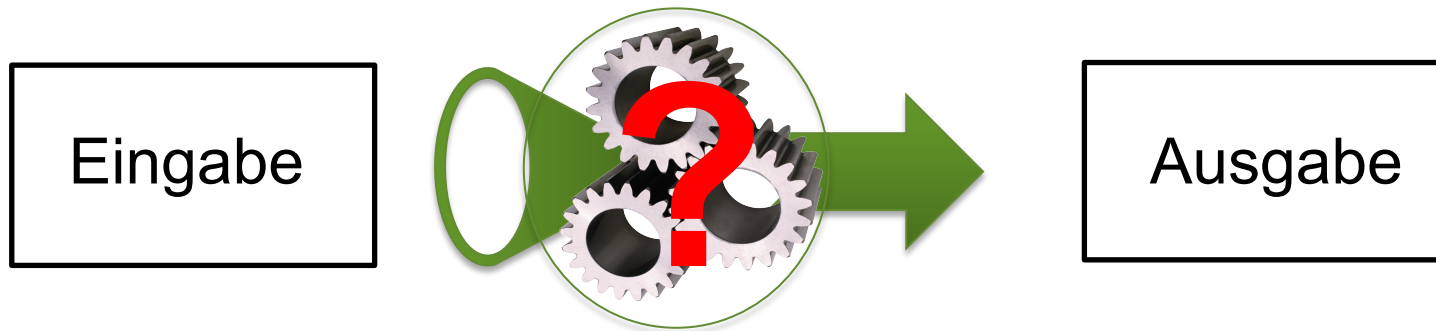
- Ein **Problem** im Sinne der Informatik:
 - Enthält eine Beschreibung der Eingabe
 - Enthält eine Beschreibung der Ausgabe
 - Gibt **keinen** Übergang von Eingabe und Ausgabe an



- Beispiele:
 - Finde den kürzesten Weg zwischen 2 Orten
 - Sortiere die Menge von Wörtern
 - Berechne die Quadratwurzel von x

Probleminstanzen

- Eine **Probleminstanz** ist eine konkrete Eingabenbelegung, für die die entsprechende Ausgabe gewünscht ist



- Beispiele von Probleminstanzen:
 - Was ist der kürzeste Weg vom Audimax in die Mensa?
 - Sortiere die Wörter alphabetisch: [Klausur, Algorithmus, Datenstruktur]
 - Berechne $x = \sqrt{77}$

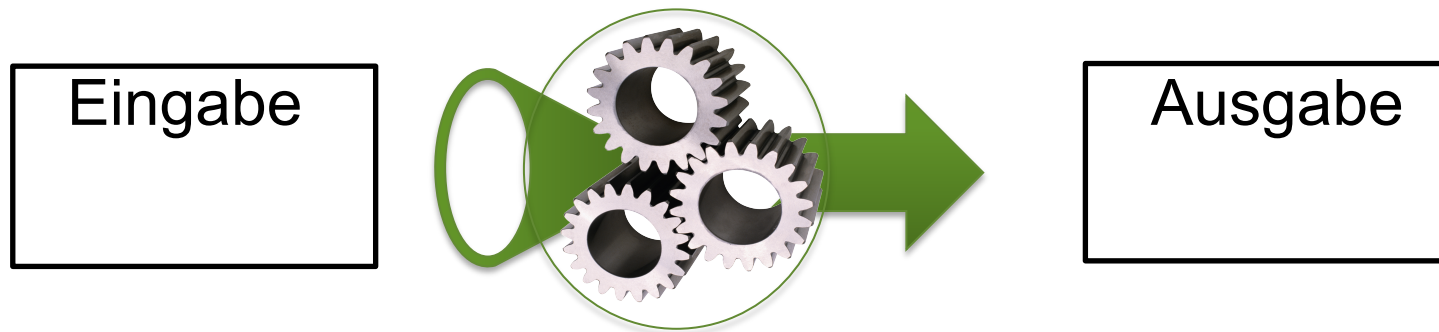
Begriff des Algorithmus

„Ein Algorithmus ist eine
endliche Folge von Rechenschritten,
die eine **Eingabe** in eine **Ausgabe** umwandelt.“

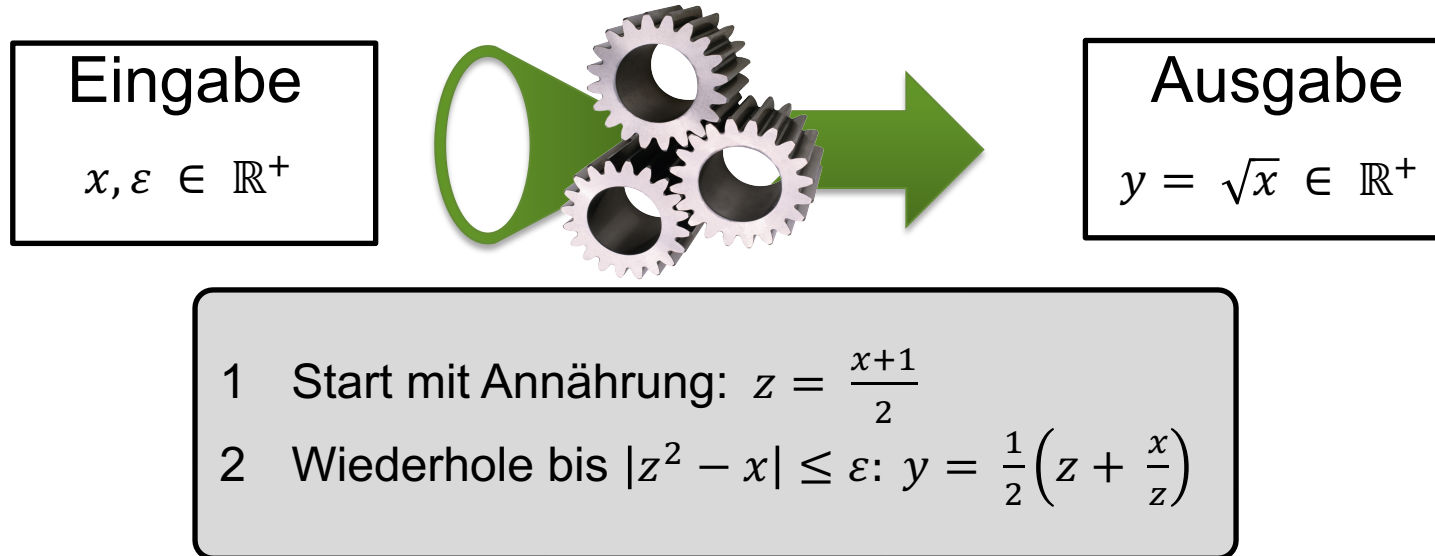
Cormen et al., 4. Auflage

Anforderungen an Algorithmen

- Spezifizierung der Eingabe und Ausgabe:
 - Anzahl und Typen aller Elemente ist definiert.
- Eindeutigkeit:
 - Jeder Einzelschritt ist klar definiert und ausführbar.
 - Die Reihenfolge der Einzelschritte ist festgelegt.
- Endlichkeit:
 - Die Notation hat eine endliche Länge.



Beispiel: Heron Verfahren



Beispiel: Summe in natürlicher Sprache

- Problem: Für ein gegebenes $n \in \mathbb{N}$, berechne die Summe $1 + 2 + \dots + n$
- Natürliche Sprache:
 - Beginne mit der Initialisierung einer Variablen **summe** mit Wert 0. Durchlaufe die natürlichen Zahlen 1 bis n mit einer weiteren Variable **zähler**. Addiere **zähler** jeweils zu **summe**. Gib nach dem Durchlauf den Text „Die Summe ist“ sowie den Wert von **summe** aus.

Beispiel: Summe in Pseudocode

- Problem: Für ein gegebenes $n \in \mathbb{N}$, berechne die Summe $1 + 2 + \dots + n$
- Pseudocode:

```
1  summe:=0;  
2  zähler:=1;  
3  WHILE zähler≤n  
4      summe=summe+zähler  
5      zähler=zähler+1  
6  ENDWHILE  
7  return „Die Summe ist“ summe
```


Beispiel: Summe in Java

- Problem: Für ein gegebenes $n \in \mathbb{N}$, berechne die Summe $1 + 2 + \dots + n$

- Java:

```
public static void main(String[] arg) {  
    int n = Integer.parseInt(arg[0]);  
    int summe = 0;  
    for (int i=1; i<=n; i++) {  
        summe = summe + i;  
    }  
    System.out.println(„Die Summe ist“ + summe);  
}
```

(Einige) Eigenschaften von Algorithmen

- **Determiniertheit:**
 - Für die gleiche Eingabe wird stets die gleiche Ausgabe berechnet (aber andere Zwischenzustände sind möglich).
- **Determinismus:**
 - Für die gleiche Eingabe ist die Ausführung und Ausgabe stets identisch.
- **Terminierung:**
 - Der Algorithmus läuft für jede Eingabe nur endlich lange.
- **Korrektheit:**
 - Der Algorithmus berechnet stets die spezifizierte Ausgabe (falls dieser terminiert).
- **Effizienz:**
 - Sparsamkeit im Ressourcenverbrauch (Zeit, Speicher, Energie, ...)

Lernziele Algorithmen

- Probleme analysieren und strukturieren
- Algorithmen auf Probleminstanzen anwenden und ausführen
- Rechenaufwand eines Algorithmus quantifizieren
- Korrektheit von Algorithmen (beispielhaft) zeigen
- Algorithmen herleiten und in Pseudocode darstellen
- Algorithmen in Java programmieren
- verschiedene Paradigmen zum Entwerfen von Algorithmen kennenlernen



Was ist eine Problemistanz?
Nennen Sie ein Beispiel.



Benennen Sie drei Anforderungen an einen Algorithmus.



Erklären Sie drei Eigenschaften, die ein Algorithmus besitzen sollte.

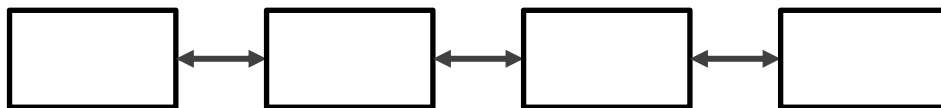
Begriff der Datenstrukturen

„Eine Datenstruktur ist eine Methode,
Daten **abzuspeichern** und zu **organisieren** sowie
den **Zugriff** auf die Daten und die **Modifikation**
der Daten zu erleichtern.“

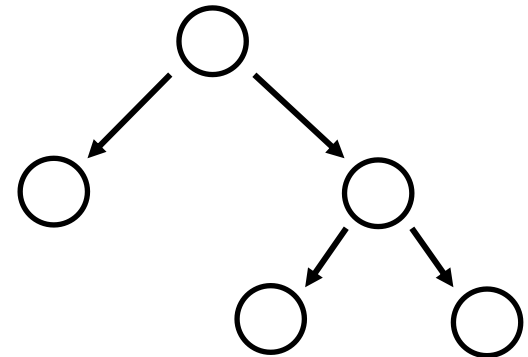
Cormen et al., 4. Auflage

Datenstrukturen

- Datenstrukturen:
 - Organisationsformen für Daten
 - Beinhalten Strukturbestandteile und Nutzerdaten (Payload)
- Beispiele:
 - Sequenzen: Arrays, Listen, ...
 - Topologische Strukturen: Bäume, Graphen, ...



Liste



Baum

Lernziele Datenstrukturen

- Datenstrukturen erkennen
- Basisoperationen auf Datenstrukturen anwenden
- Geeignete Datenstrukturen für eine Lösungsstrategie auswählen