



Technische  
Universität  
Braunschweig



# Algorithmen und Datenstrukturen

## Große Übung 0

Arne Schmidt

03.11.2022

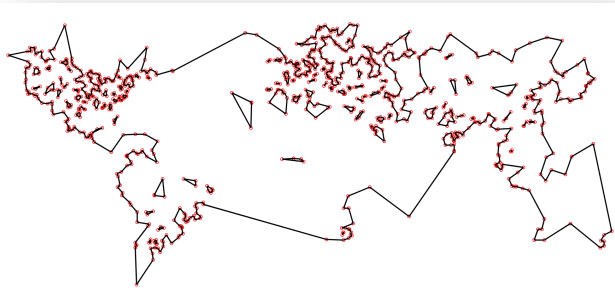
# Vorstellung

## Interessen

- Geometrische Optimierung,
- Programmierbare Materie,
- Komplexitätstheorie

### Computing Nonsimple Polygons of Minimum Perimeter

Sándor P. Fekete<sup>1</sup>, Andreas Haas<sup>1</sup>, Michael Hemmer<sup>1</sup>, Michael Hoffmann<sup>2</sup>,  
Irina Kostitsyna<sup>3</sup>, Dominik Krupke<sup>1</sup>, Florian Maurer<sup>1</sup>, Joseph S. B. Mitchell<sup>4</sup>,  
Arne Schmidt<sup>1</sup>, Christiane Schmidt<sup>5</sup>, and Julian Troegel<sup>1</sup>



### Computing MaxMin Edge Length Triangulations

Sándor P. Fekete\* Winfried Hellmann\* Michael Hemmer\* Arne Schmidt\*  
Julian Troegel\*

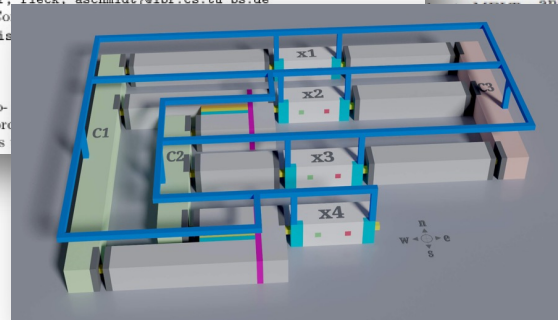
### Particle-Based Assembly Using Precise Global Control\*

Robert Keller<sup>1</sup>[0000-0001-9988-953X], Christian Rieck<sup>1</sup>[0000-0003-0846-5163],  
Florian Scheffer<sup>2</sup>[0000-0002-3471-2706], and Arne Schmidt<sup>1</sup>[0000-0001-8950-3963]

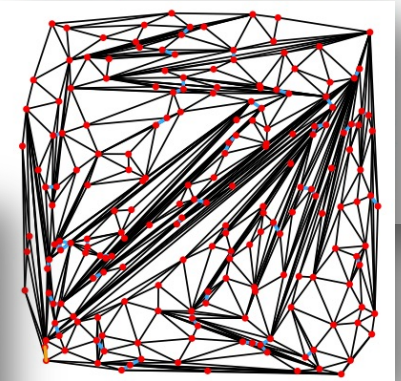
<sup>1</sup> Department of Computer Science, TU Braunschweig, Germany.  
{jkeller, rieck, aschmidt}@ibr.cs.tu-bs.de

<sup>2</sup> Department of Computer Science,  
University of Cologne, Germany

**Abstract.** In micro-robotic assembly, particles are often used to form a structure. We study the problem of using an external force to control the assembly of adhesive particles.



whik  
hull,  
) algorithm  
of a set of  
ty of finding  
as a natural  
problem by  
te. Moreover,  
polynomial-  
optim  
angle



# Organisation

# Homepage und Anmeldung

- Veranstaltungsübersicht:

<https://www.ibr.cs.tu-bs.de/courses/ws2223/aud/index.html>

## Algorithmen und Datenstrukturen

Semester Wintersemester 2022/2023 ▾

Studiengänge Bachelor Wirtschaftsinformatik, Bachelor Informations-Systemtechnik, Bachelor Informatik

IBR Gruppe ALG (Prof. Fekete)

Art Vorlesung/Übung

Dozent



**Prof. Dr. Sándor P. Fekete**

Abteilungsleiter

✉ s.fekete@tu-bs.de

☎ +49 531 3913111

📄 Raum 335

Assistent



**Dr. Arne Schmidt**

Wissenschaftlicher Mitarbeiter

✉ aschmidt@ibr.cs.tu-bs.de

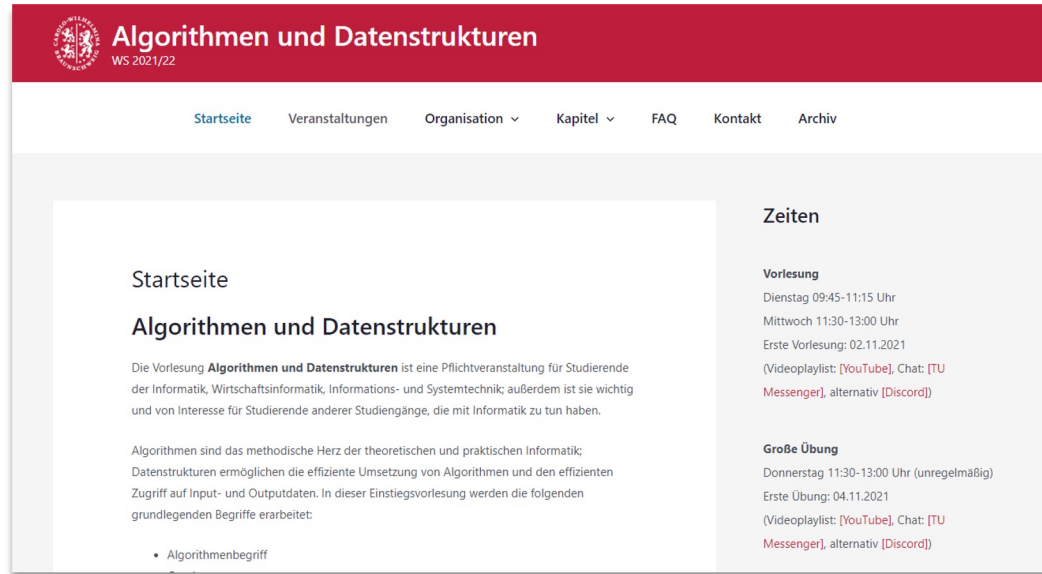
☎ +49 531 3913115


📄 Raum 319

# Homepage und Anmeldung

- Folien, Hausaufgaben, Vorlesungsvideos, Altklausuren, Übungsanmeldungen:

<https://www.aud.ibr.cs.tu-bs.de>



 **Algorithmen und Datenstrukturen**  
WS 2021/22

[Startseite](#) [Veranstaltungen](#) [Organisation](#) [Kapitel](#) [FAQ](#) [Kontakt](#) [Archiv](#)

## Startseite

### Algorithmen und Datenstrukturen

Die Vorlesung **Algorithmen und Datenstrukturen** ist eine Pflichtveranstaltung für Studierende der Informatik, Wirtschaftsinformatik, Informations- und Systemtechnik; außerdem ist sie wichtig und von Interesse für Studierende anderer Studiengänge, die mit Informatik zu tun haben.

Algorithmen sind das methodische Herz der theoretischen und praktischen Informatik; Datenstrukturen ermöglichen die effiziente Umsetzung von Algorithmen und den effizienten Zugriff auf Input- und Outputdaten. In dieser Einstiegsvorlesung werden die folgenden grundlegenden Begriffe erarbeitet:

- Algorithmenbegriff

#### Zeiten

**Vorlesung**  
Dienstag 09:45-11:15 Uhr  
Mittwoch 11:30-13:00 Uhr  
Erste Vorlesung: 02.11.2021  
(Videoplaylist: [YouTube](#), Chat: [TU Messenger](#)], alternativ [Discord](#))

**Große Übung**  
Donnerstag 11:30-13:00 Uhr (unregelmäßig)  
Erste Übung: 04.11.2021  
(Videoplaylist: [YouTube](#), Chat: [TU Messenger](#)], alternativ [Discord](#))

# Homepage und Anmeldung

- Anmeldung:

<https://www.aud.ibr.cs.tu-bs.de/organisation>

- Zuweisung erfolgt (voraussichtlich) am 09.11.22

## Anmeldung

- Mailingliste: [HIER](#). Es werden nur Mailadressen der TU Braunschweig freigeschaltet. Bei Fragen oder technischen Problemen wendet euch einfach per Mail an [Arne](#).
- Übungsgruppen: Bis einschließlich 08.11.22 auf der [Beitragsseite](#).
- Klausur: Beim jeweiligen Prüfungsamt

## Anmeldung Kleine Übungen

Allgemein / 26. Oktober 2022

Die Anmeldung für die kleinen Übungen ist ab sofort bis einschließlich 08.11.22 geöffnet.

Bitte beachtet folgende Punkte:

- Bitte wählt so viele Termine wie möglich, damit wir am Ende eine faire Zuteilung garantieren können.
- Wir werten immer die neueste Anmeldung (einfach das Formular erneut abschicken).
- Bei erfolgreicher Anmeldung erhält man eine Mail mit einer Übersicht der eingegebenen Daten. Das ist noch keine Zuweisung der Gruppen!
- Überprüft **vor** dem Absenden des Formulars, ob eure Daten (vor allem Matrikelnummer und eMail) korrekt sind.
- Die **Einteilung** in die Gruppen findet voraussichtlich am 09.11.22 statt. Ihr erhaltet dazu eine separate Mail.

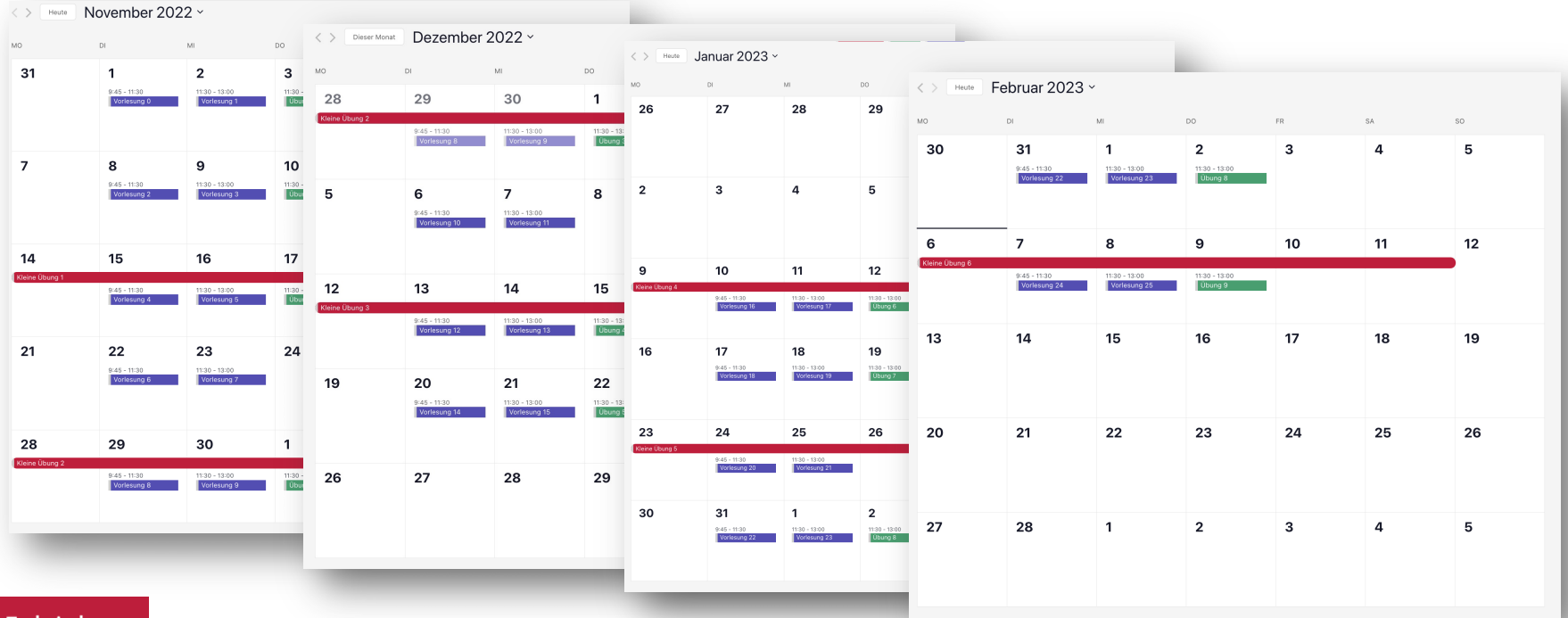
Nachname

Vorname

Matrikelnummer

# Semesterkalender

<https://aud.ibr.cs.tu-bs.de/events/>



# Semesterplan

## Semesterplan Algorithmen und Datenstrukturen WS22/23

Woche (KW)	Woche (Datum)	Vorlesung (Di./Mi.)	Gr. Übung (Do.)	Kl. Übung (Mo.-Fr.)	Ausgabe (tba)	Abgabe (tba)	Besprechung (in kl. Übung)	
44	24.10.							
45	31.10.	0,1	0					
46	07.11.	2,3	1		P0+HA1			
47	14.11.	4,5	2*	1			P0	
48	21.11.	6,7			P1+HA2	HA1		
49	28.11.	8,9	3	2			P1+HA1	
50	05.12.	10,11			P2+HA3	HA2		
51	12.12.	12,13	4	3			P2+HA2	
52	19.12.	14,15	5*		P3+HA4	HA3		
53	26.12.	Weihnachtsferien						
1	02.01.	Weihnachtsferien						
2	09.01.	16,17	6	4			P3+HA3	
3	16.01.	18,19	7		P4+HA5	HA4		
4	23.01.	20,21		5			P4+HA4	
5	30.01.	22,23	8		P5	HA5		
6	06.02.	24,25	9 <sup>(*)</sup>	6			P5+HA5	

\*: Eine Exkurs-Übung, in der spannende Probleme und Methoden aus der Algorithmik vorgestellt werden.

Sollten kurzfristig Änderungen im Zeitplan notwendig sein, werden diese über die Mailingliste bekannt gegeben.

## Lehren

Kapitel ▾

Knowledge Space

FAQ

Kontakt

Archiv

### Zeiten

#### Vorlesung

Dienstag 09:45-11:15 Uhr SN 19.1

Mittwoch 11:30-13:00 Uhr SN 19.1

Erste Vorlesung: 01.11.2022

#### Große Übung

Donnerstag 11:30-13:00 Uhr SN 19.1

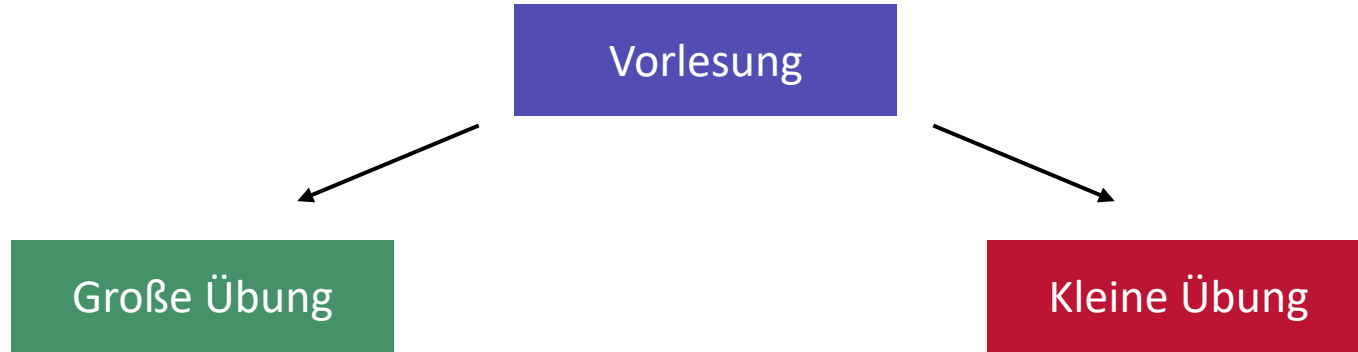
(unregelmäßig)

Erste Übung: 03.11.2021

Für Details siehe [Kalender](#) oder

[Gesamtübersicht](#).





- Aufarbeitung der Inhalte
- Exkurse
- Beantwortung von Fragen
- Interaktion!

- Vertiefung der Inhalte
- Selbständiges Arbeiten
- Besprechung von Hausaufgaben

# Hausaufgaben und Übungsblätter

5 verpflichtende  
Hausaufgabenblätter

6 freiwillige  
Übungsblätter

- Studienleistung
- 20 Punkte pro Blatt
- Studienleistung: 50% der Gesamtpunkte
  - Studienleistung ist **keine** Voraussetzung, um an der Prüfung teilzunehmen.
  - Studienleistung ist **eine** Voraussetzung, um das Modul abzuschließen.
  - Studienleistung ist nicht benotet und fließt nicht in die Prüfung ein.
- Zusätzliche Vertiefung
- Prüfungsvorbereitung

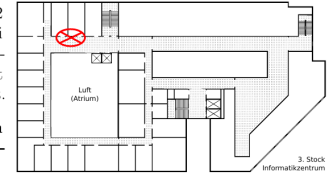
Algorithmen und Datenstrukturen  
Prof. Dr. Sándor P. Fekete  
Dr. Arne Schmidt

Winter 2022/23  
Abgabe: 21.11.2022  
Rückgabe: ab 28.11.2022

## Hausaufgabenblatt 1

Abgabe der Lösungen bis zum 21.11.2022 um 14:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet.

Bitte die Blätter zusammenheften und vorne mit Namen, Matrikelnummern versehen!



# Hausaufgaben

## Wozu Hausaufgaben?

Die Hausaufgaben dienen Euch (nicht uns) zur Vorbereitung auf die Klausur.

- Ideale Nachbereitung der Vorlesungsinhalte
- Zeitersparnis bei der Prüfungsvorbereitung
- Direktes Feedback, über euren aktuellen Lernstand.

- Zu späte Abgaben: 0 Punkte
- Zusammen überlegen, **ABER**: einzeln aufschreiben und abgeben, sonst: 0 Punkte.

# Hausaufgaben



Hausaufgabenschrank

# Klausur

- Voraussichtlich 03.03.23, zwischen 10:30 Uhr und 13:30 Uhr.
- Raumaufteilung und Beginn der Klausur folgen auf Webseite.
- Inhalt: Prinzipiell alles aus VL und Übung
- Dauer: 2 Stunden



Technische Universität Braunschweig  
Institut für Betriebssysteme und Rechnerverbund  
Abteilung Algorithmik

Wintersemester 2019/2020

Prof. Dr. Sándor P. Fekete  
Arne Schmidt

Klausur  
*Algorithmen und Datenstrukturen*  
28.02.2020

Name: .....

Vorname: .....

Matr.-Nr.: .....

Studiengang: .....

Bachelor  Master  Andere

**Klausurcode:**  
*Dieser wird benötigt, um das Ergebnis der Klausur abzurufen.*

# Mailingliste

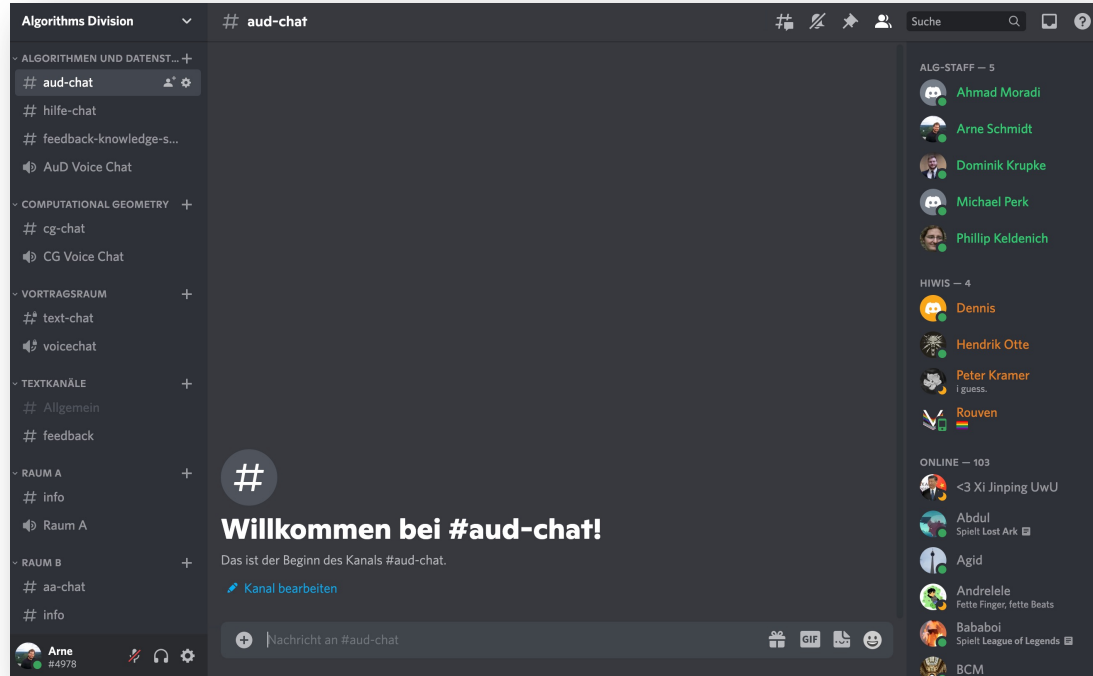
- Anmeldung über Homepage
- Für Informationen wie
  - Raumänderungen,
  - Ausfälle,
  - Etc.
- Möglichkeit für Fragen

The screenshot shows the Postorius interface for the mailing list 'Aud' (aud@ibr.cs.tu-bs.de). The page includes navigation links for 'Lists' and 'Archives', and user options for 'Login' and 'Sign Up'. The main content is divided into sections: 'Summary' (Mailingliste für Algorithmen und Datenstrukturen), contact information (aud-owner@ibr.cs.tu-bs.de), and a 'Subscription / Unsubscription' section. The subscription section contains a 'Log In' button and a form with fields for 'Your email address' and 'Your name (optional)', followed by a 'Subscribe' button. A footer at the bottom provides links to 'Postorius Documentation', 'GNU Mailman', and 'Postorius Version 1.3.6'.

# Discord



Discord



Kommunikation abseits der Vorlesung und Übung und Austausch mit Mitstudierenden.

# Fragen

Stellt Eure Fragen

- Über die Mailingliste
- Euren Tutoren
- Sprechstunde von Arne (Mo. 9:45 – 10:30 Uhr)
- Per Mail an Arne ([aschmidt@ibr.cs.tu-bs.de](mailto:aschmidt@ibr.cs.tu-bs.de))





**Fragen?**

# Pseudocode

<https://www.ibr.cs.tu-bs.de/alg/Merkzettel/pseudocode-booklet.pdf>


Me: \*starts programming without  
writing pseudocode first\*

My university:



# Pseudocode

Wörterbuch

 pseu·do

/pseúdo/

Adjektiv UMGANGSSPRACHLICH

nicht echt, nur nachgemacht, nachgeahmt

Wörterbuch

 Quell·code

/Quéllcode/

Substantiv, maskulin [der] EDV

in einer [höheren] Programmiersprache geschriebene Abfolge von Programmanweisungen, die vom Menschen gelesen, aber erst nach einer elektronischen Übersetzung vom Computer verarbeitet werden können

# Warum Pseudocode?

```
public class HalloWelt {  
    public static void main(String[] args) {  
        System.out.println("Hallo Welt!");  
    }  
}
```

Ist dieser Code effizient zu lesen?

Was können wir weglassen, wenn ein Mensch (kein Computer) diesen Code verstehen soll?

# Warum Pseudocode?

Java

```
public class HalloWelt {  
    public static void main(String[] args) {  
        System.out.println("Hallo Welt!");  
    }  
}
```

Pseudocode

```
function main(args)  
    print Hallo Welt!  
end function
```

# Problem und Instanz

Problem



Instanz

- Allgemeine Fragestellung
  - Lösung: Angabe eines Algorithmus
  - (Meist) Formuliert durch
    - Eingabe: Was ist gegeben?
    - Ausgabe: Was ist gesucht?
- Konkrete Fragestellung
  - Lösung: Angabe einer konkreten Ausgabe
  - Formuliert durch eine konkrete Eingabe eines bestimmten Problems

# Problem und Instanz Beispiel

Problem

**Größter gemeinsamer Teiler zweier Zahlen**

**Eingabe:** Zwei Zahlen  $a$  und  $b$ .

**Ausgabe:** Der größte gemeinsame Teiler  $q$  von  $a$  und  $b$ .

Lösung: Euklidischer Algorithmus (gleich mehr)



Instanz

**Größter gemeinsamer Teiler zweier Zahlen**

- ggT(5, 102); Lösung: 1
- ggT(8, 64); Lösung: 8
- ...

# Algorithmus

Eindeutige Handlungsvorschrift zur Lösung eines Problems

- Eigenschaften
  - Ausführbarkeit
  - Endlichkeit
  - Endliche Ausführzeit
  - Endlicher Speicherbedarf
- Beschrieben durch
  - Prosatext
  - Pseudocode



Donald Knuth



Alan Turing



# Beispiel: Euklidischer Algorithmus

## Prosatext

Solange die Zahl  $b$  nicht 0 ist, wählen wir  $h = a \bmod b$ , setzen  $a$  auf  $b$  und  $b$  auf  $h$ . Nachdem  $b = 0$ , geben wir  $a$  zurück.

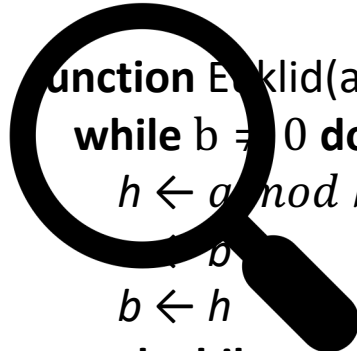
## Pseudocode

```
function Euklid( $a$ ,  $b$ )  
  while  $b \neq 0$  do  
     $h \leftarrow a \bmod b$   
     $a \leftarrow b$   
     $b \leftarrow h$   
  end while  
  return  $a$   
end function
```

(Für modulo gilt  $r = a \bmod b$ , sodass  $a = q \cdot b + r$  für ein  $q \in \mathbb{Z}$  gilt mit  $0 \leq r < b$ .)

# Beispiel: Euklidischer Algorithmus

## Pseudocode



```
function Euklid(a, b)
  while b ≠ 0 do
    h ← a mod b
    a ← b
    b ← h
  end while
  return a
end function
```

## Schlüsselwörter

- Anlehnung an höhere Programmiersprachen
- Vereinfachung zur Übersichtlichkeit (keine Klammern, Typen, etc.)
- ABER: **end** statements
- Können sowohl auf Deutsch als auch auf Englisch benutzt werden

# Pseudocode - Bausteine

## Methoden

```
function NAME (Param1, Param2, ...)  
  ...  
end function
```

## Zuweisungen

```
 $a := b$  (Weist  $a$  den Wert  $b$  zu)  
 $a \leftarrow b$   
  
 $a := b + c$  (Weist  $a$  die Summe aus  $b$  und  $c$  zu)  
 $A := B$  (Weist der Menge  $A$  die Menge  $B$  zu)
```

## Bedingungen

```
if Bedingung then  
  Aktion falls Bedingung wahr ist  
else  
  Aktion, falls Bedingung falsch ist  
end if
```

## Ausgaben / Rückgaben

```
print  $a$  (Gibt  $a$  aus)  
return  $a$  (Gibt  $a$  aus und beendet die Funktion)
```

# Pseudocode - Bausteine

## Schleifen - 1

**while** Bedingung **do**

Führe Aktion aus, *solange* eine Bedingung wahr ist.

**end while**

## Schleifen - 2

**repeat**

Führe Aktion aus, *bis* eine Bedingung wahr ist.

**until** Bedingung

## Schleifen - 3

**for**  $a$  in  $b, \dots, c$  **do**

Starte mit  $a := b$

$a$  wird nach jeder Iteration um 1 erhöht. ( $a := a + 1$ )

Wiederhole bis  $a$  den Wert  $c$  erreicht hat.

**end for**

## Schleifen - 4

**for**  $a := b$  **down to**  $c$

Starte mit  $a := b$

$a$  wird in jeder Iteration um 1 verringert. ( $a := a - 1$ )

Wiederhole bis  $a$  den Wert  $c$  erreicht hat.

**end for**

# Pseudocode – In Aktion

1. **function** Euklid( $a$ ,  $b$ )
2.     **while**  $b \neq 0$  **do**
3.          $h \leftarrow a \bmod b$
4.          $a \leftarrow b$
5.          $b \leftarrow h$
6.     **end while**
7.     **return**  $a$
8. **end function**

Instanz: Berechne ggT von  $a = 49$ ,  $b = 21$

Aufruf: Euklid(49, 21)

Rückgabe: 7

Zeile	Iteration	a	b	h
1	-	49	21	-
2	-	49	21	-
3	1	49	21	7
4	1	21	21	7
5	1	21	7	7
6	1	21	7	7
2	1	21	7	7
3	2	21	7	0
4	2	7	7	0
5	2	7	0	0
6	2	7	0	0
2	2	7	0	0
7	2	7	0	0

# Pseudocode – In Aktion

1. **function** Mult( $a, b$ )
2.      $c \leftarrow 0$
3.     **for**  $d := b$  **downto** 1
4.          $c \leftarrow c + a$
5.     **end for**
6.     **return**  $c$
7. **end function**

Instanz: Berechne das Produkt von 9 und 3

Aufruf: Mult(9, 3)

Rückgabe: 27

Variablen nach jeder Iteration der for-Schleife:

Iteration	a	b	c	d
-----------	---	---	---	---

# Pseudocode – In Aktion

1. **function** Absolute(a)
2.     **if**  $a < 0$  **then**
3.          $a \leftarrow -a$
4.     **end if**
5.     **return** a
6. **end function**

1. **function** AbsoluteDiff(a, b)
2.     **return** Absolute(a) – Absolute(b)
3. **end function**

Aufruf: AbsoluteDiff(-10, 3)

Rückgabe: 7

Zeile (AbsoluteDiff)	Zeile (Absolute)	a (AbsoluteDiff)	a (Absolute)	b
1	-	-10	-	3
2	1	-10	-10	3
2	2	-10	-10	3
2	3	-10	10	3
2	4	-10	10	3
2	5	-10	10	3
2	1	-10	3	3
2	2	-10	3	3
2	4	-10	3	3
2	5	-10	3	3

# Pseudocode - Varianten

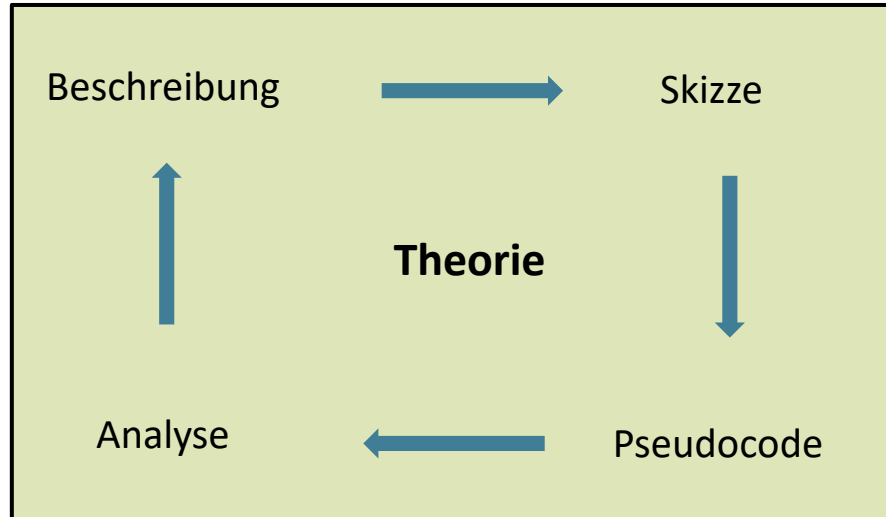
- Auf Deutsch
  - Wenn ... Dann ... Sonst ...
  - Solange ...
  - Für  $i \leftarrow 1, \dots, k$  ...
- Statt  $\leftarrow$  schreibt man gelegentlich  $:=$  oder sogar  $=$
- Mischung aus Prosa und Pseudocode:
  1. **function** Verhältnis(*liste*)
  2.     Sortiere die Zahlen in *liste* aufsteigend
  3.     **return** *liste*[0] / *liste*[länge(*liste*)]
  4. **end function**
- end... statements dürfen weggelassen werden, aber Einrückung muss korrekt sein!

## Wichtig

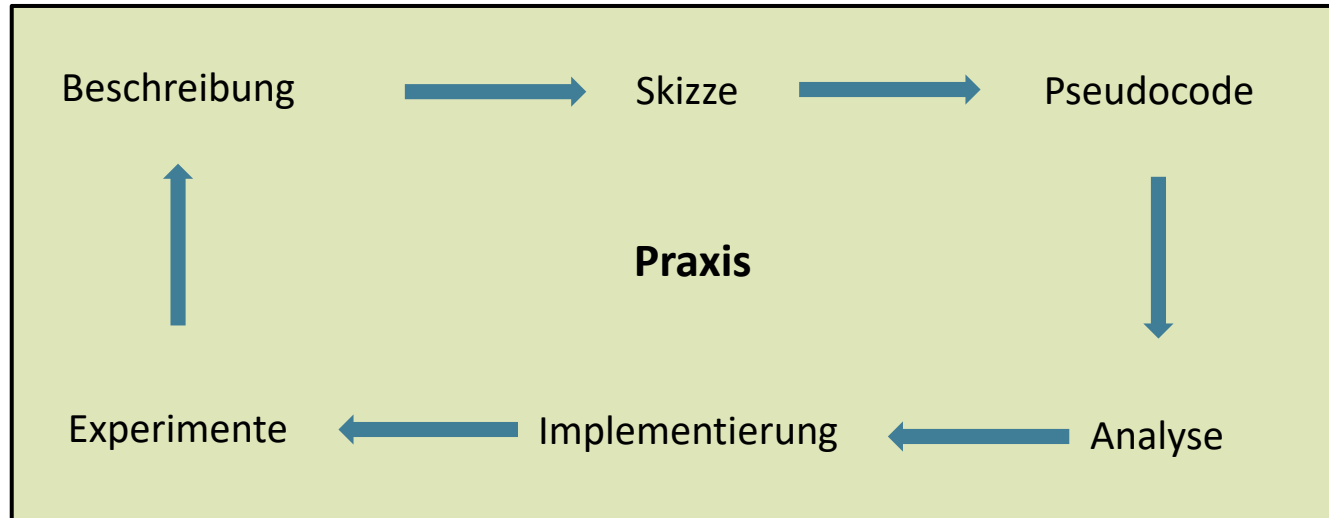
Pseudocode muss immer verständlich und eindeutig sein!



# Algorithmenentwurf in der Theorie



# Algorithmenentwurf in der Theorie



# Ein einfacher Algorithmus für den Logarithmus

Betrachten wir den Logarithmus:

**Gegeben** Zwei (ganzzahlige) Zahlen  $n$  und  $b$

**Gesucht** Eine (nicht-ganzzahlige) Zahl  $x$ , sodass  $n = b^x$  (Oder:  $x = \log_b n$ )

*Einfach ausgedrückt: Wie oft muss man  $n$  durch  $b$  teilen, damit man auf 1 kommt?*

Betrachten wir zunächst den einfachen Fall:  $x$  ist ganzzahlig

1. **function** Log( $n$ ,  $b$ )
2.      $\log \leftarrow 0$
3.     **while**  $n > 1$  **do**
4.          $\log \leftarrow \log + 1$
5.          $n \leftarrow n/b$
6.     **return**  $\log$

**Beispiel:** Log(64, 2)

<b>log</b>	0	1	2	3	4	5	6
<b>n</b>	64	32	16	8	4	2	1

# Ein Algorithmus für den Logarithmus

Was passiert, wenn  $x$  nicht ganzzahlig wird?

**Problem:** Irrationale Ergebnisse

**Lösung:** Bestimme den Logarithmus nur auf  $k$  Stellen genau.

Idee: Nutze den alten Algorithmus, indem wir die Stellen vor das Dezimalkomma verschieben.  
 $10^k \log_b n$  gibt uns  $k$  Stellen mehr vor dem Komma.

1. **function**  $\text{Log}(n, b, k)$
2.  $n \leftarrow n^{(10^k)}$       #  $10^k \log_b n = \log_b(n^{(10^k)})$
3.  $\text{log} \leftarrow \text{Log}(n, b)$
4. **return**  $\text{log} \cdot 10^{-k}$
5. **end function**

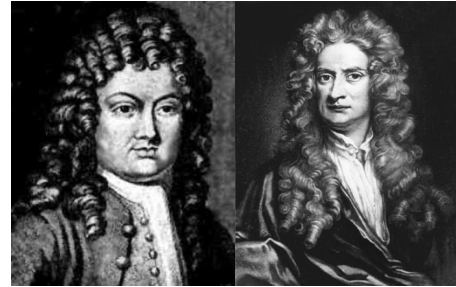
Problem 1: Schon für kleine  $k$  wird  $n$  sehr groß

Problem 2: Der Algorithmus benötigt exponentiell viele Schritte

# Mehr zu Logarithmen

Logarithmen lassen sich deutlich schneller und präziser berechnen.

- Taylor-Entwicklung (aus der Analysis)
- Newton-Verfahren (aus der Numerik)



Für diese Vorlesung ist die genaue Berechnung von (nicht ganzzahligen) Logarithmen nicht relevant.

Logarithmen werden aber dennoch benötigt, beispielsweise

- für Laufzeiten, z.B. beim Sortieren (Kapitel 5)
- zum Bestimmen der Höhe eines Suchbaumes (Kapitel 4)

**Wichtig:** Rechenregeln für Logarithmen!

**Fragen?**