



Teil 3: Algorithmen

Ein Kernkonzept zur Lösung komplexer mathematischer Probleme sind *Algorithmen*.

1 Algorithmen

Definition 1 (Algorithmus). *Ein Algorithmus ist eine Berechnungsvorschrift zur Lösung eines Problems. Er setzt sich zusammen aus einer Folge von endlich vielen, wohldefinierten Einzelschritten.*

Anmerkungen

- Algorithmen können in vielfältiger Weise dargestellt werden.
 - Beispiele: Pseudocodes, Java-Codes, Programmablaufpläne, ...
- Wir verwenden hier eine einfach zu verstehende, textuelle Darstellung.

Wir schauen uns zunächst ein einfaches Beispiel an:

Beispiel 1: Der folgende Algorithmus berechnet die Fakultät $n!$ einer natürlichen Zahl n .

Algorithmus FAK. **Eingabe:** Natürliche Zahl n .

1. Setze $k = 1$ und $i = 2$.
2. Falls $i > n$, gehe zu Zeile 6.
3. Ersetze k durch $k \cdot i$.
4. Ersetze i durch $i + 1$.
5. Gehe zu Zeile 2.
6. Gib k als Ergebnis zurück.

Erklärung:

- Ein Algorithmus besitzt eine festgelegte Eingabe (hier n).
- Ein Algorithmus besteht aus einer Abfolge von Zeilen mit textuellen *Anweisungen*.
 - Die Anweisungen definieren und verändern *Variablen* (hier k , i und n).
- Die Zeilen werden von oben nach unten ausgeführt.
 - *Sprungbefehle* ändern die Ausführungsreihenfolge (siehe Zeilen 2 und 5).

- Zeile 2 ist *bedingter Sprung* (da an Bedingung geknüpft).
- Zeile 5 ist *unbedingter Sprung* (wird immer ausgeführt).
- Ein Algorithmus terminiert (= endet) durch Rückgabe eines Ergebnisses.

Beispiel 1 (Fortsetzung): Ausführung des Algorithmus

Führen wir den Algorithmus mit $n = 3$ aus, ergibt sich folgende Folge von Anweisungen:

1. In Zeile 1 wird $k = 1$ und $i = 2$ gesetzt.
2. In Zeile 2 testet der Algorithmus, ob $2 > 3$ gilt, und stellt fest, dass dies *nicht* der Fall ist. Der Algorithmus springt daher *nicht* nach Zeile 6 sondern setzt die Ausführung in Zeile 3 fort.
3. Nach Ausführung von Zeile 3 gilt $k = 2$ (und weiterhin $i = 2$).
4. Nach Ausführung von Zeile 4 gilt $i = 3$ (und weiterhin $k = 2$).
5. Zeile 5 schickt uns zurück zu Zeile 2.
6. In Zeile 2 testet der Algorithmus $3 > 3$ und stellt erneut fest, dass dies *nicht* der Fall ist.
7. Nach Zeile 3 gilt $k = 6$ (und weiterhin $i = 3$).
8. Nach Zeile 4 gilt $i = 4$ (und weiterhin $k = 6$).
9. Zeile 5 schickt uns zurück zu Zeile 2.
10. In Zeile 2 testet der Algorithmus $4 > 3$ und stellt fest, dass dies erfüllt ist. Der Algorithmus springt daher in die Zeile 6.
11. In Zeile 6 terminiert der Algorithmus mit dem Ergebnis $k = 6$.

Anmerkungen:

- Offenbar berechnet der Algorithmus das korrekte Ergebnis für $n = 3$.
- Der Algorithmus ist nicht nur für eine bestimmte Eingabe (wie $n = 3$) konstruiert, sondern funktioniert für *beliebige* (= alle möglichen) Eingaben $n \in \mathbb{N}$. Das ist ein entscheidendes Merkmal von Algorithmen!
- Wir können den Algorithmus etwas kürzen.

Algorithmus FAK (gekürzt). **Eingabe:** Natürliche Zahl n .

1. Setze $k = 1$ und $i = 2$.
2. Falls $i > n$, gib k als Ergebnis zurück.
3. Ersetze k durch $k \cdot i$.
4. Ersetze i durch $i + 1$ und gehe zu Zeile 2.

Aufgabe 1: Führen Sie den Algorithmus mit der Eingabe $n = 0$ und $n = 1$ aus.

Aufgabe 2: Wäre der Algorithmus korrekt, wenn wir i in Zeile 1 mit 1 initialisieren?

Aufbau von Algorithmen In unserer Darstellungsform setzen sich Algorithmen aus folgenden grundlegenden Einzelschritten zusammen:

- Einführen von Variablen (z.B. Setze $i = 2$.)
- Überschreiben von Variablen (z.B. Ersetze i durch $i + 1$.)
- Arithmetische Operationen mit im Vorfeld eingeführten Variablen (z.B. $i + 1$).
- Fallunterscheidungen: Falls ... tue dies, sonst tue das.
- Sprunganweisungen (siehe Beispiel oben).
- Unterprogrammaufrufe (besprechen wir gleich).

Aufgabe 3: Entwickeln Sie einen Algorithmus ABS zur Berechnung des Betrages $|x|$ einer reellen Zahl x .

Aufgabe 4:

- (a) Entwickeln Sie einen Algorithmus POT zur Berechnung der Potenz a^n zweier Zahlen $a \in \mathbb{R}$ und $n \in \mathbb{N}$. *Tipp:* $a^n = 1 \cdot \underbrace{a \cdot a \cdot \dots \cdot a}_{n \text{ Faktoren}}$.
- (b) Berechnen Sie mit Ihrem Algorithmus die Potenzen 2^4 , 4^0 und 2^{-1} . Stellen Sie dabei die Änderungen der Variablen nachvollziehbar dar. Was fällt Ihnen auf?

2 Unterprogramme

Den oben formulierten Algorithmus FAK können wir als *Unterprogramm* zur Berechnung der Fakultät in weiteren Algorithmen verwenden.

Beispiel 2: Der Binomialkoeffizient

$$\binom{n}{k} = \frac{n!}{k! \cdot (n - k)!}$$

lässt sich mittels FAK leicht über den folgenden Algorithmus berechnen:

Algorithmus BINOM. **Eingabe:** $n, k \in \mathbb{N}$

1. Setze $x = \text{FAK}(n)$ sowie $y = \text{FAK}(k)$ und $z = \text{FAK}(n - k)$.
2. Gib $x/(y \cdot z)$ als Ergebnis zurück.

Anmerkungen:

- Der Algorithmus BINOM kann wiederum als Unterprogramm in anderen Algorithmen verwendet werden, in denen Binomialkoeffizienten eine Rolle spielen.
- Mit Unterprogrammen lassen sich beliebig komplexe Algorithmen aus einfachen Grundbausteinen zusammen setzen.

Aufgabe 5: Entwickeln Sie einen Algorithmus EULER zur Berechnung der Eulerschen Zahl e , indem Sie folgende Summe auswerten:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

Verwenden Sie den Algorithmus FAK als Unterprogramm.

- Welches Problem ergibt sich bei der Ausführung Ihres Algorithmus?
- Modifizieren Sie Ihren Algorithmus, sodass dieser die Auswertung der Summe stoppt, sobald ein Summand kleiner als 0,001 ist.

3 Algorithmen zur Teilbarkeit

Im Folgenden brauchen wir Algorithmen zur Teilbarkeit ganzer Zahlen.

Vorüberlegung:

- Im Allgemeinen lassen sich zwei ganze Zahlen a und b nicht restlos teilen.
 - Beispiel: $14/4 = 3$ Rest 2.
- Wir bezeichnen mit $\lfloor a/b \rfloor$ das abgerundete Ergebnis der Division.
- Wir bezeichnen mit $a \bmod b$ den dazugehörigen Divisionsrest.
- Für alle ganzen Zahlen a und $b \neq 0$ gilt:

$$a = a \bmod b + b \cdot \lfloor a/b \rfloor \tag{1}$$

Beispiel 3: Für $a = 14$ und $b = 4$ erhalten wir:

- $\lfloor 14/4 \rfloor = 3$
- $14 \bmod 4 = 2$
- $14 = 2 + 4 \cdot 3 = 14 \bmod 4 + 4 \cdot \lfloor 14/4 \rfloor$

Aufgabe 6: Entwickeln Sie einen Algorithmus DIV zur Berechnung des Ergebnisses $\lfloor a/b \rfloor$ der ganzzahligen Division. Wenden Sie Ihren Algorithmus auf $a = 14$ und $b = 4$ an.

Hinweis: Zählen Sie mit, wie häufig b in a „hinein passt“.

Aufgabe 7: Entwickeln Sie einen Algorithmus namens MOD zur Berechnung des Divisionsrestes $a \bmod b$. Verwenden Sie Ihren Algorithmus DIV als Unterprogramm.

Definition 2. Der größte gemeinsame Teiler zweier natürlicher Zahlen a und b ist die größte natürliche Zahl d , durch die sowohl a als auch b restlos teilbar ist (d.h. $a \bmod d = b \bmod d = 0$).

Aufgabe 8: Entwickeln Sie einen Algorithmus namens GGT zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen a und b . Verwenden Sie Ihren Algorithmus MOD als Unterprogramm.

Aufgabe 9: Entwickeln Sie einen Algorithmus SIN zur näherungsweisen Berechnung der Sinus-Funktion mittels der Taylor-Entwicklung

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \dots$$

Stoppen Sie die Auswertung sobald ein Summand betragsmäßig kleiner als 0,001 ist.

Hinweis: Verwenden Sie POT, FAK (und eventuell MOD) als Unterprogramme.

4 Rekursion

Eine *Rekursion* liegt vor, wenn ein Algorithmus sich selbst als Unterprogramm aufruft.

Beispiel 4: Wir können die Fakultät $n!$ einer natürlichen Zahl auch rekursiv berechnen. Dazu nutzen wir folgenden Zusammenhang aus:

$$n! = \begin{cases} 1, & \text{falls } n = 1, \\ n \cdot (n-1)!, & \text{sonst.} \end{cases}$$

Algorithmus REKFAK. Eingabe: natürliche Zahl n .

1. Falls $n = 1$, gib 1 als Ergebnis zurück.
2. Berechne $k = \text{REKFAK}(n-1)$.
3. Gib $n \cdot k$ als Ergebnis zurück.

Wenn wir den Algorithmus mit der Eingabe $n = 3$ ausführen, erhalten wir folgenden Ablauf:

- In Zeile 1 stellt der Algorithmus $3 \neq 1$ fest.
- In Zeile 2 ruft der Algorithmus das Unterprogramm REKFAK mit der Eingabe $n-1 = 2$ auf. Die Ausführung des Algorithmus stoppt nun an dieser Stelle, bis das Ergebnis des Unterprogramms REKFAK(2) vorliegt.
 - In Zeile 1 stellt der Algorithmus $2 \neq 1$ fest.
 - In Zeile 2 ruft der Algorithmus das Unterprogramm REKFAK mit der Eingabe $n-1 = 1$ auf. Die Ausführung des Algorithmus stoppt nun an dieser Stelle, bis das Ergebnis des Unterprogramms REKFAK(1) vorliegt.
 - * In Zeile 1 stellt der Algorithmus $1 = 1$ fest und gibt 1 als Ergebnis an das übergeordnete Unterprogramm REKFAK(2) zurück.
 - In Zeile 3 gibt der Algorithmus das Ergebnis $n \cdot k = 2 \cdot 1 = 2$ an das übergeordnete Unterprogramm REKFAK(3) zurück.
- In Zeile 3 gibt der Algorithmus das Ergebnis $n \cdot k = 3 \cdot 2 = 6$ zurück.

Anmerkungen:

- Ein rekursiver Algorithmus benutzt in mindestens einer Anweisung sich selbst als Unterprogramm.
- Wichtig ist ein *Rekursionsabbruch* (hier mit $n = 1$), sonst endet der Algorithmus nicht.
- In rekursiven Algorithmen ergibt sich eine Hierarchie verschiedener Instanzen (= Kopien) des Algorithmus. Mit Einrückung können wir die verschiedenen Stufen der Rekursion sichtbar machen. (Im Beispiel: Ebene 1: Punkte. Ebene 2: Striche. Ebene 3: Sternchen)

Aufgabe 10: Entwickeln Sie einen rekursiven Algorithmus zur Berechnung der Potenz a^n zweier Zahlen $a \in \mathbb{R}$ und $n \in \mathbb{N}$. Berechnen Sie damit 2^4 .

Aufgabe 11: Entwickeln Sie einen rekursiven Algorithmus zur Berechnung des Binomialkoeffizienten $\binom{n}{k}$. Verwenden Sie dafür den Zusammenhang

$$\binom{n}{k} = \begin{cases} 1, & \text{falls } k = 0 \text{ oder } k = n, \\ \binom{n-1}{k-1} + \binom{n-1}{k}, & \text{sonst.} \end{cases}$$