Algorithmische Mathematik I

Wintersemester 2023/2024
Prof. Dr. Barbara Verfürth
Jannik Michels und Uta Seidler

Übungsblatt 2.

Abgabe am **26.10.2023**.

Aufgabe 1. (Euklidischer Algorithmus)

(3+4+3 Punkte)

Algorithmus 1: Euklidischer Algorithmus

Input: $a, b \in \mathbb{N}$
Output: ggT von a und b

```
def euclid(a, b):
    if b = 0:
        return a
    else:
        return euclid(b, a%b)
```

Der (moderne) euklidische Algorithmus (siehe Algorithmus 1) erlaubt die effiziente Berechnung des größten gemeinsamen Teilers (ggT) zweier Zahlen. Für $a, b \in \mathbb{N}$ ist der $\mathrm{ggT}(a, b)$ definiert als die größte Zahl $t \in \mathbb{N}$, für die $t|a$ und $t|b$ gilt. Für $x, y \in \mathbb{N}$ bedeutet die Schreibweise $x|y$, dass x *Teiler von* y ist.

a) Zeigen Sie, dass für $a, b \in \mathbb{N}$ mit $a \geq b$

$$\mathrm{ggT}(a, b) = \mathrm{ggT}(a - b, b)$$

gilt.

b) Sei $k \in \mathbb{N}$ und $a, b \in \mathbb{N}$ mit $a \geq kb$. Zeigen Sie, dass für alle $l \in \mathbb{N}$ mit $l \leq k$

$$\mathrm{ggT}(a, b) = \mathrm{ggT}(a - lb, b)$$

gilt und folgern Sie

$$\mathrm{ggT}(a, b) = \mathrm{ggT}(a \mod b, b).$$

c) Beweisen Sie, dass Algorithmus 1 den größten gemeinsamen Teiler berechnet.

Aufgabe 2. (Fibonacci-Zahlen)

(3+3+3+3+3 Punkte)

Betrachten Sie die Folge der Fibonacci-Zahlen

$$F_{n+1} = F_n + F_{n-1} \qquad \text{für } n \geq 1$$

mit $F_0 = 0$ und $F_1 = 1$. Beweisen Sie die folgenden Aussagen:

a) $F_n = \frac{a^n - b^n}{\sqrt{5}}$ mit $a = \frac{1+\sqrt{5}}{2}$ und $b = \frac{1-\sqrt{5}}{2}$ für $n \geq 0$.

b) $F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$ für $n \geq 1$.

c) $F_{n+k} = F_k \cdot F_{n+1} + F_{k-1} \cdot F_n$ für $k \geq 1, n \geq 0$.

d) $F_{k \cdot n}$ ist ein Vielfaches von F_n für $k \geq 1$.

e) $\text{ggT}(F_n, F_{n+1}) = 1$ für $n \geq 1$.

Aufgabe 3. (Fehlersuche in Python)

(5 Punkte)

Bei dem Versuch in Python eine Funktion zu schreiben, die die Summe

$$S(n) = \sum_{i=1}^{n} \frac{i}{i+1}$$

berechnet, haben sich 5 Fehler eingeschlichen:

```python
def summe(n):
    ergebnis = 0
    i = 1

    while i < n:
        ergebnis = i//i+1
        i + 1

    return ergebnis
```

Finden Sie die Fehler, erklären Sie was falsch gemacht wurde und geben Sie eine richtige Variante an.

Programmieraufgabe 1. (Modulo-Operator)

(7 Punkte)

Implementieren Sie eine Funktion, die zwei natürliche Zahlen $m, n \in \mathbb{N} \setminus \{0\}$ als Parameter erhält, und die ganzen Zahlen q und r aus der Division mit Rest, das heißt $m = q\,n + r$ mit $r < n$, ausgibt. Nutzen Sie dafür nur Kontrollausdrücke, sowie Addition und Subtraktion (nicht den nativen mod-Operator oder Ganzzahldivision). Stellen Sie durch eine entsprechende Logik sicher, dass nur positive ganze Zahlen verarbeitet werden.

Programmieraufgabe 2. (Polynome)

(5+4+4 Punkte)

In dieser Aufgabe sollen einige Funktionen implementiert werden, die grundlegende Funktionalitäten für Polynome bereitstellen. Um ein Polynom

$$p(x) = p_n x^n + p_{n-1} x^{n-1} + ... + p_1 x + p_0$$

von Grad n in Python zu speichern, sollen die $n+1$ Koeffizienten des Polynoms in einer Liste gespeichert werden, d.h.

```
polynom = [p_0, p_1, ..., p_n].
```

a) Schreiben Sie eine Funktion, die eine Liste von Koeffizienten erhält und das dazugehörige Polynom auf dem Terminal ausgibt. Beipsielsweise könnte die Ausgabe für das Polynom $p(x) = 3x^4 + 4x^2 - x + 7$ so aussehen:

```
3x^4 + 4x^2 - x + 7
```

b) Schreiben Sie eine Funktion `poly_ableitung(koeffizienten)`, die als Parameter eine Liste mit den Koeffizienten erhält und eine Liste mit den Koeffizienten des abgeleiteten Polynoms zurückgibt. Testen Sie Ihre Funktion mit verschieden Polynomen.

c) Schreiben Sie eine Funktion `poly_auswertung(koeffizienten, x)`, die als Parameter die Koeffizientenliste eines Polynoms `p` und eine Auswertungstelle `x` erhält, das Polynom an dieser Stelle auswertet und `p(x)` zurückgibt. Testen Sie auch diese Funktion für verschiede Polynome.