

Informelle Algorithmen

Übung 2

Einführung in die Programmierung


Über dieses Übungsblatt

Auf diesem Übungsblatt geht es darum, das als Übung Algorithmisches Denken auf Alltagssituationen zu übertragen. Sie sollen sich zu gestellten Textproblemen Lösungen überlegen und diese in [Pseudocode](#) aufzuschreiben. Sie brauchen kein Vorwissen außer Mathematik Kenntnisse aus der Schule und gesunden Menschenverstand um die Aufgaben zu lösen. Erlaubte Bausteine sind: `if`, `for` und `while`. Die genaue Syntax ist hier nicht wichtig, achten Sie allerdings auf das **Einrücken** der Codeblöcke. Die Abgabe erfolgt als reine .txt oder .md Datei auf LMS.

Aufgabe Wegbeschreibung

Letzte Änderung: 28. April 2023, 11:34 Uhr

10 Punkte – [im Detail](#)

Ansicht:  | 

Im Folgenden sollen Algorithmen für Menschen zur Ausführung aufgeschrieben werden. Das Format ist hier nicht formal definiert – achten Sie stattdessen darauf, welcher Detailgrad notwendig ist, um für den Adressaten wohldefiniert und explizit zu sein. Adressat in a) ist ein Bewohner unseres Landes, der noch nie im Staudingerweg 9 war. In b) ein sprachkundiges Alien (s.u.).

a) Ihr sitzt in der Mathematik/Informatik Fachschaft im 4. OG. des Staudingerwegs 9 und ihr möchtet einen Eistee trinken. Ihr geht an den Kühlschrank der Fachschaftsküche und überprüft ob dort Eistee steht. Falls dies nicht der Fall ist, bewegt ihr euch in das Erdgeschoss und benutzt den Getränkeautomat dort. Falls kein Eistee verfügbar ist, trinkt ihr eine Cola Zero.



Berücksichtigt dabei auch Auswahlentscheidungen, die ihr trifft, wenn bestimmte Situationen eingetreten sind.

b) Wählt einen „Elementarschritt“ eures Algorithmus aus und skizziert umgangssprachlich, welches Wissen der „Prozessor“ haben muss, der diesen Elementarschritt ausführt. (stellt euch vor ihr müsstet diesen Schritt einem zufällig deutsch sprechenden Alien erklären. Bitte hört nach maximal einer Din A4 Seite auf!)

Aufgabe Socken

Letzte Änderung: 02. May 2022, 11:07 Uhr

15 Punkte – [im Detail](#)

Ansicht:  |  Wer kennt es nicht, man macht die Waschmaschine auf und heraus kommen eine menge Socken die nachher zu Paaren zusammengefügt werden müssen. Um dieses Problem zu lösen hat fast jeder eine eigene Strategie, mehrere Herangehensweisen sind also möglich. Wir versuchen mehrere mögliche Herangehensweisen zu verstehen und in Pseudocode zu formalisieren. Ihre Lösungen müssen hierbei nicht effizient gestaltet sein.

Wir formalisieren die Aufgabenstellung nun etwas. Sei M eine Menge bestehend aus 100 Socken, also $|M| = 100$. Eine Socke ist ein Tupel $(Farbe, Modell)$. Es gibt die Farben **blau, gelb, grün, schwarz, weiß** und von jeder Farbe gibt es genau 20 Socken. Von jedem Modell gibt es genau zwei Socken, also auch keine einzelne Socken (das ist wohl die unrealistischste Annahme hier).

1) Beschreiben Sie einen Algorithmus zum paaren der Socken in Pseudocode, die mit zufälligem Herausnehmen von Socken arbeitet. Terminiert ihr Algorithmus in jedem Fall?

2) Folgender Sockensortieralgorithmus wird tatsächlich von vielen Menschen benutzt:

```
gepaarter_stapel ist leer
ungepaarter_stapel enthält alle Socken

while(ungepaarter_sockenstapel ist nicht leer):
    nehme eine Referenzsocke vom ungepaarter_stapel
    for(alle restlichen Socken in ungepaarter_stapel):
        vergleiche die Socke mit der Referenzsocke
        if(die beiden Socken bilden ein Paar):
            falte beide Socken zusammen und lege sie auf gepaarter_stapel
            break (verlasse die for-schleife)
```

Wir möchten nun die Laufzeit unseres Algorithmus bestimmen; die Einheit die wir dafür verwenden sind Sockenvergleiche, also wieviele Male wurden zwei Socken verglichen. Wieviele Sockenvergleiche finden im **schlimmsten Fall** statt?

3) Wir möchten unseren Algorithmus verbessern und verwenden daher einen sogenannten Pre-Processing Schritt: wir sortieren zunächst die Socken nach Farben vor und verwenden dann den Algorithmus aus 2).

3a) Beschreiben Sie diesen Algorithmus in Pseudocode

3b) Wieviele Sockenvergleiche finden im **schlimmsten Fall** statt? Hat sich in dieser Metrik die Laufzeit verbessert?

Aufgabe Quadratwurzel

Letzte Änderung: 02. May 2022, 11:10 Uhr

15 Punkte — [im Detail](#)

Ansicht:  | 

Zur beliebig genauen Berechnung einer Quadratwurzel durch Iteration kann man folgende Überlegung anstellen: Man kann beispielsweise $\sqrt{2}$ so berechnen, dass man systematisch probiert.

```
Wir suchen also ein x mit  $x^2 = 2$ .
Wir beginnen als Näherung mit einer beliebigen Zahl  $x_1$ , z.B.:  $x_1 = 0,2$ .
Nun berechnen wir eine Zahl h mit  $x_1 * h = 2$ .
Daraus ergeben sich drei mögliche Fälle:
    1. Ist  $x_1$  zufällig schon genau  $\sqrt{2}$ , dann gilt  $x_1 = h = \sqrt{2}$  und wir sind fertig.
    2. Andernfalls muss  $x_1 < \sqrt{2}$  sein, dann liegt  $\sqrt{2}$  zwischen  $x_1$  und h
    3. Oder  $x_1 > \sqrt{2}$ , dann liegt  $\sqrt{2}$  zwischen h und  $x_1$ .
Ist  $x_1$  noch nicht  $\sqrt{2}$ , dann ist der Mittelwert von  $x_1$  und h sicher eine bessere Näherung. Also setzen wir  $x_2 = 1/2 (x_1 + h)$  und testen die neue Näherung genauso wie die alte.
Dies setzen wir solange fort, bis wir ausreichen genau berechnet haben.
```

1) Schreiben Sie einen Algorithmus in Pseudocode, der auf diesen Überlegungen aufbauend die Quadratwurzel einer beliebigen positiven rationalen Zahl auf fünf Nachkommastellen genau berechnet.

2) Führen Sie den Algorithmus für 7 Schritte auf per Hand aus und berechnen Sie den Fehler ihres Ergebnisses $|x_1 - \sqrt{2}|$. Sie sollten in jedem Schritt den Wert aller Variablen (hier x_1 und h) mitschreiben.