Prof. Dr. Stefan Göller
Florian Bruse
Dr. Norbert Hundeshagen

Einführung in die Informatik

WS 2018/2019

Übungsblatt 6
29.11.2018-6.12.2018

Abgabe: Bis zum 6.12. 18:00 Uhr über moodle. Reichen Sie pro Aufgabe, die Sie bearbeitet haben, genau eine Textdatei mit dem Namen `aufgabe_i.py`, bzw. `aufgabe_i.txt`, ein, (wobei i die Aufgabennummer ist) ein, welche die Lösung ihrer Gruppe enthält.

Achtung: Halten Sie sich an die vorgegebenen Dateiformate und Funktionsköpfe. Abweichungen werden mit Punktabzug sanktioniert.

Aufgabe 1 ($\mathcal{O}$-Notation) (6+6+8=20 Punkte):

Achtung: Diese Aufgabe soll als Textdatei `aufgabe_1.txt` abgegeben werden.

Zur Erinnerung: Für zwei Funktionen $f, g\colon \mathbb{N} \to \mathbb{N}$ schreiben wir $f \in \mathcal{O}(g)$, wenn es $c \in \mathbb{R}^+$ und $n_0 \in \mathbb{N}$ gibt so dass für alle $n > n_0$ gilt, dass $f(n) \leq c \cdot g(n)$.

a) Geben Sie für die folgenden Paare von Funktionen jeweils an, ob $f \in \mathcal{O}(g)$ oder ob $g \in \mathcal{O}(f)$. Begründen Sie Ihre Aussage jeweils wie folgt: Um zu zeigen, dass $f \in \mathcal{O}(g)$ geben Sie ein $c \in \mathbb{R}^+$ und ein $n_0 \in \mathbb{N}$ an, so dass $f(n) \leq c \cdot g(n)$ für alle $n > n_0$. Das Kriterium gilt analog für $g \in \mathcal{O}(f)$.

   i) $f(n) = 7n$ und $g(n) = n$
   ii) $f(n) = n^3 + 100n^2$ und $g(n) = n^3$
   iii) $f(n) = 2^{2n}$ und $g(n) = 2^n$

b) Gegeben sind jeweils zwei Funktionen f und g, sowie eine Konstante c. Es gilt jeweils, dass $f \in \mathcal{O}(g)$, und dies lässt sich mit der vorgegebenen Konstante c zeigen. Geben Sie für jede Teilaufgabe ein passendes $n_0 \in \mathbb{N}$ dazu an, also ein n_0, so dass für alle $n > n_0$ gilt, dass $f(n) \leq c \cdot g(n)$.

i) $f(n) = 10n$, $g(n) = n^2 - 10n$, $c = 1$

ii) $f(n) = n \cdot \log(n)$, $g(n) = n^2 - 4n$, $c = 0.05$

iii) $f(n) = 1000n^2$, $g(n) = 2^n$, $c = 0.25$

c) Zeigen Sie für die folgenden zwei Funktionen f und g, dass $f \in \mathcal{O}(g)$ gilt. Zusätzlich zur Angabe der Konstanten c und n_0 führen Sie dazu einen Induktionsbeweis, der zeigt, dass die Ungleichung $f(n) \leq c \cdot g(n)$ für *alle* $n > n_0$ gilt.

$$f(n) = 2^n$$
$$g(n) = 3^n - 100$$

Aufgabe 2 (Wörterbücher) (4*5=20 Punkte):

a) Wir erinnern uns an die Datei `dictionary.txt` vom Übungsblatt 2. Eine Version steht für Sie in moodle bereit. Schreiben Sie eine Methode `lies_ein()`, welche diese Datei einliest und als Liste zurück gibt. Dabei soll jede Zeile ein eigener Eintrag in der Liste sein. Entfernen Sie dabei die Zeilenumbrüche. Zur Lösung dieser Aufgabe sollen Sie *list comprehensions* verwenden.

b) Schreiben Sie dann eine Funktion `buch_1(liste)`, welche eine Liste aus Zeichenketten übergeben bekommt, und ein Wörterbuch zurückgibt, welches als Schlüsselmenge die Wörter der Liste enthält, und ihnen jeweils ihre Länge zuordnet. Nutzen Sie dazu *dictionary comprehensions*.

c) Schreiben Sie dann eine Funktion `buch_2()`, welche ein Wörterbuch zurückgibt, welches die ersten 30 natürlichen Zahlen (inklusive der 0) als Schlüsselmenge hat, und ihnen jeweils die zugehörige Fibonacci-Zahl zuordnet (also der 0 die 1, der 1 die 1, der 2 die 2 usw.). Nutzen Sie *dictionary comprehensions*.

Hinweis: Die Berechnung der Fibonacci-Zahlen wurde in der Vorlesung behandelt. Sie könenn diesen Code übernehmen.

d) Schreiben Sie schließlich eine Funktion `fibonaccilaenge(wort, buch1, buch2)`, welche eine Zeichenkette und zwei Wörterbücher übergeben bekommt. Die Funktion soll nun nach dem Schlüssel mit derm Wert `wort` im ersten Wörterbuch suchen. Falls

ein entsprechender Eintrag enthalten ist, soll der zugeordnete Wert nun als Schlüssel für eine Suche im zweiten Wörterbuch verwendet werden. Der diesem Schlüssel zugeordnete Wert soll dann zurückgegeben werden. Falls einer der beiden Aufrufe im Wörterbuch fehlschlägt, soll `None` zurückgegeben werden.

Beispiel: Falls `buch1` und `buch2` gerade die Wörterbücher aus den vorigen beiden Funktionen sind, und das übergebene Wort `"Banane"` ist, soll die Fibonacci-Zahl mit der Nummer 6 ausgegeben werden, also 13.

Aufgabe 3 (Eindeutigkeit von Sodukus) (20 Punkte):

Wir wollen den Sudokulöser aus der Vorlesung in eine Programm umbauen, welches testet, ob ein partiell ausgefülltes Feld der Größe neun mal neun (etwa ein Sudoku, das noch nicht fertig gelöst ist) ein Sudoku ist, als eindeutig lösbar ist.

Hinweis: Verwenden Sie zur Lösung dieser Aufgabe die bereitgestellte Datei `aufgabe_3.py`. Sie enthält vorgegebenen Code. Bis auf die Funktion `wieviele` dürfen Sie die vorgegebenen Funktionen nicht verändern. Sie dürfen aber neue Funktionen erstellen.

a) Schreiben Sie eine Funktion `wieviele(stand)`, welche einen Spielstand (wie in der Vorlesung beschrieben als Wörterbuch von Paaren $(i, j) \in \{1, \ldots, 9\}^2$ nach $\mathcal{P}(\{1, \ldots, 9\})$) entgegennimmt, und testet, ob der Spielstand keine, eine, oder mehr als eine Lösung unter den Sudokuregeln hat. Entsprechend soll 0, 1, oder 2 zurückgegeben werden. Sie sollen dabei die Funktion `erfuellbar` aus der Vorlesung (in `sudoku.py` enthalten) als Grundlage nehmen. Tatsächlich müssen Sie diese Funktion nur an wenigen Stellen verändern, um die gewünschte Funktion `wieviele` zu erhalten.

   Achtung: Kennzeichnen Sie Zeilen, welche Sie verändert haben, mittels eines Kommentares.

   Hinweis: Da wir nur daran interessiert sind, ob es genau eine Lösung gibt, sollen Sie die Suche nach weiteren Lösungen abbrechen, sobald Sie mindestens zwei gefunden haben.

b) (Sternchenaufgabe) Schreiben Sie eine Funktion `generiere(stand)`, welche wieder einen Spielstand (wie oben) entgegen nimmt. Ihre Funktion soll nun prüfen, ob der Spielstand ein fertig und korrekt gelöstes Sudoku darstellt. Falls nein, soll Ihr Programm eine Fehlermeldung ausgeben und dann `None` zurück geben.

   Ansonsten soll Ihre Funktion nun zufällig eine Zelle im Spielstand auswählen, und die Ziffer dort entfernen, sofern sie nicht schon entfernt ist (in diesem Fall ermitteln Sie zufällig eine neue Zelle, bis sie eine passende gefunden haben). Dazu legen Sie

eine Kopie des Wörterbuchs `stand` an und ändern den entsprechenden Eintrag auf die volle Menge $\{1, \ldots, 9\}$. Danach prüfen Sie, ob der entstandene Spielstand noch eine eindeutige Lösung hat. Falls nein, soll die Änderung verworfen werden und eine neue Zelle zufällig ermittelt werden. Falls ja, soll eine weitere Zelle entfernt werden.

Ihre Funktion stoppt, sobald entweder weniger als 20 Zahlen übrig sind, oder Sie bei 10 Löschversuchen nacheinander auf nicht eindeutige Lösungen gestoßen sind (damit sind *nicht* Treffer auf bereits gelöschte Zellen gemeint). Dann geben Sie den Spielstand zurück.