Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel — Universität Freiburg
Dr. Stefan Wölfl, Thorsten Engesser, Tim Schulte — Institut für Informatik
Wintersemester 2016/2017

Übungsblatt 13

Abgabe: Freitag, 03. Februar 2017, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet13` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann in Dateien in diesem Unterverzeichnis erwartet. Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

`http://gki.informatik.uni-freiburg.de/teaching/ws1617/info1/guide/hinweise.html`

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 13.1 (Tupel vs. Listen; Punkte: 3+3, Dateien: `test_tuple_list.py`)

Tuples are faster than lists. Diese Feststellung findet sich in einigen Online-Dokumentationen zu Python, aber stimmt das? Um diese Frage zu analysieren, interessiert uns:

(a) Ist der Zugriff auf einen Listeneintrag `x = lst[k]` langsamer als der Zugriff auf einen Tupeleintrag `x = tpl[k]`?

(b) Ist die Erzeugung eines Listen-Slices `x = lst[k:n]` langsamer als die Erzeugung eines Tupel-Slices `x = tpl[k:n]`?

Diskutieren Sie obige Fragestellung anhand eigener Testergebnisse (als Kommentar in der Datei `test_tuple_list.py`). Verwenden Sie dazu das `timeit`-Modul und eines der beiden auf der Webseite der Übungen bereitgestellten Templates. Gestalten Sie die Ausgabe so, dass ersichtlich wird, für welche der Fragen (a) oder (b) und welchen Datentyp Ihre Zeitmessungen gelten.

Aufgabe 13.2 (Komplexitätsanalyse; Punkte: 2+3+3+4; Dateien: `komplexitaet.txt`, `search.py`)

Betrachten Sie die beiden Funktionen `search1` und `search2`, die zusammen mit den Hilfsfunktionen `noop` und `fib` in der Datei `search.py` (zu finden auf der Übungswebseite) definiert sind:

```
def noop(elm):
    return elm

def fib(elm):
    return elm if elm <= 1 else fib(elm-1) + fib(elm-2)
```

```python
def search1(elm, lst, fct):
    for elm1 in lst:
        for elm2 in lst:
            if elm == fct(elm1 + elm2):
                return True
    return False

def search2(elm, lst, fct):
    length = len(lst)
    for i in range(length):
        for j in range(i, length):
            if elm == fct(lst[i] + lst[j]):
                return True
    return False
```

(a) Erklären Sie, inwiefern sich die beiden Suchfunktionen prinzipiell unterscheiden. Sind sie semantisch äquivalent (d.h., geben sie bei gleicher Eingabe das Gleiche zurück)?

(b) Schätzen Sie mit Hilfe der Landauschen $\mathcal{O}$-Notation die asymptotische Laufzeit der beiden Implementierungen in Abhängigkeit von der *Länge der Eingabeliste* `lst` ab. Für Ihre Analyse sollen Sie davon ausgehen, dass die Aufrufe der Funktion `fct` konstante Kosten aufweisen.

(c) Im Folgenden wollen wir den Einfluss des Laufzeitverhaltens von `fct` empirisch testen: Benutzen Sie dazu wieder eines der beiden auf der Übungswebseite bereitgestellten Templates. Passen Sie es so an, dass Sie die obigen Funktionen `search1` und `search2` auf etwaige Laufzeitunterschiede testen. Binden Sie dazu beide Hilfsfunktionen `noop` und `fib` nacheinander ein und vergleichen Sie. Diskutieren Sie die Ergebnisse.

(d) Betrachten Sie folgende Tabelle. Geben Sie an, welcher der angegebenen Komplexitätsklassen die Funktionen $a(n), b(n), c(n)$ und $d(n)$ jeweils angehören.

(e)

	$O(1)$	$O(n)$	$O(n\sqrt{n})$	$O(n^3)$	$O(2^n)$
$a(n) = 2^n + n^3$					
$b(n) = 42^{42}$					
$c(n) = (\log n)^3$					
$d(n) = 0.9^{2n}$					

Aufgabe 13.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet13` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Bezug zur Vorlesung, Interessantes, etc.).