Technische Hochschule Ingolstadt, WI
Prof. A. Schiendorfer, L. Lodes, M.Sc.

Software Development (Winter 24/25)

Übungsblatt 05

Bonuspunkte können bis 26.11.2023, 11:00 Uhr geltend gemacht werden.

Aufgabe 1 *(Sets, 3 Punkte)*

Betrachten Sie das Beispiel auf Folie 26 von Foliensatz 4 "Data Structures". Hier wird demonstriert, dass das Suchen von 1000 zufälligen Zahlen in einer Liste von 100 000 Zahlen um mehrere Größenordnungen langsamer ist als in einem gleich großen Set. Erklärt werden kann dies mit der sogenannten Laufzeitkomplexität der darunterliegenden Suchalgorithmen. Während in einer Liste bei Verdopplung der Größe auch der Suchaufwand ca. doppelt so groß wird, steigt der Suchaufwand in Sets nur in ca. log_2-facher Beziehung an. In dieser Aufgabe sollen Sie den dazugehörigen Code selbst schreiben und Experimente damit durchführen. Gehen Sie dazu wie folgt vor:

- Recherchieren Sie, wie Sie mit Hilfe des `random` Moduls eine Liste und ein Set mit einer bestimmten Anzahl von Zufallszahlen *ohne zurücklegen* aus einem festgelegten Intervall ziehen können.
- Recherchieren Sie, wie Sie mit Hilfe des `time` Moduls die Ausführungszeit einiger Code-Zeilen messen können.
- Erstellen Sie sich Listen/Sets mit jeweils 10 000, 100 000, 500 000 und 1 000 000 Zufallszahlen, die *ohne zurücklegen* aus dem Intervall $[0, 10000000]$ gezogen wurden
- Prüfen Sie mit Hilfe des `in` Befehls 1000 mal, ob eine zufällig gezogene Zahl aus dem Intervall $[0, 10000000]$ in der Liste/dem Set enthalten ist. Messen Sie dabei jeweils die benötigte Zeit.

Vergleichen Sie die Ergebnisse der unterschiedlich großen Listen und Sets miteinander. Wie äußern sich die Auswirkungen der beiden Datenstrukturen?

Aufgabe 2 *(Dictionaries, 2 Punkte)*

Betrachten Sie nochmals die Stückliste aus Aufgabe 1b) von Übungsblatt 3. Erweitern Sie die beispielhafte Stückliste um weitere Produkte. Schreiben Sie ein Programm, das Ihnen auf Basis der erweiterten Stückliste jede Artikelnummer auf die Menge und Einheit des jeweiligen Produktes abbildet.

Beispiel: Aus der Liste `parts = [(1, '500-1', 'Hammer', 2, 'Pieces'), (2, '503', 'Screwdriver', 3, 'Pieces'), (3, '555', 'Nail', 10, 'Pieces')]` soll das Dictionary `inventory = {'500-1': (2, 'Pieces'), '503': (3, 'Pieces), '555': (10, 'Pieces')}` erzeugt werden.

Aufgabe 3 *(Zip, 1 Punkte)*

Stellen Sie sich vor, Sie müssen in einem Programm die Vornamen und Nachnamen von Personen verwalten. Dazu finden Sie bereits existierenden Code vor, in dem alle Vor- und Nachnamen in jeweils einer Liste abspeichert (s. Beispiel unten). Erklären Sie kurz, ob dies eine optimale Lösung ist oder ob eine Lösung mit Tupeln möglicherweise sinnvoller ist. Verwenden Sie dann `zip()`, um die beiden Listen in eine Liste aus Tupeln zu überführen.

Beispiel: Aus den Listen `first_names = ['Jane', 'Max', 'Giannis', 'Juan']` und `last_names = ['Doe', 'Mustermann', 'Karamitros', 'Perez']` soll die Liste `names = [('Jane', 'Doe'), ('Max', 'Mustermann'), ('Giannis', 'Karamitros'), ('Juan', 'Perez')]` erzeugt werden.