SS 2014

Prof. Dr. Margarita Esponda

ALP2

2. Übungsblatt (Abgabe: Mi., den 07.05.)

Ziel: Auseinandersetzung mit Rekursion vs. Iteration und erste Analysen der Klassenkomplexität von Algorithmen.

1. Aufgabe (4 Punkte)

Definieren Sie eine Python-Funktion **apply_if**, die als Eingabe eine einstellige Funktion **f**, ein Prädikat **p** und eine Liste **xs** bekommt und die Funktion **f** nur auf die Elemente der Liste, die das Prädikat **p** erfüllen, anwendet.

Anwendungsbeispiel:

>>> apply_if (**ungerade**, **quadrat**, [2, 3, 1, 9, 4, 0, 5])

[9, 1, 81, 25]

2. Aufgabe (8 Punkte)

Betrachten Sie folgende Haskell-Funktion höherer Ordnung:

```
zipWith  f  (x:xs)  (y:ys)  =  (f x y) : (zipWith  f  xs  ys )
zipWith  f  _  _  =  []
```

a) Programmieren Sie eine Python-Funktion, die genau der gleichen Semantik entspricht. D.h. das Ergebnis soll in eine neue Liste zurückgegeben werden.

b) Programmieren Sie eine effiziente Version der Funktion, die aber trotzdem korrekte Ergebnisse liefert.

c) Verwenden Sie das Time-Modul von Python und messen Sie damit die Ausführungszeit beider Lösungen. Beispiele zur Verwendung des Time-Moduls werden in der Vorlesung gezeigt.

3. Aufgabe (5 Punkte)

Schreiben Sie eine Python-Funktion, die die Funktion **randint** des **random**-Moduls von Python benutzt und berechnet, nach wie vielen Zufallszahlen die erste Wiederholung vorkommt.

a) Verwenden Sie dafür eine Dictionary-Datenstruktur.

b) Sie müssen die zwei Argumente für die **randint**-Funktion in Ihre Funktion miteingeben.

4. Aufgabe (7 Punkte)

Programmieren Sie eine rekursive Variante der collatz-Funktion aus dem 1. Übungsblatt.

a) Eine Hilfsfunktion, die bei Eingabe einer Zahl C_i die nächste Zahl C_{i+1} der Collatz-Folge berechnet, soll als lokale Funktion innerhalb Ihrer rekursiven Funktion definiert werden.

b) Welche Vorteile bzw. Nachteile hat die rekursive Lösung gegenüber der Lösung mit Hilfe einer **while**-Schleife?

5. Aufgabe (12 Punkte) Geburtstagsparadox

Die Wahrscheinlichkeit, dass 2 Personen am gleichen Tag Geburtstag haben, kann mit Hilfe eines Zufallsgenerators empirisch approximiert werden.

a) Schreiben Sie eine Funktion **double_birthday**, die zufällige Geburtstage produziert und zählt, nach wie vielen Versuchen eine Wiederholung vorkommt. Verwenden Sie für die Zwischenspeicherung der Daten ein Wörterbuch (Dictionary) als Datenstruktur.

b) Programmieren Sie eine zweite Funktion **repeat_double_birthday**, die das Experiment lange genug wiederholt und mit Hilfe eines Arrays aufzählt, nach wie vielen Versuchen (2, 3, 4,...,bis max. 365) eine Wiederholung vorkommt.

c) Mit der Ergebnisliste aus b) können Sie dann eine Funktion **birthday_paradox** programmieren, die die Wahrscheinlichkeit, dass auf einer Party mit **n** Gästen zwei Personen am gleichen Tag Geburtstag haben, empirisch berechnet.