Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser, Tim Schulte
Wintersemester 2016/2017

Universität Freiburg
Institut für Informatik

Übungsblatt 6

Abgabe: Freitag, 02. Dezember 2016, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet06` 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 6.1 (Mengen-Operationen; Punkte: 5+4; Datei: `sets.py`)

In dieser Aufgabe geht es um die Berechnung von Teilmengen und Potenzmengen. Die Teilaufgaben dürfen in beliebiger Reihenfolge bearbeitet werden, wobei die zuerst definierte Funktion in der Definition der anderen verwendet werden darf. Schreiben Sie für die beiden Funktionen in (a) und (b) jeweils vier Testfunktionen (`test_subsets_xyz()` und `test_powerset_xyz()`, `xyz` ist dabei durch Namen Ihrer Wahl zu ersetzen).

(a) Definieren Sie eine Funktion `subsets(s, k)`, die angewendet auf eine Menge `s` und eine natürliche Zahl `k` ≥ 0, die Menge gerade jener Teilmengen von `s` zurückgibt, die genau `k` Elemente enthalten.

(b) Definieren Sie eine Funktion `powerset(s)`, die für eine beliebige Menge `s` die Potenzmenge berechnet und zurückgibt.

Hinweis: In dieser Aufgabe sollen ausschließlich Mengen (keine Listen!) benutzt werden. Das Importieren von anderen Modulen ist bei der Bearbeitung beider Teilaufgaben nicht erlaubt. Achten Sie ferner darauf, dass die an diese Funktionen übergebenen Mengen nach Ausführung der Funktion noch dieselben Mengen sind! Denken Sie an die Dokumentation ihrer Funktionen mittels docstrings.

Aufgabe 6.2 (PageRank; Punkte: 5+4, Datei: `pagerank.py`)

Mit dem von den Google-Gründern Larry Page und Sergey Brin entwickelten PageRank-Algorithmus sollen sich Webseiten nach ihrer Relevanz ordnen lassen, wobei diese anhand der Verlinkungen abgeschätzt wird. Dabei werden den Webseiten Gewichte zugeordnet und daraus das Ranking bestimmt. Das Gesamtranking besteht aus den Gewichten aller Webseiten im Netzwerk.
Wir repräsentieren ein solches Ranking als `dict`, das jeder Webseite ein Gewicht (vom Typ `float`) zwischen 0 und 1 zuordnet. Die Gewichte sollen dabei so normalisiert sein, dass sie aufsummiert (im Rahmen der Fließkomma-Ungenauigkeit) 1 ergeben.
Dies ermöglicht uns eine Interpretation des Rankings anhand des sogenannten *Zufallssurfer-Modells*. Hierbei entspricht das einer Webseite zugeordnete Gewicht der Wahrscheinlichkeit, dass eine zufällig surfende Person sich gerade auf der Webseite befindet.
Ein Ranking für ein Netzwerk bestehend aus den Seiten `spam.com`, `egg.org` und `ham.de` könnte zum Beispiel folgendermaßen aussehen:

```
>>> ranking = {'spam.com': 0.2,
...            'egg.org': 0.5,
...            'ham.de': 0.3}
```

Die Struktur des Netzwerkes soll als `dict` repräsentiert werden. Dieses ordnet jeder Webseite ein `set` mit den von der Webseite verlinkten Seiten zu. Zum Beispiel:

```
>>> network = {'spam.com': {'egg.org'},
...            'egg.org': set(),
...            'ham.de': {'spam.com', 'egg.org'}}
```

(a) Implementieren Sie zunächst die Funktion `nextranking(ranking, network)`, die für ein übergebenes Ranking `ranking` und für ein übergebenes Netzwerk `network` das Nachfolgeranking berechnet. Dabei wird folgende Annahme getroffen: Von jeder Webseite aus wird mit 85%iger Wahrscheinlichkeit zu einer beliebigen verlinkten Webseite gewechselt. Andernfalls wird zu einer beliebigen Seite gewechselt, diese muss nicht verlinkt sein und darf auch die ursprüngliche Seite sein. Sei n die Anzahl der von einer Webseite i verlinkten Seiten und m die Anzahl aller Seiten im Netzwerk. Dann ist die Wahrscheinlichkeit p_{ij} von Webseite i auf eine Webseite j zu wechseln $\frac{0.85}{n} + \frac{0.15}{m}$, falls j von i verlinkt ist, und $\frac{0.15}{m}$ andernfalls. Für den Spezialfall, in dem Seite i auf keine andere Seite verlinkt, wird mit uniformer Wahrscheinlichkeit $p_{ij} = \frac{1}{m}$ auf eine beliebige Seite j gewechselt. Die Wahrscheinlichkeit, von Seite i zu starten und auf Seite j zu wechseln, ist $x_{ij} =$ `ranking[i]` $* p_{ij}$. Die Wahrscheinlichkeit, von einer beliebigen Seite zu starten und auf Seite j zu wechseln, ist die Summe der x_{ij} über alle Webseiten i. Diese Wahrscheinlichkeit entspricht dem neuen Gewicht von j.

```
>>> from pprint import pprint
>>> pprint(nextranking(ranking, network))
{'egg.org': 0.48916666666666675,
 'ham.de': 0.19166666666666665,
 'spam.com': 0.3191666666666667}
```

(b) Implementieren Sie die Funktion `ranking(network)`, die für alle Seiten des Netzwerks zunächst ein uniformes Ranking erstellt, das allen Seiten das gleiche Gewicht zuweist. Danach soll das Ranking sukzessive verfeinert werden, indem es jeweils durch das mittels `nextranking` erzeugte Nachfolgerranking ersetzt wird. Dieser Vorgang wird wiederholt, solange sich mindestens ein Gewicht des Rankings um mehr als `1e-6` vom vorherigen Gewicht unterscheidet. Danach soll das „konvergierte“ Rating zurückgegeben werden.

```
>>> pprint(ranking(network))
{'egg.org': 0.5208692975273158,
 'ham.de': 0.19757959373228617,
 'spam.com': 0.2815511087403979}
```

Aufgabe 6.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet06` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).