Informatik I WS 2005/2006

Übungsblatt 6

Abgabe: 12.12.05 - 14.12.05

Aufgabe 1 (Dictionaries, Strings, Listen, Umgang mit Dateien) Punkte: 6

Die Weihnachtszeit ist die Zeit der Geschenke. Nun gibt es gute Geschenke, schlechte Geschenke und Bücher. Von der letzten Sorte bekommt man meist mehr, als man lesen kann oder will. Peinlich wird es aber, wenn der Schenkende dann am Telefon nachfragt: „Du hast doch sicher schon das Buch „Kritik der reinen Vernunft“ gelesen, das ich dir letzte Weihnachten geschenkt habe und über das du dich so sehr gefreut hast. Was hältst du denn eigentlich von Kants Meinung zu Leibnitz?“.
Der Nichtinformatiker müsste an dieser Stelle verschämt zugeben, das Buch nie gelesen zu haben, während der Informatiker lässig seinen Python-Interpreter anwirft und sich einen Index des Buches erstellen lässt. Darin kann er dann blitzschnell nachschlagen, an welchen Stellen des Buches sich Kant über Leibnitz äussert und somit weiterhin als belesen gelten.
In dieser Aufgabe soll ein Programm zum Indizieren eines Textes erstellt werden. Das Programm erhält eine Textdatei per Kommandozeile. Dann soll für jedes Wort des Textes gespeichert werden, in welcher Zeile es vorkommt. Beispiel:

('ab', [2101, 4059, 5933, 6451, 10647, 10835, 12415, 15038, 16016, 17878, 19758])
('abaenderung', [6454])
('abaenderungen', [1022, 19833])
('abbruch', [5149, 7715, 14154, 14254, 14455, 15045, 17938, 18978, 19866, 19879, 21562, 21653])
...
('zwischenzustandes', [7037])
('zwist', [12564, 13431])
('zwoelf', [1509, 1513])

Verwenden Sie dazu ein Dictionary. Das zeilenweise Einlesen einer Datei und das Aufspalten eines Strings in einzelne Wörter wurde bereits in der Vorlesung beschrieben. Leider bleiben nach dem Aufspalten mit der split()-Methode die Satzzeichen erhalten. So wird die Zeile *“machen soll, einhellig zu machen: so!“* in die Teilstrings *“machen“* , *“soll,“* , *“einhellig“* , *“zu“* , *“machen:“* und *“so!“* aufgespalten. Beim einfachen Einfügen in den Index führt das dazu, dass *“machen“* und *“machen:“* als zwei unterschiedliche Schlüssel gespeichert würden. Dies sollen Sie vermeiden.
Schreiben Sie dazu eine Funktion, die einen String als Input erhält und auch einen String als return-Wert hat. Im zurückgegebenen String sollen die Sonderzeichen des Input-Strings durch Leerzeichen ersetzt werden. Beispiel: Der Rückgabewert von *“soll, einhellig zu machen: so!“* ist demnach *“soll einhellig zu machen so “*.
Die Ausgabe des Index soll wie im Beispiel oben, in alphabetischer Reihenfolge erfolgen. Um die Sortierung zu vereinfachen, wandeln Sie alle Buchstaben in Kleinbuchstaben um. Zum Testen des Programms steht die Datei kant.txt auf der Vorlesungs-Homepage bereit.
Tips: Verwenden Sie die nützlichen Methoden *isalpha()* und *isspace()* für einzelne ASCII-Zeichen, sowie die Stringmethoden *replace(old, new)* sowie *lower()*. Außerdem benötigen Sie noch die Listen-Methode *sort()* und die Methode *items()* für Dictionaries. Die genauen Funktionsweisen dieser Methoden werden in der *Python Library Reference* auf *www.python.org* erklärt.

Aufgabe 2 (Listen, Binärdarstellung von Zahlen) Punkte: 9

Das *perfekte Mischen (oder auch „Perfect Shuffle“)* eines Kartendecks funktioniert folgendermassen: Man teilt das Kartendeck in zwei gleich grosse Stapel auf und fügt sie dann wieder zusammen, indem man von jedem Stapel abwechselnd eine Karte nimmt. Versierte Pokerspieler machen dies normalerweise in Sekundenschnelle mit den Daumen. Dabei kann man zwei Arten des Mischens unterscheiden, je nachdem, ob man beim Zusammenfügen die erste Karte vom oberen oder vom unteren Teilstapel nimmt:

- Beim „Perfect-In-Shuffle“ nimmt man die erste Karte vom unteren Stapel: Also wird der ursprüngliche Stapel A B C D E F G H erst aufgeteilt in A B C D und E F G H und dann zu E A F B G C H D zusammengefügt.
- Beim „Perfect-Out-Shuffle“ nimmt man die erste Karte vom oberen Stapel: Der ursprüngliche Stapel A B C D E F G H wird wieder zuerst in die Teilstapel A B C D und E F G H aufgeteilt und dann zu A E B F C G D H zusammengefügt.

(Anmerkung: Perfektes Mischen ist eine wichtige Technik bei vielen parallelen Algorithmen in der Signalverarbeitung, wie z.B. bei der schnellen Fourier-Transformation).

a) Schreiben Sie ein Programm, das eine natürliche Zahl N von der Kommandozeile einliest und anschliessend ein Kartendeck mit 52 Karten N mal nach der „Perfect-Out-Shuffle“-Methode mischt. Das Kartendeck kann einfach durch eine Liste mit den Zahlen 0-51 repräsentiert werden.

b) Schreiben Sie ein Programm, das berechnet, wie oft man ein Kartendeck nach „Perfect-In-Shuffle“ bzw „Perfect-Out-Shuffle“-Methode mischen muss, bis die Karten wieder die ursprüngliche Reihenfolge haben.

c) Einige Kartentricks basieren auf dem perfekten Mischen, denn es ist möglich, eine Karte durch perfektes Mischen an jede beliebige Stelle eines Kartenstapels zu befördern.
Schreiben Sie ein Programm, das eine natürliche Zahl N (zwischen 0 und 51) von der Kommandozeile einliest und 8 mal perfekt mischt, so dass die oberste Karte anschliessend an Position N im Stapel steht.
Tip: Betrachten sie die Binärdarstellung von N (von links nach rechts). Wenn ein Bit den Wert 0 hat, führen Sie ein „Perfect-Out-Shuffle“ durch, wenn ein Bit den Wert 1 hat, ein „Perfect-In-Shuffle“.

Aufgabe 3 (Komplexe Zahlen, Umgang mit externen Modulen) Punkte: 5

In der Vorlesung haben Sie bereits die Mandelbrotmenge kennengelernt. Die Mandelbrotmenge wurde dort definiert als: M=$\{c \in \mathbb{C}$: Folge $(z_i)_i$ bleibt beschränkt$\}$ mit $z_0 = 0$ und $z_{i+1} = z_i^2 + c$.
In dieser Aufgabe sollen Sie die Mandelbrotmenge berechnen und als Bild mit Hilfe der „Python Imaging Library“ ausgeben. Gehen Sie dabei, wie in der Vorlesung beschrieben vor:
Für jeden Punkt c wird die Folge $(z_i)_n$ bis n = 256 berechnet. Ist $|z_i| > 2$ für ein $i < 256$, so gehört c nicht zur Mandelbrotmenge und der Pixel erhält eine Farbe, die abhängig von der Zahl i gewählt werden soll, ist $|z_{256}| \leq 2$, so gehört c wahrscheinlich zur Menge und der entsprechende Pixel wird schwarz gefärbt.
Als Punktemenge wählen Sie den Bereich (-1.259, -1.277)x(0.047, 0.065) auf der komplexen Ebene (Das ist die sogenannte „Seahorse-Region“). Die Bildgröße soll dem Programm in Form von zwei Zahlen für die Höhe und Breite des Bildes gemessen in Pixeln per Kommandozeile übergeben werden.