Stefan Kurtz **Universität Hamburg**
Emanuel Ehmki **Zentrum für Bioinformatik**

Programmierung in der Bioinformatik
Wintersemester 2019/2020
Übungen zur Vorlesung: Ausgabe am 27.11.2019

...und immer wieder am Mittwochvergügbar:
`https://feedback.informatik.uni-hamburg.de/PfN1/wise2019-2020`

Aufgabe 6.1 (2 Punkte) In Textverarbeitungssystemen und auf Webseiten wird Text in Paragraphen häufig dynamisch umgebrochen. Wenn man mit Paste/Copy aus solchen Kontexten Text kopiert und in den Editor einfügt, entstehen meist sehr lange Zeilen, die jeweils den Inhalt eines Paragraphen darstellen. Ein Beispiel finden Sie in der Datei `python_history.txt` in den Materialien zu dieser Aufgabe.

Entwickeln Sie ein Python-Programm `fold_text.py`, das einen Text zeilenweise einliest und die Zeilen durch Einfügen des newline-Zeichens \n aufbricht und ausgibt. Beim Aufruf des Programms ist das erste Argument die maximale Anzahl der Zeichen pro Zeile in der Ausgabe und das zweite Argument ist der Dateiname. Beispiel: Durch den Aufruf

```
./fold_text.py 70 python_history.txt
```

wird der Inhalt der Datei `python_history_fold70.txt` ausgegeben. In dieser Datei haben alle Zeilen eine maximale Länge von 70 Zeichen.

Damit Sie sich auf das Wesentliche konzentrieren können, finden Sie in den Materialien eine Datei `fold_text_template.py`. Benennen Sie die Datei in `fold_text.py` um. Der vorhandene Programmcode dient zur Verifikation der Werte in `sys.argv` und zum Öffnen einer Datei. Nutzen Sie die Variable `stream` zum zeilenweisen Einlesen des Inhalts der geöffneten Datei.

In den Materialien finden Sie ein Makefile. Durch `make test` verifizieren Sie die Korrektheit Ihres Programms.

Aufgabe 6.2 (4 Punkte) Die *Root Mean Square*-Distanz (RMSD) wird häufig als Abstandsmaß für drei-dimensionale Strukturen, z.B. von Proteinen, verwendet. Für zwei Vektoren $\overrightarrow{v} = (v_0, v_1, \ldots, v_{n-1})$ und $\overrightarrow{w} = (w_0, w_1, \ldots, w_{n-1})$ mit jeweils n Punkten ist die RMSD wie folgt definiert:

$$\text{rmsd}(\overrightarrow{v}, \overrightarrow{w}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(v_i.x - w_i.x)^2 + (v_i.y - w_i.y)^2 + (v_i.z - w_i.z)^2}$$

Dabei steht $.x$ für die x-Koordinate eines Punktes. Entsprechendes gilt für $.y$ und $.z$.

In Python kann man einen Punkt durch eine Liste mit 3 numerischen Werten (in unserem Fall Fließkommawerten) repräsentieren, so dass der erste Wert die x-Koordinate, der zweite Wert die y-Koordinate und der dritte Wert die z-Koordinate ist. Ein Vektor ist dann eine Liste von Punkten, d.h. eine Liste von Listen jeweils der Länge 3. In der Datei `rmsd_input.csv` finden Sie in jeder

Zeile einen Vektor, der in dieser Form als gültiger Python-Ausdruck angegeben ist. In der Vorlesung wurde gezeigt, dass man einen String, der einen gültigen Python-Ausdruck darstellt, durch die Funktion `eval` in den entsprechenden Ausdruck konvertieren kann.

Sie sollen nun ein Python-Skript `rmsd.py` entwickeln, das eine auf der Kommandozeile angegebene Datei mit Vektoren in der obigen Notation einliest und jeweils den RMSD berechnet. Genauer gesagt, soll der RMSD für die beiden Vektoren in Zeile 1 und 2, für die beiden Vektoren in Zeile 3 und 4, etc. bestimmt werden (siehe auch `rmsd_out.solution.txt`).

Beispiel: Hier sind die beiden ersten Zeilen aus `rmsd_input.csv`:

```
[[4.0,5.0,6.0],[8.0,8.0,8.0]]
[[1.0,2.0,3.0],[6.0,6.0,6.0]]
```

Damit ist $\overrightarrow{v} = (v_0, v_1)$ mit $v_0 = [4.0, 5.0, 6.0]$ und $v_1 = [8.0, 8.0, 8.0]$ sowie $\overrightarrow{w} = (w_0, w_1)$ mit $w_0 = [1.0, 2.0, 3.0]$ und $w_1 = [6.0, 6.0, 6.0]$. Damit ergibt sich die folgende RMSD-Berechnung:

$$\text{rmsd}(\overrightarrow{v}, \overrightarrow{w}) = \sqrt{\frac{1}{2}\left((4.0-1.0)^2 + (5.0-2.0)^2 + (6.0-3.0)^2 + 3 \cdot (8.0-6.0)^2\right)} = 4.41588$$

Die einzelnen Schritte Ihrer Lösung sollen durch die folgenden Funktionen implementiert werden:

- Die Funktion `listofvectors_read(filename)` erhält als Parameter den Namen der Datei mit den Vektoren. Diese Funktion öffnet die Datei, liest sie zeilenweise ein, konvertiert mit `eval` jede einzelne Zeile in einen Vektor (wie oben beschrieben) und speichert die Vektoren in einer Liste, die durch eine `return`-Anweisung zurückgeliefert wird.
- Eine Funktion `rmsd_evaluate(v_vector, w_vector)`, die zwei Vektoren als Parameter erhält und hierfür den RMSD-Wert berechnet. Benutzen Sie `assert`, um zu verifizieren, dass `v_vector` und `w_vector` die gleiche Länge haben, und dass die einzelnen Punkte, aus denen die Vektoren bestehen, Listen mit genau 3 Elementen sind.
- Eine Funktion `listofvectors_rmsd_print(listofvectors)`, die für die Paare von Vektoren aus `listofvectors`, wie oben definiert, den RMSD-Wert berechnet und als Zeile der Form

  `rmsd(vector<i>,vector<i+1>)=<rmsd-wert>`

  ausgibt. Dabei sind die durch die spitzen Klammern gekennzeichneten Werte Platzhalter für konkrete Werte. Der RMSD-Wert wird mit 5 Nachkommastellen ausgegeben.

Im Material finden Sie die genannte Eingabedatei `rmsd_input.csv`, eine Datei `rmsd_out.solution.txt` mit der erwarteten Ausgabe sowie ein Makefile. Durch `make test` verifizieren Sie die Korrektheit Ihres Programms. Wenn das nicht auf Anhieb klappt, testen Sie zunächst, ob Ihr Programm für die ersten beiden Vektoren funktioniert. Falls nicht, geben Sie Zwischenergebnisse aus und verifizieren schrittweise die Berechnung.

Aufgabe 6.3 (4 Punkte) Wir betrachten ein Alphabet der Größe r mit den Zeichen $a_1, a_2, \ldots, a_r$. Für alle i, $0 \leq i \leq r$ ist der i-te Skyline-String s_i wie folgt definiert:

$$s_i = \begin{cases} \varepsilon & \text{falls } i = 0 \\ s_{i-1} a_i s_{i-1} & \text{falls } 1 \leq i \leq r \end{cases}$$

Beispiel: Für das Alphabet $\{a, b, c, d\}$ ergeben sich die folgenden Skyline-Strings $s_1 = a$, $s_2 = aba$, $s_3 = abacaba$ und $s_4 = abacabadabacaba$.

Hintergrund: Für bestimmte Algorithmen zur Sortierung von Strings bilden Skyline-Strings die Eingabe, die zur maximalen Laufzeit führt. Dieser Aspekt spielt in dieser Aufgabe jedoch keine Rolle.

Hier sind die Teilaufgaben, die Sie lösen sollen:

- Geben Sie an, welche Länge s_i hat. Sie sollen hier keine konkreten Zahlen angeben, sondern einen mathematischen Ausdruck, durch den, abhängig von i, die Länge von s_i bestimmt werden kann. Wenn Sie noch keine Idee haben, wie der Ausdruck gebildet werden kann, können Sie erst den zweiten Teil der Aufgabe lösen und sich die Längen der s_i ausgeben lassen.
- Wir betrachten das Alphabet $\{a, b, c, \ldots, z\}$ von 26 Kleinbuchstaben, d.h. $r = 26$. Schreiben Sie ein Python-Skript `skyline.py`, das für dieses Alphabet nacheinander die Skyline-Strings der Längen i, $1 \leq i \leq 9$ zeilenweise ausgibt. D.h. die i-te Zeile enthält s_i. Das i-te Zeichen des obigen Alphabets können Sie mit Hilfe der Methoden `ord` und `chr` berechnen. Im Material zu dieser Übung finden Sie die erwartete Ausgabe Ihres Programms sowie ein Makefile. Durch `make test` können Sie verifizieren, ob Ihr Programm korrekt funktioniert.

Bitte die Lösungen zu diesen Aufgaben bis zum 02.12.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 04.12.2019.