**Arbeitsgruppe für Genominformatik** **Universität Hamburg**
**Stefan Kurtz** **Zentrum für Bioinformatik**

# Genominformatik
# Sommersemester 2017
# Übungen zur Vorlesung: Ausgabe am 09.05.2017

Punkteverteilung: Aufgabe 3.1: 5 Punkte, Aufgabe 3.2: 4 Punkte. Aufgabe 3.3: 4 Punkte.

Abgabe bis zum 18.05.2017, 23:59 Uhr.

**Aufgabe 3.1** In der Vorlesung wurde ein Algorithmus beschrieben, der für eine gegebene RNA-Sequenz die minimale freie Energie liefert, unter der Annahme, dass Basen-Paarungen unabhängig voneinander sind. Dieser Algorithmus wird auch „Nussinov Algorithmus“ genannt. Schreiben Sie in C, Ruby oder Python ein Programm `nussinov`, das eine RNA-Sequenz von der Kommandozeile liest und darauf den Nussinov-Algorithmus anwendet. Es soll eine RNA-Sekundärstruktur mit minimaler freier Energie (mfe) bestimmt und ausgegeben werden. Als Nebenbedingung soll gelten, dass Hairpin loops mindestens drei ungepaarte Basen enthalten müssen, d.h. $\ell_{\min} = 3$. Die Basenpaare sollen durch die Energiefunktion $\alpha$ unterschiedlich bewertet werden: $\alpha(\mathtt{G},\mathtt{C}) = \alpha(\mathtt{C},\mathtt{G}) = -3$, $\alpha(\mathtt{A},\mathtt{U}) = \alpha(\mathtt{U},\mathtt{A}) = -2$, $\alpha(\mathtt{G},\mathtt{U}) = \alpha(\mathtt{U},\mathtt{G}) = -1$ und $\alpha(x,y) = \infty$ für alle anderen Basenkombinationen $x$ und $y$. Berechnen Sie dazu die Matrix $E$ und wenden den Traceback-Algorithmus, wie in der Vorlesung beschrieben, an. In den Materialien zur Übung finden Sie eine Datei `phenyalanines.txt`, die drei ähnliche Sequenzen enthält, und zwar in den Zeilen, die nicht mit # oder > beginnen. Ihr Programm soll exakt das Ergebnis in der xml-Datei `fold-phenyalanines.xml` reproduzieren. Das können Sie durch `make test-nussinov` überprüfen.

**Aufgabe 3.2** Ein Punkt-Klammer-String (auch Vienna-Notation genannt) ist eine einfache Repräsentation der Sekundärstruktur einer RNA-Sequenz. Die Vienna-Notation besteht aus den drei Zeichen `.`, `(` und `)`. Zu jeder geöffneten Klammer `(` gibt es eine dazugehörige geschlossene Klammer `)`. Jedes Klammerpaar steht für ein Paar von gepaarten Positionen. Das Zeichen `.` repräsentiert eine ungepaarte Basenposition. Als Beispiel repräsentiert der Vienna-Notation-String

```
.((.((...)).((...)).))
```

die RNA-Sekundärstruktur

$$\{(2,22),(3,21),(5,11),(6,10),(13,19),(14,18)\}$$

für eine Sequenz der Länge 22. Implementieren Sie nun zwei Programme `vienna2pairlist` und `pairlist2vienna`. Das erste Programm liefert für eine Vienna-Notation die Länge der repräsentierten Sequenz und die geordnete Folge der Basenpaar-Positionen. Das zweite Programm liefert für eine Sequenzlänge und eine geordnete Folge von Basenpaar-Positionen

die entsprechende Vienna-Notation. Testen Sie Ihre Implementierung, in dem Sie für alle Vienna-Notationen aus der Datei `Vienna-examples.txt` (siehe Material) die entsprechende Folge von Basenpaar-Positionen und die Länge der RNA-Sequenz berechnen, um daraus dann die Vienna-Notation zu bestimmen. Diese muss mit dem Original übereinstimmen.

**Aufgabe 3.3** Sei $S(n)$ die Anzahl der potentiell möglichen RNA-Sekundärstrukturen einer Sequenz der Länge $n \geq 1$. Dabei ist keine konkrete Sequenz gemeint, sondern $S(n)$ soll für alle Sequenzen der Länge $n$ gelten. Die minimale Länge eines Hairpin-Loops soll $1$ sein. Entwickeln Sie eine Rekurrenz für $S(n)$ und begründen Sie nachvollziehbar, warum die Rekurrenz korrekt ist. Implementieren Sie diese Rekurrenz durch eine rekursive Funktion (DP-Verfahren ist nicht notwendig). Ihr Programm soll `rna_count` heissen und für den einzigen Parameter $n$ die Anzahl der Strukturen der Länge $\ell$, für $1 \leq \ell \leq n$ ausgeben. Ihr Programm sollte für $n = 16$ das gleiche Ergebnis liefern, das Sie auch in der Datei `numstructs.tsv` (siehe Materialien) finden. Sie können das durch den Test `make test-numstructs` überprüfen.

**Die Lösungen zu diesen Aufgaben werden am 23.05.2017 besprochen.**