Abteilung für Genominformatik **Universität Hamburg**
S. Kurtz, F. Meyer und S. Steinbiß **Zentrum für Bioinformatik**

Grundlagen der Sequenzanalyse
Wintersemester 2012/2013
Übungen zur Vorlesung: Ausgabe am 20.11.2012

Punkteverteilung: Aufgabe 6.1: 2 Punkte, Aufgabe 6.2: 4 Punkte, Aufgabe 6.3: 4 Punkte

Abgabe bis zum 26.11.2012.

Aufgabe 6.1 Der bekannte DP-Algorithmus zum globalen Alignment kann leicht wie folgt modifiziert werden:

1. statt Kosten werden Scores verwendet, die auch negativ sein können.
2. statt zu minimieren wird in der Rekurrenz maximiert.

Der so entstandene Algorithmus wird auch als Needleman-Wunsch-Algorithmus bezeichnet (NW).

Sei nun folgende Scorefunktion σ gegeben:

$$\sigma(a \to b) = \begin{cases} 2 & \text{if} \quad a, b \in \mathcal{A}, a = b \\ -4 & \text{if} \quad a, b \in \mathcal{A}, a \neq b \\ -5 & \text{if} \quad a = \epsilon \text{ oder } b = \epsilon \end{cases}$$

Ein Match wird hier also mit Score 2 bewertet, ein Mismatch mit -4 und ein Indel mit -5.

Betrachten Sie nun die folgenden Ausschnitte aus Matrizen optimaler Alignment-Scores, die beim Smith-Waterman (SW), bzw. NW-Algorithmus gefüllt werden:

1)

		$v[j]$	
		$\vdots$	
		2	4
$u[i]$	$\ldots$	0	

2)

		$v[j]$	
		$\vdots$	
		0	8
$u[i]$	$\ldots$	2	

3)

		$v[j]$	
		$\vdots$	
		2	5
$u[i]$	$\ldots$	4	

Welche Scores erhalten die leeren Felder in den folgenden vier Fällen?

(a) $u[i] \neq v[j]$, NW

(b) $u[i] = v[j]$, NW

(c) $u[i] \neq v[j]$, SW

(d) $u[i] = v[j]$, SW

Aufgabe 6.2 Implementieren Sie eine Funktion, die eine Scorematrix aus einer Datei einliest und in einer Datenstruktur speichert, die sich dafür eignet, den Score für eine Ersetzungsoperation von zwei Aminosäuren auf einfache Weise zu extrahieren. Bitte vermeiden Sie es, bei jedem Zugriff auf einen Score-Wert die Datei neu einzulesen! Versuchen Sie außerdem, den Zugriff auf die Datenstruktur von ihrer tatsächlichen Speicherung zu entkoppeln, z.B. durch eine Zugriffsfunktion.

Verwenden Sie zum Testen Ihrer Implementierung die folgende Scorematrix für Proteinsequenzen. Sie finden die komplette Matrix in der Textdatei `/projects/lehre/gsa/BLOSUM62` bzw. in STiNE.

```
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -10 -2 -3
D -2 -2  1  6 -3  0  2 -1 -1 -3 -10 -1 -3 -3 -1  0 -1 -10 -3 -3
...
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4
```

Versuchen Sie so gut wie möglich, alle Fehler abzufangen, die durch Scorematrizen entstehen können, welche nicht dem obigen Format entsprechen.

Aufgabe 6.3 Ein wichtiges Problem in der Massenspektrometrie besteht darin, alle Proteinsequenzen in einer Datenbank zu finden, die ein bestimmtes Molekulargewicht haben. Formal lässt sich dieses Problem wie folgt formulieren:

Sei $\mathcal{A}$ ein Alphabet und $\sigma : \mathcal{A} \to \mathbb{N} \setminus \{0\}$ eine Gewichtsfunktion, die jedem Symbol aus $\mathcal{A}$ eine positive natürliche Zahl zuordnet. Für alle Sequenzen $u \in \mathcal{A}^*$ definieren wir das *Gewicht* $\sigma(u) = \sum_{i=1}^{|u|} \sigma(u[i])$.

Sei $S \in \mathcal{A}^n$ eine Sequenz der Länge n und $w > 0$ ein vorgegebenes Gewicht. Das *gewichtete Substring-Matching Problem* besteht darin, die folgende Menge zu bestimmen:

$$Sol(\sigma, S, w) = \{(j, l) \mid 1 \leq j \leq n, 1 \leq l \leq n + 1 - j, \sigma\left(S[j \ldots j + l - 1]\right) = w\}$$

d.h. die Menge aller Substrings in S (repräsentiert durch eine Startposition j und die Länge l), deren Gewicht genau w ist.

Beispiel: Sei $\mathcal{A} = \{a, c, g, t\}$ das DNA-Alphabet, sei $\sigma(a) = 1$, $\sigma(c) = \sigma(t) = 2$ sowie $\sigma(g) = 3$. Für $S = acgtagctc$ ist dann

$$Sol(\sigma, S, 9) = \{(1, 5), (3, 4), (6, 4)\}$$

Diese Menge repräsentiert die Substrings $acgta$, $gtag$ und $gctc$ und es gilt $\sigma(acgta) = \sigma(gtag) = \sigma(gctc) = 9$.

Entwickeln und formulieren Sie einen möglichst effizienten Algorithmus in Pseudocode oder einer Programmiersprache, um das gewichtete Substring-Matching Problem zu lösen. Dabei sollten Sie ausnutzen, dass die Gewichte immer positive ganze Zahlen sind. Welche Laufzeit hat Ihr Algorithmus?

Die Lösungen zu diesen Aufgaben werden am 27.11.2012 besprochen.