Abteilung für Genominformatik
S. Kurtz, F. Meyer und S. Steinbiß

Universität Hamburg
Zentrum für Bioinformatik

Grundlagen der Sequenzanalyse
Winter
Übungen zur Vorlesung: Ausgabe am 22.10.2013

Punkteverteilung: Aufgabe 2.1: 4 Punkte, Aufgabe 2.2: 3 Punkte, Aufgabe 2.3: 3 Punkte

Abgabe bis zum 28.10.2013. Hinweis zu Aufgaben 2.1 und 2.3: Anstatt Pseudocode kann auch lauffähiger Programmcode abgegeben werden.

Aufgabe 2.1 Seien u und v zwei beliebige Sequenzen der Länge m bzw. n. Für alle $i, 0 \leq i \leq m$ und $j, 0 \leq j \leq n$ bezeichne $Aligns(i,j)$ die Anzahl der Alignments von $u[1 \ldots i]$ und $v[1 \ldots j]$. In der Vorlesung wurde die folgende Rekursionsvorschrift für $Aligns$ angegeben:

$$Aligns(i,j) \quad = \quad \begin{cases} 1 & \text{falls } i = 0 \text{ oder } j = 0 \\ Aligns(i-1,j)+ & \\ Aligns(i,j-1)+ & \\ Aligns(i-1,j-1) & \text{sonst} \end{cases}$$

1. Schreiben Sie in einer Programmiersprache Ihrer Wahl oder Pseudocode ein Programm, das direkt nach obiger Rekursionsvorschrift den Wert $Aligns(m,n)$ berechnet. D.h. schreiben Sie $Aligns$ direkt als rekursive Funktion auf. Das Programm soll die Parameter m und n soll das Programm als Kommandozeilen-Argumente bekommen. Die maximal möglichen Eingaben für m und n sollen jeweils 15 sein. Bei Eingaben größer als 15 soll das Programm mit einer Fehlermeldung abbrechen. Bei korrekten Parametern soll das Programm eine Ausgabe in folgendem Format liefern:

   ```
   m n result (c calls)
   ```

   Dabei ist `c` die Anzahl der rekursiven Aufrufe. Beispiel: Der Aufruf von `aligns 4 4` sollte das folgende Ergebnis liefern:

   ```
   4 4 321 (481 calls)
   ```

2. (*Nur für Lösungen, die in einer Programmiersprache abgegeben werden*)
   Berechnen Sie $Aligns(15,11) = 921406335$. Wieviele Aufrufe der Funktion $Aligns$ benötigt Ihr Programm, um diesen Wert zu bestimmen?

3. Schreiben Sie nun ein Programm (bzw. geben Sie Pseudocode an), welches $Aligns(m,n)$ nach der angegebenen Rekursionsvorschrift mit Hilfe einer $(m+1) \times (n+1)$-Matrix $Alignstab$ berechnet. Dabei gilt für alle $(i,j) \in [0,m] \times [0,n]$ die Gleichung

   $$Alignstab[i,j] = Aligns(i,j)$$

   Bitte beachten Sie dabei, dass das Füllen einer Matrix nach einer Rekursionsvorschrift *nicht* zwangsweise bedeutet, dass die Matrix durch rekursive Funktionsaufrufe gefüllt wird! Es ist bei dieser Teilaufgabe eine *nicht*-rekursive Lösung gefordert.

   Bei korrekten Parametern soll das Programm eine Ausgabe in folgendem Format liefern:

```
m n result (a accesses)
```

Dabei ist `a` die Anzahl der Zugriffe auf *Alignstab*. Beispiel: Ein Aufruf von `alignstab 3 3` sollte folgendes Ergebnis liefern:

```
3 3 63 (27 accesses)
```

4. (*Nur für Lösungen, die in einer Programmiersprache abgegeben werden*)
   Berechnen Sie $Aligns(15, 11)$ mit Ihrem zweiten Programm. Wieviele Zugriffe auf die Matrix *Alignstab* benötigt Ihr Programm, um obigen Wert zu bestimmen?
5. Beschreiben Sie den konzeptionellen Unterschied zwischen einer echt rekursiven Implementierung einer Rekurrenzvorschrift und einer Implementierung mit Hilfe einer Matrix.

Aufgabe 2.2 Geben Sie drei Beispiele für Funktionen an, welche die Laufzeiten von Algorithmen beschreiben, die zeigen, dass ein $O(n^3)$ Algorithmus schneller sein kann als ein $O(n \log n)$ Algorithmus (für kleine Werte von n).
(Hinweis: Es ist nicht nach konkreten Algorithmen gefragt, sondern nach beispielhaften Laufzeiten hypothetischer Algorithmen!)

Aufgabe 2.3 Sei $\mathcal{A} = \{a, c, g, t, n\}$ das DNA-Alphabet. Dabei steht n für eine sog. *Wildcard*, d.h. eine unbestimmte Base. Sei zudem $s \in \mathcal{A}^*$ eine DNA-Sequenz.

Geben Sie einen Algorithmus an, welcher in einer Laufzeit von $O(|s|)$ für ein gegebenes k mit $1 \leq k \leq |s|$ alle Substrings der Länge k in s bestimmt, die keine Wildcard-Symbole enthalten. Dabei reicht die Ausgabe der Startpositionen der entsprechenden Substrings in s aus.

Die Lösungen zu diesen Aufgaben werden am 29.10.2013 besprochen.