# Bioinformatik I, WS00/01
# Übungsblatt zu Alignments

Besprechung am 16. November 2000

## Anzahl von Alignments

Ein *Alignment der Wörter $a$ und $b$* über einem beliebigen Alphabet $\mathcal{A}$ ist ein Paar von Wörtern $a^\diamond$ und $b^\diamond$, so dass gilt

$$a^\diamond, b^\diamond \in (\mathcal{A} \uplus \{-\})^* \tag{1}$$

$$|a^\diamond| = |b^\diamond| =: k \tag{2}$$

$$a^\diamond_{|\mathcal{A}} = a,\ b^\diamond_{|\mathcal{A}} = b \tag{3}$$

$$\text{für alle } 1 \leq i \leq k : \text{nicht } (a^\diamond_i = - \text{ und } b^\diamond_i = -). \tag{4}$$

Dies soll heissen, $a^\diamond$ und $b^\diamond$ sind Wörter über dem Alphabet $\mathcal{A} \cup \{-\}$, wobei das Symbol $-$ nicht in $\mathcal{A}$ vorkommt (1). Diese Wörter sind gleichlang; bezeichne $k$ diese *Länge des Alignments* (2). Wenn man das Symbol $-$ aus den Wörtern $a^\diamond$ und $b^\diamond$ entfernt erhält man wieder die Wörter $a$ und $b$ (3). Ausserdem soll in $a^\diamond$ und $b^\diamond$ an keiner Position $1 \leq i \leq k$ das Symbol $-$ in beiden Wörtern gleichzeitig vorkommen (4).

Uns interessiert nun die Frage, wieviele Alignments von Wörtern der Länge $n$ und $m$ existieren. Diese Anzahl sei $\#A(n,m)$. D.h. formaler

$$\#A(n,m) = |\{(a^\diamond, b^\diamond) | (a^\diamond, b^\diamond) \text{ ist ein Alignment von } a \text{ und } b\}|,$$

wobei $a$ und $b$ beliebige, aber fest gewählte Wörter über einem festen Alphabet $\mathcal{A}$ sind mit $|a| = n$ und $|b| = m$ .

### Aufgabe 1

1. Rekursionsformel

   (a) Geben Sie eine Rekursionsformel zur Berechnung von $\#A(n,m)$ an. Achten Sie insbesondere auf die Terminierung dieser Formel.

   (b) Implementieren Sie diese Rekursionsformel in C/C++.

   (c) Wie effizient ist der naheliegendste Algorithmus. Entwickeln Sie einen verbesserten Algorithmus und implementieren Sie auch diesen. [Der gesuchte Algorithmus "merkt" sich schon berechnete Zwischenergebnisse].

2. Betrachte nun das speziellere Problem, die Anzahl $\#A(n,m,k)$ der Alignments fester Länge $k$ zu berechnen. D.h. wir definieren

   $$\#A(n,m,k) = |\{(a^\diamond, b^\diamond) | (a^\diamond, b^\diamond) \text{ ist Alignment von } a \text{ und } b, |a^\diamond| = k\}|,$$

   wobei $a$ und $b$ beliebige, aber fest gewählte Wörter über einem festen Alphabet $\mathcal{A}$ sind mit $|a| = n, |b| = m$.

   (a) Geben Sie eine Rekursionsformel für $\#A(n,m,k)$ an.

   (b) Geben Sie eine nicht-rekursive Formel für $\#A(n,m,k)$ an.

3. Nicht-rekursive Formel

   (a) Die nicht-rekursive Formel für $\#A(n,m,k)$ kann verwendet werden, um eine nicht-rekursive Formel für $\#A(n,m)$ zu entwickeln. Überlegen Sie dazu vor allem, welche Gesamtlängen für Alignments zulässig sind.

   (b) Entwickeln Sie den aus der nicht-rekursiven Formulierung resultierenden Algorithmus und implementieren Sie diesen in C/C++.

4. Diskussion

   (a) Vergleichen Sie die Algorithmen bezüglich ihrer Effizienz.

   (b) Welches Problem eines Brute-Force Ansatzes zur Berechnung optimaler Alignments wird aus den Formeln deutlich?

## Aufgabe 2

**Ein Brute-Force Algorithmus zur Bestimmung optimaler Alignments.** Ein Alignment $(a^\diamond, b^\diamond)$ von $a$ und $b$ werde durch eine Funktion $W$ bewertet. $W$ ist definiert als

$$W(a^\diamond, b^\diamond) = \sum_{1 \le i \le |a^\diamond|} w(a_i^\diamond, b_i^\diamond),$$

wobei

$$w(x,y) = \begin{cases} 0 & x = y \\ c_1 & x = - \text{ oder } y = - \\ c_2 & \text{sonst.} \end{cases}$$

1. Entwickeln Sie einen Algorithmus der für gegebene Wörter $a$ und $b$ sämtliche Alignments generiert, diese mit $W$ bewertet und so ein bezüglich $W$ optimales Alignment berechnet.

2. Implementieren Sie diesen Algorithmus in C/C++, die Wörter $a$ und $b$, sowie die Konstanten $c_1$ und $c_2$, sollen im Programm veränderbar sein.

3. Probieren Sie aus, für welche Längen von $a$ und $b$ sich mit diesem Programm optimale Alignments berechnen lassen.