Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Volker Heun

SS 2000
Lösungsblatt 6
13. Juli 2000

Algorithmen der Bioinformatik

Abgabetermin: 4. Juli

Aufgabe 1

In der Stern-Methode zur approximativen Konstruktion eines mehrfachen Sequenzen Alignments wurden beim Hinzufügen der neuen Zeichenreihen zu dem bereits existierenden MSA keine Bedingung an die Reihenfolge vorgegeben. Zeigen Sie, dass unabhängig von der Reihenfolge immer dasselbe mehrfache Sequenzen Alignment konstruiert wird.

Lösungsvorschlag: Die Center Star Methode ist eine Spezialisierung des allgemeinen induktiven Verfahrens zur Konstruktion von mehrfachen Sequence Alignments, die zu einem gegebenen Baum konsistent sind. Dieser Baum ist speziell ein Sterngraph, in dessen Zentrum der Center String S_c liegt. Die Kanten verlaufen von S_c zu allen anderen Strings der Menge. Der Konstruktionsschritt fügt zum aktuellen Alignment wie folgt einen String S' hinzu: Wähle einen bereits enthaltenen String S_i (wobei $\overline{S_i}$ das um die im momentanen Alignment vorliegenden Blanks erweiterte Wort S_i ist) und erzeuge ein optimales Paar-Alignment zwischen $\overline{S_i}$ und S'. Wenn dabei $\overline{S_i}$ neue Blanks erhält, füge sie in alle bereits behandelten Wörter ein. (Wir nehmen an: $\delta(-,-) = 0$.) Das Wort $\overline{S'}$ wird dann ins Alignment eingefügt bzw. an den Baum angehängt. Die Reihenfolgenunabhängigkeit folgt nun aus der Tatsache, daß die Nachbarn von S_c im Baum im wesentlichen unabhängig voneinander behandelt werden. D.h. Matches und Substitute-Operationen sind von einer Änderung der Reihenfolge nicht betroffen. Zum Eifügen der Blanks ist jedoch zu bemerken, daß der Index, an dem diese eingefügt werden, nicht eindeutig bestimmt ist. In benachbarten Spalten, die durch solche propagierten Einfügungen von Blanks entstehen, könne i.a. Zeichen vertauscht werden.

Beispiel: $S_1 = S_c = AT$, $S_2 = AGT$, $S_3 = ATTT$
Es sind folgende Alignments äquivalent:

$$
\begin{array}{cccc||cccc}
A & - & - & T & A & - & - & T \\
A & - & G & T & A & G & - & T \\
A & T & T & T & A & T & T & T
\end{array}
$$

Aufgabe 2

Sei $S = \{s_1, \ldots, s_k\}$ eine Menge von Strings, so dass kein String Teilstring eines anderen Strings ist. Zeigen Sie, dass für einen kürzesten Superstring $\mathrm{SSP}(S)$ gilt:

$$\max\{|s_i| \,:\, i \in [1:k]\} + k - 1 \;\leq\; \mathrm{SSP}(S) \;\leq\; \sum_{i=1}^{k} |s_k|.$$

Geben Sie Beispiele für beliebiges k an, die zeigen, dass die Schranken scharf sind.

Lösungsvorschlag: Die erste Ungleichung läßt sich wie folgt zeigen: Sei o.B.d.A. $|s_1| \geq |s_2| \geq \cdots \geq |s_k|$, und sei keines der Wörter in einem anderen als Teilwort enthalten. Dann wird die rechte Seite der ersten Ungleichung minimal, wenn sich diese Strings maximal überlappen lassen. Dies ist genau dann der Fall, wenn die letzten $|s_2| - 1$ Zeichen von s_2 ein Suffix von s_1 sind, die letzten $|s_3| - 1$ Zeichen ein Suffix von s_2, u.s.w. Der längste gemeinsame Superstring besteht dann aus s_1, konkateniert mit genau einem Zeichen für jeden weiteren String $s_2, s_3, \ldots, s_k$. Dies ist auch zugleich die Anleitung zum Bau eines Beispiels, für dies hier ein "=" gilt. Daß diese Ungleichung "scharf" ist, sieht man aber nicht, wie in der Aufgabenstellung gefordert, an einem Beispiel, sondern mit Hilfe folgender Überlegung: Gäbe es in der oben dargestellten Situation einen kürzeren gemeinsamen Superstring, so wäre entweder $s_1 = s_i$ für ein $i > 1$, oder es würde eines der Wörter s_i $(i > 1)$ keinen Beitrag liefern, wäre also in einem Wort s_j $(j \neq i)$ enthalten.

Die zweite Ungleichung ist noch leichter zu sehen: Es ist klar, daß $s_1 s_2 \cdots s_k$ ein Superstring der gegebenen Wörter ist. Er hat die Länge $\sum_{i=1}^{k} |s_k|$. Ein kürzester Superstring hat also höchstens diese Länge. Ein Beispiel für eine Menge von Wörtern, bei denen der kürzeste gemeinsame Supderstring diese Länge hat, ist folgendes:

$$\begin{array}{lcl}
s_1 & = & 01010101\ldots01010101 \\
s_2 & = & 00110011\ldots00110011 \\
s_3 & = & 00011100\ldots11000111 \\
\ldots & & \ldots \\
s_3 & = & 000000000111111111
\end{array}$$

Die Länge der Strings sei $\mathrm{kgV}(2k, 2k-2, 2k-2, \ldots, 2)$, damit jeder der Strings mit einer 1 endet.

Aufgabe 3

Sei $S = \{s_1, \ldots, s_m\}$ eine Menge von Strings. Zeigen Sie, dass Sie in Zeit $O(\sum_{i=1}^{m} |s_i|)$ die folgende Menge bestimmen können:

$$\{s \in S \,:\, \exists i \in [1:m] : s \text{ ist ein Teilwort von } s_i\}.$$

Dies ist die Menge aller Strings, die in einem anderen String enthalten sind.

Lösungsvorschlag:

Konstruiere den kanonischen *gemeinsamen Suffixbaum* für die Strings s_i, $1 \leq i \leq m$, wie folgt: Sei T_i der Suffixbaum für die Strings $s_1, \ldots, s_i$, dann ist T_1 der gewöhnliche Suffixbaum für das Wort s_1, und T_i entsteht aus T_{i-1} durch Weiterverwenden der Wurzel von T_{i-1} und aller in T_{i-1} bereits enthaltenen Suffixe für das Wort s_i. Für die Labels der Blätter gilt folgendes: Ein Blatt erhält Label (i, p) genau dann, wenn die Konkatenation der Kantenlabels auf seinem zugehörigen Pfad das Suffix ab Position i innerhalb von s_p ist. Nun wird der gemeinsame Suffixbaum durch Elimination des Terminationszeichens so modifiziert, daß auch interne Knoten labels haben können. (Mit "\$" markierte Kanten zu Blättern fallen fort). Dann gilt für jedes p folgende Beobachtung: Es taucht das Label $(1, p)$ genau dann an einem internen Knoten auf, wenn String s_p Teilstring eines anderen Strings der Menge ist. Die Erfülltheit unserer Zeitschranke ist offensichtlich.