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

SS 2000
Lösungsblatt 7
20. Juli 2000

# Algorithmen der Bioinformatik

*Abgabetermin: 11. Juli*

## Aufgabe 1

Sei $s = s_1 \cdots s_n$ eine Zeichenreihe der Länge $n$. Ein $k$-mer $t$ von $s$ ist eine Teilwort der Länge $k$ von $s$, d.h. es existiert ein $i \in [1 : n-k+1]$ mit $s_i \cdots s_{i+k-1} = t$. Für eine Zeichenreihe $s$ sei $\mathcal{M}_k(s) = \{s_i \cdots s_{i+k-1} : i \in [1 : n-k+1]\}$ die Multimenge der $k$-mere von $s$.

Zeigen Sie, dass es zwei Zeichenreihe $s$ und $s'$ gibt, so dass $\mathcal{M}_4(s) = \mathcal{M}_4(s')$ ist. Dies bedeutet, dass man aus einer Menge der $k$-mere die ursprüngliche Zeichenreihe nicht mehr eindeutig rekonstruieren kann.

Wie sieht es für beliebiges $k$ aus?

**Lösungsvorschlag:** Für $k = 4$ erhält man etwa im Falle des Wortes 00001000 keine eingeutige Rekonstruierbarkeit mehr. Denn für 00010000 erhält man dieselbe Menge von 4-meren:

$$\{0000, 0001, 0010, 0100, 1000\}.$$

Für allgemeine $k$ ist $0^k10^{k-1}$ bzw. $0^{k-1}10^k$ ein mögliches Gegenbeispiel, denn die Menge der $k$-mere ist in beiden Fällen

$$\{0^t, 0^{t-1}1, 0^{t-2}10, \ldots, 10^{t-1}\}.$$

## Aufgabe 2

Konstruieren Sie mit Hilfe des in der Vorlesung vorgestellen Approximationsalgorithmus einen Superstring für

$$\begin{aligned} s_1 &= ACGACC, \\ s_2 &= ACCGAC, \\ s_3 &= CCGACG, \\ s_4 &= CGACGA, \\ s_5 &= CGACCG. \end{aligned}$$

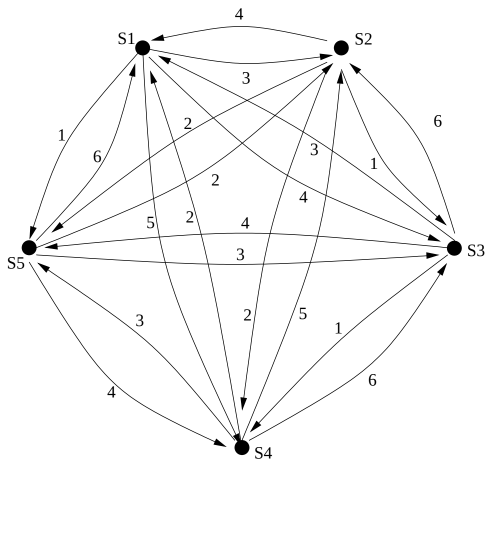
Ist der konstruiere Superstring der kürzest mögliche?

**Lösungsvorschlag:** Wir lösen die Aufgabe schrittweise nach dem Algorithmus aus der Vorlesung: Im ersten Schritt stellen wir eine $5 \times 5$-Tabelle $T$ auf, die an der Position $(i, j)$ die Anzahl der Zeichen von $s_j$ enthält, die bei einer maximalen Überlagerung eines Suffixes von $s_j$ mit einem Präfix von $s_i$ übrigbleiben, d.h. $T[i, j] = |s_j| -$ max. overlap zwischen $s_j$ und $s_i$. Für $i = j$ setzen wir den $s_j$ statt 0, denn vollständige overlaps eines Strings mit sich selbst helfen hier nicht weiter.

In diesem Fall gilt:

| $T$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|
| $s_1$ | 6 | 3 | 4 | 5 | 1 |
| $s_2$ | 4 | 6 | 1 | 2 | 2 |
| $s_3$ | 3 | 6 | 6 | 1 | 4 |
| $s_4$ | 2 | 5 | 6 | 6 | 3 |
| $s_5$ | 6 | 2 | 3 | 4 | 6 |

Dann wird $T$ als gewichtetet Adjazenzmatrix eines Graphen $G$ mit Knotenmenge $V = \{s_1, \dots, s_5\}$ betrachtet und ein minimales cycle cover von $G$ gesucht.



In diesem Fall sieht man leicht, daß $\langle s_2, s_3, s_4, s_1, s_5 \rangle$ ein minimales cycle cover ist. Dieses cycle cover definiert sofort einen minimalen gemeinsamen Superstring, nämlich $ACCGACGACCG$. Der Überlapp ist bei allen Wörtern, mit der Ausnahme von $s_1$, der maximal Mögliche. Somit erkennt man sofort, daß es in diesem Fall einen kürzeren Superstring nicht geben kann. (Dieser wäre nur durch einen anderen Startpunkt beim Duchlaufen des Kreises möglich!)

## Aufgabe 3

In der Übung erhalten Sie eine Kopie des Artikels *Combinatorial Algorithms for DNA Sequence Assembly* von *J.D. Kececioglu* und *E.W. Myers*, erschienen in *Algorithmica* Band 13 auf den Seiten 7–51 im Jahre 1995.

Lesen Sie den Artikel, so dass er in der Übung diskutiert werden kann.