Übungen zur Vorlesung Sequenzanalyse

Universität Bielefeld, SS 2018
Dr. Daniel Dörr

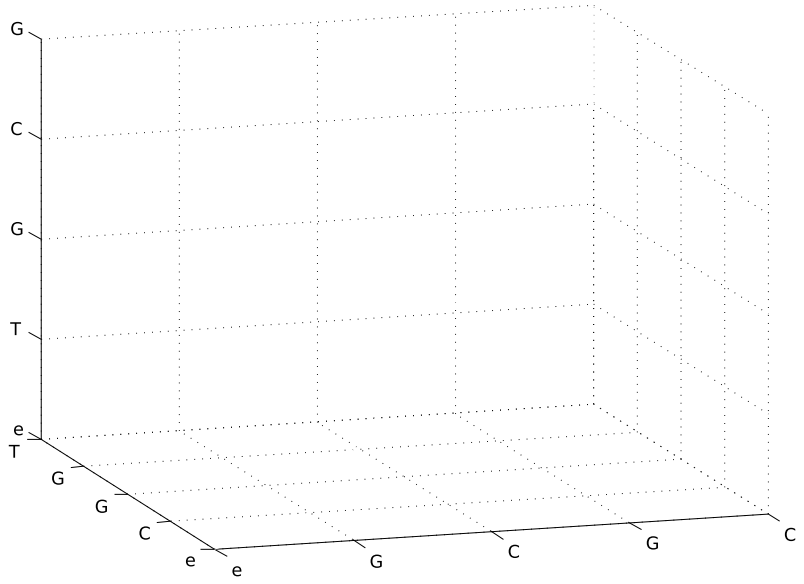
https://gi.cebitec.uni-bielefeld.de/teaching/2018summer/sa

Übungsblatt 9 vom 14.06.2018
Abgabe bis spätestens Sonntag, 24.06.2018 23:59 GMT+2 via Email oder Postfach (U10-151)

Aufgabe 1 (Geometrische Projektion des Alignments) (2 Punkte)
Gegeben sei das multiple Alignment der Sequenzen $s_1 =$ `GCGC`, $s_2 =$ `TGCG`, $s_3 =$ `CGGT`:

```
-  G  C  G  C
T  G  C  G  -
C  G  -  G  T
```

Zeichne das Alignment im 3-Dimensionalen Raum und die jeweiligen Projektionen auf den 2-Dimensionalen Unterräumen ein:



Aufgabe 2 (Sum-of-Pairs Score) (4 Punkte)
Gegeben sind die vier Sequenzen $s_1 =$ `TCG`, $s_2 =$ `AGCCT`, $s_3 =$ `TTGT`, $s_4 =$ `AACGT`, sowie lineare Gapkosten von 2 und die folgende Substitutions-Scorematrix:

	A	C	G	T
A	3	−2	−3	−1
C	−2	5	−1	−2
G	−3	−1	3	−3
T	−1	−2	−3	5

1. Berechne den Sum-of-Pairs Score des folgenden multiplen Alignments:

```
–  T  C  G  –
A  G  C  C  T
–  T  T  G  T
A  A  C  G  T
```

2. Finde ein multiples Alignment mit höherem Sum-of-Pairs Score.

Aufgabe 3 Polynomielle Reduktion (4 Punkte)

Das Kantenüberdeckungsproblem ist NP-schwer. Zeige, dass auch das Mengenüberdeckungsproblem NP-schwer ist, indem du eine polynomielle Reduktion findest, die das Kantenüberdeckungsproblem auf das Mengenüberdeckungsproblem reduziert.

Knotenüberdeckungsproblem: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine Zahl k, gibt es eine Teilmenge $V' \subseteq V$ der Größe $|V'| \leq k$ sodass für alle Kanten $(u, v) \in E$ gilt: $\{u, v\} \cap V' \neq \emptyset$?

Mengenüberdeckungsproblem: Gegeben seien eine Menge M von Elementen, Teilmengen $S_1, S_2, \ldots, S_n \subseteq M$ und Zahl $k \leq n$, gibt es eine Kollektion von k Teilmengen $\{S_{i_1}, \ldots S_{i_k}\} \subseteq \{S_1, S_2, \ldots S_n\}$ sodass $M = S_{i_1} \cup \ldots \cup S_{i_k}$?

Bonusaufgabe 1 (Exakte Textsuche in BW-transformierten Texten) (8* Punkte)

Implementiere in der Programmierspache Java den in der Vorlesung besprochenen Algorithmus zur exakten Textsuche in Burrows-Wheeler-Transformierten Texten. Gehe dabei wie folgt vor:

1. Lese Muster p, Burrows-Wheeler-Transfomierten Text $L = bwt(t)$ und Suffixarray *pos* ein.
2. Iteriere durch L, merke dabei für jede Position i, $0 \leq i < |t|$ die Anzahl vorkommen des Zeichens $L[i]$ in Präfix $L[0..i-1]$ (wobei $L[0..-1] = \varepsilon$). Benutze dabei eine Hilfsdatenstruktur, damit dieser Schritt nicht quadratische Laufzeit benötigt. Merke dir zusätzlich die absoluten Häufigkeiten aller Zeichen in L.
3. Erstelle mithilfe der im vorherigen Schritt erstellten Hilfsdatenstrukturen ein sogenanntes „LF-Mapping“, d.h. ein Mapping von Positionen von L auf (eine implizite Darstellung von) F.
4. Implementiere eine Funktion, welche mittels L, des LF-Mappings und des Musters m alle Vorkommen von m in L sucht.
5. Gebe die Startpositionen der Vorkommen mittels Suffixarray *pos* aus.