Stefan Kurtz **Universität Hamburg**
Abteilung für Genominformatik **Zentrum für Bioinformatik**

Grundlagen der Sequenzanalyse
Wintersemester 2022/2022
Übungen zur Vorlesung: Ausgabe am 25.10.2022

Beginnen Sie die Übungen mit den folgenden Schritten (das Zeichen $ steht für das Eingabeprompt im Terminal):

- falls Sie noch kein Repository gecloned haben:

  ```
  $ git clone https://gitlab.rrz.uni-hamburg.de/Bae4410/gsa_2022.git
  ```

- Beachten Sie den Unterstrich in `gsa_2022`. Es hat sich gezeigt, dass manchmal der Copy/Paste Mechanismus den Unterstrich nicht wiedergibt. In einem solchen Fall muss man den Unterstrich vor der Ausführung des Kommandos hinzufügen.
- Wechseln in das Repository: `$ cd gsa_2022`
- Aktualisieren: `$ git pull`
- Wechseln in das HOME-Verzeichnis. Dazu kann man `cd` ohne Argument aufrufen:

  ```
  $ cd
  ```

- Erzeugen eines eigenen Verzeichnisses für die eigenen Lösungen und Materialien aus der Vorlesung, falls es noch nicht existiert. Damit Sie es später nicht vergessen, verwenden Sie gleich den Verzeichnisnamen, der die Nachnamen der Gruppenmitglieder beinhaltet. Unten werden beispielhaft die Platzhalter `Name1` und `Name2` für die Namen der Gruppenmitglieder verwendet. Diese müssen Sie natürlich durch reale Nachnamen ersetzen. Umlaute ü werden durch ue ersetzt und ß durch ss: Falls ein Nachname aus mehreren Worten besteht, werden diese Worte durch einen Unterstrich verbunden.

  ```
  $ mkdir -p my_gsa_2022/Uebungen/Blatt01.Name1.Name2
  ```

- Wechseln in das neue Verzeichnis:

  ```
  $ cd my_gsa_2022/Uebungen/Blatt01.Name1.Name2
  ```

- Kopieren der Materialien für Blatt01:

  ```
  $ cp -r ../../../gsa_2022/Uebungen/Blatt01/* .
  ```

- Erstellen der Lösung in Dateien, entsprechend der Benennungen aus der Übungsaufgabe (falls es Vorgaben gibt).
- Falls Sie noch keinen Texteditor beherrschen, müssen Sie sich mit einem vertraut machen, z.B. mit `vim`. Hierzu gibt es ein interaktives Tutorial: `https://www.openvim.com`

Aufgabe 1.1 (2 Punkte) Sei $\exp$ mit $\exp(x) = \mathrm{e}^x$ die Exponentialfunktion und bezeichne $\log(x)$ den Logarithmus von x zur Basis e. Sie kennen sicher alle die folgenden Eigenschaften dieser beiden Funktionen,

die für alle $q, p \in \mathbb{R}$ gelten:

$$\begin{aligned}
\exp(\log p) &= p \\
\log(\exp p) &= p \\
\log(pq) &= \log p + \log q \\
\log \frac{p}{q} &= \log p - \log q \\
\exp(p+q) &= \exp p \cdot \exp q \\
\exp(p-q) &= \frac{\exp p}{\exp q}
\end{aligned}$$

Beweisen Sie, ggf. unter Verwendung obiger Eigenschaften, die folgende Gleichung:

$$\log(p+q) = \log p + \log(1 + \exp(\log q - \log p)).$$

Aufgabe 12 (4 Punkte) In der Vorlesung werden an verschiedenen Stellen Mengen definiert, z.B.

$$\mathcal{A}^i = \begin{cases} \{\varepsilon\} & \text{if } i = 0 \\ \{aw \mid a \in \mathcal{A}, w \in \mathcal{A}^{i-1}\} & \text{if } i > 0 \end{cases}$$

Diese Definition läßt sich wie folgt lesen:

Die Menge $\mathcal{A}^0$ aller Strings der Länge 0 ist $\{\varepsilon\}$, wobei ε das leere Wort bezeichnet. Die Menge $\mathcal{A}^i$ aller Strings der Länge $i > 0$ ist definiert als die Menge aller aw, wobei a ein Element aus der Menge $\mathcal{A}$ ist und w ein Element aus der Menge $\mathcal{A}^{i-1}$, also der Menge der Strings der Länge $i - 1$.

1. Beschreiben Sie, analog zum obigen Beispiel aus der Vorlesung, folgende Mengendefinitionen in ganzen Sätzen:
   a) $E_k = \{2^i \mid 0 \leq i \leq k\}$
   b) $P_w = \{(i,j) \mid 1 \leq i < j \leq n, (w[i], w[j]) \in B, j - i - 1 \geq \ell\}$. w steht hierbei für einen String über dem Alphabet $\mathcal{A} = \{A, C, G, U\}$, B bezeichnet die Menge $\{(A,U), (U,A), (C,G), (G,C), (G,U), (U,G)\}$ und ℓ und n sind ganze Zahlen.
2. Geben Sie zu folgenden in ganzen Sätzen definierten Mengen $H_{p,q}(u)$ und $M_k(w,v)$ formale Definitionen in Mengen-Notation an:
   a) Sei f eine Funktion, die Paare von Strings gleicher Länge auf natürliche Zahlen abbildet. Für positive ganze Zahlen p, q, g und einen String u ist $H_{p,q}(u)$ definiert als die Menge aller Paare (t,i), wobei t ein String der Länge p über dem Alphabet $\mathcal{A}$ ist, $1 \leq i \leq |u| - q + 1$ gilt und f, angewendet auf das Paar $(t, u[i \ldots i+q-1])$ mindestens den Wert g hat.
   b) Für zwei Strings w und v und eine positive ganze Zahl k ist $M_k(w,v)$ die Menge aller Paare von Positionen in w und v, an denen ein Treffer (d.h. ein gemeinsamer Substring) der Länge k beginnt.

Erstellen Sie Ihre Lösung als LaTeX-Datei `set_notation.tex` und legen Sie nur diese im entsprechenden Material-Verzeichnis ab. Verifizieren Sie, dass man durch Anwendung von `pdflatex` aus der `.tex`-Datei eine PDF-Datei erzeugen kann. Ihre PDF-Datei wird dann nicht als Teil der Lösung abgegeben. Diese Regel gilt für alle zukünftigen Aufgaben mit Lösungen, die LaTeX-Formatierungen erfordern.

Aufgabe 13 (2 Punkte)

Man unterscheidet bei doppelsträngiger DNA zwischen der Basensequenz auf dem *plus*- und dem *minus*-Strang. Häufig wird auch vom Vorwärts und Rückwärtsstrang gesprochen. Durch Basenpaarung bedingt

sind diese *revers komplementär* zueinander. Sei $\mathcal{A} = \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$ das DNA-Alphabet. Das *Komplement* $\overline{x}$ eines Zeichens $x \in \mathcal{A}$ ist definiert als $\overline{\mathtt{a}} = \mathtt{t}$, $\overline{\mathtt{t}} = \mathtt{a}$, $\overline{\mathtt{c}} = \mathtt{g}$ und $\overline{\mathtt{g}} = \mathtt{c}$.

Für eine DNA Sequenz $s \in \mathcal{A}^*$ der Länge n ist das *reverse Komplement* $\overleftarrow{s}$ von s definiert als $\overleftarrow{s} = \overline{s[n]}\ \overline{s[n-1]} \ldots \overline{s[2]}\ \overline{s[1]}$. Zum Beispiel ist $\overleftarrow{\mathtt{gagctgaa}} = \mathtt{ttcagctc}$.

Geben Sie einen Algorithmus an, der für eine gegebene Sequenz s der Länge n das reverse Komplement in konstantem Speicherplatz berechnet. Der Speicherplatz für die ursprüngliche Sequenz s soll durch das reverse Komplement überschrieben werden, so dass am Ende des Algorithmus die ursprüngliche Sequenz nicht mehr gespeichert vorliegt. Konstanter Speicherplatz, etwa zum temporären Zwischenspeichern einzelner Zeichen oder ganzer Zahlen, ist zulässig.

Der Algorithmus soll in Form von Pseudocode in LaTeX (siehe Beispiel am Ende des Übungsblattes) in einer Datei `rev_compl_inplace.tex` angegeben werden. Eine Darstellung als Textdatei mit passender Einrückung wäre auch in Ordnung. Verwenden Sie dann bitte für den Dateinamen den Suffix `.txt` statt `.tex`.

Aufgabe 1*4* (3 Punkte)

Seien S und T zwei Sequenzen der Längen m und n. Die Trefferstatistik von T bzgl. S ist eine Tabelle M der Länge n, indiziert von 1 bis n, so dass $M[i] = (\ell_i, P_i)$ für alle i, $1 \leq i \leq n$ und es gilt:

- ℓ_i ist die Länge des längsten Präfix von $T[i \ldots n]$, der Substring von S ist.
- P_i ist die Menge aller Positionen p, $1 \leq p \leq m$ so dass $T[i \ldots i + \ell_i - 1] = S[p \ldots p + \ell_i - 1]$

Beispiel: Sei $S = \mathtt{caccaccac}$ und $T = \mathtt{aacaccaacaca}$, hier mit den entsprechenden Indexpositionen angegeben:

caccaccac	aacaccaacaca
123456789	123456789012

Dann wird die Trefferstatistik von T bzgl. S durch die Tabelle rechts dargestellt:

i	$M[i]$	
	ℓ_i	P_i
1	1	{2, 5, 8}
2	2	{2, 5, 8}
3	5	{1, 4}
4	4	{2, 5}
5	3	{3, 6}
6	2	{1, 4, 7}
7	1	{2, 5, 8}
8	2	{2, 5, 8}
9	3	{1, 4, 7}
10	2	{2, 5, 8}
11	2	{1, 4, 7}
12	1	{2, 5, 8}

Berechnen Sie in einer Datei `matching_stat_SvsT.tsv` die Trefferstatistik von S bzgl. T. Die Datei soll aus drei Spalten, die jeweils durch einen Tabulator getrennt sind, bestehen, Die erste Spalte enthält den Index i, die zweite Spalte den Wert von ℓ_i und die dritte Spalte die Elemente der Menge P_i, in aufsteigender Reihenfolge und jeweils durch ein Komma getrennt. Die entsprechende Datei für die Trefferstatistik von T bzgl. S finden Sie in den Materialien.

Bitte die Lösungen zu diesen Aufgaben bis zum 30.10.2022 um 22:00 Uhr an gsa@zbh.uni-hamburg.de schicken.

Beispiel für Pseudocode

Hinweis für Studierende, die LaTeX nutzen: verwenden Sie die Pakete `algorithm` und `algpseudocode` in der Präambel und schreiben Sie den Pseudocode nach folgendem Schema. Wenn Sie nicht LaTeX nutzen, dann orientieren Sie sich bei der Syntax am Beispiel unten. Dabei stehen die Bezeichner *Input* bzw. *Output* für Beschreibungen der Ein- bzw. Ausgabe. Die Bezeichner *Anweisungen* sind Platzhalter für eine oder mehrere Anweisungen. Der Bezeichner *Bedingung* steht für einen Ausdruck, der zu einem Wahrheitswert ausgewertet wird, z.B. $x < y$ oder $i = 0$.

```
\begin{algorithm}
  \caption{Ein Beispiel f\"ur Pseudocode}
  \begin{algorithmic}[1]
    \State \textit{Input} \Comment{Beschreibung der Eingabe}
    \State \textit{Output} \Comment{Beschreibung der Ausgabe}
    \Statex
    \State \(\mathit{variable}\gets \mathit{value}\) \Comment{Wertzuweisung}
    \Statex
    \If{\textit{Bedingung}} \Comment{bedingte Anweisungen}
      \State \textit{Anweisungen}
    \ElsIf{\textit{Bedingung}}
      \State \textit{Anweisungen}
    \Else \Comment{alternative Anweisungen}
      \State \textit{Anweisungen}
    \EndIf
    \Statex
    \For{\(i \gets 0\) \textbf{upto} \(n-1\)}\Comment{for-Schleife}
      \State $j\gets 2\cdot i$
      \State \textit{Anweisungen}
    \EndFor
    \Statex
    \While{\textit{Bedingung}}\Comment{while-Schleife}
      \State \textit{Anweisungen}
    \EndWhile
  \end{algorithmic}
\end{algorithm}
```

Hier ist das Ergebnis, das man sicher auch anders erzeugen könnte:

Algorithm 1 Ein Beispiel für Pseudocode

1: *Input* ▷ Beschreibung der Eingabe
2: *Output* ▷ Beschreibung der Ausgabe

3: $variable \leftarrow value$ ▷ Wertzuweisung

4: **if** *Bedingung* **then** ▷ bedingte Anweisungen
5: *Anweisungen*
6: **else if** *Bedingung* **then**
7: *Anweisungen*
8: **else** ▷ alternative Anweisungen
9: *Anweisungen*
10: **end if**

11: **for** $i \leftarrow 0$ **upto** $n - 1$ **do** ▷ for-Schleife
12: $j \leftarrow 2 \cdot i$
13: *Anweisungen*
14: **end for**

15: **while** *Bedingung* **do** ▷ while-Schleife
16: *Anweisungen*
17: **end while**