Stefan Kurtz **Universität Hamburg**
Christina de Bruyn Kops **Zentrum für Bioinformatik**

Grundlagen der Sequenzanalyse
Wintersemester 2016/2017
Übungen zur Vorlesung: Ausgabe am 01.11.2016

Punkteverteilung: Aufgabe 3.1: 3 Punkte, Aufgabe 3.2: 3 Punkte, Aufgabe 3.3: 4 Punkte

Abgabe bis zum 7.11.

Aufgabe 3.1 Überlegen Sie sich eine möglichst speichereffiziente Datenstruktur zur Repräsentation von Alignments und beschreiben Sie diese informell. Begründen Sie Ihre Entscheidung bezüglich der Datenstruktur und diskutieren Sie Vor- und Nachteile. Wie wird mit Ihrer Repräsentation ein Alignment im Allgemeinen gespeichert? Mit welchem Speicherplatzaufwand geschieht dies? Gibt es Redundanzen in der Darstellung? Welche Operationen unterstützt Ihre Datenstruktur und in welcher Laufzeit sind diese durchführbar?

Wie sieht in Ihrer Datenstruktur eine konkrete Repräsentation des folgenden Alignments aus:

```
acgtaga--tataga-gat
|||  |||   ||  |||  |  |
acgaagaggta-agagggt
```

Aufgabe 3.2 Seien $u = agtgcacaca$ und $v = atcacactta$ zwei Sequenzen. Berechnen Sie die Dynamic-Programming-Matrizen für den Algorithmus zur Lösung des Edit-Distanz-Problems für die verschiedenen Kostenfunktionen δ und die Sequenzen u und v:

1. Einheitskosten:
$$\delta(\alpha \to \beta) = \begin{cases} 0 & \text{falls } \alpha, \beta \in \mathcal{A} \text{ und } \alpha = \beta \\ 1 & \text{sonst} \end{cases}$$

2. Hammingkosten:
$$\delta(\alpha \to \beta) = \begin{cases} 0 & \text{falls } \alpha, \beta \in \mathcal{A} \text{ und } \alpha = \beta \\ 2 & \text{falls } \alpha, \beta \in \mathcal{A} \text{ und } \alpha \neq \beta \\ \infty & \text{sonst} \end{cases}$$

3.
$$\delta(\alpha \to \beta) = \begin{cases} 0 & \text{falls } \alpha, \beta \in \mathcal{A} \text{ und } \alpha = \beta \\ 3 & \text{falls } \alpha, \beta \in \mathcal{A} \text{ und } \alpha \neq \beta \\ 1 & \text{sonst} \end{cases}$$

In der Matrix zu der Einheitskostenfunktion tragen Sie bitte zusätzlich einen minimierenden Pfad ein. Geben Sie außerdem das daraus resultierende Alignment an. Die Berechnungen können manuell erfolgen.

Aufgabe 3.3 Sei δ eine Kostenfunktion, so dass für alle Editoperationen $\alpha \to \beta$ die folgenden Eigenschaften gelten:

$$\begin{aligned} \delta(\alpha \to \beta) &= \delta(\beta \to \alpha) \\ \delta(\alpha \to \beta) = 0 &\iff \alpha = \beta \end{aligned}$$

Zeigen Sie, dass dann für alle $u, v \in \mathcal{A}^*$ die folgenden Eigenschaften gelten:

1. $edist_\delta(u, v) = edist_\delta(v, u)$
2. $edist_\delta(u, v) = 0 \iff u = v$

Wichtig: Der Beweis soll in ähnlicher Weise geführt werden wie der Beweis zu

$$edist_\delta(u, v) = edist_\delta(u^{-1}, v^{-1})$$

im GSA-Skript.

Die Lösungen zu diesen Aufgaben werden am 08.11.2016 besprochen.

Lösung zu Aufgabe 3.1:

Aus der Vorlesung wissen wir, dass ein Alignment der Sequenzen u, v mit $|u| = m$, $|v| = n$ die Form $A = (\alpha_1 \to \beta_1, \ldots, \alpha_h \to \beta_h)$ hat. Würde man dies so im Computer abbilden, hätte man einen Speicherverbrauch von h, also $O(m + n)$. Gewöhnlich kommt dazu noch der Speicherplatz für beide Sequenzen, da diese bei Alignmentprogrammen meist unverändert bleiben. Beim intuitiven Speichern von Alignments hätten wir also die Sequenzinformation doppelt hinterlegt, da gilt: $u = \alpha_1 \ldots \alpha_h$, $v = \beta_1 \ldots \beta_h$.

Eine erste Verbesserung besteht nun darin, lediglich die entsprechenden Editoperationen für eine Transformation zu speichern, ohne wiederholt die Zeichen abzulegen. D.h. man hinterlegt nur noch eine Sequenz aus Editoperationen $eop \in \{\mathtt{R}, \mathtt{D}, \mathtt{I}\}$ (Replacement, Deletion, Insertion), wobei jede Position der Transformationssequenz mit einer Position in einer beliebigen aber festen der beiden Sequenzen korrespondiert. Damit wird der Speicherplatzbedarf auf $\frac{1}{3}$ der vorherigen Variante reduziert. Das in der Aufgabe angegebene Beispiel wird mit einer Transformationssequenz wie folgt codiert:

$$L_{eop} = \mathtt{RRRRRRRIIRRDRRRIRRR}$$

Eine weitere Reduktion erhält man, wenn anstelle von einzelnen Operationen für Folgen der gleichen Transformation Tupel $(eop, n), n \in \mathbb{N}$ aus Editoperation und Länge der Folge gespeichert werden. Das Beispiel aus der Aufgabenstellung lässt sich damit wesentlich kompakter darstellen:

$$L_{Tupel} = ((\mathtt{R}, 7), (\mathtt{I}, 2), (\mathtt{R}, 2), (\mathtt{D}, 1), (\mathtt{R}, 3), (\mathtt{I}, 1), (\mathtt{R}, 3))$$

Der asymptotische Platzbedarf bleibt hierbei weiterhin bei $O(m+n)$ und wird z.B. für Alignments mit wechselnden Folgen aus Replacements und Deletions der Länge 1 erreicht. Da solche Alignments eher selten auftreten, ist der durchschnittliche Speicherverbrauch wesentlich geringer.

Um ein Alignment im Computer als Sequenz von Tupeln aus Editoperation und Schrittweite zu speichern, verwenden wir einen Byte-Array. Dabei codieren die ersten 2 Bit eines jeden Eintrages die Editoperation und die verbleibenden 6 Bit die Länge der Folge. Sei i ein solches Byte, dann lässt sich die Editoperation des Tupels mit einer Funktion $eop(i)$ bestimmen:

$$eop(i) = \begin{cases} \mathtt{R} & \text{if} \quad i \,\&\, \mathtt{11000000} = \mathtt{11000000} \\ \mathtt{D} & \text{if} \quad i \,\&\, \mathtt{11000000} = \mathtt{01000000} \\ \mathtt{I} & \text{if} \quad i \,\&\, \mathtt{11000000} = \mathtt{10000000} \end{cases}$$

Die Elemente des Arrays zu dem Beispiel aus der Aufgabenstellung sehen dann wie folgt aus:

$$L_{binaer} = (11000111, 10000010, 11000010, 01000001, 11000011, 10000001, 11000011)$$

Lösung zu Aufgabe 3.2:

```
      a  t  c  a  c  a  c  t  t  a

   0  1  2  3  4  5  6  7  8  9 10
    \
a  1  0  1  2  3  4  5  6  7  8  9
      |
g  2  1  1  2  3  4  5  6  7  8  9
       \
t  3  2  1  2  3  4  5  6  6  7  8
         |
g  4  3  2  2  3  4  5  6  7  7  8
          \
c  5  4  3  2  3  3  4  5  6  7  8
             \
a  6  5  4  3  2  3  3  4  5  6  7
                \
c  7  6  5  4  3  2  3  3  4  5  6
                   \
a  8  7  6  5  4  3  2  3  4  5  5
                      \
c  9  8  7  6  5  4  3  2--3--4  5
                               \
a 10  9  8  7  6  5  4  3  3  4  4
```

```
agtgcacac--a
 | |     ||
a-t-cacactta
```

```
      a  t  c  a  c  a  c  t  t  a
   0 i. i. i. i. i. i. i. i. i. i.
a i.  0 i. i. i. i. i. i. i. i. i.
g i. i.  2 i. i. i. i. i. i. i. i.
t i. i. i.  4 i. i. i. i. i. i. i.
g i. i. i. i.  6 i. i. i. i. i. i.
c i. i. i. i. i.  6 i. i. i. i. i.
a i. i. i. i. i. i.  6 i. i. i. i.
c i. i. i. i. i. i. i.  6 i. i. i.
a i. i. i. i. i. i. i. i.  8 i. i.
c i. i. i. i. i. i. i. i. i. 10 i.
```

```
a i. i. i. i. i. i. i. i. i. i. 10

i. = infinite

       a  t  c  a  c  a  c  t  t  a
   0   1  2  3  4  5  6  7  8  9 10
a  1   0  1  2  3  4  5  6  7  8  9
g  2   1  2  3  4  5  6  7  8  9 10
t  3   2  1  2  3  4  5  6  7  8  9
g  4   3  2  3  4  5  6  7  8  9 10
c  5   4  3  2  3  4  5  6  7  8  9
a  6   5  4  3  2  3  4  5  6  7  8
c  7   6  5  4  3  2  3  4  5  6  7
a  8   7  6  5  4  3  2  3  4  5  6
c  9   8  7  6  5  4  3  2  3  4  5
a 10   9  8  7  6  5  4  3  4  5  4
```

Lösung zu Aufgabe 3.3:

1. Sei $A = (\alpha_1 \to \beta_1, \ldots, \alpha_h \to \beta_h)$ ein optimales Alignment von u und v. Dann ist $A' = (\beta_1 \to \alpha_1, \ldots, \beta_h \to \alpha_h)$ ein Alignment von v und u. Wegen der Symmetrie von δ gilt $\delta(A) = \delta(A')$ und damit $edist_\delta(u, v) = \delta(A) = \delta(A')$. Wenn A' optimal ist gilt $edist_\delta(u, v) = \delta(A) = \delta(A') = edist_\delta(v, u)$.

   Beweis durch Widerspruch: Annahme A' ist nicht optimal. Dann gibt es ein Alignment $A'^\star = (\beta_1^\star \to \alpha_1^\star, \ldots, \beta_k^\star \to \alpha_k^\star)$ mit $\delta(A'^\star) < \delta(A')$. Dann ist $A^\star = (\alpha_1^\star \to \beta_1^\star, \ldots, \alpha_k^\star \to \beta_k^\star)$ ein Alignment von u und v. Wegen der Symmetrie von δ gilt $\delta(A^\star) = \delta(A'^\star) < \delta(A') = \delta(A) \Rightarrow$ Widerspruch zu der Annahme das $\delta(A)$ optimal ist.

2. Sei $edist_\delta(u, v) = 0$. Dann gibt es ein Alignment $A = (\alpha_1 \to \beta_1, \ldots, \alpha_h \to \beta_h)$ mit $\delta(A) = 0$. Damit ist $\delta(\alpha_i \to \beta_i) = 0$ für alle $i, 1 \leq i \leq h$. Laut Definition gilt dann auch $\alpha_i = \beta_i$, also $u = v$.

   Sei nun $u = v$ und $h = |u|$. Dann ist $A = (u_1 \to u_1, \ldots, u_h \to u_h)$ ein Alignment von u und v und es gilt $\delta(A) = 0$ nach Definition. Damit ist $edist_\delta(u, v) \leq \delta(A) = 0$. Da $edist_\delta(u, v) \geq 0$, gilt $edist_\delta(u, v) = 0$.