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

Grundlagen der Sequenzanalyse
Wintersemester 2022/2023
Übungen zur Vorlesung: Ausgabe am 24.01.2023

Aufgabe 12.1 (2 Punkte)

Aus Zeitgründen können wir in der Vorlesung das Thema „Significance of local alignments“ nicht besprechen. Dieser Abschnitt wird daher im Peer-Teaching Format behandelt.

Für diese Übungsaufgabe muss sich aus jeder Kleingruppe ein Student bzw. eine Studentin anhand der Vorlesungsfolien (siehe `Vorlesung/evalues_slides.pdf`, frame 1-7 Mitte, frame 11-14, frame 17-20) auf dieses Thema vorbereiten. Sie sollten untereinander frühzeitig klären, wer diese Vorbereitung in dieser Woche übernimmt.

Das in der Vorbereitung erworbene Wissen soll an die anderen Mitglieder der jeweiligen Kleingruppe dadurch weitergegeben werden, dass man zusammen den Inhalt der Folien zum genannten Thema bespricht. Das kann entweder in den ersten 20-25 Minuten der Übung erfolgen oder zu einem anderen Zeitpunkt rechtzeitig vor Abgabe des Übungsblattes.

Der oder die Studierende, die sich vorbereitet hat, kann ggf. bei Unklarheiten Fragen, zumindest zu den Grundbegriffen, beantworten oder Erläuterungen mündlich ergänzen.

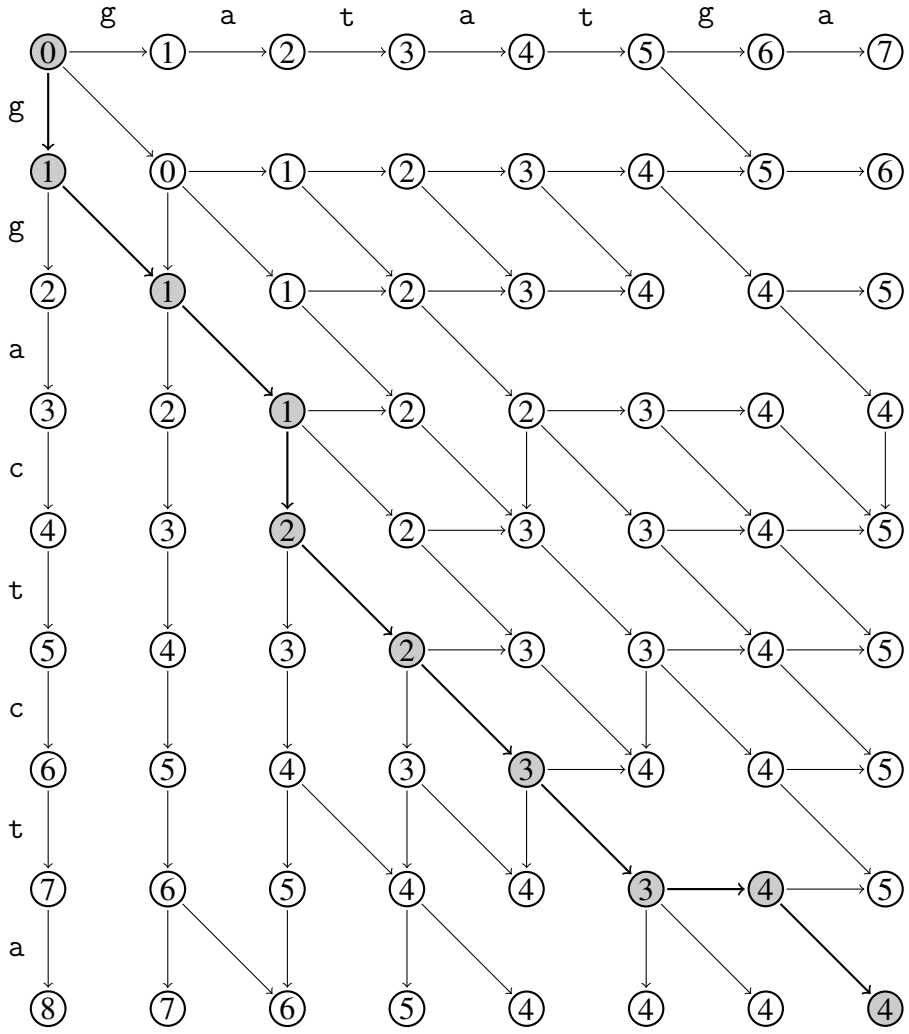
Nach Bearbeitung dieser Aufgabe dokumentiert jede Kleingruppe in der Textdatei `bearbeitung.txt` in höchstens 15 Zeilen mit maximal 80 Zeichen pro Zeile ihr Vorgehen bei der Erarbeitung des Themas und ggf. noch bestehende Verständnisfragen oder Hinweise zu Unklarheiten in den Folien. Willkommen sind natürlich auch Bemerkungen zur Lehrform selbst. Es soll nicht der Inhalt der Folien wiedergegeben werden. Selbstverständlich müssen Sie in der Datei auch die üblichen Metadaten angeben.

Aufgabe 12.2 (4 Punkte) Geben Sie die DP-Rekurrenz an, um das optimale *Multiple Sequenz-Alignment* (*MSA*) von $k = 3$ Sequenzen zu berechnen. D.h. leiten Sie aus der allgemeinen DP-Rekurrenz für k Sequenzen (siehe Vorlesungsfolien zum Multiplen Alignment) die Rekurrenz für $k = 3$ ab, wobei die jeweiligen Bedingungen für die einzelnen Fälle explizit angegeben werden sollen. Gehen Sie analog zum Beispiel für die partielle Auswertung der Rekurrenz für $k = 2$ aus der Vorlesung vor. Den LaTeX-Code dieser partiellen Auswertung finden finden Sie in der Datei `Mrecurrence2.tex`.

Aufgabe 12.3 (6 Punkte) In der Vorlesung wurde ein Algorithmus zur Berechnung eines optimalen Alignments von Sequenzen u und v beschrieben, der nur linearen Speicherplatz benötigt. Dieser Algorithmus berechnet eine Tabelle C von Kreuzungspunkten. Diese Tabelle ist indiziert von 0 bis n, wobei n die Länge der Sequenz v ist. C speichert Zeilenindices, d.h. Werte zwischen 0 und m, wobei m die Länge der Sequenz u ist. Es gilt: Falls $C(j) = i$, dann liegt der Knoten (i, j) auf einem minimierender Pfad von $(0, 0)$ nach (m, n) im Edit-Graphen $G(u, v)$ von u und v. C wird im Algorithmus so konstruiert, dass es ein optimales Alignment repräsentiert. Ein Beispiel finden Sie in Abbildung 1.

In dieser Aufgabe geht es darum, in linearer Zeit aus einer Liste von $n + 1$ Kreuzungspunkten ein optimales Alignment zu konstruieren. Zur Lösung dieser Aufgabe müssen Sie jeweils zwei

Abbildung 1: Sei u =ggactcta, v =gatatga und C =[0,2,4,5,6,7,7,8]. Die Kreuzungspunkte beschreiben den mit fetten Kanten dargestellten Pfad durch den Edit-Graphen und damit das rechts davon stehende Alignment. Im Beispiel oben ist $C(0) = 0$ und $C(1) = 2$. Also muss $u[1 \dots 2] = gg$ mit $v[1] = g$ aligniert werden. Das optimale Alignment nach unserer Konvention ist $g \to \varepsilon, g \to g$. Es ist $C(1) = 2$ und $C(2) = 4$. Also muss $u[3 \dots 4] = ac$ mit $v[2] = a$ aligniert werden. Da $u[3] = a = v[2]$ ergibt sich das optimale Alignment $a \to a, c \to \varepsilon$. Es ist $C(5) = C(6) = 7$. Also muss der leere String mit dem vorletzten Zeichen von v aligniert werden, d.h. das optimale Teilalignment ist $\varepsilon \to g$.



Optimales Alignment entsprechend der Kreuzungspunkte aus C:

```
ggactct-a
 ||  |  |  |
-ga-tatga
```

aufeinander folgende Kreuzungspunkte $C(j)$ und $C(j+1)$ betrachten. Aufgrund der Konstruktion von C gilt $C(j) \leq C(j+1)$. Aus der Relation von $C(j)$ und $C(j+1)$ ergibt sich, welcher Substring u' von u mit dem Zeichen $v[j+1]$ aligniert werden muss.

Da das zu konstruierende Alignment von u und v optimal ist, muss auch das Teilalignment von u' und $v[j+1]$ optimal sein. Führen Sie nun eine Fallunterscheidung für u' durch und überlegen Sie sich, wie man ohne Verwendung dynamischer Programmierung ein optimales Alignment A von u' mit dem Zeichen $v[j+1]$ konstruiert. Da in der Rekurrenz für die Matrix R_{j_0} (siehe Vorlesungsfolien) bei gleichen Kosten die Präferenzen „Einfügung vor Ersetzung“ und „Ersetzung vor Löschung“ gelten, müssen Sie bei der Alignment-Konstruktion wie folgt vorgehen (nur dann können Sie die Alignments aus den Testfällen rekonstruieren):

- Falls im Alignment A ein Match vorkommt, muss das letzte Zeichen in u', das gleich $v[j+1]$ ist, ersetzt werden.
- Falls im Alignment A kein Match vorkommt, muss das letzte Zeichen in u' ersetzt werden.

Im Beispiel aus Abbildung 1 kommt `g` an den ersten beiden Positionen in u vor. Nach obiger Regel wird das Alignment so konstruiert, dass das zweite `g` in u und das erste `g` in v matcht. Entsprechend wir dieser Match im Alignment in einer Spalte dargestellt.

Dokumentieren Sie Ihre Überlegungen zur Konstruktion der Teilalignments nachvollziehbar in Textform am Anfang des Programms oder in einer eigenen Datei. 2 Punkte

Implementieren Sie nun, entsprechend Ihren Überlegungen, in einer Datei `crosspoints2al.py` eine Funktion `crosspoints2alignment(u,v,ctab)` in Python, die für die beiden Strings `u` und `v` und die entsprechende Liste `ctab` der $n+1$ Kreuzungspunkte in linearer Zeit ein optimales Alignment konstruiert und durch eine **`return`**-Anweisung zurückliefert.

Zur Repräsentation von Alignments verwenden Sie die Klasse `Alignment`, siehe `aligntype.py`. Die Editoperationen, aus denen die optimalen Teilalignments bestehen, müssen nacheinander durch die Anwendung der passenden Methoden an eine Instanz der Klasse `Alignment` „angehängt“ werden. 2 Punkte

In den Materialien finden Sie Testdaten und Unittests für 103 Paare von Sequenzen. Beschreiben Sie durch Kommentare in der Datei `crosspoints2al_unittest.py`, welche Daten jeweils für ein Sequenzpaar vorliegt und welche Eigenschaften bzgl. dieser Daten jeweils durch die Tests verifiziert werden. 1 Punkt

Durch `make test` führen Sie zwei Unittests durch. Für das Bestehen der Tests erhalten Sie jeweils 0.5 Punkte. 1 Punkt

Die Punkteverteilung ergibt sich aus den Randmarkierungen oben.

Bitte die Lösungen zu diesen Aufgaben bis zum 29.01.2023 um 22:00 Uhr an gsa@zbh.uni-hamburg.de schicken.