Institut für Informatik
Praktische Informatik und Bioinformatik
Prof. Dr. Caroline Friedel
Michael Kluge, Markus Joppich

Wintersemester 2017/18

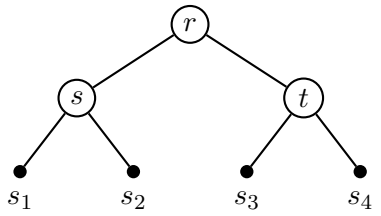
Übungen zur Algorithmischen Bioinformatik II

Blatt 9

Abgabetermin: Donnerstag, 11.01.2018, vor Beginn der Vorlesung

1. Aufgabe:

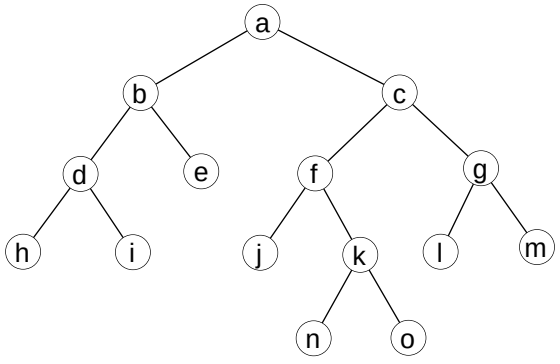
Berechnen Sie für den unten angegeben Baum die Sequenzen an den inneren Knoten für ein optimales uniform geliftetes Alignment gemäß der dynamischen Programmierung in Abschnitt 6.6.6 (insbesondere Seite 383) des Skripts.



d	s_1	s_2	s_3	s_4
s_1	0	1	1	2
s_2		0	2	2
s_3			0	3
s_4				0

2. Aufgabe (Bonus-Aufgabe):

Bestimmen Sie für den unten abgebildeten Baum T sowie $t = 2$ und $i \in [0 : t-1]$ die Darstellung T_i, wie auf Seite 384 des Skripts beschrieben. Dabei sind für jedes T_i die Menge von Teilbäumen von T anzugeben, aus der sich T_i zusammensetzt.



3. Aufgabe (Bonus-Aufgabe):

Bestimmen Sie je einen Baum für die unten angegebene Distanzmatrix nach den folgenden Verfahren: Single-Linkage-Clustering, Average-Linkage-Clustering und Complete-Linkage-Clustering.

Geben Sie dabei auch immer alle Zwischenschritte an.

	A	B	C	D
A	0	1	3	2
B	1	0	7	9
C	3	7	0	6
D	2	9	6	0

Programmieraufgaben

Bei den folgenden drei Programmieraufgaben handelt es sich um zusätzliche Bonus-Aufgaben. Die erzielten Punkte werden für den Noten-Bonus gewertet, die zu erzielenden Punkte jedoch nicht.

Für alle Programmieraufgaben gilt:

- Die Implementierung erfolgt in Java oder alternativ in Python3.
- Die Abgabe erfolgt per **Email** an algo2@bio.ifi.lmu.de als ausführbare *jar*-Datei in der auch die **Source**-Files enthalten sind.
- Überlegen Sie sich geeignete Testfälle um Ihren Code zu überprüfen.

1. Programmieraufgabe: Vorberechnungen (10 Punkte)

Implementieren Sie (für eine gegebene Menge von Sequenzen $s_1, \ldots, s_k \in \Sigma^*$, Gap-Kosten $g \in \mathbb{N}$, Mismatch-Kosten $m \in \mathbb{N}$ und eine globale obere Schranke $C \in \mathbb{N}$) die Berechnung der Vorwärts- und Rückwärts-Matrix $P_{xy}, S_{xy}, \in \mathbb{N}^{|x|\times|y|}$ sowie der oberen Schranken $C_{xy} \in \mathbb{Z}$ für alle paarweisen Alignments der Sequenzen $x = s_i$ und $y = s_j$, $1 \leq i < j \leq k$. Verwenden Sie den Needleman-Wunsch-Algorithmus für globale Alignments mit **linearen** Gapkosten.

Als Ausgabe erwarten wir eine Datei *matrices.mats* im Ausgabeordner *output* in der für alle paarweisen Alignments die Indizes der Sequenzen *i_j*, sowie deren obere Schranke C_{ij} samt der Matrix $\Delta_{xy} = P_{xy} + S_{xy}$ ausgegeben wird. Die Matrix soll Komma-separiert ausgegeben werden. Ein Beispiel für $k = 3$, $g = 3$ und $m = 2$ ist hier angegeben:

```
0_1,-5
_,_,G,A,T,T,C
_,9,9,15,21,27,33
A,13,11,9,15,21,27
T,17,13,11,9,15,21
```

```
T,21,17,13,11,9,15
C,23,21,17,11,11,9
G,27,23,21,15,11,9
G,33,27,23,19,15,9

0_2,-2
_,_,G,G,A,T,T,C
_,12,12,12,18,24,30,36
A,16,12,14,12,18,24,30
T,20,16,12,14,12,18,24
T,22,20,16,12,14,12,18
C,24,22,20,16,12,14,12
G,30,24,22,20,16,12,12
G,36,30,24,22,20,16,12

1_2,-11
_,_,G,G,A,T,T,C
_,3,3,9,15,21,27,33
G,9,3,3,9,15,21,27
A,15,9,5,3,9,15,21
T,21,15,11,7,3,9,15
T,27,21,17,13,7,3,9
C,33,27,23,19,13,9,3
```

Beachten Sie folgende Vorgehensweise bei der **Implementierung**:

- Für Sequenzen $\{s_1, \ldots, s_k\}$ berechnen Sie die paarweisen Alignments $(s_1, s_2), (s_1, s_3), \ldots, (s_1, s_k), (s_2, s_3), \ldots (s_{k-1}, s_k)$. Geben Sie die Alignments in dieser **Reihenfolge** aus!

Beantworten Sie folgende **Fragen**:

(a) Was ist die Interpretation eines Wertes in der Matrix Δ_{ij}?

(b) Welche Werte erwarten Sie auf dem Backtracking-Pfad eines optimalen Alignments?

Beachten Sie bitte folgendes **Interface** zum Aufruf dieser Aufgabe:
`java -jar` *ihrnachname*`.jar --seqs` <$s_1, \ldots, s_k$> `--g` <g> `--m` <m> `--C` <C> `--out` <output-folder> `--only-matrices`.

2. Programmieraufgabe: Carillo-Lipman (10 Punkte)

Im zweiten Aufgabenteil implementieren Sie das eigentliche Alignment nach Carillo-Lipman.

Geben Sie hier bitte eine Datei *alignments.aln* aus, in der das Mehrfache Sequenzalignment für alle Sequenzen angegeben ist. Schreiben Sie in die erste Zeile den Alignment Score. Die nächste Zeile soll enthalten, wie viele Zellen bei der Berechnung besucht wurden. Darunter soll das Mehrfache Sequenzalignment ausgegeben werden. Dabei muss das Alignment die Sequenzen in Ihrer Eingabereihenfolge enthalten. Eine Beispielausgabe wäre also:

```
20
294
--ATTCGG
-GATTC--
GGATTC--
```

Beachten Sie bitte folgendes **Interface** zum Aufruf dieser Aufgabe:
`java -jar` *ihrnachname*`.jar --seqs <`$s_1, \ldots, s_k$`> --g <g> --m <m> --C <C> --out <output-folder>`.

3. Programmieraufgabe: Center-Star-Alignment (10 Punkte)

Implementieren sie das Center-Star Alignment aus der Vorlesung. Verwenden Sie allgemeine Gap-Kosten $g \in \mathbb{N}$ sowie Mismatch-Kosten $m \in \mathbb{N}$.

Geben Sie auf der Konsole den Index der Center-Star-Sequenz aus, sowie alle paarweisen Alignments zu dieser Sequenz.

Schreiben Sie in die Ausgabedatei das Mehrfache Sequenzalignment aller Sequenzen das konsistent ist mit einem Stern mit der Center-Star Sequenz als Zentrum.

Beachten Sie bitte folgendes **Interface** zum Aufruf dieser Aufgabe:
`java -jar` *ihrnachname*`.jar --seqs <`$s_1, \ldots, s_k$`> --g <g> --m <m> --out <alignment-file>`.