Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Volker Heun

Sommersemester 2016
Semestralklausur
21. Juli 2016

Algorithmische Bioinformatik I

Vorname	Name	Matrikelnummer

Reihe	Platz	Unterschrift

Hiermit stimme ich einer Veröffentlichung meines Klausurergebnisses dieser Semestralklausur unter Verwendung meiner Matrikelnummer im Internet zu. *Ja* ☐ *Nein* ☐

(Unterschrift)

Allgemeine Hinweise zur Semestralklausur

- Vor der Prüfung ist diese Seite mit Vornamen, Namen, Matrikelnummer, Reihe und Platz **leserlich mit Druckbuchstaben** zu versehen und zu unterschreiben.
- Bitte **nicht** in *roter* oder *grüner* Farbe bzw. **nicht** mit Bleistift schreiben.
- Der Studentenausweis und ein amtlicher Lichtbildausweis sind bereit zu halten.
- Die reine Bearbeitungszeit beträgt **120 Minuten**.
- Es sind insgesamt 40 Punkte zu erreichen, zum Bestehen sind 17 Punkte nötig.

Viel Erfolg!

Hörsaal verlassen von ..... bis ..... von ..... bis .....

Vorzeitig abgegeben um .....

	Hz	A1	A2	A3	A4	A5	$\sum$
Erstkorrektur							
Nachkorrektur							
Zweitprüfer							

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 1 (8 Punkte)

Gib das Master-Theorem aus der **Vorlesung** an. Spezifiziere hierzu insbesondere die drei verschiedenen Fälle und gib an, welche Lösung der jeweilige Fall besitzt.

Bestimme die Asymptotik von $T(n)$ mithilfe des Master-Theorems aus der **Vorlesung** unter Angabe einer der drei Fälle (siehe oben) mit Begründung bzw. begründe, warum das Master-Theorem nicht anwendbar ist. Es gilt dabei immer $T(1) = 1$:

a) $T(n) = 2 \cdot T(n/2) + \sqrt{n}$,

b) $T(n) = 2 \cdot T(n/4) + \sqrt{n}\log(n)$,

c) $T(n) = 3 \cdot T(n/3) + n\sqrt{n}$.

Lösungsskizze

Seien $a, b, d \in \mathbb{N}$ mit $b > 1$, sei $f(n)$ eine Funktion und sei $T(n)$ definiert durch die Rekursionsgleichung $T(n) = a \cdot T(n/b) + f(n)$ für $n > 1$ und $T(1) = d$. Dann gilt:

$$T(n) = \begin{cases} \Theta(n^{\log_b(a)}) & \text{falls } f(n) = O(n^{\log_b(a)-e}) \text{ für ein konstantes } e > 0 \\ \Theta(n^{\log_b(a)}\log(n)) & \text{falls } f(n) = \Theta(n^{\log_b(a)}) \\ \Theta(f(n)) & \text{falls } f(n) = \Omega(n^{\log_b(a)+e}) \text{ für ein konstantes } e > 0 \\ & \text{und } a \cdot f(n/b) \leq c \cdot f(n) \text{ für ein konstantes } c < 1 \end{cases}$$

a) Für das Master-Theorem erhalten wir $a = 2$, $b = 2$ und $f(n) = \sqrt{n}$. Es gilt $\log_2(2) = 1$ und somit $f(n) = \sqrt{n} = n^{1/2} = O(n^{\log_2(2)-e}) = O(n^{\log_b(a)-e})$ für $e \in (0, 1/2)$. Somit gilt der erste Fall und es ist $T(n) = \Theta(n^b \log(a)) = \Theta(n^{\log_2(2)}) = \Theta(n)$.

b) Für das Master-Theorem erhalten wir $a = 2$, $b = 4$ und $f(n) = \sqrt{n}\log(n)$. Es gilt $\log_b(a) = \log_4(2) = 1/2$.

Also ist $f(n) = \sqrt{n}\log(n) = \omega(n^{1/2})$ und damit $f(n) \neq O(n^{1/2-e}) = O(n^{\log_b(a)-e})$ für alle $e > 0$ und Fall 1 ist nicht zutreffend.

Weiter ist $f(n) = \sqrt{n}\log(n) = \omega(n^{1/2})$ und daher $f(n) \neq \Theta(n^{1/2}) = \Theta(n^{\log_b(a)})$. Fall 2 trifft also nicht zu.

Auch ist $f(n) = \sqrt{n}\log(n) \neq \Omega(n^{1/2+e})$ für alle $e > 0$, da $\sqrt{n}\log(n) = o(n^{1/2+e})$ für alle $e > 0$. Das Master-Theorem ist also auch im Fall 3 nicht anwendbar.

c) Für das Master-Theorem erhalten wir $a = 3$, $b = 3$ und $f(n) = n\sqrt{n}$. Es gilt $\log_3(3) = 1$ und somit $f(n) = n\sqrt{n} = n^{3/2} = \Omega(n^{\log_3(3)+e}) = \Omega(n^{\log_b(a)+e})$. für ein $e \in (0, 1/2)$, Weiter ist

$$a \cdot f(n/b) = 3 \cdot (n/3)\sqrt{\frac{n}{3}} \leq \frac{1}{\sqrt{3}} \cdot n^{3/2} \overset{!}{\leq} c \cdot f(n)$$

und somit gilt der dritte Fall des Master-Theorems mit $c = 1/\sqrt{3} < 1$ (da $\sqrt{3} > 1$). Damit gilt $T(n) = \Theta(f(n)) = \Theta(n\sqrt{n})$.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 2 (8 Punkte)

Betrachte den unter a) abgebildeten Suffix-Baum für $s = s_1 \cdots s_7 = aababab$. Der besseren Lesbarkeit wegen sind hierbei immer explizit die Kantenlabels statt der Referenzen angegeben.

a) Zeichne alle Suffix-Links ein, die Ukkonens Algorithmus hierfür konstruiert hat.

b) Gib die Kantenlabels so an, wie sie in Ukkonens Algorithmus verwendet werden.

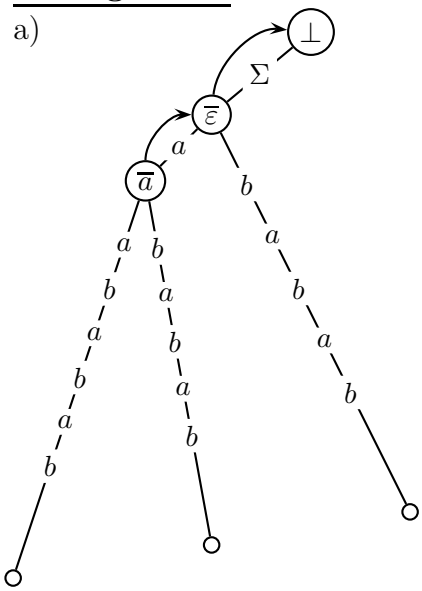
c) Führe Ukkonens Algorithmus für den Übergang von s auf $s' = s{\cdot}a = aabababa$ aus.

d) Führe Ukkonens Algorithmus für den Übergang von s' auf $s'' = s'{\cdot}a = aababababaa$ aus.

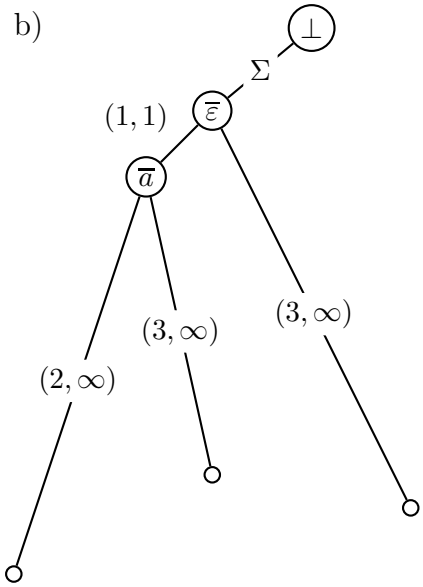
Gib für c) und d) alle Zwischenschritte an, markiere insbesondere die Position des aktiven Knotens und Endknotens im jeweiligen Suffix-Baum. Zeichne dabei nur die verwendeten und neu eingetragenen Suffix-Links mit jeweils einer anderen Farbe ein und nummeriere die neuen Blätter in der Reihenfolge der Einfügung.

Lösungsskizze

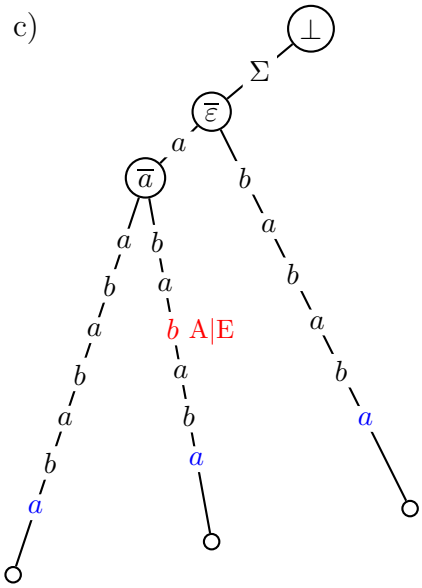
a)



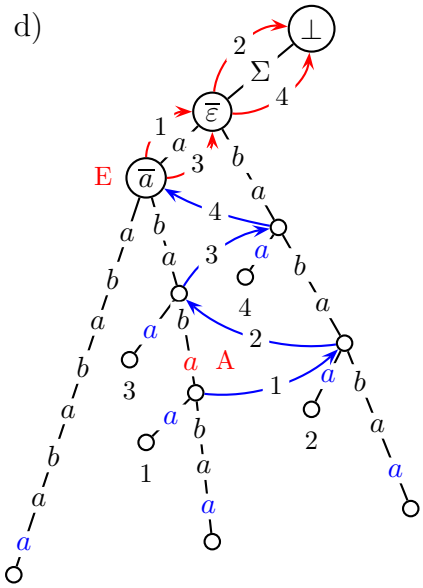
b)



c)



d)



Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 3 (8 Punkte)

Löse die folgende Rekursionsgleichung **mit Hilfe der allgemeinen Lösung für lineare Rekursionsgleichungen**:

$$f_n = 4 \cdot f_{n-1} - 2 \quad \text{für } n \geq 1, \qquad \text{und} \qquad f_0 = 1.$$

Lösungsskizze

Betrachte die Rekursionsgleichung für n und $n-1$:

$$\begin{aligned} f_n &= 4 \cdot f_{n-1} - 2, \\ f_{n-1} &= 4 \cdot f_{n-2} - 2. \end{aligned}$$

Ziehen wir die zweite von der ersten Gleichung ab, so erhalten wir eine homogene lineare Rekursionsgleichung

$$f_n = 5 \cdot f_{n-1} - 4 \cdot f_{n-2} \qquad \Longleftrightarrow \qquad f_n - 5 \cdot f_{n-1} + 4 \cdot f_{n-2} = 0$$

mit den Anfangsbedingungen $f_0 = 1$ und $f_1 = 4 \cdot f_0 - 2 = 2$.

Wir betrachten nun das charakteristische Polynom $\chi(n)$ für diese Rekursionsgleichung:

$$\chi(n) = n^2 - 5n + 4.$$

Die Nullstellen ergeben sich (beispielsweise mit der p, q-Formel) zu

$$x_{1,2} = \frac{5}{2} \pm \sqrt{\frac{25}{4} - 4} = \frac{5}{2} \pm \sqrt{\frac{9}{4}}.$$

Also sind 1 und 4 die beiden Nullstelle des charakteristischen Polynoms. Somit hat die Lösung der Rekursionsgleichung die Form:

$$f_n = a \cdot (1)^n + b \cdot (4)^n = a + b \cdot 4^n.$$

Mit den Anfangsbedingungen ergibt sich für a und b:

$$\begin{aligned} 1 &= f_0 = a \cdot (1)^0 + b \cdot (4)^0 = a + b, \\ 2 &= f_1 = a \cdot (1)^1 + b \cdot (4)^1 = a + 4 \cdot b. \end{aligned}$$

Subtraktion der ersten von der zweiten Gleichung liefert $3b = 1$, also $b = \frac{1}{3}$. Somit ist $a = \frac{2}{3}$, also ist $f_n = \frac{2}{3} + \frac{1}{3} \cdot 4^n$.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 4 (8 Punkte)

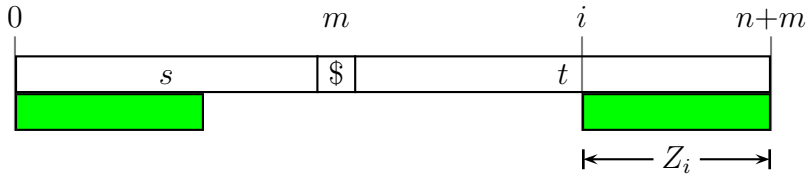
Gegeben seien zwei Wörter $s = s_1 \cdots s_m \in \Sigma^m$ und $t = t_1 \cdots t_n \in \Sigma^n$. Gib einen Algorithmus mit Laufzeit $O(n + m)$ an, der das längste Präfix von s findet, das auch ein Suffix von t ist.

Beispiel: Für $s = ababbaaaa$ und $t = bbbababb$ sind ε und $ababb$ jeweils sowohl ein Präfix von s als auch ein Suffix von t.

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen!

Lösungsskizze

Zuerst schauen wir uns an, wie ein Präfix von s, das auch ein Suffix von t ist in der Zeichenreihe $s\$t$ aussieht:



Sei $\ell := |s\$t| = n + m + 1$. Gilt also $i + Z[i] = \ell$ für ein $i \in [m + 1 : \ell - 1]$, dann endet ein Präfix von s als Suffix von t an Position $i - m$ in t.

Wir konstruieren also für $s\$t$ zuerst die zugehörigen Z-Werte in Zeit $O(n + m)$. Dann testen wir für jedes $i \in [m + 1 : \ell - 1]$ in aufsteigender Reihenfolge, ob $i + Z[i] = \ell$ gilt. Dies ist in Zeit $O(n + m)$ möglich. Der erste Wert, für den das gilt, ist nach obiger Erläuterung ein Suffix von t, der ein Präfix von s ist. Nach Wahl von i, muss dieses dann auch das längste Präfix von s und wir haben die gesuchte Lösung unseres Problems.

[Alternative Lösungsmöglichkeiten mit Suffix-Bäumen oder der Border-Tabelle.]

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 5 (8 Punkte)

Ein *spezielles Alignment* für zwei Sequenzen $s \in \Sigma^n$ und $t \in \Sigma^m$ ist ein globales paarweises Sequenzen-Alignment $(\overline{s}, \overline{t}) \in \mathcal{A}(s,t)$ mit der Einschränkung, dass auf ein Indel (Insertion bzw. Deletion) keine Substitution folgen darf (also rechts davon stehen darf), jedoch ein Match oder ein Indel.

Beispiel: Für $s = AAAAC$ und $t = ATTC$ ist $\binom{AAA-AC}{-ATT-C}$ ein spezielles Alignment (allerdings nicht notwendigerweise ein optimales), $\binom{AA\boldsymbol{A}-AC}{A-\boldsymbol{T}T-C}$ oder $\binom{AAA-\boldsymbol{A}C}{A--T\boldsymbol{T}C}$ jedoch nicht.

Finde einen möglichst effizienten Algorithmus, der für zwei gegebene Sequenzen $s \in \Sigma^n$ und $t \in \Sigma^m$ ein optimales spezielles Alignment bzgl. der Alignment-Distanz mit linearer Lückenstrafe findet. Hierbei trägt jede Insertion und jede Deletion 2 sowie jede Substitution 3 zur Alignment-Distanz bei, ein Match wie üblich 0.

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen!

Lösungsskizze

Wir konstruieren wie beim Gotoh-Algorithmus verschiedene Matrizen D, E, F und G. Dabei ist $D[i,j]$ bzw. $E[i,j]$ bzw. $F[i,j]$ bzw. $G[i,j]$ die Alignment-Distanz eines speziellen Alignments von $s_1 \cdots s_i$ mit $t_1 \cdots t_j$, das keine weitere Einschränkung besitzt bzw. mit einem Indel bzw. mit einem Match bzw. einer Substitution endet. Für die Rekursionsgleichungen erhalten wir dann:

$$
\begin{aligned}
E[i,j] &= \min\{D[i,j-1]+2, D[i-1,j]+2\} \\
F[i,j] &= \begin{cases} D[i-1,j-1] & \text{falls } s_i = t_j \\ \infty & \text{sonst} \end{cases} \\
G[i,j] &= \begin{cases} \infty & \text{falls } s_i = t_j \\ \min\{F[i-1,j-1]+3, G[i-1,j-1]+3\} & \text{sonst} \end{cases} \\
D[i,j] &= \min\{E[i,j], F[i,j], G[i,j]\}
\end{aligned}
$$

Es stellt sich nun noch die Frage, welche Werte jeweils in der 1. Zeile bzw. in der 1. Spalte der Matrizen stehen. Es gilt für $i > 0$ und $j > 0$:

$$
\begin{array}{lll}
E[0,j] = 2*j, & F[i,0] = \infty, & G[i,0] = \infty, \\
E[i,0] = 2*i, & F[0,j] = \infty, & G[0,j] = \infty, \\
E[0,0] = \infty, & F[0,0] = 0, & G[0,0] = 0.
\end{array}
$$

Die Werte für D ergeben sich aus der Minimumsbildung.

Ein spezielles Alignment selbst findet man wieder über den Traceback ausgehend von $D[n,m]$, wobei man berücksichtigen muss, aus welcher Matrix das Minimum stammt.

Die Korrektheit folgt aus der Tatsache, dass ein optimales spezielles Alignment in der letzten Spalte entweder ein Indel, ein Match oder eine Substitution besitzt. Hier werden gemäß der Definition eines speziellen Alignment für alle drei Fälle nach in den obigen Matrizen E, F und G jeweils eine korrekte Alignment-Distanz für das entsprechende spezielle Alignment berechnet.

Jeder Eintrag der vier Tabellen lässt sich in konstanter Zeit ermitteln, von daher wird für das Erstellen der Tabellen Zeit $O(nm)$ benötigt. Ein spezielles Alignment kann mit dem Traceback in Zeit $O(n+m)$ erstellt werden, so dass die Gesamtlaufzeit $O(nm)$ ist.