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

Wintersemester 2010/11
Semestralklausur
27. Juli 2010

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 **150 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	A6	A7	$\sum$
Erstkorrektur									
Nachkorrektur									
Zweitprüfer									

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 1 (4 Punkte)

Bestimme die Asymptotik von $T(n)$ mithilfe des Master-Theorems (es gilt dabei immer $T(1) = 1$):

a) $T(n) = 4 \cdot T(n/2) + n^3$,

b) $T(n) = 8 \cdot T(n/2) + n^2$.

Lösungsskizze

a) Es gilt $n^3 = \Omega(n^{\log_2(4)+\epsilon})$, da $\log_2(4) = 2$ und für ein hinreichend kleines $\epsilon > 0$, beispielsweise mit $\epsilon = \frac{1}{2}$. Weiter ist

$$4(n/2)^3 = \frac{1}{2} \cdot n^3$$

und somit gilt der dritte Fall des Master-Theorems mit $c = 1/2 < 1$ und es ist $T(n) = \Theta(n^3)$.

b) Da $\log_2(8) = 3$, gilt $n^2 = O(n^{3-\epsilon})$, beispielsweise mit $\epsilon = \frac{1}{2}$. Also gilt der erste Fall des Master-Theorems und es ist $T(n) = \Theta(n^{\log_2(8)}) = \Theta(n^3)$.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 2 (6 Punkte)

Betrachte den unter a) abgebildeten Suffix-Baum für $s = s_1 \cdots s_6 = abaaba$. 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.

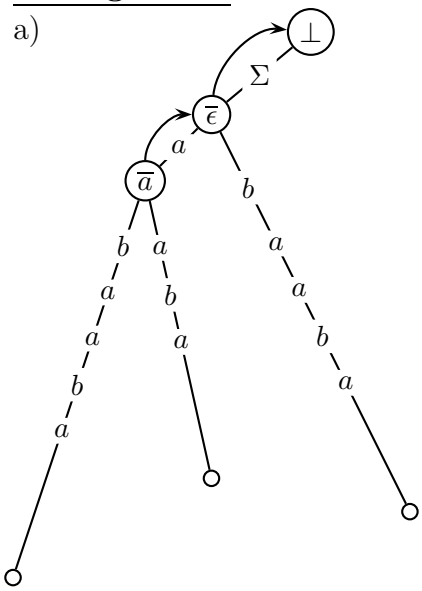
c) Führe Ukkonens Algorithmus für den Übergang von s auf $s' = s \cdot a = abaabaa$ aus.

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

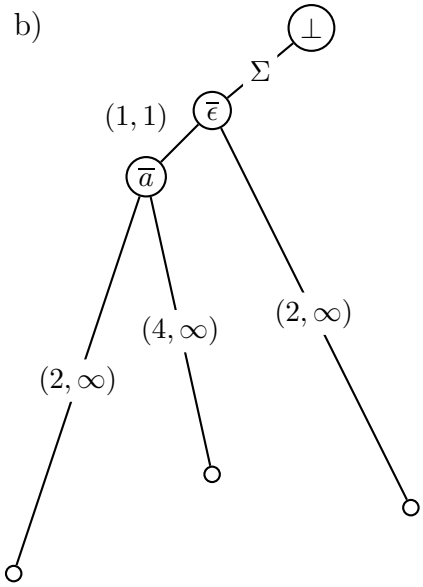
Gib für c) und d) alle Zwischenschritte an, markiere insbesondere die Position des aktiven Knotens und Endknotens im jeweiligen Suffix-Baum. Zeichne dabei nur die verwendeten und neu eingetragenen Suffix-Links mit jeweils einer anderen Farbe ein und nummeriere die neuen Blätter in der Reihenfolge der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

Lösungsskizze

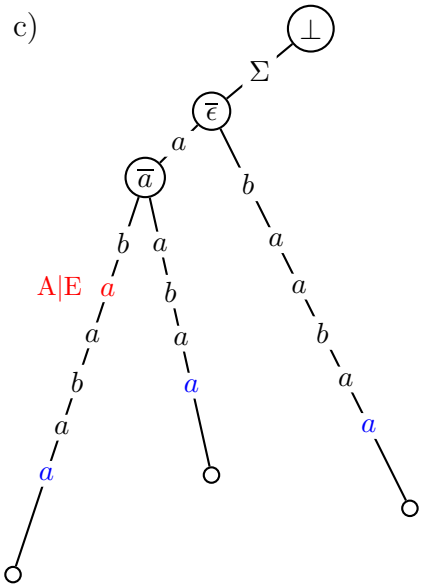
a)



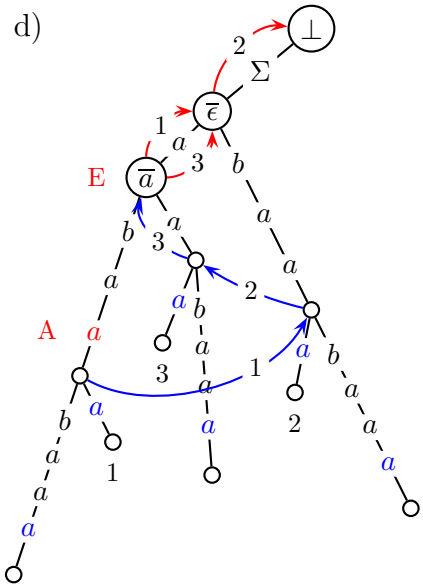
b)



c)



d)



Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 3 (6 Punkte)

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

$$f_n = 3f_{n-1} + 1 \quad \text{für } n \geq 1, \qquad \text{und} \qquad f_0 = 0.$$

Lösungsskizze

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

$$\begin{aligned} f_n &= 3f_{n-1} + 1, \\ f_{n-1} &= 3f_{n-2} + 1. \end{aligned}$$

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

$$f_n = 4f_{n-1} - 3f_{n-2}$$

mit den Anfangsbedingungen $f_0 = 0$ und $f_1 = 3 \cdot f_0 + 1 = 1$.

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

$$\chi(n) = n^2 - 4n + 3.$$

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

$$x_{1,2} = 2 \pm \sqrt{4-3}.$$

Also sind 3 und 1 die Nullstellen des charakteristischen Polynoms. Somit hat die Lösung der Rekursionsgleichung die Form:

$$f_n = a \cdot 3^n + b \cdot 1^n.$$

Mit den Anfangsbedingungen ergibt sich für a und b:

$$\begin{aligned} 0 &= f_0 = a \cdot 3^0 + b \cdot 1^0 = a + b, \\ 1 &= f_1 = a \cdot 3^1 + b \cdot 1^1 = 3a + b. \end{aligned}$$

Zieht man die erste von der zweiten Gleichung ab, so erhält man $2a = 1$ und somit $a = 1/2$ sowie weiter mit der ersten Gleichung $b = -a = -1/2$. Also ist $f_n = \frac{1}{2}(3^n - 1)$.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 4 (6 Punkte)

Berechne

$$\sum_{i=0}^{n-1} \frac{i}{(i+1)(i+2)}$$

mithilfe der diskreten Stammfunktion und partieller Integration.

Lösungsskizze

Nach der partiellen Integration gilt $\sum (f \cdot \Delta g) = fg - \sum (\Delta f)(Eg)$ mit $f(x) = x$ und $g(x) = -x^{\underline{-1}}$ und daher $(\Delta g)(x) = x^{\underline{-2}} = \frac{1}{(x+1)(x+2)}$:

$$\begin{aligned}
\sum_{i=0}^{n-1} \frac{i}{(i+1)(i+2)} &= \sum_{i=0}^{n-1} i \cdot \frac{1}{(i+1)(i+2)} \\
&= \sum_{i=0}^{n-1} i \cdot i^{\underline{-2}} \\
&= \left[i \cdot (-i^{\underline{-1}})\right]_0^n - \sum_{i=0}^{n-1} 1 \cdot (-(i+1)^{\underline{-1}}) \\
&= \left[-\frac{i}{i+1}\right]_0^n - \sum_{i=1}^{n} 1 \cdot (-i^{\underline{-1}}) \\
&= -\frac{n}{n+1} + \sum_{i=1}^{n} \frac{1}{i+1} \\
&= -\frac{n+1-1}{n+1} + \sum_{i=2}^{n+1} \frac{1}{i} \\
&= -1 + \frac{1}{n+1} + H_n + \frac{1}{n+1} - 1 \\
&= H_n - 2 + \frac{2}{n+1}
\end{aligned}$$

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 5 (6 Punkte)

Sei $s \in \Sigma^m$ und $t \in \Sigma^n$. Weiter sei b' die Border-Tabelle für $s\$t$ mit $\$ \notin \Sigma$. Entwirf einen Algorithmus, der in Linearzeit alle Vorkommen von s in t findet und dabei nur auf die Border-Tabelle b' zurückgreift.

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen!

Lösungsskizze

Der Wert von $b'[m+1+i]$ ist die Länge des eigentlichen Randes von $s\$t_1 \cdots t_{i-1}$. Da $\$ \notin \Sigma$ gilt $b'[m+1+i] \leq m$ für alle $i \in [0:n]$. Wenn jedoch $b'[m+1+i] = m$ ist der eigentliche Rand von $s\$t_1 \cdots t_m$ gerade s. Somit endet genau dann ein Treffer von s an Position $i-1$ in t, wenn $b'[m+1+i] = m$.

Der Algorithmus durchsucht also ab Position m alle Einträge in b' auf den Wert m. Ist $b'[m+1+i] = m$, so endet an Position $i-1$ in t ein Vorkommen von s, d.h. an Position $i-1-(m-1) = i-m$ beginnt ein Treffer von s in t.

Das Erstellen der Border-Tabelle kostet $O(|s\$t|) = O(n+m)$. Der Durchlauf und Test kann in Zeit $O(n)$ erledigt werden. Der Algorithmus hat also eine Laufzeit von $O(n+m)$. Da die Eingabegröße $|s| + |t| = n + m$ ist, ist die Laufzeit linear.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 6 (6 Punkte)

Gegeben seien zwei Sequenzen $s, t \in \Sigma^*$ mit $|s| = n$ und $|t| = m$. Ein Alignment $(\overline{s}, \overline{t}) \in \overline{\Sigma}^* \times \overline{\Sigma}^*$ für s und t heißt *schief*, wenn gilt:

- $\overline{s} = s$,
- $|\overline{s}| = |\overline{t}|$,
- $\exists j, k \in [0, m] : \overline{t}|_\Sigma = t_j \cdots t_k$.

Anschaulich ist ein schiefes Alignment ein Alignment eines Teilwortes von t mit dem Wort s, so dass in der Sequenz s keine Leerzeichen eingefügt werden dürfen (im Alignment $(\overline{s}, \overline{t})$ gibt es also keine Insertionen).

Gib einen Algorithmus mit Laufzeit $O(nm)$ an, der ein optimales schiefes Alignment berechnet, wobei $w : \overline{\Sigma}_0^2 \to \mathbb{R}$ die Kostenfunktion für das Ähnlichkeitsmaß sei und eine lineare Lückenstrafe zugrunde gelegt wird.

Begründe sowohl die Korrektheit als auch die Laufzeit Deines Algorithmus.

Lösungsskizze

Wir verwenden wieder dynamische Programmierung. Dazu sei für $i \in [0 : n]$ und $j \in [0 : m]$ die Matrix S wie folgt definiert: $S[i, j]$ ist der Wert eines optimalen Alignments von $s_1 \cdots s_i$ mit $t_{j'} \cdots t_j$ (für alle $j' \in [0 : j + 1]$). Da keine Einfügungen erlaubt sind, endet ein solches Alignment mit einer Substitution oder einer Deletion.

$$S(i, j) = \max\{S(i - 1, j - 1) + w(s_i, t_j), s(i - 1, j) + w(s_i, -)\}.$$

Da Präfixe von t nicht zu alignieren sind, kosten Einfügungen zum Beginn nichts (ansonsten sind sie verboten) und die Initialisierung der ersten Zeile sieht wie folgt aus:

$$S(0, j) = 0.$$

Löschungen von s werden hingegen ganz normal bezahlt:

$$S(i, 0) = \sum_{k=1}^{i} w(s_k, -) = S(i - 1) + w(s_i, -).$$

Den Score einer optimalen Lösung finden wir im maximalen Wert der letzten Zeile (um die Einfügungen der Suffixe von t ignorieren zu können).

Für das Traceback starten wir im Maximum der letzten Zeile und verfolgen die Sieger der Maxima rückwärts bis wir in der ersten Zeile ankommen. Den Restweg zum Punkt $(0, 0)$ ignorieren wir.

Die Laufzeit zum Ausfüllen des Matrix s ist $O(nm)$, da jede Zelle in konstanter Zeit berechnet werden kann. Das Alignment selbst kann auch wieder in Zeit $O(n+m)$ gefunden werden, sofern die Matrix S gegeben ist.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 7 (6 Punkte)

Für ein Wort $w = w_1 \cdots w_m \in \Sigma^m$ bezeichnet $w^R = w_m \cdots w_1 \in \Sigma^m$ das *gespiegelte Wort* zu w.

Konstruiere einen Algorithmus mit linearer Laufzeit, der die Anzahl aller Teilwörter (nicht Vorkommen!) ausgibt, deren gespiegeltes Wort ebenfalls in $t \in \Sigma^n$ vorkommt (das leere Wort braucht hierbei nicht berücksichtigt zu werden).

Beispiel: Für ANANAS lautet die Antwort 10 (A, N, S, AN, NA, ANA, NAN, ANAN, NANA, ANANA)

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen!

Lösungsskizze

Wir betrachten das Wort $t\sharp t^R\$$. Hierfür konstruieren wir den Suffix-Baum T, dessen Größe $O(|t\sharp t^R\$|) = O(2n+2) = O(n)$ ist. In diesem entfernen wir alle Teilbäume, die über das Zeichen $\sharp$ erreichbar sind (das Zeichen $\sharp$ selbst belassen wir dabei im Baum). Dabei müssen wir für jede Kanten mit Label (p, q) testen, ob $n+1 \in [p : q]$ gilt ($n+1$ ist die Position von $\sharp$ in $t\sharp t^R\$$). Dieser Baum T' enthält nun alle Suffixe von $t\sharp$ sowie $t^R\$$. Man überlege sich, dass jeder Knoten weiterhin mindestens zwei Kinder besitzt (was für die weitere Betrachtung aber nicht wichtig ist). Dieser Suffix-Baum und seine Beschneidung kann mit Hilfe von Ukkonens Algorithmus in Zeit $O(n)$ konstruiert werden.

Jeder Knoten und jede Position, von dem man jetzt ein Blatt erreichen kann, das auf das Zeichen $\sharp$ endet, beschreibt ein Präfix eines Suffixes von t, also einem Teilwort von t. Jeder Knoten und jede Position, von dem man jetzt ein Blatt erreichen kann, das auf das Zeichen \$ endet, beschreibt ein Präfix eines Suffixes von t^R, also einem Teilwort von t^R. Wir müssen also nur alle Knoten (und implizit alle Positionen) markieren, von denen sowohl ein Blatt erreichbar ist, das mit $\sharp$ endet, als auch ein Blatt erreichbar ist, das mit \$ endet. Dies kann mit einer Tiefensuche in Zeit $O(n)$ erreicht werden.

Alle Positionen die sich nun oberhalb solcher markierter Knoten befinden, sind Teilwörter von t, die auch Teilwörter von t^R sind, d.h. deren Spiegelung ein Teilwort von t ist. Man überlege sich, dass die markierten Knoten die Wurzel beinhalten und selbst einen Teilbaum von T' bilden müssen.

Da wir nicht die Anzahl der Vorkommen, sondern die Anzahl solcher Wörter finden wollen, müssen wir nur für jede Kante zu einem markierten Knoten mit Kantenlabel (p, q) den Wert $q - p + 1$ addieren. Falls $q = \infty$ sein sollte, gilt hier natürlich $q = 2n + 2$, die Länge des Wortes für den der Suffix-Baum konstruiert wurde. Die kann wieder mit einer weiteren Tiefensuche in Zeit $O(n)$ erledigt werden. Insgesamt ist der Zeitbedarf also linear in n.