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

Sommersemester 2016
Wiederholungsklausur
19. Oktober 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) $T(n) = 2 \cdot T(n/4) + \sqrt{n}$,

b) $T(n) = 4 \cdot T(n/3) + n^2$.

c) $T(n) = 3 \cdot T(n/3) + n/\log(n)$.

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 $a = 2$, $b = 4$ und $f(n) = \sqrt{n}$. Es gilt $\log_4(2) = \frac{1}{2}$ und somit $f(n) = \sqrt{n} = n^{1/2} = \Theta(n^{\log_2(4)}) = \Theta(n^{\log_b(a)})$. Also gilt der zweite Fall des Master-Theorems und es ist $T(n) = \Theta(f(n) \log(n)) = \Theta(\sqrt{n} \log(n))$.

b) Für das Master-Theorem erhalten wir $a = 4$, $b = 3$ und $f(n) = n^2$. Da offensichtlich $1 < \log_3(4) < 2$ gilt, gilt auch $\log_3(4) = 1 + e$ für ein geeignetes $e \in (0, 1)$. Damit erhalten wir $f(n) = n^2 = \Omega(n^{1+e}) = \Omega(n^{\log_b(a)+e})$ für das obige $e \in (0, 1/2)$. Weiter ist

$$4 \cdot \left(\frac{n}{3}\right)^2 \leq \frac{4}{9} \cdot n^2 \overset{!}{\leq} c \cdot f(n)$$

und somit gilt der dritte Fall des Master-Theorems mit $c = \frac{4}{9} < 1$ und wir erhalten $T(n) = \Theta(f(n)) = \Theta(n^2)$.

c) Für das Master-Theorem erhalten wir $a = 3$, $b = 3$ und $f(n) = n/\log(n)$. Es gilt $\log_b(a) = \log_3(3) = 1$.

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

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

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

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 2 (8 Punkte)

Betrachte die folgende Suchwortmenge $S = \{a, abcbcbc, b, bcbca, cb, cbca\}$.

a) Konstruiere einen Suchwort-Baum für S nach Aho-Corasick;

b) Konstruiere die Failure-Links in diesem Suchwort-Baum;

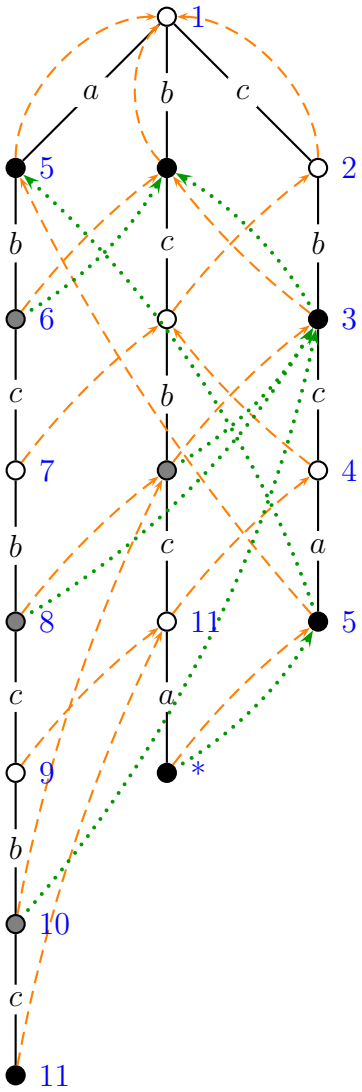
c) Markiere nach dem in der Vorlesung angegebenen Algorithmus von Aho und Corasick alle Knoten darin, die einem Suchwort aus S entsprechen;

d) Markiere nach dem in der Vorlesung angegebenen Algorithmus von Aho und Corasick die Knoten, für die Treffer ausgegeben werden und gebe die zugehörigen Hit-Links an.

e) Wende den Algorithmus von Aho-Corasick mit dem konstruierten Suchwort-Baum auf das folgende Wort an: $t_1 \cdots t_{11} = cbcabcbcbca$.

Hinweis: Verwende verschiedene Farben, aus denen ersichtlich wird, welche Teile des Baumes (bzw. welche Annotationen) zu welchem Aufgabenteil gehören (zeichne ggf. den Baum mehrmals).

Lösungsskizze



Failure-Links sind orange und gestrichelt, Hit-Links grün und gepunktet dargestellt. Die blaue Zahl i gibt den Knoten an, an dem versucht wird, mit s_i weiterzuarbeiten; Bei * befindet sich der Algorithmus am Ende.

Es werden dabei die folgenden Treffer an der entsprechenden Endposition ausgegeben:

cb, b	@	2
$cbca, a$	@	4
b	@	5
cb, b	@	7
cb, b	@	9
$abcbcbc$	@	10
$bcbca, cbca, a$	@	11

Der Lesbarkeit wegen werden hier die Worte der Treffermenge ausgegeben, normalerweise wird die Länge bzw. die daraus berechnete Startposition ausgegeben.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 3 (8 Punkte)

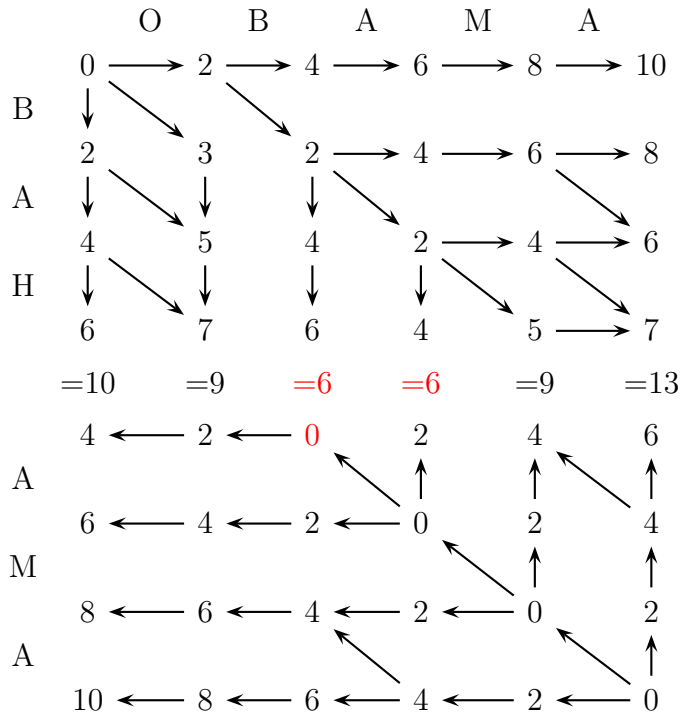
Betrachte die Wörter s = BAHAMA und t = OBAMA. Berechne den ersten Schritt des **Hirschberg-Algorithmus** bei einem globalen Sequenzen-Alignment für s und t zur Rekonstruktion des Tracebacks. Bestimme insbesondere den bzw. die Schnittpunkte der Wörter s und t, d.h. die Teilwörter, für die der Hirschberg-Algorithmus rekursiv aufgerufen wird und gib das bzw. die zugehörigen Alignments an.

Gib dazu sowohl die beiden Tabellen zur Ermittlung der Schnittpunkte an und zeichne in diesen Tabellen auch die Traceback-Pfeile ein (die vom Hirschberg-Algorithmus nicht verwendet werden). Die benötigten rekursiv konstruierten Alignments dürfen aus diesen Tabellen abgelesen werden.

Die Kostenfunktion für ein Distanzmaß sei dabei mit 0 für ein Match, mit 3 für eine Substitution und mit 2 für eine Indel-Operation gegeben.

Begründe kurz in eigenen Worten, warum die Rekonstruktion eines optimalen Alignments beim Hirschberg-Algorithmus linearen Platz benötigt.

Lösungsskizze



Damit liegt der Schnittpunkt bei $(3, 2)$, d.h. $BAH|AMA$ versus $OB|AMA$, oder $(3, 3)$, d.h. $BAH|AMA$ versus $OBA|MA$. Die beiden Alignments lauten:

$$\begin{pmatrix} -BAHAMA \\ OB--AMA \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} -BAHAMA \\ OBA--MA \end{pmatrix}.$$

Der Divide-Schritt benötigt jeweils linearen Platz $O(n)$, da nur die letzten beiden Zeilen der DP-Matrix zu speichern sind. Da der Platz in den rekursiven Aufrufen wiederverwendet werden kann ist der Platzbedarf hierfür insgesamt linear. Im Conquer-Schritt wird nur das Alignment aufgebaut, das insgesamt linearen Platz $O(n + m)$ benötigt.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 4 (8 Punkte)

Zeige, dass gilt:

$$\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 \ \in \ \Theta(n^9).$$

Hinweis: Verwende dabei geeignete Abschätzungen.

Lösungsskizze

Es gilt:

$$\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 \ \leq \ \sum_{i=1}^{n} n^4 \cdot n^4 \ = \ \sum_{i=1}^{n} n^8 \ = \ n^9.$$

Also ist $\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 \ \in \ O(n^9)$.

Weiter gilt

$$\begin{aligned}
\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 &\geq \sum_{i=\lceil \frac{n}{3} \rceil}^{\lfloor \frac{2n}{3} \rfloor} i^4 \cdot (n-i+1)^4 \\
&\geq \sum_{i=\lceil \frac{n}{3} \rceil}^{\lfloor \frac{2n}{3} \rfloor} \left\lceil \frac{n}{3} \right\rceil^4 \cdot \left(n - \left\lfloor \frac{2n}{3} \right\rfloor + 1 \right)^4 \\
&\geq \sum_{i=\lceil \frac{n}{3} \rceil}^{\lfloor \frac{2n}{3} \rfloor} \left(\frac{n}{3} \right)^4 \cdot \left(n - \frac{2n}{3} \right)^4 \\
&\geq \sum_{i=\lceil \frac{n}{3} \rceil}^{\lfloor \frac{2n}{3} \rfloor} \left(\frac{n}{3} \right)^8 \\
&= \left(\left\lfloor \frac{2n}{3} \right\rfloor - \left\lceil \frac{n}{3} \right\rceil + 1 \right) \cdot \left(\frac{n}{3} \right)^3 \\
&\geq \left(\left(\frac{2n}{3} - 1 \right) - \left(\frac{n}{3} + 1 \right) + 1 \right) \cdot \left(\frac{n}{3} \right)^3 \\
&\geq \left(\frac{n}{3} - 1 \right) \cdot \left(\frac{n}{3} \right)^8 \\
&\quad \tfrac{n}{3} - 1 \geq \tfrac{n}{6} \text{ für } n \geq 6 \\
&\geq \frac{n}{6} \cdot \left(\frac{n}{3} \right)^8 \\
&\geq \left(\frac{n}{6} \right)^9
\end{aligned}$$

Also gilt $\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 \ \in \ \Omega(n^9)$.

Zusammen gilt also $\sum_{i=1}^{n} i^4 \cdot (n-i+1)^4 \ \in \ \Theta(n^9)$.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 5 (8 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 möglichst effizienten Algorithmus, der für ein Wort $t \in \Sigma^n$ ein längstes Teilwort von t findet, dessen gespiegeltes Wort ebenfalls in t vorkommt.

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen!

Lösungsskizze

Wir konstruieren nun für $t\sharp t^R\$$ mit $\sharp, \$ \notin \Sigma$ mit $\sharp \neq \$$ den zugehörigen Suffix-Baum T. Die Größe des Suffix-Baumes beträgt $O(2n) = O(n)$. Mit einer Tiefensuche entfernen wir alle Teilbäume, die über das Zeichen $\sharp$ erreichbar sind. Somit enden nun alle Pfade im Suffix-Baum mit dem Zeichen $\sharp$ oder \$.

Mit einer weiteren Tiefensuche markieren wir alle Knoten des Baumes, von denen sowohl ein Blatt erreichbar ist, dessen Kantenlabel mit $\sharp$ endet, als auch ein Blatt erreichbar ist, dessen Kantenlabel mit \$ endet. Somit ist jedes Wort w, das zu einem Pfad von der Wurzel zu einem markierten Knoten des Suffix-Baumes korrespondiert, sowohl ein Präfix eines Suffixes von t als auch ein Präfix eines Suffixes von t^R. Das Wort w ist also sowohl Teilwort von t als auch von t^R. Damit taucht neben dem Wort w in t auch w^R in $(t^R)^R = t$ auf, da w auch in t^R auftaucht.

Somit müssen wir mit einer weiteren Tiefensuche nur einen tiefsten (bzgl. der String-Tiefe) Knoten in diesem Suffix-Baum finden. Da sowohl die Konstruktion des Suffix-Baumes in Zeit $O(2n)$ als auch die drei Tiefensuchen nur jeweils $O(n)$ Zeit benötigen, ist die Gesamtaufzeit $O(n)$.