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

Sommersemester 2022
Wiederholungsklausur
11. Oktober 2022

Algorithmische Bioinformatik I

Vorname	Name	Matrikelnummer

Reihe	Platz	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) $T(n) = 3 \cdot T(n/4) + \log(n)$.

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

c) $T(n) = 8 \cdot T(n/2) + n^3$.

Lösungsskizze

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

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

a) Für das Master-Theorem erhalten wir $a = 3$, $b = 4$ und $f(n) = \log(n)$. Es gilt $\log_4(3) \in (0,1)$ und somit $f(n) = \log(n) = O(n^{\log_4(3)-e}) = O(n^{\log_b(a)-e})$ für ein geeignetes $e \in (0, 1 - \log_4(3))$. Somit gilt der erste Fall des Master-Theorems und es ist $T(n) = \Theta(n^b \log(a)) = \Theta(n^{\log_4(3)})$.

b) Für das Master-Theorem erhalten wir $a = 3$, $b = 2$ und $f(n) = n^2$. Es gilt daher $\log_2(3) \in (1,2)$ und somit $f(n) = n^2 = \Omega(n^{\log_2(3)+e}) = \Omega(n^{\log_b(a)+e})$. für ein geeignetes $e \in (0, 2 - \log_2(3))$. Weiter ist

$$a \cdot f(n/b) = 3 \cdot (n/2)^2 \leq \frac{3}{4} \cdot n^2 \overset{!}{\leq} c \cdot f(n)$$

und somit gilt der dritte Fall des Master-Theorems mit $c = 3/4 < 1$. Damit gilt $T(n) = \Theta(f(n)) = \Theta(n^2)$.

c) Für das Master-Theorem erhalten wir $a = 8$, $b = 2$ und $f(n) = n^3$. Es gilt $\log_2(8) = 3$ und somit $f(n) = n^3 = \Theta(n^{\log_2(8)}) = \Theta(n^{\log_b(a)})$. Also gilt der zweite Fall des Master-Theorems und es ist $T(n) = \Theta(f(n)\log(n)) = \Theta(n^3 \log(n))$.

Vorname: ______________ Name: ______________ Matrikelnummer: ____________

Aufgabe 2 (8 Punkte)

Betrachte die folgende Suchwortmenge $S = \{a, aabcaa, bcac, ca, cac\}$.

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

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

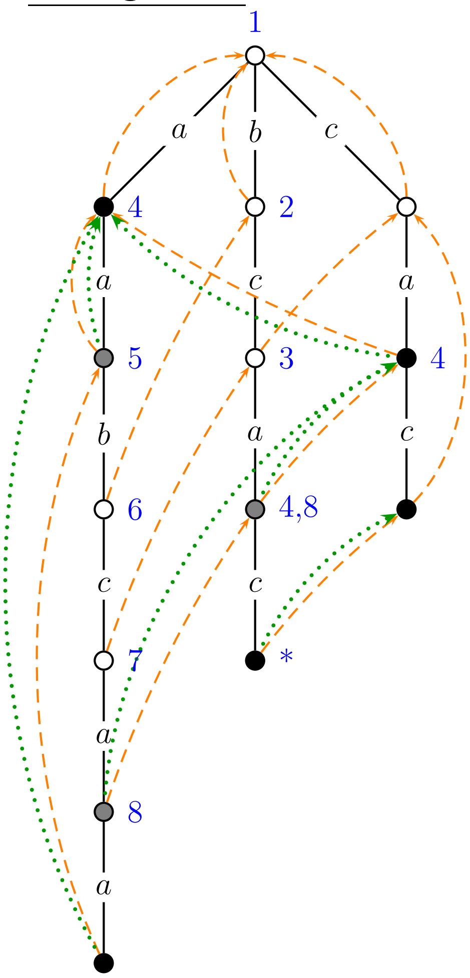
c) Markiere nach dem in der Vorlesung angegebenen Algorithmus von Aho und Corasick alle Knoten darin, die einem Suchwort aus S entsprechen;

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: $s_1 \cdots s_8 = bcaabcac$.

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:

$$\begin{array}{rcl} ca & @ & \overline{bac} \\ a & @ & \overline{bac} \\ a & @ & \overline{aa} \\ ca & @ & \overline{aabaca} \\ a & @ & \overline{aabaca} \\ bcac & @ & \overline{bcac} \\ cac & @ & \overline{bcac} \end{array}$$

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)

Bestimme für das Wort $s = s_0 \cdots s_7 = abbaabbb$ die Shift-Tabelle für den Boyer-Moore-Algorithmus gemäß der Strong-Good-Suffix-Rule aus der Vorlesung.

Gib dabei nicht nur die Tabelle, sondern auch alle Zwischenschritte für die Berechnung an.

Lösungsskizze

In jedem Durchlauf des ersten Teils der Berechnung der Shift-Tabelle beim Boyer-Moore-Algorithmus wird die folgende Aktualisierung vorgenommen (hier mit $m = 8$):

$$S[m-i-1] := \min\{S[m-i-1], j' - i - 1\}.$$

Dies wird im Folgenden kurz durch $S[\cdot] \leftarrow j' - i - 1$ angezeigt. Alle diese Aktualisierungen des ersten Teils sind in der Abbildung auf der folgenden Seite dargestellt. Wenn die Aktualisierung rot dargestellt ist, wird die Shift-Tabelle $S[\cdot]$ dabei nicht verändert.

j'	i	$s_{m-j} \cdots s_{m-1}$	$S[\cdot]$	$border[\cdot]$
		$a\ b\ b\ a\ a\ b\ b\ b$		
0		ε		-1
1		b		0
2	0	$b\ b$		1
3	1	$b\ b\ b$		2
4	2	$a\ b\ b\ b$	$\boxed{S[5] \leftarrow 1}$	
	1	$a\ b\ b\ b$	$\boxed{S[6] \leftarrow 2}$	
	0	$a\ b\ b\ b$	$\boxed{S[7] \leftarrow 3}$	
	-1	$a\ b\ b\ b$		0
5	0	$a\ a\ b\ b\ b$	$S[7] \leftarrow 4$	
	-1	$a\ a\ b\ b\ b$		0
6	0	$b\ a\ a\ b\ b\ b$		1
7	1	$b\ b\ a\ a\ b\ b\ b$		2
8	2	$a\ b\ b\ a\ a\ b\ b\ b$	$S[5] \leftarrow 5$	
	1	$a\ b\ b\ a\ a\ b\ b\ b$	$S[6] \leftarrow 6$	
	0	$a\ b\ b\ a\ a\ b\ b\ b$	$S[7] \leftarrow 7$	
	-1	$a\ b\ b\ a\ a\ b\ b\ b$		0

Da der eigentliche Rand von s gerade ϵ ist, werden alle Werte in der Shift-Tabelle $S[i]$ für $i \in [0 : 10]$ mit 11 aktualisiert, also ändern sich die Werte nicht. Damit bekommen wir letztendlich die folgende Shift-Tabelle:

0	1	2	3	4	5	6	7
8	8	8	8	8	1	2	3

Vorname: ______________ Name: ______________ Matrikelnummer: ____________

Aufgabe 4 (8 Punkte)

Beweise, dass für alle Funktionen $f, g : \mathbb{N}_0 \to \mathbb{R}_+$ gilt:

$$O(f^2) + O(g^2) = O((f+g)^2)$$

Hinweis: Beachte, dass das Gleichheitszeichen hier Mengengleichheit bedeutet.

Es gilt $O(f^2) + O(g^2) = \left\{\hat{f} + \hat{g} \;:\; \hat{f} \in O(f^2) \wedge \hat{g} \in O(g^2)\right\}$.

Lösungsskizze

$\Rightarrow$: Sei $\hat{f} \in O(f^2)$ und $\hat{g} \in O(g^2)$, dann gilt:

$$\exists c_f > 0, n_f \in \mathbb{N} : \; \forall n \geq n_f : \; \hat{f}(n) \leq c_f \cdot (f(n))^2$$

sowie

$$\exists c_g > 0, n_g \in \mathbb{N} : \; \forall n \geq n_g : \; \hat{g}(n) \leq c_g \cdot (g(n))^2.$$

Dann gilt für $n \geq n_0 := \max\{n_f, n_g\}$ und $c := \max\{c_f, c_g\}$:

$$\begin{aligned}
(\hat{f} + \hat{g})(n) &= \hat{f}(n) + \hat{g}(n) \\
&\leq c_f \cdot (f(n))^2 + c_g \cdot (g(n))^2 \\
&\leq c \cdot (f(n))^2 + c \cdot (g(n))^2 \\
& \text{ da } f(n) \geq 0 \text{ und } g(n) \geq 0 \\
&\leq c \cdot (f(n))^2 + 2c \cdot f(n) \cdot g(n) + c \cdot (g(n))^2 \\
&= c \cdot (f(n) + g(n))^2 .
\end{aligned}$$

Also ist $\hat{f} + \hat{g} \in O((f+g)^2)$.

$\Leftarrow$: Aus $\hat{h} \in O((f+g)^2)$ folgt:

$$\exists c > 0, n_0 \in \mathbb{N} : \; \forall n \geq n_0 : \; \hat{h}(n) \leq c \cdot (f(n) + g(n))^2.$$

Für $a, b \in \mathbb{R}$ gilt offensichtlich $(a-b)^2 \geq 0$ und somit $a^2 + b^2 \geq 2ab$. Damit folgt, dass

$$(a+b)^2 = a^2 + 2ab + b^2 \leq a^2 + (a^2 + b^2) + b^2 = 2a^2 + 2b^2.$$

Es gilt daher

$$\exists c > 0, n_0 \in \mathbb{N} : \; \forall n \geq n_0 : \; \hat{h}(n) \leq c \cdot (f(n) + g(n))^2 \leq 2c((f(n))^2 + (g(n))^2).$$

Wir definieren nun

$$\begin{aligned}
\hat{f}(n) &:= 2c(f(n))^2 \\
\hat{g}(n) &:= \hat{h} - \hat{f}(n)
\end{aligned}$$

Dann gilt für $n \geq n_0$:

$$\begin{aligned}
\hat{g}(n) &= \hat{h} - \hat{f}(n) \\
&\leq 2c(f(n))^2 + 2c(g(n))^2 - 2c(f(n))^2 \\
&\leq 2c(g(n))^2
\end{aligned}$$

Somit ist offensichtlich $\hat{f} \in O(f^2)$ und $\hat{g} \in O(g^2)$ und somit ist $\hat{h} \in O(f) + O(g)$.

Aufgabe 5 (8 Punkte)

Für ein Wort $w = w_1 \cdots w_n \in \Sigma^n$ ist seine *Spiegelung* definiert als $w^R := w_n \cdots w_1 \in \Sigma^n$. Ein Wort $w' \in \Sigma^*$ ist eine *partielle Doppelspiegelung* eines Wortes $w \in \Sigma^*$, wenn es drei Wörter $x, y, z \in \Sigma^*$ mit $w = xyz$ gibt, so dass $w' = x^R z^R$ (beachte, dass im Allgemeinen $y \neq \epsilon$ sowie $w' \neq w^R$ gilt).

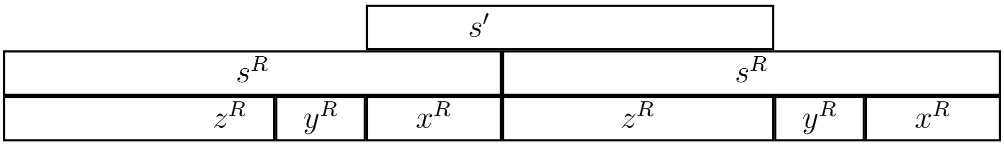
Beispielsweise ist LAGERTOR eine partielle Doppelspiegelung von REGALDUNKELROT.

Entwirf einen Algorithmus, der für $s \in \Sigma^n$ und $t \in \Sigma^m$ in Zeit $O(n + m)$ feststellt, ob t eine partielle Doppelspiegelung von s enthält.

Hinweis: Korrektheitsbeweis und Laufzeitanalyse nicht vergessen.

Lösungsskizze

Wir nehmen an, dass für das Wort $s \in \Sigma^*$ mit $s = xyz$ das Wort $s' = x^R z^R$ eine partielle Doppelspiegelung ist (mit geeigneten $x, y, z \in \Sigma^*$). Dann betrachten wir das Wort $s^R s^R$ und stellen fest, dass dann s' ein Teilwort von $s^R s^R$ sein muss und umgekehrt (siehe auch folgende Abbildung).



Um nun festzustellen, ob eine partielle Doppelspiegelung s' von s ein Teilwort von t ist, müssen also $s^R s^R$ und t ein gemeinsames Teilwort der Länge maximal $|s^R| = |s|$ haben, wobei dieses die Position $|s^R| = |s|$ in $s^R s^R$ enthalten muss (die Indexierung von $s^R s^R$ beginnt hier bei 1).

Daher erstellen wir das Wort $x = x_1 \cdots x_{2n+m+2} = s^R s^R \sharp t \$$ und konstruieren den zugehörigen Suffix-Baum $\hat{T} = \hat{T}(x)$. Wenn es einen Knoten v in $\hat{T}$ gibt, der Blätter als Nachfolger hat, die zu einer Position aus $[1 : |s^R|] = [1 : n]$ sowie zu einer Position aus $[|s^R s^R| + 2 : |s^R s^R \sharp t|] = [2n + 2 : 2n + m + 1]$ gehören (oder auch mehreren Positionen), dann haben wir mit dem zu v korrespondierenden Wort einen Kandidaten für s' gefunden.

Wenn für den größten Blattindex i eines Blattes, das Nachfolger eines solchen Knotens v ist, gilt, dass $i + \ell(v) - 1 \geq m$ ist (wobei $\ell(v)$ die String-Tiefe von v ist), dann überdeckt dieser Kandidat die Position $|s| = m$ in $s^R s^R$ und wir haben eine partielle Doppelspiegelung von s als Teilwort von t gefunden. Wenn die String-Tiefe dieses Knotens v größer als $|s| = m$ ist, dann verwenden wir den Präfix der Länge m und wir haben die partielle Doppelspiegelung gefunden.

Die Konstruktion des Suffix-Baumes $\hat{T}$ ist mit Ukkonens Algorithmus in Zeit $O(n + m)$ möglich. Die inneren Knoten, die jeweils Blätter als Nachfolger sowohl aus dem Intervall $[1 : |s^R|] = [1 : n]$ als auch aus dem Intervall $[|s^R s^R| + 2 : |s^R s^R \sharp t|] = [2n+2 : 2n+m+1]$ besitzen, können mit einer Tiefensuche auch in Zeit $O(|\hat{T}|) = O(n+m)$ gefunden werden. Die String-Tiefe eines Knotens kann ebenfalls in dieser Tiefensuche mitbestimmt werden. Ebenso kann für jeden Knoten, der maximale Index aus $[1 : n]$ eines Blattes, der Nachfolger von v ist, in Zeit $O(|\hat{T}|) = O(n + m)$ mitbestimmt werden (oder 0, wenn es kein Nachfolgerblatt mit Index aus $[1 : n]$ gibt). Somit ist die Gesamtlaufzeit $O(n + m)$.