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

Wintersemester 2017/18
Semestralklausur
15. Februar 2018

Algorithmen auf Sequenzen

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: ____________

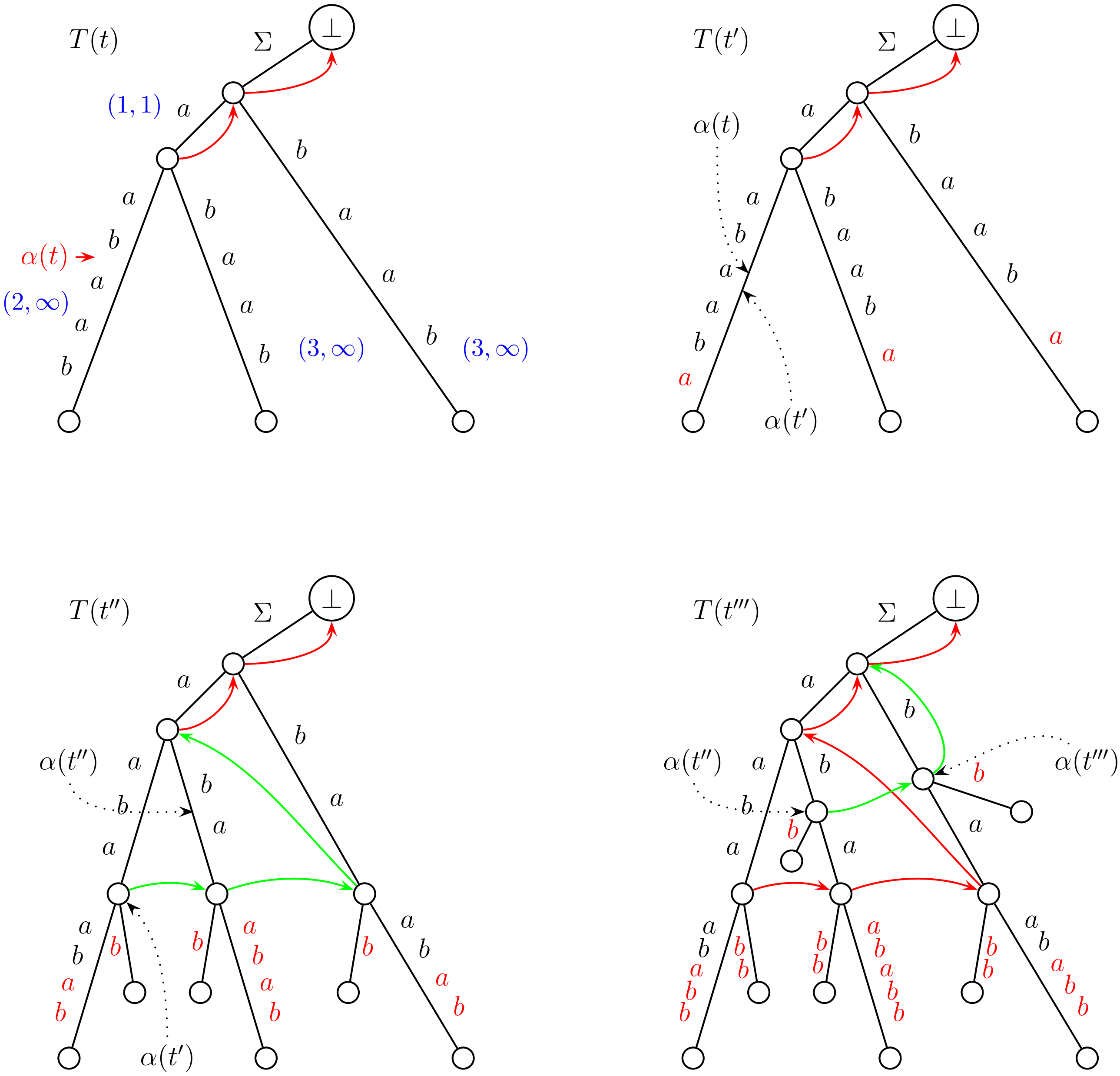
Aufgabe 1 (8 Punkte)

Zeichne im unteren Suffix-Baum für $t = t_1 \cdots t_6 = aabaab$ die Suffix-Links ein und erweitere diesen nach Ukkonens Algorithmus für $t' = aabaab\underline{a}$, $t'' = aabaab\underline{ab}$ und $t''' = aabaab\underline{abb}$. Ergänze hierzu die unten angegebenen Suffix-Bäume.

Es sind auch jeweils die neuen Suffix-Links und die Position des aktiven Suffixes sowohl vor als auch nach Ukkonens Erweiterungsschritt einzuzeichnen.

Gib für den Baum $T(t)$ die Kantenmarkierungen an, die bei einer echten Implementierung hierfür verwendet werden.

Lösungsskizze



Vorname: ______________ Name: ______________ Matrikelnummer: ______________

Aufgabe 2 (8 Punkte)

Betrachte das Wort $t\$ = t_1 \cdots t_9\$ =$ SLEEPLESS\$.

a) Konstruiere die Burrows-Wheeler-Transformierte $\hat{t}$ zu $t\$$.

b) Gib die zugehörige LF-Funktion für $\hat{t}$ an.

c) Bestimme die Werte $C(\cdot)$ und $Occ(\cdot,\cdot)$ für b).

d) Suche nach $s = s_1 \cdots s_3 =$ PLE im FM-Index für t mit Hilfe des in der Vorlesung angegebenen Algorithmus.

Hinweise: Für Teil a) und b) fülle die unten angegebene Tabelle korrekt aus. In dieser Aufgabe gilt: $\$ < E < L < P < S$.

Lösungsskizze

i	$A[i]$	$t^{A[i]}$	$\hat{t}_i$	LF[i]
0	10	\$	S	7
1	3	EEPLESS\$	L	4
2	4	EPLESS\$	E	1
3	7	ESS\$	L	5
4	2	LEEPLESS\$	S	8
5	7	LESS\$	P	6
6	6	PLESS\$	E	2
7	9	S\$	S	9
8	1	SLEEPLESS\$	\$	0
9	8	SS\$	E	3

$Occ(\cdot,\cdot)$	\$	E	L	P	S
0	0	0	0	0	1
1	0	0	1	0	1
2	0	1	1	0	1
3	0	1	2	0	1
4	0	1	2	0	2
5	0	1	2	1	2
6	0	2	2	1	2
7	0	2	2	1	3
8	1	2	2	1	3
9	1	3	2	1	3

$C(\cdot)$	i
\$	0
E	1
L	4
P	6
S	7

$[\ell, r] = [0:9]$, $i = 3$, $s_i =$ E
$\ell' = C(E) + Occ(E, 0-1) = 1 + 0 = 1$
$r' = C(E) + Occ(E, 9) - 1 = 1 + 3 - 1 = 3$

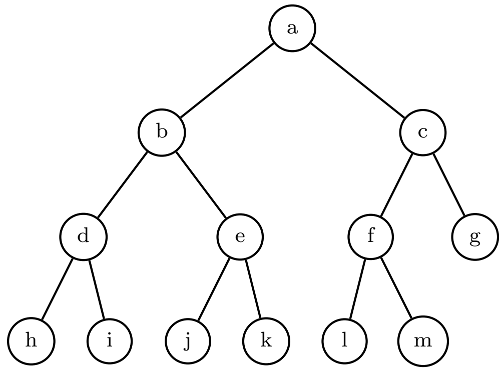
$[\ell, r] = [1, 3]$, $i = 2$, $s_i =$ L
$\ell' = C(L) + Occ(L, 1-1) = 4 + 0 = 4$
$r' = C(L) + Occ(L, 3) - 1 = 4 + 2 - 1 = 5$

$[\ell, r] = [4, 5]$, $i = 1$, $s_i =$ P
$\ell' = C(P) + Occ(P, 4-1) = 6 + 0 = 6$
$r' = C(P) + Occ(P, 5) - 1 = 6 + 1 - 1 = 6$

$[\ell, r] = [6, 6]$.

Vorname: ______________ Name: ______________ Matrikelnummer: ____________

Aufgabe 3 (8 Punkte)



Wende auf den rechten Baum die lineare Vorverarbeitung aus der Vorlesung für $k = 5$ an. Fülle insbesondere auch die unten angegebenen Tabellen aus der Vorverarbeitung für RMQ-Anfragen aus. Beantworte dann die LCA-Anfrage $\text{lca}(h, g)$ gemäß dem Algorithmus aus der Vorlesung. Es müssen nur die Bit-Vektoren $V^{B,j}$ angegeben werden, die für die Abfrage benötigt werden (die explizite Herleitung ist nicht notwendig).

Lösungsskizze

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Euler	a	b	d	h	d	i	d	b	e	j	e	k	e	b	a	c	f	l	f	m	f	c	g	c	a
Tiefe	0	1	2	3	2	3	2	1	2	3	2	3	2	1	0	1	2	3	2	3	2	1	2	1	0

	1	2	3	4	5
F'	0	1	0	1	0
P'	1	3	5	1	5

Q'	1	2	3	4	5
0	1	2	3	4	5
1	1	3	3	5	–
2	1	3	–	–	–

Q' beantwortet dabei $\text{RMQ}(i, i + 2^j - 1)$ im Feld F'.

$\text{lca}(h, g)$ entspricht $\text{RMQ}_F(4, 23)$. Dazu benötigen wir die Ergebnisse von

- $\text{RMQ}_F(4, 5)$: Dazu benötigen wir den Bitvektor $V^{1,5} = 1 \cdot 1 \cdot 1 \cdot 0 \cdot 1$. Damit ist die linkeste 1 von $V^{1,5} \wedge 0 \cdot 0 \cdot 0 \cdot 1 \cdot 1 = 0 \cdot 0 \cdot 0 \cdot 0 \cdot 1$ an Position 5 in der Euler-Tour mit Wert 2.
- $\text{RMQ}_F(6, 20)$: Zuerst beantworten wir dazu $\text{RMQ}_{F'}(2, 3)$ und $\text{RMQ}_{F'}(3, 4)$ in F' mittels $Q'(2, 1) = 3$ und $Q'(3, 1) = 3$. Damit ist das Minimum gerade $F'[3] = 0$, also an Position 15 in der Euler-Tour mit Wert 0.
- $\text{RMQ}_F(21, 23)$:Dazu benötigen wir den Bitvektor $V^{5,3} = 0 \cdot 1 \cdot 1 \cdot 0 \cdot 0$. Damit ist die linkeste 1 von $V^{5,3} \wedge 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 0 \cdot 1 \cdot 1 \cdot 0 \cdot 0$ an Position 2 im 5. Block, als an Position 22 in der Euler-Tour mit Wert 1.

Insgesamt ist das Minimum an Position 15 mit Wert 0. Dies entspricht dem Knoten a.

Aufgabe 4 (8 Punkte)

Beweise formal, dass sich zwei LCP-Intervalle nicht überlappen können (d.h. dass diese entweder disjunkt sind oder eines im anderen enthalten sein muss).

Lösungsskizze

Wiederholen wir zuerst die Definition von LCP-Intervallen:

Sei $t = t_1 \cdots t_n \in \Sigma^*$ und sei A bzw. L das zugehörige Suffix-Array bzw. die zugehörige LCP-Tabelle. Ein Intervall $[i : j]$ mit $i < j \in [0 : n]$ heißt ein LCP-Intervall vom Typ ℓ oder kurz ein ℓ-Intervall, wenn

1. $L[i] < \ell$,
2. $L[k] \geq \ell$ *für alle* $k \in [i + 1 : j]$,
3. $L[k] = \ell$, *für mindestens ein* $k \in [i + 1 : j]$ *und*
4. $L[j + 1] < \ell$.

Hierbei gelte $L[0] = L[n + 1] = -1$.

Für einen Widerspruchsbeweis sei $[i : j]$ ein ℓ-Intervall und $[i' : j']$ ein ℓ'-Intervall mit $i < i' \leq j < j'$.

Fall 1; Sei zuerst $\ell' \leq \ell$.

Nach Definition von ℓ-$[i : j]$ gilt $L[i'] \geq \ell$, da $i' \in [i + 1 : j]$. Dies ist nach Definition von ℓ'-$[i' : j']$ aber ein Widerspruch zu $L[i'] < \ell' \leq \ell$.

Fall 2; Sei nun $\ell' > \ell$.

Nach Definition von ℓ'-$[i' : j']$ gilt $L[j + 1] \geq \ell'$, da $j + 1 \in [i' + 1 : j']$. Dies ist nach Definition von ℓ-$[i : j]$ aber ein Widerspruch zu $L[j + 1] < \ell < \ell'$.

Vorname: ______________ Name: ______________ Matrikelnummer: ____________

Aufgabe 5 (8 Punkte)

Sei Σ ein Alphabet und zwei Wörter $t \in \Sigma^n$ und $t' \in \Sigma^m$. Konstruiere einen Algorithmus mit Laufzeit $O(n+m+\ell)$, der alle Wörter $w \in \Sigma^*$ findet, die in t genau einmal und in t' mindestens zweimal auftreten; dabei ist ℓ die Anzahl dieser Wörter.

Zeige die Korrektheit des Algorithmus und analysiere dessen Laufzeit.

Lösungsskizze

Konstruiere einen Suffix-Baum T für $t\sharp t'\$$. Im Folgenden betrachten wir diesen Suffix-Baum T', in dem alle Kanten nach $\sharp$ abgeschnitten wurden. Dies ist mit einer Tiefensuche über den Suffix-Baum T in linearer Zeit in der Größe des Suffix-Baumes T möglich, also in Zeit $O(n+m)$.

Mit Hilfe einer Tiefensuche in T' zähle getrennt für jeden inneren Knoten die Anzahl nachfolgenden Blätter mit einem Suffix in t und in t'. Blätter gehören zu t bzw. t', wenn Kanten zum Blatt mit \$ endet bzw. in der Kante ein $\sharp$ vorkommt. Die zu inneren Knoten korrespondierenden Wörter, deren Zähler für t genau 1 und für t' mindestens 2 ist, kommen eben in t genau einmal und in t' mindestens zweimal vor. Dies gilt auch für alle Wörtern, die auf der eingehenden Kante zu diesem Knoten enden.

Die Laufzeit für Ukkonens Algorithmus ist linear in der Länge von $t\sharp t'\$$, also $O(n+m)$. Auch eine Tiefensuche in T bzw. T' kann in Zeit $O(n+m)$ ausgeführt werden.

Die Ausgabe sind nun alle Wörter ausgibt, die zu Kanten gehören, deren Endknoten beide Zähler wie oben beschreiben gesetzt sind. Die Ausgabe selbst sind allerdings nur die Start- und Endposition im String t. Für die Start-Positionen sind wie in der Vorlesung die Blattlisten an den inneren Knoten von t' mitzukonstruieren, dies ist Zeit linear in T' möglich. Somit ist die Laufzeit für die Ausgabe selbst durch die Ausgabegröße bestimmt. Sei also ℓ die Anzahl aller gesuchten Wörter, dann ist die Laufzeit $O(|t\sharp t'\$| + \ell) = O(n+m+\ell)$, was optimal ist.