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

Wintersemester 2018/19
Semestralklausur
13. Februar 2019

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)

Konstruiere für $t\$ = t_1 \cdots t_{10}\$ = abbababbab\$$ das zugehörige Suffix-Array mit dem Algorithmus von Manber und Myers. Hierbei gilt: $\$ < a < b$.

Lösungsskizze

Zuerst sortieren der Suffixe mit Bucketsort:

$$(11 \mid 1, 4, 6, 9 \mid 2, 3, 5, 7, 8, 10).$$

i	1	2	3	4	5	6	7	8	9	10	11
A	11	1	4	6	9	2	3	5	7	8	10
11						10					
1											
4						10	3				
6						10	3	5			
9						10	3	5	8		
2		1									
3						10	3	5	8	2	
5		1	4								
7		1	4	6							
8						10	3	5	8	2	7
10		1	4	6	9						
	11	1	4	6	9	10	3	5	8	2	7

i	1	2	3	4	5	6	7	8	9	10	11
A	11	1	4	6	9	10	3	5	8	2	7
11		9									
1											
4										2	
6		9	4								
9										2	7
10							8				
3		9	4	1							
5							8	3			
8		9	4	1	6						
2											
7							8	3	5		
	11	9	4	1	6	10	8	3	5	2	7

i	1	2	3	4	5	6	7	8	9	10	11
A	11	9	4	1	6	10	8	3	5	2	7
11										7	
9									5		
4											
1											
6										7	2
10				6							
8			4								
3											
5				6	1						
2											
7								3			
	11	9	4	6	1	10	8	3	5	7	2

i	1	2	3	4	5	6	7	8	9	10	11
A											

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 2 (8 Punkte)

Betrachte das Wort $t\$ = t_1 \cdots t_{10}\$ = \text{ANNABANANA\$}$.

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 $Occ(\cdot, \cdot)$.

d) Suche nach $s = s_1 \cdots s_3 = \text{BAN}$ 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: $\$ < A < B < N$.

Lösungsskizze

i	$A[i]$	$t^{A[i]}$	$\hat{t}_i$	LF[i]
0	11	\$	A	1
1	10	A\$	N	7
2	4	ABANANA\$	N	8
3	8	ANA\$	N	9
4	6	ANANA\$	B	6
5	1	ANNABANANA\$	\$	0
6	5	BANANA\$	A	2
7	9	NA\$	A	3
8	3	NABANANA\$	N	10
9	7	NANA\$	A	4
10	2	NNABANANA\$	A	5

$Occ(\cdot, \cdot)$	\$	A	B	N
0	0	1	0	0
1	0	1	0	1
2	0	1	0	2
3	0	1	0	3
4	0	1	1	3
5	1	1	1	3
6	1	2	1	3
7	1	3	1	3
8	1	3	1	4
9	1	4	1	4
10	1	5	1	4

$C(\cdot)$	i
\$	0
A	1
B	6
N	7

$[\ell, r] = [0 : 10]$, $i = 3$, $s_i = \text{N}$
$\ell' = C(N) + Occ(N, 0 - 1) = 7 + 0 = 7$
$r' = C(N) + Occ(N, 10) - 1 = 7 + 4 - 1 = 10$

$[\ell, r] = [7, 10]$, $i = 2$, $s_i = \text{A}$
$\ell' = C(A) + Occ(A, 7 - 1) = 1 + 2 = 3$
$r' = C(A) + Occ(A, 10) - 1 = 1 + 5 - 1 = 5$

$[\ell, r] = [3, 5]$, $i = 1$, $s_i = \text{B}$
$\ell' = C(B) + Occ(B, 3 - 1) = 6 + 0 = 6$
$r' = C(B) + Occ(B, 5) - 1 = 6 + 1 - 1 = 6$

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

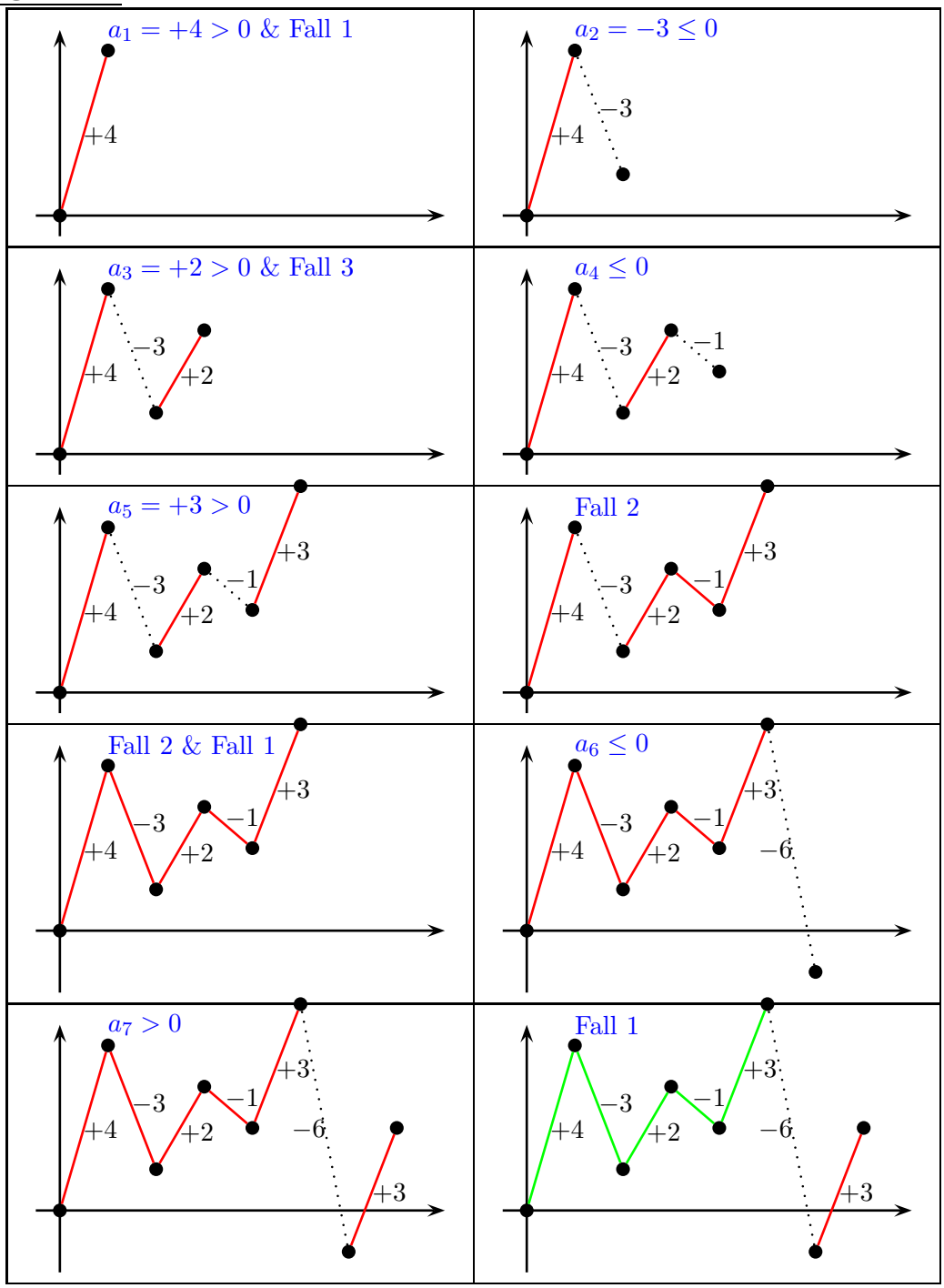
Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 3 (8 Punkte)

Ermittle mit dem in der Vorlesung angegebenen Linearzeit-Algorithmus für AMSS alle maximal bewerteten Teilfolgen von a und gib dabei alle Zwischenschritte sowie die jeweils angewendeten Fälle an. Gib auch an, wann welche maximal bewertete Teilfolge vom Algorithmus ausgegeben wird.

$$a = (+4, -3, +2, -1, +3, -6, +3)$$

Lösungsskizze



Nachdem im letzten Schritt bereits $(1, 5)$ ausgegeben wurde, wird am Ende noch $(7, 7)$ ausgegeben.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 4 (8 Punkte)

Entwirf für das LCE-Problem einen alternativen Algorithmus basierend auf Extended Suffix-Arrays (ohne Verwendung von Bäumen), so dass Anfragen in konstanter Zeit nach einer linearen Vorverarbeitung beantwortet werden können. Versuche dabei, möglichst wenig Bestandteile des Extended Suffix-Arrays zu verwenden.

Hinweis: Sei $t \in \Sigma^n$ und sei $t' = t\$$. Für $i < j \in [1 : n]$ ist die longest common (forward) extension *von i und j in t definiert als die Zahl $\ell \in \mathbb{N}_0$ mit $t'_i \cdots t'_{i+\ell-1} = t'_j \cdots t'_{j+\ell-1}$ und $t'_{i+\ell} \neq t'_{j+\ell}$ (in Zeichen $lce(i, j) = \ell$).*

Lösungsskizze

Wir konstruieren zuerst das Suffix-Array A für t und die LCP-Tabelle L in Zeit $O(|t|)$. Da alle Suffixe lexikographisch sortiert sind, gilt:

$$\mathrm{lce}(i, j) = \mathrm{lcp}(R[i], R[jj).$$

Dabei ist R das zu A inverse Suffix-Array, damit wir wissen an welchen Stellen t^i bzw. t^j im Suffix-Array stehen.

Nach der Vorlesung gilt das folgende Lemma.

Lemma *Sei $t = t_1 \cdots t_n \in \Sigma^*$ und sei A das zugehörige Suffix-Array. Dann gilt für $i < j \in [0 : n]$, dass*

$$lcp(i, j) = \min \{ lcp(k - 1, k) \ : \ k \in [i + 1 : j] \}.$$

Wir bereiten also die LCP-Tabelle L für RMQ-Anfragen in Zeit $O(|t|)$ vor. Somit ist dann

$$\mathrm{lce}(i, j) = \mathrm{lcp}(R[i], R[j]) = \mathrm{RMQ}_L(R[i] + 1, R[j]).$$

Somit haben wir eine lineare Vorverarbeitung mit Anfragen in konstanter Zeit.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 5 (8 Punkte)

Sei Σ ein Alphabet und $t \in \Sigma^n$. Konstruiere einen Algorithmus mit Laufzeit $O(n)$, der alle Wörter $w \in \Sigma^*$ findet, die in t mindestens zweimal und höchstens viermal auftreten, wobei kein echtes Präfix von w auch maximal viermal in t auftritt.

Zeige die Korrektheit des Algorithmus und analysiere dessen Laufzeit.

Hinweis: Für $w \in \Sigma^*$ ist $u \in \Sigma^*$ ein *echtes Präfix*, wenn $w = uv$ mit $v \in \Sigma^+$ ist.

Lösungsskizze

Konstruiere einen Suffix-Baum T für $t\$$. Mit Hilfe einer Tiefensuche in T zähle für jeden inneren Knoten die Anzahl nachfolgenden Blätter. Die zu inneren Knoten korrespondierenden Wörter, deren Zähler für t im Intervall von $[2:4]$ liegen sind für uns von Interesse, da alle Wörter, die auf der Kante zu diesem Knoten enden, mindestens zweimal und maximal viermal in t vorkommen. Wir markieren daher während der Tiefensuche alle inneren Knoten mit dieser Eigenschaft. Damit die zugehörigen echten Präfixe mindestens fünfmal vorkommen, darf der Elter-Konten nicht markiert sein. Betrachten wir also die Wörter, die auf einer Kante von einem nichtmarkierten Knoten $\overline{w}$ zu einem markierten Knoten $\overline{wx}$ (mit $w, x \in \Sigma^+$) liegt. Das Wort wx_1 (mit $x = x_1 \cdots x_{|x|}$) ist dann ein Wort mit der gesuchten Eigenschaften, da w mindestens fünfmal vorkommt und jedes andere Präfix von wx einen echten Präfix besitzt, das ebenfalls maximal viermal vorkommt.

Die Laufzeit für Ukkonens Algorithmus ist linear in der Länge von $t\$$, also $O(n)$. Auch die benötigte Tiefensuche in T kann in Zeit $O(n)$ ausgeführt werden.

Die Ausgabe sind nun alle oben genannten Wörter, die zu Kanten gehören, deren Elterkonten unmarkiert und deren Kindknoten markiert ist. Die Ausgabe selbst sind allerdings nur die Start- und Endposition im String t für das entsprechende Wort w.

Da für jede Kante maximal eine Referenz für das zugehörige Wort ausgegeben wird, ist die Ausgabegröße $O(|E(T)|)$. Da in einem Baum die Anzahl der Kanten durch Anzahl der Knoten des Baumes und in einem Suffix-Baum die Anzahl innereren Knoten durch die Anzahl Blätter beschränkt ist (die $|t| + 1 = n + 1$ ist), gibt es maximal $O(n)$ Kanten. Die Ausgabe ist also linear in der Eingabe.