Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 1

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

Lösungsskizze

Erstes sortieren mit Bucketsort:

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

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

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

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

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 2

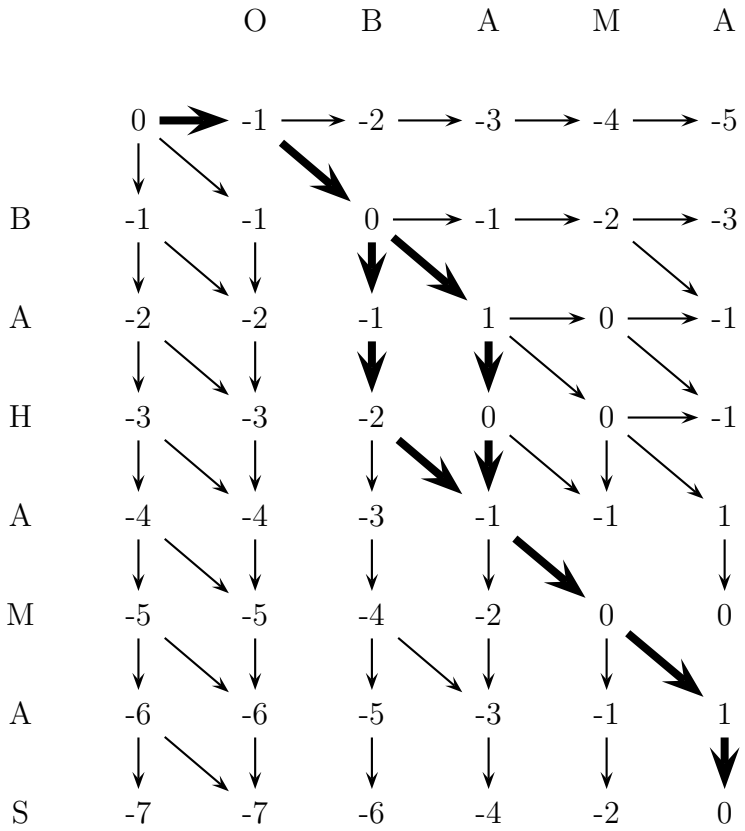
Betrachte die Wörter s = BAHAMAS und t = OBAMA. Konstruiere mit Hilfe des Algorithmus von Needleman und Wunsch ein globales optimales Alignment.

Gib dazu die Tabelle zur Ermittlung der Teil-Lösungen an und zeichne in diesen Tabellen auch alle Backtracking-Pfeile ein. Kennzeichne alle optimalen Backtracking-Pfade und gib die optimalen Alignments an!

Die Scoring-Funktion sei gegeben mit:

$$\sigma(a,b) = \begin{cases} 1 & \text{für } a = b \\ -1 & \text{für } a \neq b \end{cases}$$

Lösungsskizze



Die beiden optimalen Alignments lauten:

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

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 3

Betrachte das Wort t = ANANAS.

1. Konstruiere die Burrows-Wheeler-Transformierte $\hat{t}$ zu $t\$$ (Das Suffix-Array muss dabei nicht konstuiert werden, sondern kann von Hand bestimmt werden).
2. Bestimme die Tabellen C und *occ*.
3. Suche nach s = ANA mit Hilfe des in der Vorlesung angegebenen Algorithmus (backward-search).

Lösungsskizze

Das Suffix-Array und die Burrows-Wheeler-Transformierte ist:

i	$A[i]$	$\tilde{t}$	$t_{A[i]}..t_n$
0	6	S	\$
1	0	\$	ANANAS\$
2	2	N	ANAS\$
3	4	N	AS\$
4	1	A	NANAS\$
5	3	A	NAS\$
6	5	A	S\$

Die Tabellen C und *occ* sind:

$occ(.,.)$	\$	A	N	S
0	0	0	0	1
1	1	0	0	1
2	1	0	1	1
3	1	0	2	1
4	1	1	2	1
5	1	2	2	1
6	1	3	2	1

	\$	A	N	S
$C(.)$	0	1	4	6

Schritt 1:

$$\begin{aligned} [l,r] &= [0:6], i=3, s_i = A \\ l' &= C(A) + occ(A,-1) = 1+0 = 1 \\ r' &= C(A) + occ(A,6) - 1 = 1+3-1 = 3 \end{aligned}$$

Schritt 2:

$$\begin{aligned} [l,r] &= [1:3], i=2, s_i = N \\ l' &= C(N) + occ(N,0) = 4+0 = 4 \\ r' &= C(N) + occ(N,3) - 1 = 4+2-1 = 5 \end{aligned}$$

Schritt 3:

$$\begin{aligned} [l, r] &= [4:5], i = 1, s_i = A \\ l' &= C(A) + occ(A, 3) = 1 + 0 = 1 \\ r' &= C(A) + occ(A, 5) - 1 = 1 + 2 - 1 = 2 \end{aligned}$$

Das Ergebnis ist also der Bereich $[1, 2]$ im Suffix-Array, also die Suffixe 0 und 2.

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 4

Betrachte den Bit-Vektor $t = (1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1)$. Konstruiere die succincte Datenstruktur zum Beantworten von *rank* queries. Benutze eine Block-Größe von $b = 3$ und eine Superblock-Größe von $s = 6$!

Beantworte damit die queries $rank_1(9)$, $rank_1(16)$ und $rank_0(14)$ (Die Indizes sind 0-basiert zu verstehen)!

Lösungsskizze

Die letzten Einträge der Superblöcke und Blöcke sind:

```
11001010110011001
AAAAAABBBBBB        (Superblocks)
AAABBBCCCDDDEEE     (Blocks)
i=2  5  8 11 14     (Positionen des letzten Eintrags)
```

Der Superblock-Index ist hiermit:

i	5	11
$S(i)$	3	6

Und der Block-Index:

i	2	5	8	11	14
$B(i)$	2	0	2	0	2

Blocklänge ist $b = 3$, der Vier-Russen-Index F ist also:

Vektor	$p = 0$	$p = 1$	$p = 2$
000	0	0	0
001	0	0	1
010	0	1	1
011	0	1	2
100	1	1	1
101	1	1	2
110	1	2	2
111	1	2	3

$$
\begin{aligned}
rank_1(9) &= S(5) + B(8) + F(100, 0) = 3 + 2 + 1 = 6 \\
rank_1(16) &= S(11) + B(14) + F(01., 1) = 6 + 2 + 1 = 9 \\
rank_0(14) &= 14 + 1 - rank_1(14) = 14 + 1 - (S(11) + B(15)) = 14 + 1 - (6 + 2) = 7
\end{aligned}
$$

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 5

Füge sukzessive folgende Werte in einen anfangs leeren Rot-Schwarz-Baum ein und gib alle Zwischenschritte an:

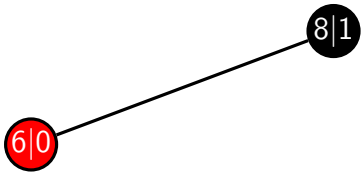
8,6,5,3,4,1

Lösungsskizze

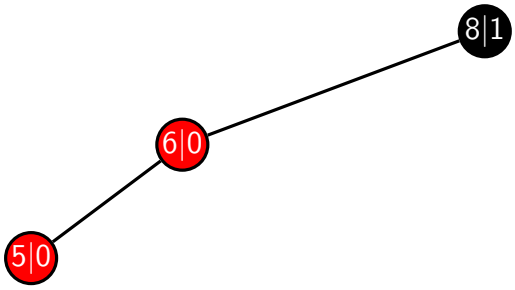
Die entsprechenden Schwarztiefen sind neben den Werten angegeben!

Insert(8):

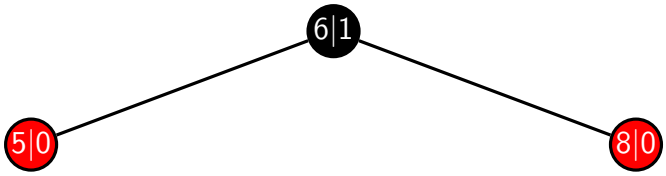
Insert(6):



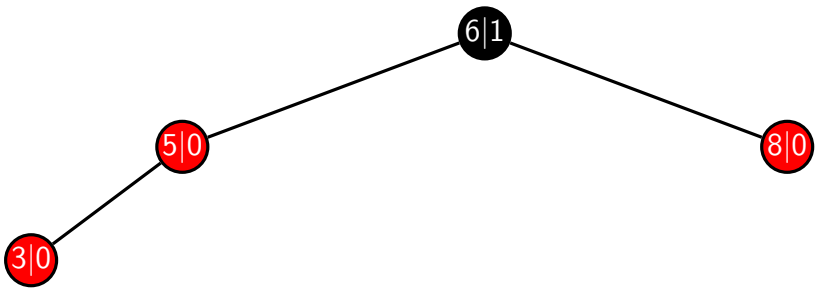
Insert(5):



Case 2, right-rotate and recolor:

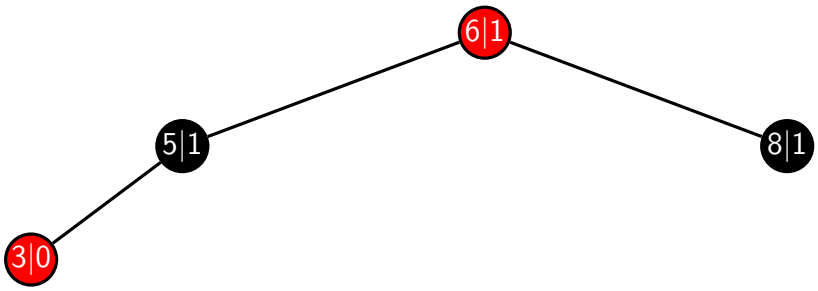


Insert(3):

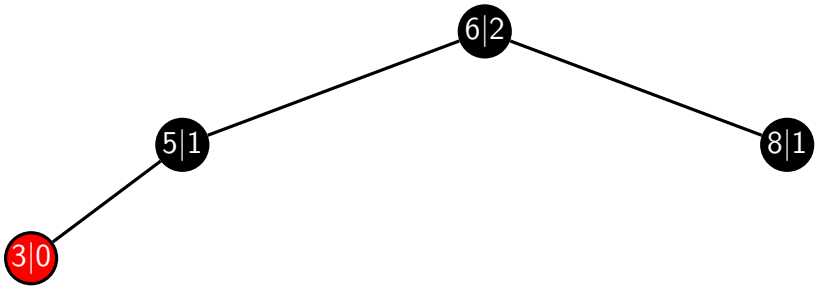


Vorname: _____ Name: _____ Matrikelnummer: _____

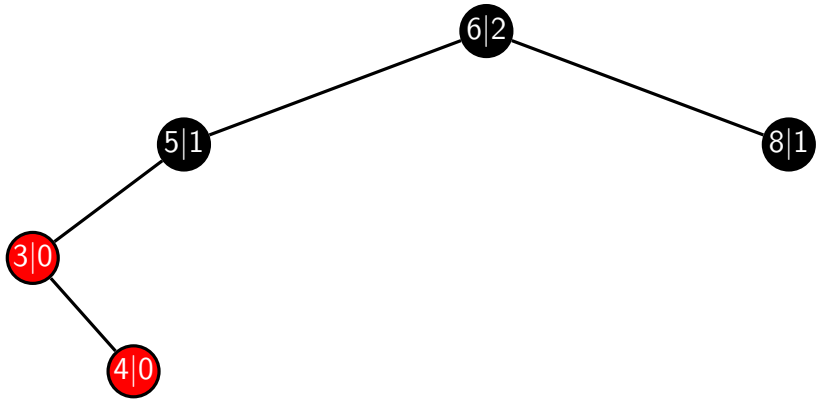
Case 1, uncle is red



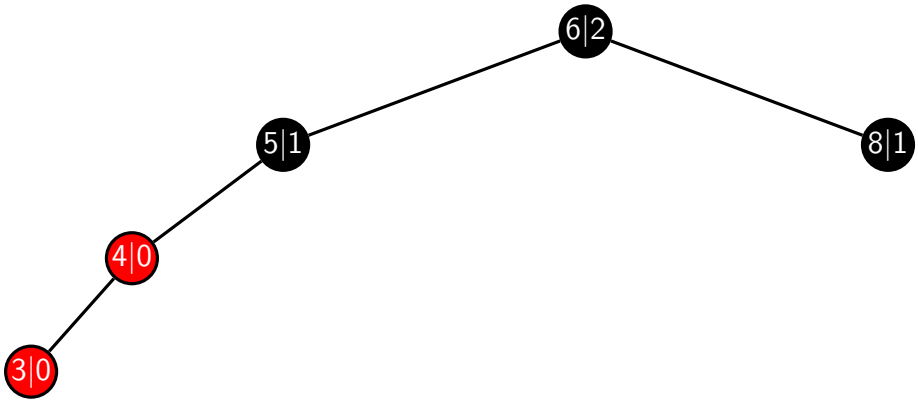
correct root



Insert(4):

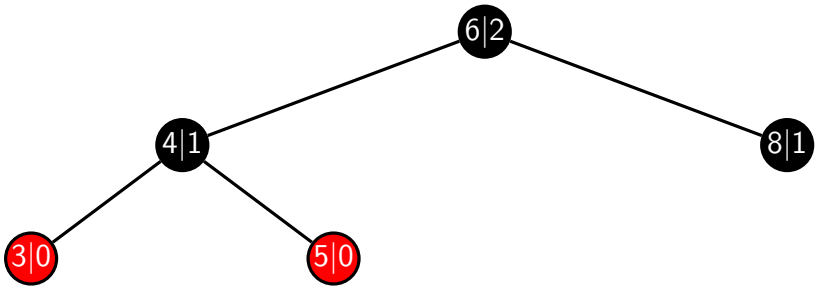


Case 2, left-rotate:

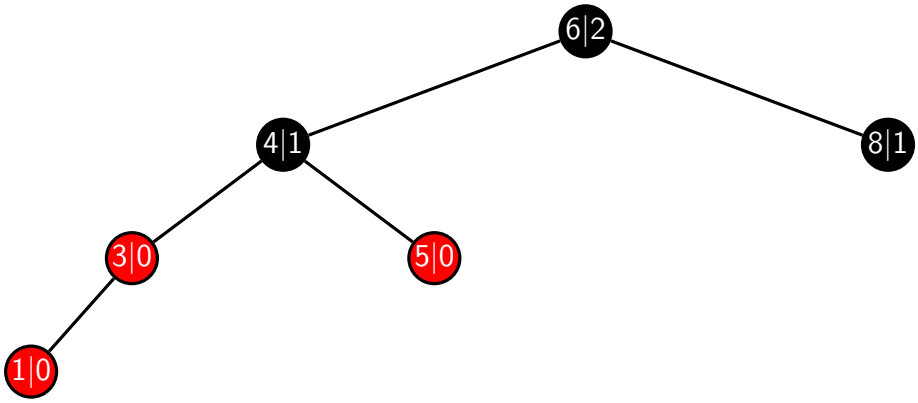


right-rotate, recolor:

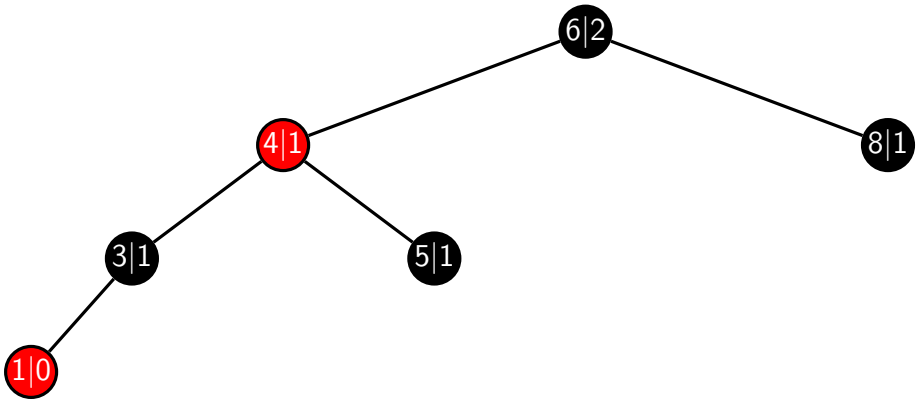
Vorname: _____ Name: _____ Matrikelnummer: _____



Insert(1):



Case 1, recolor:

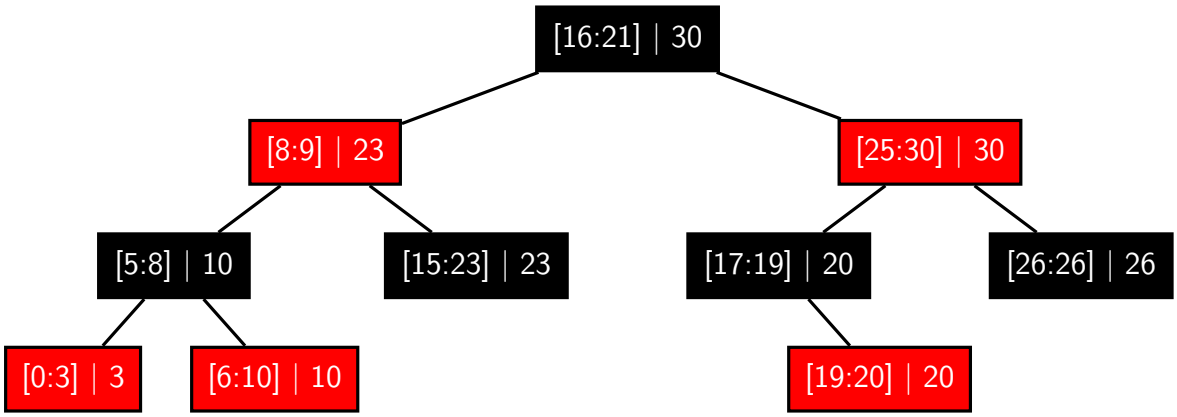


Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 6

Suche folgende Intervalle im gegebenen Intervalbaum und gib alle Zwischenschritte an:

$A = [22 : 24]$
$B = [11 : 13]$



Lösungsskizze

Suche nach $[22 : 24]$:

1. Starte mit $[16 : 21]$, teste $22 < 23 \rightarrow$ links
2. Bei $[8 : 9]$, teste $22 \geq 10 \rightarrow$ rechts
3. Bei $[15 : 23]$, überlappt, gib aus!

Suche nach $[11 : 13]$:

1. Starte mit $[16 : 21]$, teste $11 < 23 \rightarrow$ links
2. Bei $[8 : 9]$, teste $11 \geq 10 \rightarrow$ rechts
3. Bei $[15 : 23]$, überlappt nicht, kein Kind, gib *null* aus!

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 7

Es seien $Y_1, ..., Y_m$ unabhängige und identisch verteilte Zufallsvariablen aus einer Binomialverteilung mit Wahrscheinlichkeitsfunktion:

$$P_Y(y) = \binom{n}{y} \cdot p^y \cdot (1-p)^{n-y}$$

Bestimme den Maximum-Likelihood-Schätzer für p, wenn $y_1, ..., y_m$ beobachtet wurden.

Lösungsskizze

Die Likelihood und (abgeleitete) Log Likelihood Funktionen sind:

$$\begin{aligned}
L(p; y_1, ...y_m, n) &= \prod_{i=1}^{m} \binom{n}{y_1} \cdot p^{y_i} \cdot (1-p)^{n-y_i} \\
l(p; y_1, ...y_m, n) &= \sum_{i=1}^{m} log\binom{n}{y_1} + y_i log(p) + (n-y_i) log(1-p) \\
l'(p; y_1, ...y_m, n) &= \sum_{i=1}^{m} \frac{y_i}{p} - \frac{n-y_i}{1-p}
\end{aligned}$$

Ableitung gleich Null setzen:

$$\begin{aligned}
\sum_{i=1}^{m} \frac{y_i}{\hat{p}} - \frac{n-y_i}{1-\hat{p}} &= 0 \\
\Leftrightarrow \sum_{i=1}^{m} (1-\hat{p}) y_i - \hat{p}(n-y_i) &= 0 \\
\Leftrightarrow \sum_{i=1}^{m} y_1 - \hat{p} y_i - n\hat{p} + y_i \hat{p} &= 0 \\
\Leftrightarrow \left(\sum_{i=1}^{m} y_i \right) - mn\hat{p} &= 0 \\
\Leftrightarrow mn\hat{p} &= \sum_{i=1}^{m} y_1 \\
\Leftrightarrow \hat{p} &= \frac{1}{mn} \sum_{i=1}^{m} y_1
\end{aligned}$$

Vorname: _____ Name: _____ Matrikelnummer: _____

Aufgabe 8 (8 Punkte)

Gegeben sei eine Zeichenreihe t. Ein Wort w heißt minimal rechts bzw. links-eindeutiges Teilwort von t, wenn w genau einmal in t auftritt und wenn jedes echte (d.h. nicht w selbst) Präfix bzw. Suffix von w mindestens zweimal in t auftritt.

1. Entwirf einen effizienten Algorithmus zum Auffinden aller minimal rechts-eindeutigen Teilwörter der Länge mindestens l, beweise seine Korrektheit und analysiere seine Laufzeit.

2. Entwirf einen effizienten Algorithmus zum Auffinden aller minimal links-eindeutigen Teilwörter der Länge mindestens l, beweise seine Korrektheit und analysiere seine Laufzeit.

Lösungsskizze

1. Zunächst konstruieren wir das Suffix-Array A und LCP array L für t. Jedes Teilwort w von t ist im Suffix-Array als Präfix eines der sortierten Suffixe zu finden. Wir testen also für jedes Suffix $A[i]$ zwei Eigenschaften: (A) w tritt genau einmal in t auf und (B) jedes Präfix von w tritt mehr als einmal auf. Damit (A) gilt, muss die Teilwort-Länge $|w| > L(i)$ und $|w| > L(i+1)$ sein. Für (B) muss $|w| - 1 \leq L(i)$ oder $|w| - 1 \leq L(i+1)$. D.h. es muss entweder folgendes gelten:

$$\begin{aligned} |w| > L(i) \wedge |w| - 1 \leq L(i) \quad &\wedge \quad |w| > L(i+1) \\ \Leftrightarrow |w| = L(i) + 1 \quad &\wedge \quad |w| > L(i+1) \end{aligned}$$

oder folgendes gelten

$$\begin{aligned} |w| > L(i+1) \wedge |w| - 1 \leq L(i+1) \quad &\wedge \quad |w| > L(i) \\ \Leftrightarrow |w| = L(i+1) + 1 \quad &\wedge \quad |w| > L(i) \end{aligned}$$

Damit ist die Korrektheit gezeigt. Für die lineare Laufzeit bleibt festzuhalten, dass A und L in $O(n)$ konstruiert werden können, und der Test eines jeden Suffixes in $O(1)$ ist.

2. Für links-eindeutige Teilwörter müssen nur rechts-eindeutige Teilwörter der invertierten Zeichenreihe $t_n ... t_1$ gefunden werden.