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

Wintersemester 2020/21
Semestralklausur
25. Februar 2021

# Algorithmen auf Sequenzen

| 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 | | | | | | | |

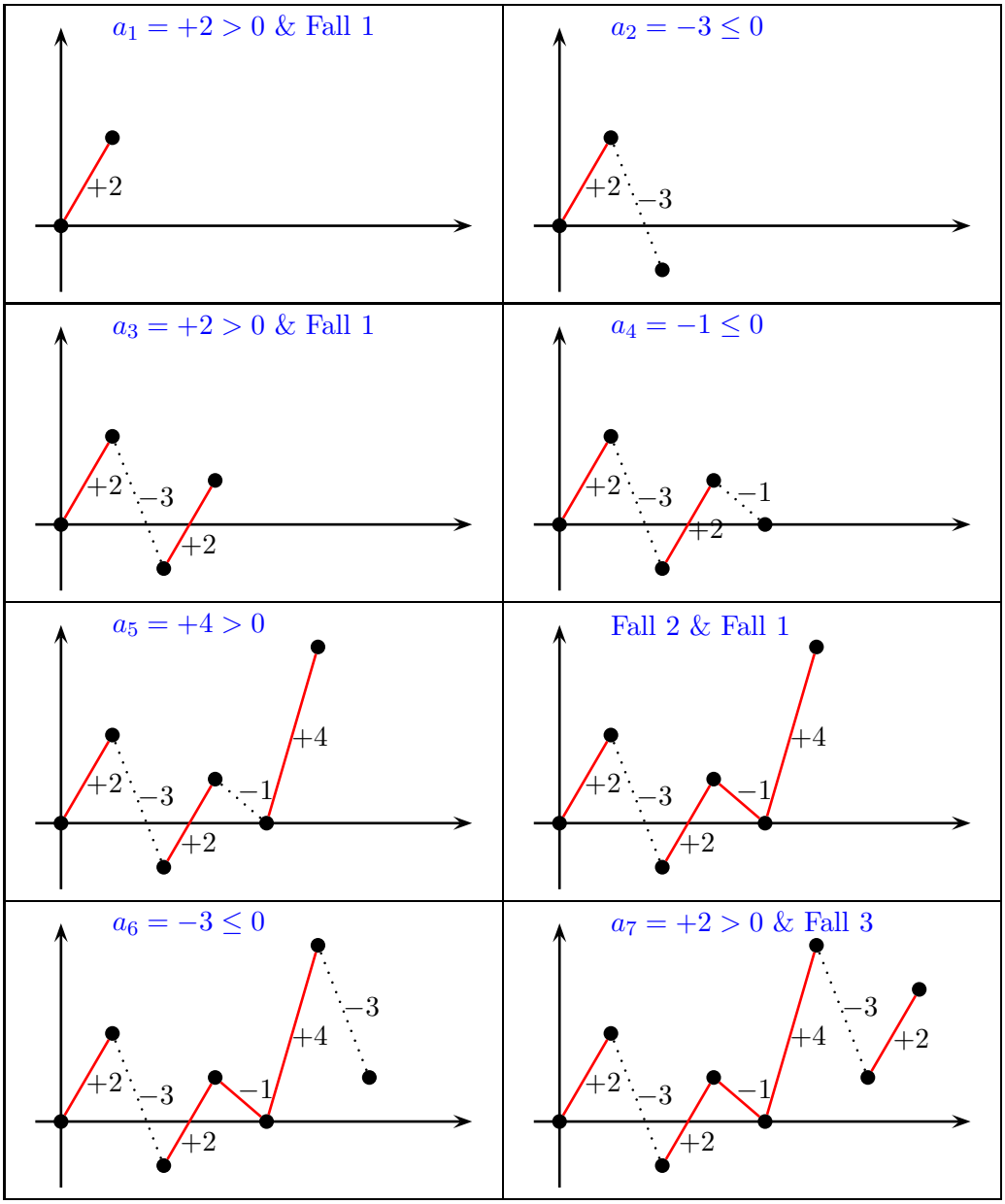
Vorname: _____ Name: _____ Matrikelnummer: _____

## Aufgabe 1 (8 Punkte)

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, welche maximal bewerteten Teilfolgen vom Algorithmus ausgegeben werden.

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

**Lösungsskizze** (nicht ausreichend für die volle Punktzahl)




Ausgegeben wird also $(1, 1)$, $(3, 5)$ und $(7, 7)$.

Vorname: _____ Name: _____ Matrikelnummer: _____

## Aufgabe 2 (8 Punkte)

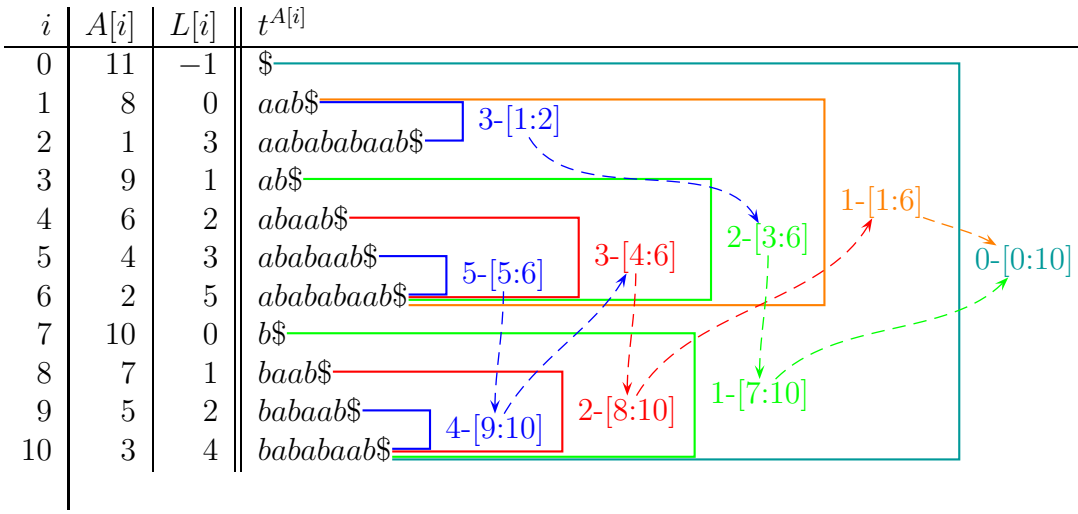
Gib für $t\$ = t_1 \cdots t_{10}\$ = aababababaab\$$ das zugehörige Suffix-Array $A$ inklusive der zugehörigen LCP-Tabelle $L$ an.

Zeichne dort die LCP-Intervalle mit den Suffix-Links (ohne die Suffix-Links von einelementigen LCP-Intervallen) ein.

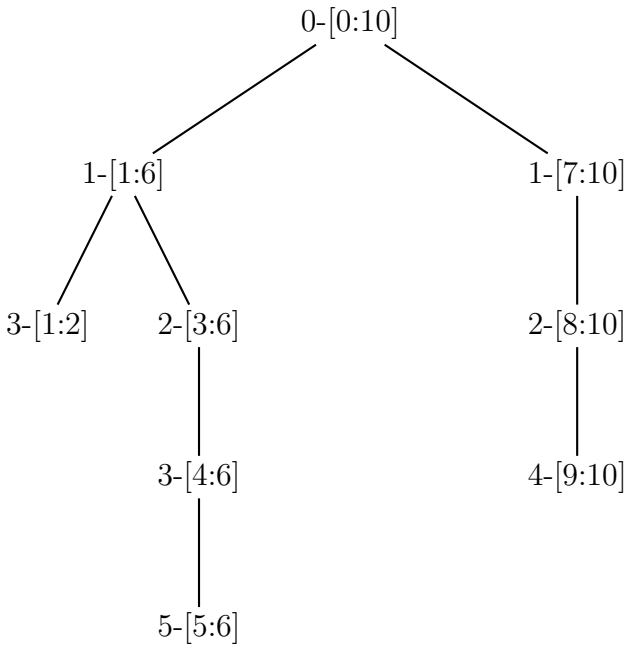
Zeichne weiterhin den zugehörigen LCP-Intervall-Baum.

*Bemerkung:* Die Ordnung auf dem Alphabet $\Sigma$ sei $\$ < a < b$.

**Lösungsskizze** (nicht ausreichend für die volle Punktzahl)




Im Folgenden ist der LCP-Intervall-Baum (ohne Blätter) dargestellt.



Vorname: _____ Name: _____ Matrikelnummer: _____

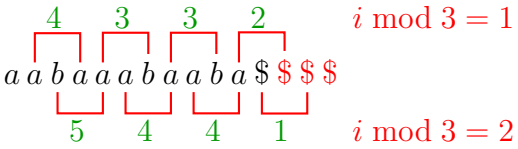
## Aufgabe 3 (8 Punkte)

Erstelle für das Wort $t\$ = t_0 \cdots t_{10}\$ = aabaaabaaba\$$ ein Suffix-Array nach dem Algorithmus von Kärkkäinen und Sanders. Gib dabei alle Zwischenschritte an, wobei der rekursive Aufruf von Hand sortiert werden darf.

*Hinweis:* Beim Mischen von $A_0$ mit $A_{12}$ soll für jede Ergebnisposition in $A$ angegeben werden, ob 1 oder 2 Zeichenvergleiche erforderlich sind oder ob auf die Ordnung von $A_{12}$ zurückgegriffen wird.

**Lösungsskizze** (nicht ausreichend für die volle Punktzahl)

Zuerst konstruieren wir alle Tripel, die an Position $i \bmod 3 \neq 0$ beginnen, und sortieren diese




Wir erhalten nun

$$\begin{aligned} t^{(1)} &= 4 \cdot 3 \cdot 3 \cdot 2 \\ t^{(2)} &= 5 \cdot 4 \cdot 4 \cdot 1 \\ \tilde{t} &= 4 \cdot 3 \cdot 3 \cdot 2 \cdot 5 \cdot 4 \cdot 4 \cdot 1 \\ \tilde{t} \cdot 0 &= 4 \cdot 3 \cdot 3 \cdot 2 \cdot 5 \cdot 4 \cdot 4 \cdot 1 \cdot 0 \end{aligned}$$

Wir erhalten nach dem rekursiven Sortieren der Suffixe von $\tilde{t} \cdot 0 = 2 \cdot 2 \cdot 6 \cdot 5 \cdot 3 \cdot 4 \cdot 2 \cdot 1 \cdot 0$:

$$\begin{array}{lcll} A'[0] & = & 8 & \hat{=}\ 0 \quad \hat{=}\ \epsilon \\ A'[1] & = & 7 & \hat{=}\ 10 \quad \hat{=}\ \$\$\$ \\ A'[2] & = & 3 & \hat{=}\ 254410 \quad \hat{=}\ a\$\$\cdots \\ A'[3] & = & 2 & \hat{=}\ 3254410 \quad \hat{=}\ aab\,a\$\$\cdots \\ A'[4] & = & 1 & \hat{=}\ 33254410 \quad \hat{=}\ aab\,aab\,a\$\$\cdots \\ A'[5] & = & 6 & \hat{=}\ 410 \quad \hat{=}\ aba\,\$\$\$ \\ A'[6] & = & 0 & \hat{=}\ 433254410 \quad \hat{=}\ aba\,aab\,aab, a\$\$\cdots \\ A'[7] & = & 5 & \hat{=}\ 4410 \quad \hat{=}\ aba\,aba\,\$\$\$ \\ A'[8] & = & 4 & \hat{=}\ 54410 \quad \hat{=}\ baa\,aba\,aba\,\$\$ \end{array}$$

Damit erhalten wir $A_{12}$:

$$\begin{array}{lcllll} A_{12}[0] & = & 2+3\cdot(\mathbf{7}-4) & = & 11 & \hat{=}\ \$ \\ A_{12}[1] & = & 1+3\cdot(\mathbf{3}) & = & 10 & \hat{=}\ a\$ \\ A_{12}[2] & = & 1+3\cdot(\mathbf{2}) & = & 7 & \hat{=}\ aaba\$ \\ A_{12}[3] & = & 1+3\cdot(\mathbf{1}) & = & 4 & \hat{=}\ aabaaba\$ \\ A_{12}[4] & = & 2+3\cdot(\mathbf{6}-4) & = & 8 & \hat{=}\ aba\$ \\ A_{12}[5] & = & 1+3\cdot(\mathbf{0}) & = & 1 & \hat{=}\ abaaabaaba\$ \\ A_{12}[6] & = & 2+3\cdot(\mathbf{5}-4) & = & 5 & \hat{=}\ abaaba\$ \\ A_{12}[7] & = & 2+3\cdot(\mathbf{4}-4) & = & 2 & \hat{=}\ baaabaaba\$ \end{array}$$

Vorname: _____ Name: _____ Matrikelnummer: _____

Nun bestimmen wir $A_0$:

| $A_0$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | 0 | 3 | 6 | 9 |
| 11 | | | | |
| 10 | | | 9 | |
| 7 | | | 9 | 6 |
| 4 | 3 | | | |
| 8 | | | | |
| 1 | 3 | 0 | | |
| 5 | | | | |
| 2 | | | | |
| | 3 | 0 | 9 | 6 |
| | $a$ | $a$ | $b$ | $b$ |
| | $a$ | $a$ | $a$ | $a$ |
| | $a$ | $b$ | \$ | $a$ |

Nun müssen wir noch $A_0$ und $A_{12}$ mischen (hierbei sind in Blau bzw. Rot die Werte mit $i \bmod 3 = 1$ bzw. $i \bmod 3 = 2$ in $A_{12}$):

| $A_0$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | 3 | 0 | 9 | 6 |
| | $a$ | $a$ | $b$ | $b$ |
| | $a$ | $a$ | $a$ | $a$ |

| $A_{12}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| | 11 | 10 | 7 | 4 | 8 | 1 | 5 | 2 |
| | \$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $b$ |
| | | \$ | $a$ | $a$ | $b$ | $b$ | $b$ | $a$ |

| $A$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $11^1$ | $10^*$ | $3^*$ | $7^*$ | $0^*$ | $4^1$ | $8^1$ | $1^1$ | $5^1$ | $9^*$ | $2^*$ | 6 |

Bei den Vergleichen gibt $\cdot^1$ bzw. $\cdot^2$ an, dass ein Vergleich des ersten bzw. zweiten Zeichens der entsprechenden Suffixe zur Bestimmung der Reihenfolge beim Mischen ausreichend war, $\cdot^*$ zeigt an, dass die relative Ordnung der Suffixe in $A_{12}$ erforderlich war. Bei den Werten ohne Zusatzinformation war kein Vergleich mehr erforderlich.

Vorname: _____ Name: _____ Matrikelnummer: _____

## Aufgabe 4 (8 Punkte)

Betrachte das Wort $t = t_1 \cdots t_{12} \in \{A, I, N, S\}^*$ und die zugehörige Burrows-Wheeler-Transformierte $\hat{t} = \hat{t}_0 \cdots \hat{t}_{12} = \text{SNSN\$NANAAAAI} \in \{\$, A, I, N, S\}$.

a) Gib die Tabellen $C(\cdot)$ und $Occ(\cdot, \cdot)$ für $\hat{t}$ an.

b) Rekonstruiere $t$ aus $\hat{t}$.

c) Konstruiere den Wavelet-Tree zu $\hat{t}$.

d) Bestimme die Werte von $Occ(A, 11)$ und $Occ(N, 7)$ nach der in der Vorlesung vorgestellten Methode aus dem Wavelet-Tree aus c).

*Hinweis:* Das zu betrachtende Alphabet ist $\Sigma \cup \{\$\} = \{\$, A, I, N, S\}$, wobei die Ordnung auf dem Alphabet durch die hier angegebene Reihenfolge gegeben ist.

**Lösungsskizze** (nicht ausreichend für die volle Punktzahl)

Wir erhalten die folgenden Tabellen:

| $Occ(\cdot,\cdot)$ | \$ | A | I | N | S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 2 |
| 3 | 0 | 0 | 0 | 2 | 2 |
| 4 | 1 | 0 | 0 | 2 | 2 |
| 5 | 1 | 0 | 0 | 3 | 2 |
| 6 | 1 | 1 | 0 | 3 | 2 |
| 7 | 1 | 1 | 0 | 4 | 2 |
| 8 | 1 | 2 | 0 | 4 | 2 |
| 9 | 1 | 3 | 0 | 4 | 2 |
| 10 | 1 | 4 | 0 | 4 | 2 |
| 11 | 1 | 5 | 0 | 4 | 2 |
| 12 | 1 | 5 | 1 | 4 | 2 |

| $C(\cdot)$ | $i$ |
|---|---|
| \$ | 0 |
| A | 1 |
| I | 6 |
| N | 7 |
| S | 11 |

Der letzte Buchstabe von $t$ ist immer $\hat{t}_0$, also $t_{12} = \hat{t}_0 = S$. Der Buchstabe unmittelbar vor dem zuletzt berechneten Buchstaben $\hat{t}_i$ in $t$ ist $\hat{t}_{i'}$ und berechnet sich mittels

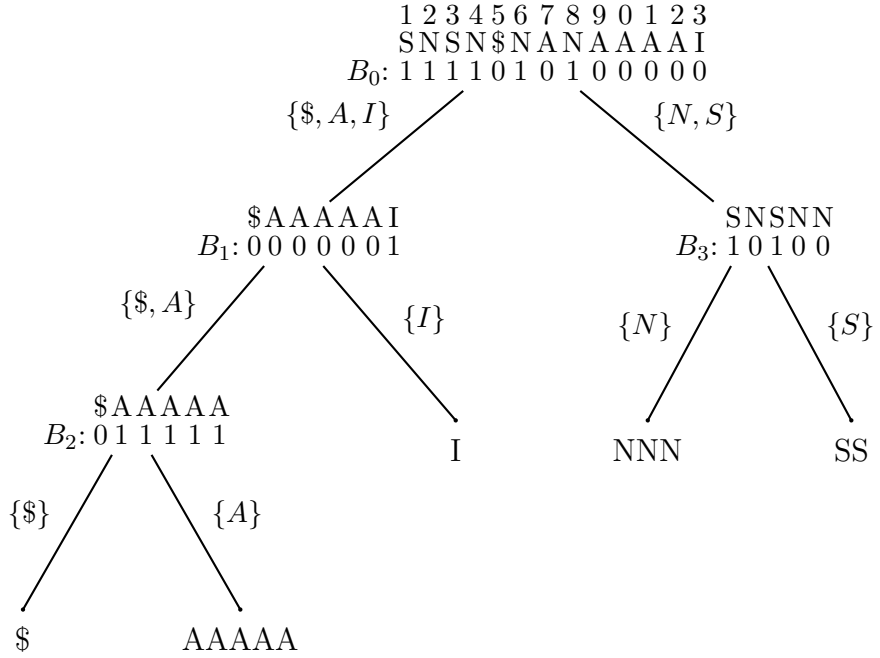
$$i' = \text{LF}(i) = C(\hat{t}_i) + Occ(\hat{t}_i, i-1).$$

Somit gilt:

| $i$ | $\hat{t}_i$ | $C(\hat{t}_i) + Occ(\hat{t}_i, i-1) =$ | $i'$ |
|---|---|---|---|
| 0 | S | $C(S) + Occ(S, -1)$ | $= 11 + 0 = 11$ |
| 11 | A | $C(A) + Occ(A, 10)$ | $= 1 + 4 = 5$ |
| 5 | N | $C(N) + Occ(N, 4)$ | $= 7 + 2 = 9$ |
| 9 | A | $C(A) + Occ(A, 8)$ | $= 1 + 2 = 3$ |
| 3 | N | $C(N) + Occ(N, 2)$ | $= 7 + 1 = 8$ |
| 8 | A | $C(A) + Occ(A, 7)$ | $= 1 + 1 = 2$ |
| 2 | S | $C(S) + Occ(S, 1)$ | $= 11 + 1 = 12$ |
| 12 | I | $C(I) + Occ(I, 11)$ | $= 6 + 0 = 6$ |
| 6 | A | $C(A) + Occ(A, 5)$ | $= 1 + 0 = 1$ |
| 1 | N | $C(N) + Occ(N, 0)$ | $= 7 + 0 = 7$ |
| 7 | N | $C(N) + Occ(N, 6)$ | $= 7 + 3 = 10$ |
| 10 | A | $C(A) + Occ(A, 9)$ | $= 1 + 3 = 4$ |
| 4 | \$ | Fertig! | $=$ |

Somit ist $t =$ ANNAISANANAS.

Wir erhalten den folgenden Wavelet Tree für Teil c.




Für den Teil d erhalten wir Folgendes.

1) Zuerst zur Bestimmung von $Occ(A, 11)$. Da A in der kleineren Hälfte des Alphabets $\{\$, A, I, N, S\}$ liegt, bestimmen wir an der Wurzel für $B_0 = 1111010100000$

$$\text{rank}_0^{B_0}(11) = 5$$

Da A nun wieder in der kleineren Hälfte des betrachteten Alphabets $\{\$, A, I\}$ liegt, bestimmen wir für $B_1 = 0000001$

$$\text{rank}_0^{B_1}(5) = 5$$

Da A nun in der größeren Hälfte von $\{\$, A\}$ liegt, bestimmen wir an der Wurzel für $B_2 = 011111$

$$\text{rank}_1^{B_2}(5) = 4$$

Also ist $Occ(A, 11) = 4$.

2) Nun zur Bestimmung von $Occ(N, 7)$. Da N in der größeren Hälfte des Alphabets $\{\$, A, B, I, N, S\}$ liegt, bestimmen wir an der Wurzel für $B_0 = 1111010100000$

$$\text{rank}_1^{B_0}(7) = 5$$

Da N nun in der kleineren Hälfte des betrachteten Alphabets $\{I, N, S\}$ liegt, bestimmen wir für $B_3 = 10100$

$$\text{rank}_0^{B_3}(5) = 3$$

Also ist $Occ(N, 7) = 3$.

Vorname: _____ Name: _____ Matrikelnummer: _____

## Aufgabe 5 (8 Punkte)

Sei $\Sigma$ ein Alphabet, $t = t_1 \cdots t_n \in \Sigma^n$ und $1 < k \in \mathbb{N}$. Konstruiere einen Algorithmus mit optimaler Laufzeit, der alle kürzesten Wörter $w \in \Sigma^*$ findet, die in $t$ mindestens $k$-mal auftreten, aber das zugehörige gespiegelte Wort $w^R$ nicht in $t$ auftritt.

Zeige die Korrektheit des Algorithmus und analysiere dessen Laufzeit.

*Hinweis:* Für $w = w_1 \cdots w_\ell \in \Sigma^\ell$ ist das *zugehörige gespiegelte Wort* $w^R = w_\ell \cdots w_1$.

**Lösungsskizze** (nicht ausreichend für die volle Punktzahl)

Konstruiere einen Suffix-Baum $T$ für $t' = t'_1 \cdots t'_{2n+2} = t\sharp t^R\$$. Mit Hilfe einer Tiefensuche in $T$ zähle getrennt für jeden inneren Knoten die Anzahl der nachfolgenden Blätter mit einem Suffix in $t$ (also Blätter mit Index aus $[1 : n]$) und in $t^R$ (also Blätter mit Index aus $[n + 2 : 2n + 1]$) und bestimme ebenfalls die Worttiefe jedes Knotens in $T$.

Das zu einem inneren Knoten korrespondierende Wort $w$, dessen Zähler für Blätter in $t$ größer gleich $k$ und dessen Zähler für $t'$ gleich 0 ist, kommen eben in $t$ mindestens $k$-mal vor und treten in $t^R$ nicht auf. Beachte, dass ein Wort $w$ in $t^R$ genau dann nicht auftritt, wenn $w^R$ nicht in $t$ auftritt. Dies gilt auch für alle Wörter, die auf der eingehenden Kante zu diesem Knoten enden. Markiere also in $T$ (mit einer weiteren Tiefensuche oder parallel zur ersten) alle inneren Knoten mit den oben genannten Eigenschaften.

Für die Ausgabe kürzester solcher Wörter kommen nun zunächst nur die Wörter in Frage, die zu Kanten gehören, deren Endknoten markiert sind und deren Anfangsknoten eben nicht markiert ist. Das korrespondiere Teilwort zum Anfangsknoten ist nämlich kürzer als die korrespondierenden Wörter zu Positionen auf der Kante zum oder am Endknoten. Da wir nur kürzeste Teilwörter ausgeben wollen, sind nur Teilwörter relevant, die zur ersten Position auf einer Kante korrespondieren, für die der zugehörigen Anfangsknoten nicht markiert und der Endknoten markiert ist. Des Weiteren muss der Anfangsknoten unter all solchen eine niedrigste Worttiefe besitzen. Mit einer weiteren Tiefensuche können wir nun die minimale Worttiefe von inneren unmarkierten Knoten ermitteln, die mindestens ein markiertes Kind besitzen. Solche inneren Anfangsknoten mit minimaler Worttiefe bezeichnen wir im Folgenden als *relevant*.

Für die Ausgabe ermitteln wir mit einer weiteren Tiefensuche für die Kinder von relevanten Knoten die zugehörigen Blattlisten beschränkt auf Teilwörter aus $t$, also mit Blattmarkierungen aus $[1 : n]$. Tatsächlich müssen wir keine vollständigen Blattlisten speichern, sondern nur einen Index aus dieser Blattliste, z.B. den kleinsten. Für die als relevant markierten Knoten mit Worttiefe $d$ geben wir nun zu jedem markierten Kind die zugehörige Referenz $(s, e)$ des Wortes aus, das nach dem ersten Zeichen auf der Kante zu dem markierten Kind endet, wobei $s$ ein (und nur ein) Knoten aus der Blattliste des Kindes ist und $e = s + d$ (das Wort entspricht also $t_s \cdots t_e$).

Die Laufzeit für Ukkonens Algorithmus zur Kontrsuktion von $T$ ist linear in der Länge der Zeichenreihe $t\sharp t^R\$$, also $O(2n) = O(n)$. Auch die oben genannten Tiefensuchen in $T$ können jeweils in Zeit $O(n)$ ausgeführt werden. Die Konstruktion der Blattlisten an den Kindern von relevanten Knoten ist nach Vorlesung ebenfalls in linearer Zeit möglich. Da relevante Knoten nicht Vorfahren eines anderen relevanten Knotens sein können, kann maximal an jedem markierten Kind eines relevanten Knoten eine Referenz ausgegeben werden, also insgesamt maximal $O(n)$. Somit ist die Laufzeit $O(n)$, was optimal ist.