1.3 Suffixbäume

Der KMP-Algorithmus und auch der BM-Algorithmus haben den Musterstring vorbereitet um dann in Linearzeit alle Vorkommen des Musterstrings im Textstring zu finden. Falls viele verschiedene Musterstrings im Textstring gesucht werden, ist es mitunter sinnvoller, den Textstring vorzubereiten, so dass die einzelnen Musterstrings effizient im Textstring gefunden werden.

Ziel:
Entwicklung einer Datenstruktur für den Textstring, die eine effiziente Lösung von Stringmatchingproblemen ermöglicht. Insbesondere sollen alle Vorkommen eines Musterstrings y mit $|y| = m$ in Zeit $O(m + A(y))$ bestimmt werden können, wobei $A(y)$ die Anzahl der Vorkommen von y im Textstring bezeichnet.

Literatur:

Dan Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997, 89 - 107.

Esko Ukkonen, Online-Construktion of Suffix Trees, Algorithmica 14 (1995), 249 - 260.

Zunächst benötigen wir einige Definitionen:

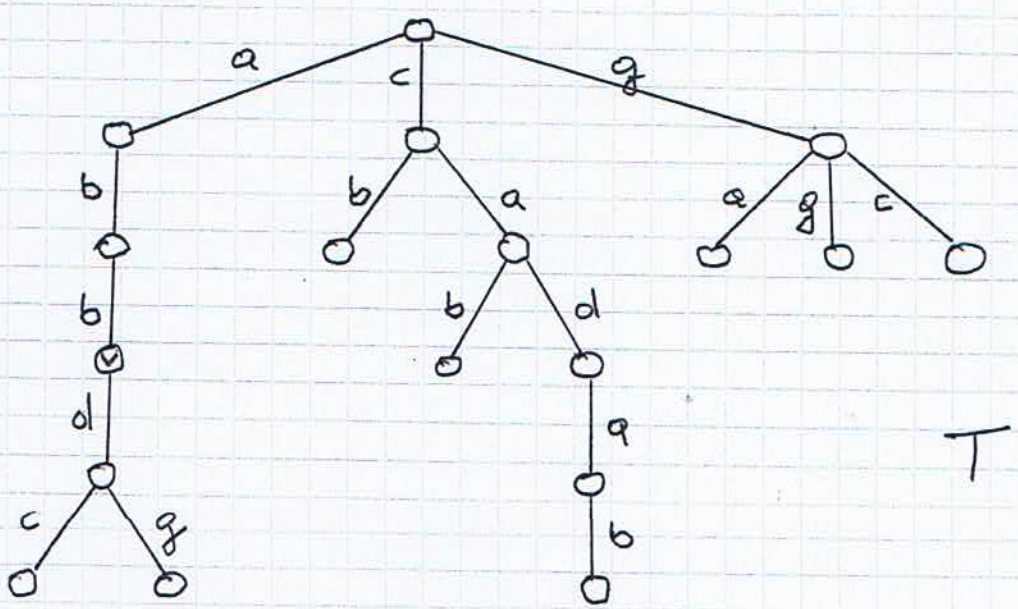
Ein <u>Trie</u> bezüglich eines Alphabets Σ, $|\Sigma| =: k$ ist ein Baum $T = (V, E)$, für den gilt:

i) Jeder innere Knoten hat Ausgangsgrad $\leq k$.
ii) Die ausgehenden Kanten eines inneren Knotens sind mit paarweise verschiedenen Elementen aus Σ beschriftet

Ein Pfad P von der Wurzel des Tries T zu einem Blatt v korrespondiert zu demjenigen String, den wir durch Konkatenation der Kantenbeschriftungen auf P in der Reihenfolge, in der diese auf P vorkommen, erhalten.

Somit repräsentiert ein Trie T eine Menge $S(T)$ von Strings. Diese korrespondiert zur Gesamtheit aller Pfade von der Wurzel von T zu einem Blatt.

<u>Beispiel</u>:



$S(T) = \{ abbdc, abbdg, cb, cab, cadab, ga, gg, gc \}$

◊

Obige Menge S(T) ist präfixfrei, d.h., kein Element von S(T) ist Präfix eines anderen Elementes von S(T). Falls wir zu S(T) noch das Element abb hinzunehmen, dann ist die resultierende Menge nicht mehr präfixfrei, da der String abb auch Präfix des Strings abbdc ist.

Folgende Erweiterungen der Definition eines Tries erlauben auch die Repräsentation von Stringmengen S, die nicht präfixfrei sind:

1) Innere Knoten, in denen solch ein Präfix endet, werden markiert, so dass die Pfade von der Wurzel des Tries zu einem markierten Knoten oder zu einem Blatt eindeutig zu den Elementen von S korrespondieren.

2) Die eingehenden Kanten von Blättern erhalten keine Beschriftung mit einem Symbol aus dem Alphabet Σ und der Trie ist so organisiert, dass auch bei nicht präfixfreien Mengen S die Pfade von der Wurzel zu einem Blatt eindeutig zu den Elementen von S korrespondieren.

Wir werden nachfolgend stets die erste Erweiterung der Definition eines Tries verwenden.

Beispiel (Fortführung)

Wenn wir im obigen Trie T den Knoten v markieren, dann erhalten wir einen Trie T', der die Menge

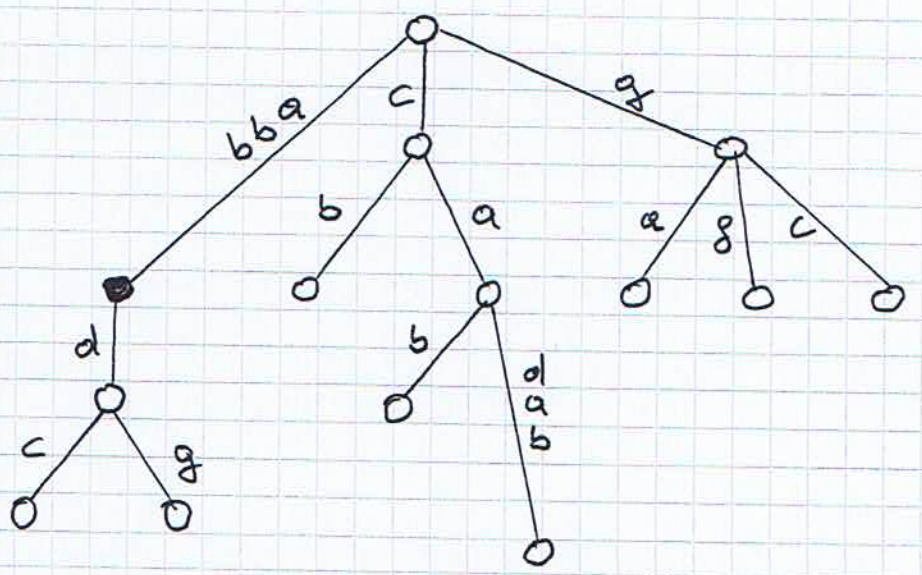
$$S(T') = \{abb, abbdc, abbdg, cb, cab, cadab, ga, gg, gc\}$$

repräsentiert

◊

Aus einem Trie T erhalten wir den korrespondierenden kompakten Trie T_c, indem wir alle Pfade P, auf denen nur nichtmarkierte Knoten mit Ausgangsgrad eins liegen, zu einer Kante zusammenfassen und diese mit dem zu P korrespondierenden String beschriften.

Beispiel (Fortführung):



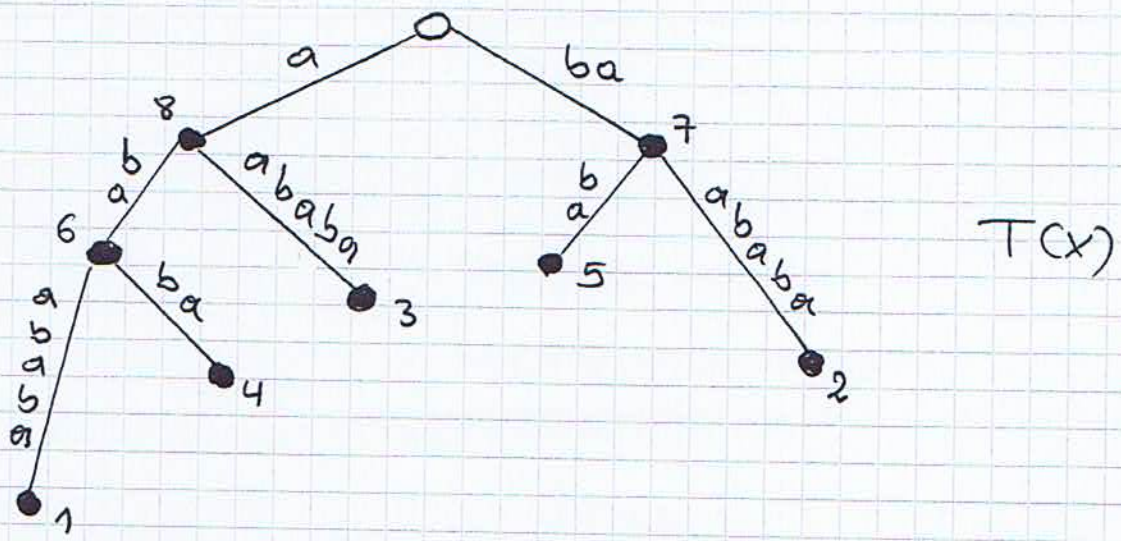
Der zu T' korrespondierende kompakte Trie T_c'.

◆

Ein Suffixbaum $T(x)$ für einen String $x = a_1 a_2 \dots a_n \in \Sigma^n$ ist ein kompakter Trie bezüglich des Alphabets Σ, der n markierte Knoten, die mit den Zahlen $1, 2, \dots, n$ numeriert sind, enthält. Der Pfad von der Wurzel zum Knoten mit Nummer i korrespondiert zu demjenigen Suffix von x, der in der i-ten Position von x startet. D.h., zu $a_i a_{i+1} \dots a_n$.

Beispiel:

Sei $x = abaababa$



$T(x)$

Bevor wir einen effizienten Algorithmus zur ~~Effizienten~~ Berechnung eines Suffixbaumes $T(x)$ für einen gegebenen Textstring x entwickeln, zeigen wir, wie mit Hilfe des Suffixbaumes das Stringmatchingproblem effizient gelöst werden kann.

gegeben: Textstring $x := a_1 a_2 \dots a_n$, ein Suffixbaum $T(x)$ und ein Musterstring $y := b_1 b_2 \dots b_m$.

gesucht: Alle Positionen in x, in denen der Musterstring y beginnt.

Lösungsmethode:

- Starte in der Wurzel von $T(x)$ und konstruiere den längstmöglichen Pfad P in $T(x)$ mit Kantenmarkierungen $b_1 b_2 \dots b_j$, $0 \leq j \leq m$.
- Zwei Fälle können eintreten:

1. Fall: $j < m$

Dann ist der Musterstring y kein Teilstring von x.

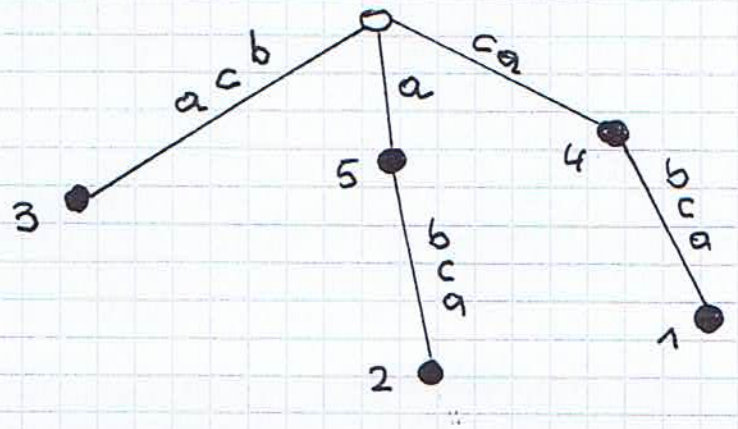
2. Fall: $j = m$

Sei v derjenige Knoten in $T(x)$, in dem der konstruierte Pfad P endet. Dann bezeichnen die Nummern der markierten Knoten im Teilbaum mit Wurzel v exakt diejenigen Positionen in x, in denen der Musterstring y beginnt.

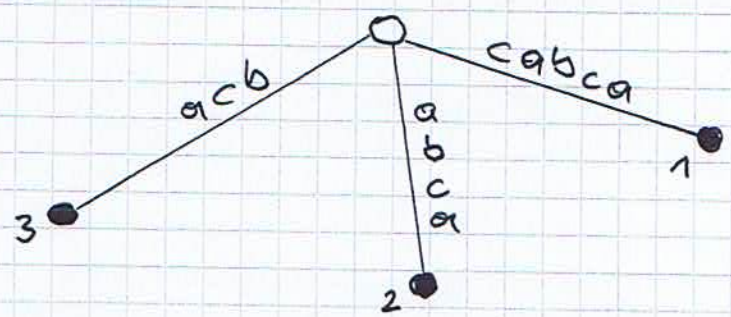
Übung:
Entwickeln Sie einen formalen Beweis für die Korrektheit des obigen Verfahrens.

05.11.

Betrachten wir für den String $x = cabca$ folgenden Suffixbaum $T(x)$



$T(x)$ enthält Knoten, die den Ausgangsgrad 1 haben. Da $T(x)$ kompakt ist, sind alle Knoten mit Ausgangsgrad 1 markiert. Falls wir in $T(x)$ alle Pfade, auf denen nur Knoten mit Ausgangsgrad 1 liegen, zu einer Kante zusammenfassen und diese mit derjenigen String, der zu dem Pfad korrespondiert, beschriften, dann erhalten wir folgenden Baum $T'(x)$:



Allgemein definieren wir: Sei $T(z)$ ein Suffixbaum für einen String z. Aus $T(z)$ erhalten wir einen <u>impliziten Suffixbaum</u> $T'(z)$ für z, indem wir Pfade, auf denen nur Knoten mit Ausgangs-

grad eins liegen, zu einer Kante zusammenfassen und diese mit demjenigen String, der zu dem Pfad korrespondiert, beschriften.

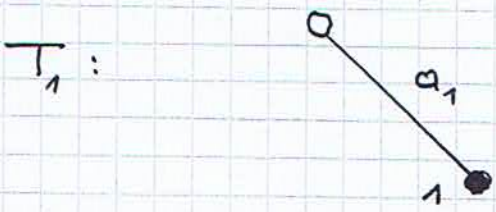
Sei $x := a_1 a_2 \dots a_n$ derjenige String, für den wir einen Suffixbaum konstruieren möchten. Die Idee besteht darin, zunächst einen impliziten Suffixbaum für x zu konstruieren und dann diesen zu einen Suffixbaum zu erweitern.

Hierzu konstruieren wir mit dem Präfix a_1 beginnend sukzessive für jeden Präfix $a_1 a_2 \dots a_i$ von x einen impliziten Suffixbaum T_i. Gegeben einen impliziten Suffixbaum T_n für x konstruieren wir dann einen Suffixbaum für x.

Demzufolge ist der Algorithmus in n Phasen unterteilt. In Phase i, $1 \leq i \leq n$ wird ein impliziter Suffixbaum T_i für den Präfix $a_1 a_2 \dots a_i$ von x konstruiert. Die Phasen selbst wie folgt aus:

<u>Phase 1:</u>

T_1 besitzt nur eine Kante. Diese ist mit a_1 markiert. D.h.,



Annahme:

Für $a_1 a_2 \dots a_i$, $i \geq 1$ ist ein impliziter Suffixbaum T_i konstruiert.

Phase $i+1$:

Die $(i+1)$-te Phase ist in $i+1$ Erweiterungs-Schritte unterteilt. Dabei korrespondiert der j-te Erweiterungsschritt, $1 \leq j \leq i+1$ zum Suffix $a_j a_{j+1} \dots a_{i+1}$ von $a_1 a_2 \dots a_{i+1}$.

Nehmen wir an, dass die Erweiterungsschritte $1, 2, \dots, j-1$ bereits erfolgt sind. Wir werden nun den j-ten Erweiterungsschritt beschreiben.

Erweiterung j:

Ziel:

Dafür zu sorgen, dass in der Wurzel des impliziten Suffixbaumes T_{i+1} startend, ein Pfad mit Markierung $a_j a_{j+1} \dots a_{i+1}$ existiert.

Da T_{i+1} ein impliziter Suffixbaum ist, kann solch ein Pfad auf einer Kante, in einem inneren Knoten oder in einem Blatt enden.

Durchführung:

Finde, in der Wurzel des aktuellen Baumes startend, das Ende des Pfades mit Markierung $a_j a_{j+1} \dots a_i$.

Da T_i solch einen Pfad enthielt, befindet sich auch solcher im aktuellen Baum. Jenachdem, wo dieser Pfad endet, unterscheiden wir drei Fälle:

1. Fall: Pfad mit Markierung $a_j a_{j+1} \dots a_i$ endet in einem Blatt.

- Konkateniere a_{i+1} an das Ende der Markierung derjenigen Kante, die in dem Blatt des Pfades endet.

2. Fall: Pfad mit Markierung $a_j a_{j+1} \dots a_i$ endet in einem inneren Knoten v.

Zwei Unterfälle können eintreten.

2.1 Die Markierung einer der ausgehenden Kanten von v hat Präfix a_{i+1}.

Dann ist der gewünschte Pfad mit Markierung $a_j a_{j+1} \dots a_{i+1}$ bereits im aktuellen Suffixbaum.

$\rightsquigarrow$

Tue nichts.

2.2 Alle Markierungen der ausgehenden Kanten von v beginnen mit einem Symbol $\neq a_{i+1}$.

$\rightsquigarrow$

v erhält einen neuen Nachfolgerknoten w. Die

Kante (v,w) erhält die Markierung a_{i+1} und w die Nummer j. Der Knoten w ist dann ein Blatt im aktuellen Suffixbaum.

3. Fall Pfad mit Markierung $a_j a_{j+1} \dots a_i$ endet auf einer Kante e.

Sei $\beta c \gamma$ die Markierung der Kante e, wobei β der Suffix von $a_j a_{j+1} \dots a_i$ ist. D.h., c ist das erste Symbol der Kantenmarkierung, das nicht zu $a_j a_{j+1} \dots a_i$ korrespondiert. Da der Pfad auf der Kante e und nicht in einem Knoten endet, existiert c.

Zwei Unterfälle können eintreten:

3.1 $c = a_{i+1}$

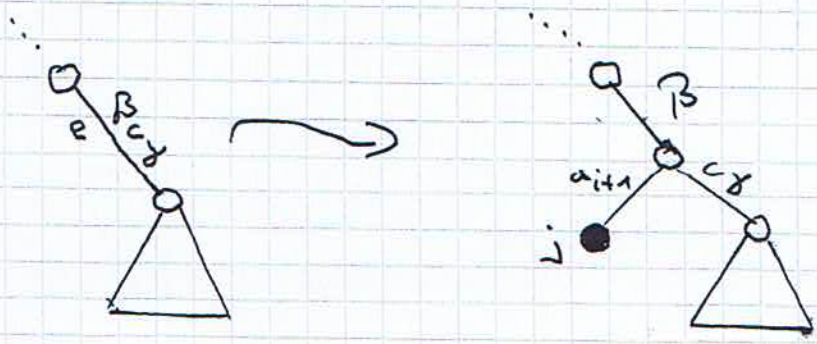
Dann ist der gewünschte Pfad mit Markierung $a_j a_{j+1} \dots a_{i+1}$ bereits im aktuellen Suchbaum

$\rightsquigarrow$

Tue nichts.

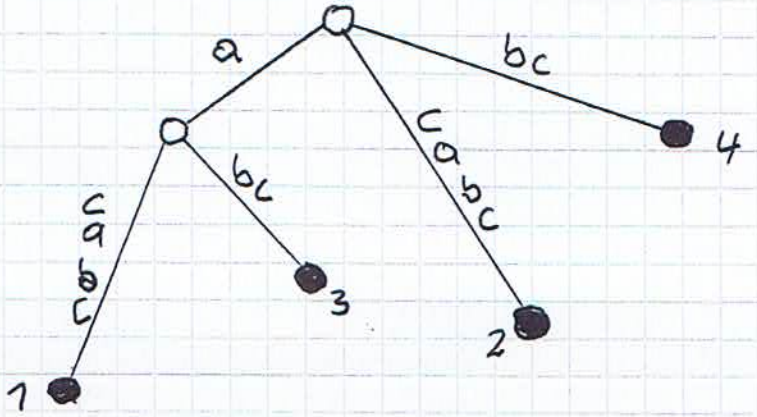
3.2 $c \neq a_{i+1}$

Führe folgende Transformation durch:

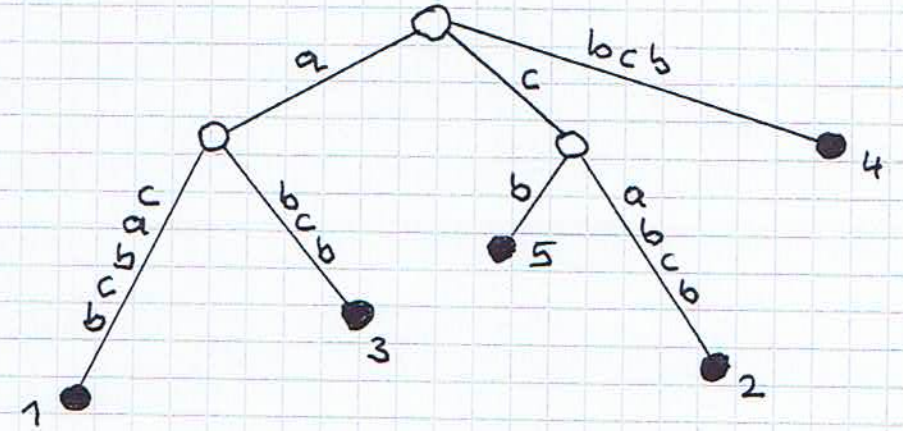


Beispiel:

Betrachten wir folgenden impliziten Suffixbaum für acabc:



Nach Hinzufügen des Symbols b erweitert sich obiger implizite Suffixbaum zu folgenden impliziten Suffixbaum für acabcb:



Ziel:

Abschätzung des Zeit- und Platzbedarfes des obigen Algorithmus.

Platzbedarf:

In jedem Blatt von T_n endet eines der möglichen n Suffixe. Also enthält T_n maximal n Blätter und somit auch maximal $n-1$ innere Knoten und höchstens $2n-2$ Kanten. Eine Kante ist mit einem Teilstring von x der Länge $\leq n$ beschriftet.

$\Rightarrow$

Der Platzbedarf von T_n ist $O(n^2)$.

Zeitanalyse:

- Kosten $K(T_{i+1})$ für die Konstruktion von T_{i+1}:

Für die Erweiterung j ist zunächst das Ende des Pfades mit Markierung $a_j a_{j+1} \dots a_i$ zu finden. Falls man hierzu in der Wurzel des aktuellen Baumes startet und den Pfad abläuft, dann benötigt man Länge des Pfades, also $O(i-(j-1)) = O(i+1-j)$ Zeit. Ansonsten benötigt man für die Erweiterung konstante Zeit.

$\Rightarrow$

$$K(T_{i+1}) = O\left(\sum_{j=1}^{i} i+1-j\right)$$

$$= O\left(\sum_{j=0}^{i-1} j+1\right)$$

$$= O(i^2).$$

- Gesamtkosten:

$$O\left(\sum_{i=1}^{n} i^2\right) = O(n^3).$$

Unser Ziel ist nun die Reduktion der benötigten Zeit und des benötigten Platzes.

Ziel 1:

Reduktion auf $O(n^2)$ Zeit und $O(n^2)$ Platz.

Falls wir für jedes j das Ende des Pfades mit Markierung $a_j a_{j+1} \dots a_i$ in konstanter Zeit finden, dann reduzieren sich die Kosten für die Kosten von T_i auf $O(i)$.

$\Rightarrow$

Die Gesamtkosten betragen dann

$$O\left(\sum_{i=1}^{n} i\right) = O(n^2).$$

Idee:

Nach der Konstruktion von T_i, $1 \le i < n$ kennen wir für $1 \le j \le i$ das Ende des Pfades von der Wurzel aus beginnend mit Markierung $a_j a_{j+1} \dots a_i$. Wenn wir uns dieses merken, dann erhalten wir bei der Konstruktion von T_{i+1} dieses in konstanter Zeit.

Lediglich für $j = i+1$ muss das Ende des Pfades mit Markierung q_{i+1} explizit durch Ablaufen des Pfades ermittelt werden, falls dieser überhaupt existiert. Hierfür reicht konstante Zeit.

$\Rightarrow$

Wir haben somit die Zeit auf $O(n^2)$ reduziert. Da wir an der Struktur von T_n nichts geändert haben, hat sich der benötigte Platzbedarf nicht geändert und ist nach wie vor $O(n^2)$.

Ziel 2:
Reduktion auf $O(n)$ Zeit und $O(n)$ Platz.

Folgendes Beispiel zeigt, dass der bisherige Suffixbaum in der Tat $\Omega(n^2)$ Platz benötigen kann

Beispiel:

Sei

$$x = abcdefghijklmnopqrstuvwxyz$$

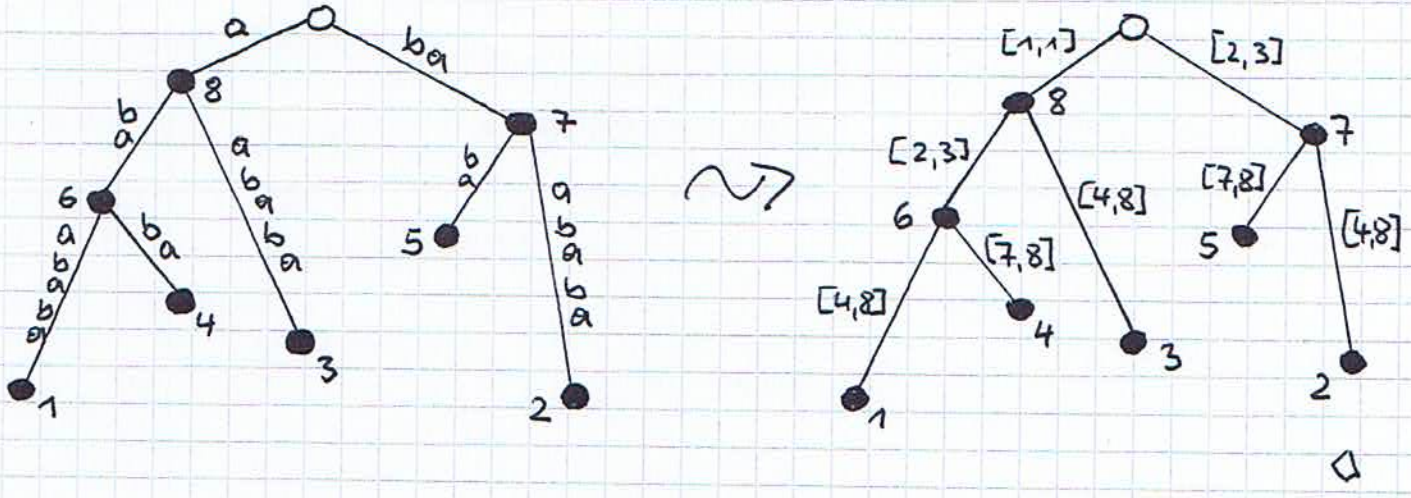
Jedes Suffix beginnt mit einem unterschiedlichen Symbol. Also besitzt die Wurzel des impliziten Suffixbaumes 26 Söhne. Jede korrespondierende Kante ist mit einem kompletten Suffix markiert. Somit benötigen wir für die Markierungen $\sum_{j=1}^{26} j = \frac{26 \cdot 27}{2}$ Platz

Idee:

Jede Kantenmarkierung korrespondiert zu einem Teilstring des Textstrings x. Anstatt diesen explizit hinzuschreiben genügt es, den Anfang und das Ende dieses Teilstrings zu spezifizieren.

Beispiel:

Sei x = abaababa. Wir erhalten folgende Suffixbäume:



$\Rightarrow$

Wir haben den Platzbedarf von $O(n^2)$ auf $O(n)$ reduziert.

Für die Reduktion der benötigten Zeit betrachten wir nochmals die Durchführung von Erweiterung j. Je nachdem, welche Aktion der Erweiterungsschritt durchführt, fassen wir die Fälle und Unterfälle von Erweiterung j zusammen. Wir erhalten dann:

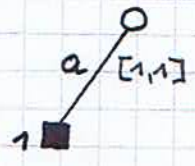
Aktion	Fall bzw. Unterfall
1. Tue nichts	2.1 und 3.1
2. Erweitere Blattmarkierung	1.
3. Füge eventuell inneren Knoten plus Blatt ein.	2.2 und 3.2

Um zu verdeutlichen, was im einzelnen passiert, führen wir die Konstruktion anhand obigen Beispiels durch.

Beispiel (Fortführung):

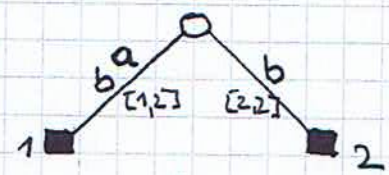
Sei $x = abaababa$. Wir zeichnen Blätter als Quadrate und innere Knoten als Kreise.

T_1:



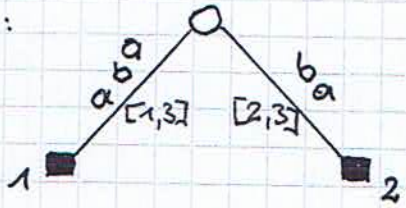
3

T_2:



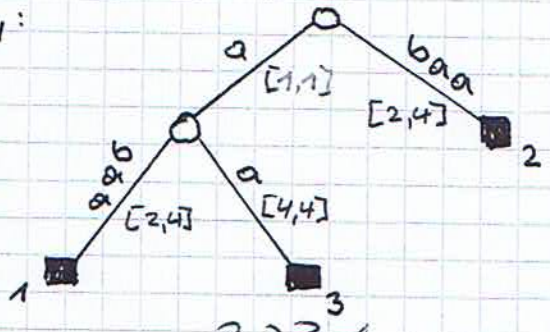
23

T_3:



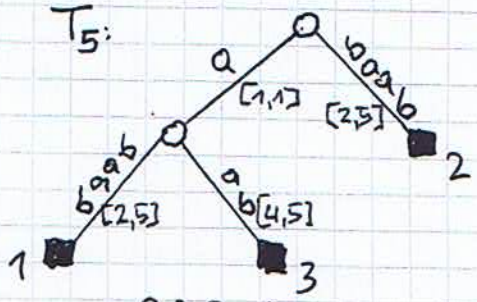
221

T_4:



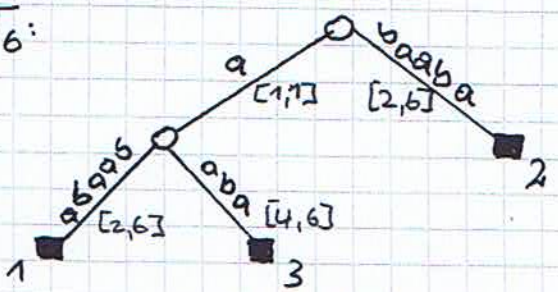
2231

T_5:



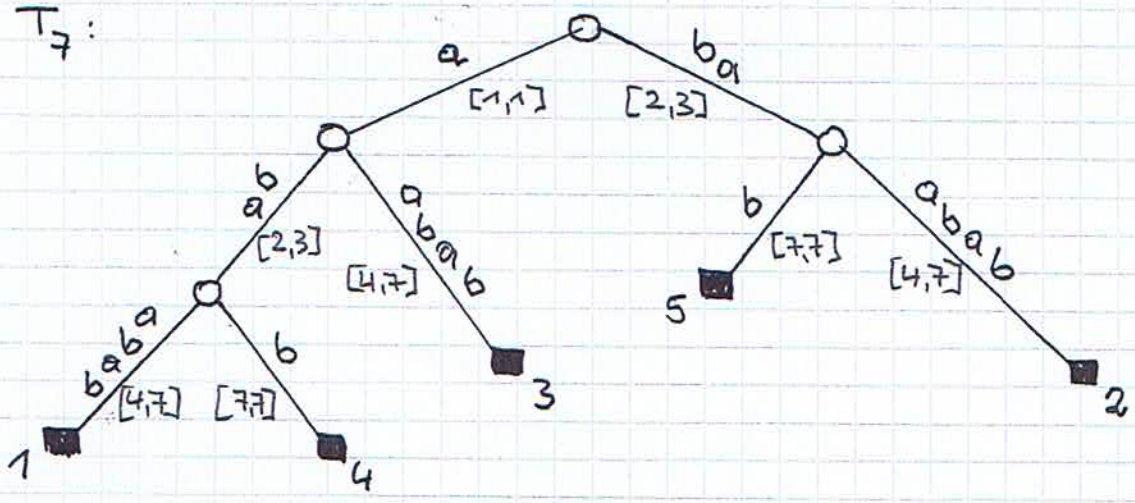
22211

T_6:

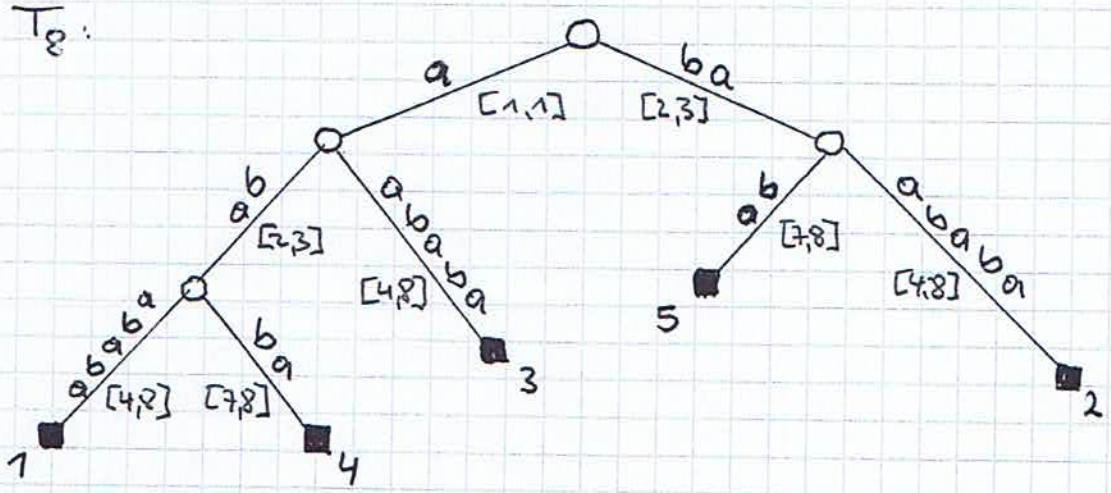


222111

T_7:



T_8:



Beobachtung:

1) Falls im aktuellen Suffixbaum T ein Pfad in der Wurzel - startend mit Markierung $c_1 c_2 \dots c_k$ existiert, dann existiert in T für je- des $1 \leq \ell \leq k$ ein Pfad, der in der Wurzel anfängt, mit Markierung $c_\ell c_{\ell+1} \dots c_k$.

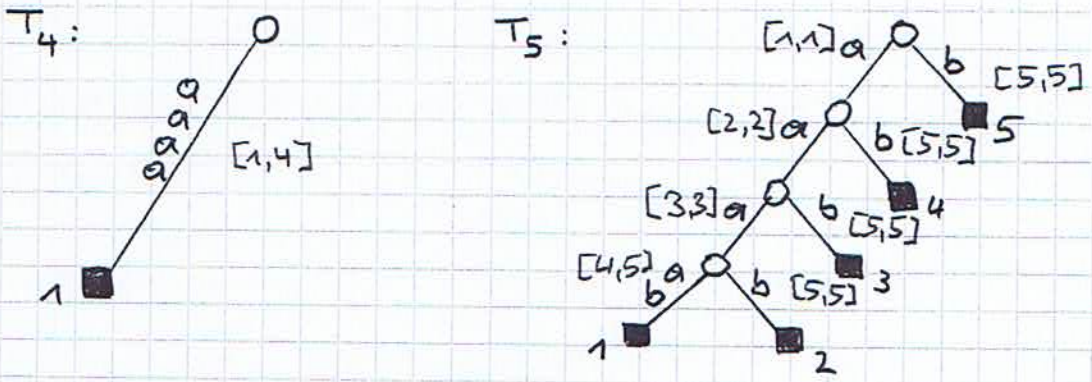
$\Rightarrow$

Wenn bei der (Konstruktion) von T_{i+1} bei der Erweiterung j die 1. Aktion "tue nichts" durchgeführt wird, dann wird auch bei jeder nachfolgenden Er=

weiterung ℓ, $j+1 \leq \ell \leq i+1$ diese Aktion durchgeführt.

2) Wenn ein Blatt mit Nummer k kreiert wird, dann bleibt dieser Knoten auch in den nachfolgenden Suffixbäumen ein Blatt mit Nummer k.

Zur Verdeutlichung der nächsten Beobachtung betrachten wir für $x = aaaab$ die Konstruktion von T_5 aus T_4.



Beobachtung:

3) Bei der Konstruktion von T_{i+1} aus T_i hat die Folge der Erweiterungsschritte folgende Struktur:

- Folge von Aktionen vom Typ 2 "Erweitere Blattmarkierung"

gefolgt von

- Folge von Aktionen vom Typ 3 "Füge eventuell inneren Knoten plus Blatt ein."

gefolgt von

- Folge von Aktionen vom Typ 1 "Tue nichts".

Dabei können eine oder zwei der oben definierten Folgen leer sein.

$\Rightarrow$

Für $1 \leq k \leq n$ gilt: Wenn ein Blatt mit Nummer k kreiert wird, dann gilt für $1 \leq \ell < k$, dass das Blatt mit Nummer ℓ bereits existiert.

Für unseren Konstruktionsalgorithmus haben obige Beobachtungen folgende Implikationen:

a) Die Erweiterungen der Blattmarkierungen müssen nicht explizit durchgeführt werden. D.h., für eingehende Kanten der Blätter ist lediglich der Anfang des Teilstrings von x, der zur Kantenmarkierung korrespondiert, zu spezifizieren. Das Ende ist stets das aktuelle Ende in t.

$\Rightarrow$

Die Folge von Aktionen vom Typ 2 wird nicht explizit durchgeführt.

b) Annahme:

Aus T_i soll der Suffixbaum T_{i+1} konstruiert werden. In T_i existieren die Blätter mit Nummern $1, 2, \ldots, k$. Blatt mit Nummer $k+1$

existiert nicht.

Um im Fall $k < i$ bei der Konstruktion von T_{i+1} mit der expliziten Durchführung der Konstruktion beginnen zu können, benötigen wir

- in T_i das Ende des Pfades P, beginnend in der Wurzel mit Markierung $a_{i+1} a_{i+2} \dots a_i$.

Gemäß der obigen Überlegungen endet P in einem inneren Knoten oder auf einer Kante.

Beobachtung:
Wegen obiger Beobachtung 1 und der Wahl von k ist P genau derjenige Pfad, mit dem die Konstruktion von T_i beendet wurde.

$\Rightarrow$

Das Ende von P ist bekannt, so dass die Konstruktion von T_{i+1} am Ende von P gestartet werden kann.

Zwei Fälle können nun eintreten:

1. Fall: Aktion vom Typ 1 wird durchgeführt.

Wegen Beobachtung 3 sind wir mit der Konstruktion von T_{i+1} fertig. Des Weiteren kennen wir nun das Ende des Pfades, beginnend in der Wurzel mit der Markierung $a_{i+1} a_{i+2} \dots a_{i+1}$, so dass die explizite Konstruktion von T_{i+2} hier gestartet werden kann.

2. Fall Aktion vom Typ 3 wird durchgeführt.

Je nachdem, wo der Pfad P endet, unterscheiden wir zwei Unterfälle.

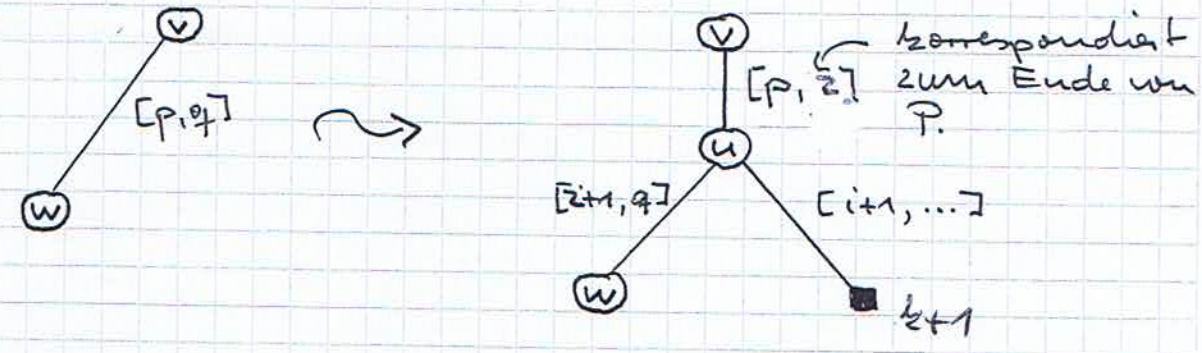
2.1 P endet in einem inneren Knoten v.

Gemäß unserer Konstruktion beginnen alle Markierungen der ausgehenden Kanten von v mit einem Symbol $\neq a_{i+1}$. Der Knoten v erhält ein neues Blatt b mit Nummer $k+1$ als weiteren Sohn. Die Kante (v,b) wird mit $[i+1,\dots]$ markiert.

2.2 P endet auf einer Kante $e := (v,w)$.

Sei $[p,q]$ die Markierung von e. Falls w ein Blatt ist, dann interpretieren wir q als $i+1$.

Situation nebst Modifikation:



Um die Konstruktion fortsetzen zu können, benötigen wir das Ende desjenigen Pfades P', der in der Wurzel anfängt und Markierung $a_{i+2}\, a_{i+3} \dots a_i$ hat. Gemäß obiger Beobachtung 1 existiert P'.

Ziel:
Entwicklung einer Datenstruktur, die den Zugriff auf das Ende von P' in konstanter Zeit ermöglicht.

Zunächst benötigen wir einige Bezeichnungen. Sei v ein Knoten im aktuellen Suffixbaum T. Dann bezeichnet $m(v)$ die zum Pfad von der Wurzel von T zum Knoten v korrespondierende Kantenmarkierung. Sei $m(v) := a\alpha$ mit $a \in \Sigma$ und $\alpha \in \Sigma^*$. Dann definieren wir

$$s(v) := \begin{cases} \text{Knoten } w \text{ in } T \text{ mit } m(w) = \alpha & \text{falls } w \text{ existiert} \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

Beobachtung:
Sei v ein Blatt mit Nummer p und u ein Blatt mit Nummer $p+1$ in T. Dann gilt $s(v) = u$.

Annahme:
Jeder Knoten v im aktuellen Suffixbaum T enthält einen Zeiger auf den Knoten $s(v)$, falls dieser existiert.

Dann können wir wie folgt das Ende des Pfades P' finden. Je nachdem, ob oben Fall 2.1 oder Fall 2.2 zutreffend war, unterscheiden wir zwei Fälle:

2.1 P endet in einem inneren Knoten v.

Da jeder innere Knoten mindestens zwei Söhne besitzt, gibt es Pfade mit Markierungen

$a_{z+1} a_{k+2} \dots a_i c \dots$ und auch $a_{z+1} a_{k+2} \dots a_i d \dots$

mit $c \neq d$ von der Wurzel zu einem Blatt.

Beobachtung 1 $\Rightarrow$

Es gibt Pfade mit Markierungen

$a_{z+2} \dots a_i c \dots$ und $a_{z+2} \dots a_i d \dots$

von der Wurzel zu einem Blatt.

$\Rightarrow$

Der Pfad P^4, der in der Wurzel anfängt und die Markierung $a_{z+2} \dots a_i$ hat, endet in einem inneren Knoten.

$\Rightarrow$

Der Knoten $s(v)$ existiert im aktuellen Suffixbaum T.

Annahme $\Rightarrow$

Wir haben in T einen sogenannten <u>Suffixzeiger</u> von v nach $s(v)$.

<u>Übung</u>

Geben Sie ein Beispiel an, in dem $s(v)$ eine aus=

gehende Kante, deren Markierung mit einem Symbol $g \in \Sigma$ beginnt, besitzt, obwohl dies für v nicht der Fall ist.

Obige Übungsaufgabe besagt, dass nun zwei Fälle eintreten können:

2.1.1 Eine der ausgehende Kanten von $s(v)$ hat Markierung, die mit a_{i+1} beginnt.

$\rightsquigarrow$

Führe Aktion vom Typ 1 "Tue nichts" durch.

Wegen Beobachtung 3 sind wir mit der Konstruktion von T_{i+1} fertig. Des Weiteren kennen wir nun das Ende des Pfades, beginnend in der Wurzel mit der Markierung $a_{k+2} a_{k+3} \cdots a_{i+1}$, so dass die Konstruktion von T_{i+2} hier gestartet werden kann.

2.1.2 Die Markierungen aller ausgehenden Kanten von $s(v)$ beginnen mit einem Symbol $\neq a_{i+1}$.

$\rightsquigarrow$

$s(v)$ erhält ein neues Blatt b mit Nummer $k+2$ als weiteren Sohn. Die Kante $(s(v), b)$ erhält die Markierung $[i+1, \ldots]$.

Wir folgen dem Zeiger auf $s(s(v))$. Dieser

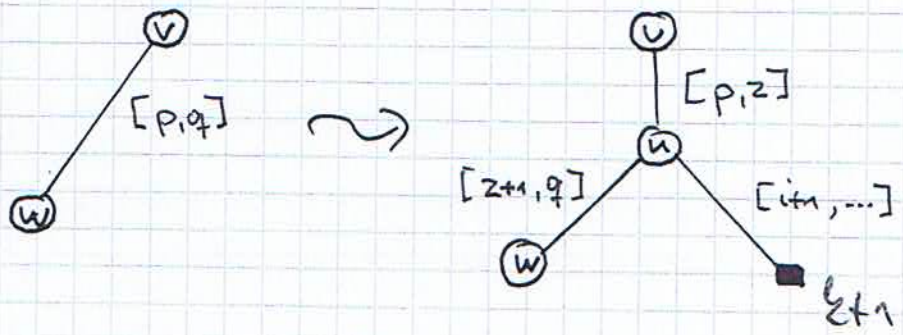
existiert, falls s(v) nicht die Wurzel des aktuellen Suffixbaumes ist, was in der Definition von s(v) $\alpha = \varepsilon$ implizieren würde.

Falls der Zeiger auf s(s(v)) nicht existiert, dann ist lediglich noch Erweiterung i+1 durchzuführen. Oben haben wir gezeigt, wie diese in konstanter Zeit durchführbar ist.

2.2 P endet auf einer Kante $e = (v, w)$.

Dann ist v ein innerer Knoten. Oben haben wir uns überlegt, dass s(v) existiert.

Situation nebst Modifikation:



Sei

$$a_{z+1} \alpha \beta = a_{z+1} a_{z+2} \cdots a_i ,$$

wobei $a_{z+1}\alpha$ die Markierung des Pfades $\bar{p}$ von der Wurzel zum Knoten v und β die Markierung der Kante (v, u) sind.

Es gilt: $|\beta| = z - p + 1$

Gemäß unserer Annahme enthält v einen Zeiger auf s(v).

- Wir folgen nun dem Zeiger auf s(v) und dann von s(v) aus dem Pfad mit Markierung β, bis das Ende dieses Pfades gefunden ist.

Ziel: Analyse dieser Pfadverfolgung.

Sei

$$\beta = a_e a_{e+1} \dots a_i .$$

Immer wenn ein innerer Knoten eingefügt wird, erhält dieser ein neues Blatt als Sohn. Dieses Blatt verbleibt während der gesamten Konstruktion als Blatt im Suffixbaum und ändert auch niemals seine Nummer.

Somit folgt aus der Tatsache, dass P auf einer Kante endet und der Konstruktion, dass s(v) genau eine ausgehende Kante

$$e_1 := (s(v), w_1)$$

besitzt, deren Markierung mit a_e beginnt.

Sei $[p_1, q_1]$ die Markierung der Kante e_1.

Im Folgenden bezeichnet L stets die Länge des Suffixes von β, oder noch zu folgenden Teilpfades. D.h., zu Be-

genau gilt: $L = z - p + 1$.

Jenachdem, ob der noch zu betrachtende Pfad über die aktuelle Kante hinausgeht oder nicht unterscheiden wir zwei Fälle.

1. Fall: $L \leq q_1 - p_1 + 1$

Dann endet der Pfad $\overline{\overline{P}}$ mit Markierung β

$$\begin{cases} \text{im Knoten } w_1 & \text{falls } L = q_1 - p_1 + 1 \\ \text{auf der Kante } e_1 & \text{falls } L < q_1 - p_1 + 1 \end{cases}$$

2. Fall: $L > q_1 - p_1 + 1$

Dann entspricht der zur Kante $e_1 = (s(v), w_1)$ korrespondierende Teilpfad dem echten Präfix der Länge $q_1 - p_1 + 1$ von β

Wir modifizieren L durch

$$L := L - (q_1 - p_1 + 1)$$

und folgen von w_1 aus $\overline{\overline{P}}$ weiter.

Wir verfahren nun bezüglich des Pfades mit Markierung $a_{z+2} a_{z+3} \dots a_i$ wie oben bezüglich des Pfades mit Markierung $a_{z+1} a_{z+2} \dots a_i$ beschrieben.

Zwei Dinge haben wir noch herauszuarbeiten. Zum einen müssen wir uns noch überlegen, wie wir die Suffixzeiger der inneren Knoten setzen.

(Für Blätter benötigen wir keine Suffixzeiger.) Zum anderen ist noch abzuschätzen, welchen Aufwand wir insgesamt für das Zurückverfolgen der Teilpfade haben.

Zunächst überlegen wir uns, wie wir die Suffixzeiger der inneren Knoten effizient setzen können.

Idee:
Falls wir uns davon überzeugen könnten, dass nach Hinzufügen eines neuen inneren Knotens u in einem Erweiterungsschritt einer Phase der Knoten $s(u)$ im nächsten Erweiterungsschritt derselben Phase sowieso besucht wird, dann könnte nach Hinzunahme eines neuen inneren Knotens u der Suffixzeiger auf $s(u)$ im nachfolgenden Erweiterungsschritt mit lediglich konstantem Mehraufwand gesetzt werden.

Dies leistet gerade folgendes Lemma:

Lemma 1.12
Falls bei der Konstruktion von T_{i+1} bei der Erweiterung j ein neuer innerer Knoten u mit $m(u) = a_j a_{j+1} \dots a_i$ dem aktuellen Suffixbaum als Sohn des Knotens v hinzugefügt wird, dann endet bei der Erweiterung $j+1$ die Zurückverfolgung des Pfades von $s(v)$ aus beginnend in einem inneren Knoten w mit $m(w) = a_{j+1} \dots a_i$,

falls solcher Knoten existiert, oder es wird ein Knoten w mit $u(w) = a_{j+1} a_{j+2} \dots a_i$ dem aktuellen Suffixbaum hinzugefügt.

Beweis:

Übung ■

Es verbleibt also noch die Analyse des Gesamtaufwands für das Zurückverfolgen der Teilpfade. Falls hierbei ein Gesamtaufwand von $O(n)$ herauskommt, dann folgt aus obigen Überlegungen, dass der implizite Suffixbaum T_n in $O(n)$ Zeit konstruierbar ist.

Erinnerung: (Definition Tiefe eines Knotens u):

Sei u ein Knoten eines Baumes T. Dann ist die Tiefe von u bzgl. T $Tiefe(u,T)$ definiert durch

$$Tiefe(u,T) := \begin{cases} 0 & \text{falls } u = Wurzel(T) \\ 1 + Tiefe(u,T_i) & \text{sonst, wobei } T_i \text{ derjenige direkte Unterbaum von T ist, der u enthält.} \end{cases}$$

$Tiefe(u,T)$ ist somit die Anzahl der Knoten auf dem Pfad von Wurzel(T) nach u, wobei der Knoten u selbst nicht mitgezählt wird.

Idee:

Wir interpretieren den Algorithmus als einen Agenten, der in der Wurzel des anfänglichen Suffixbaums startet und wie der Algorithmus die Pfade zurückverfolgt, indem er sich von Knoten zu Knoten bewegt. Unser Ziel ist es, den Gesamtaufwand für die Wanderung des Agenten durch den Suffixbaum mit Hilfe der Tiefe des Knotens, in dem sich der Agent am Schluss befindet, abzuschätzen.

Beobachtung:

1) Die Tiefe des Startknotens ist null.
2) Das Passieren einer Baumkante erhöht die Tiefe des aktuellen Knotens für den Agenten um eins.
3) Nur das Durchschreiten eines Suffixzeigers kann die Tiefe des aktuellen Knotens für den Agenten verringern.
4) Nach Terminierung des Algorithmus ist die aktuelle Knotentiefe für den Agenten $\leq n$.

Somit gilt für die Gesamtlänge GL durchschrittenen Pfades

$$GL \leq n + 2 \cdot GR,$$

wobei GR die gesamte Verringerung der aktuellen Tiefe bezeichnet.

Somit müssen wir nur noch GR abschätzen. Dies ist mit Hilfe der folgenden Lemmas einfach.

Lemma 1.13

Sei $(v, s(v))$ ein Suffixzeiger, den der Agent durchschreitet. Dann gilt zum Zeitpunkt des Durchschreitens für den aktuellen Suffixbaum T

$$\text{Tiefe}(v,T) \leq \text{Tiefe}(s(v),T) + 1.$$

Beweis:

Seien

P der Pfad von Wurzel(T) nach v in T

und

P' der Pfad von Wurzel(T) nach $s(v)$ in T.

Sei w ein beliebiger Vorgängerknoten von v mit

$$m(w) = a\beta \quad , a \in \Sigma, \beta \in \Sigma^+$$

Lemma 1.12 $\Rightarrow$

$s(w)$ existiert und w besitzt Suffixzeiger $(w, s(w))$

Seien w und w' zwei verschiedene Vorgängerknoten von v mit $|m(w)| \geq 2$ und $|m(w')| \geq 2$.

$\Rightarrow$

$s(w) \neq s(w')$ und sowohl $s(w)$ als auch $s(w')$ liegen auf P'.

Lediglich der direkte Nachfolger von Wurzel(T) auf P besitzt eventuell keinen Suffixzeiger.

Insgesamt folgt aus obigen Betrachtungen

$$|P| \leq |P'| + 1$$

$$\Rightarrow$$

$$\text{Tiefe}(v,T) \leq \text{Tiefe}(s(v),T) + 1.$$

∎

Da maximal n-mal ein Suffixzeiger durchlaufen wird, folgt direkt aus Lemma 1.13

$$GR \leq n.$$

Insgesamt haben wir folgenden Satz bewiesen:

Satz 1.3

Die Länge des Pfades, den der Agent während der Konstruktion des impliziten Suffixbaumes durchschreitet, ist $\leq 3 \cdot n$.

Da die sonstige Arbeit pro Phase konstant ist und n Phasen durchlaufen werden, wird der implizite Suffixbaum in $O(n)$ Zeit konstruiert.

Offen ist noch die Konstruktion eines Suffixbaumes aus dem impliziten Suffixbaum. Hierzu konkatenieren wir den Textstring x und den

String, bestehend aus dem Sonderzeichen $. Der Effekt ist, dass kein Suffix von x$ Präfix eines anderen Suffix ist. Also ist der implizite Suffixbaum für x$ auch der Suffixbaum für x$.

Übung:
Konstruieren Sie in Linearzeit aus einem Suffixbaum für x$ einen Suffixbaum für x.

Insgesamt haben wir folgenden Satz bewiesen:

Satz 1.4
Seien Σ ein endliches Alphabet und $x \in \Sigma^+$ ein String der Länge n über Σ. Ein Suffixbaum für x kann in $O(n)$ Zeit konstruiert werden.

1.4 Anwendungen von Suffixbäumen

Suffixbäume finden unmittelbar ihre Anwendung bei der Implementierung eines Indexes für einen Text. Dies überrascht nicht, da jeder Teilstring eines Textes Präfix eines Suffixes des Textes ist.

Sei $x := a_1 a_2 a_3 \dots a_n$ ein Text. Der kürzeste Präfix von $a_i a_{i+1} \dots a_n$, $1 \le i \le n$, so dass dieser nirgendwo sonst im Text x vorkommt, heißt Identifizierer der Position i im Text x. Falls der Text x mit einem Sonderzeichen endet, dann ist für jede Position i der Identifizierer definiert. Der Positionsbaum eines Textes

$x = a_1 a_2 \dots a_n \in \Sigma^n$ ist ein kompakter Trie bezüglich des Alphabets Σ, der $\leq n$ markierte Blätter enthält. Die Blätter sind mit paarweise verschiedenen Zahlen aus $\{1, 2, \dots, n\}$ numeriert. Der Pfad von der Wurzel zum Blatt mit Nummer i korrespondiert zum Identifizierer der Position i in Text x.

Der Positionsbaum für x kann leicht aus dem Suffixbaum für x konstruiert werden.

Übung:
Entwickeln Sie einen Algorithmus, der in Linearzeit aus dem Suffixbaum für x den Positionsbaum für x konstruiert.

Folgende vier Operationen soll der Index eines Textes unterstützen:

1) Für $y \in \Sigma^*$ finde den längsten Präfix, der in Text x enthalten ist.
2) Finde das erste Vorkommen eines Strings $y \in \Sigma^*$ in Text x.
3) Bestimme die Anzahl des Vorkommen von $y \in \Sigma^*$ in Text x.
4) Berechne eine Liste aller Vorkommen von $y \in \Sigma^*$ in Text x.

Übung:
Entwickeln Sie Algorithmen für die vier Operationen, die der Index eines Textes unterstützen soll. Zeigen Sie insbesondere, dass der Suffixbaum für einen Text x derart in Linearzeit vorbereitet werden kann, dass die Anzahl der Vorkommen eines Strings y im Text x in Zeit $O(|y|)$ ermittelt werden kann.

1970 hat Don Knuth vermutet, dass es für folgendes Problem keinen Linearzeitalgorithmus gibt.

Längste gemeinsame Teilstringproblem

gegeben: Strings x_1 und x_2 über einem endlichen Alphabet Σ.
gesucht: ein längster gemeinsamer Teilstring von x_1 und x_2.

Die Verwendung von Suffixbäumen ermöglicht einen einfachen Linearzeitalgorithmus für dieses Problem.

Beobachtung:
Ein längster gemeinsamer Teilstring von x_1 und x_2 ist Präfix sowohl eines Suffixes von x_1 als auch eines Suffixes von x_2.

$\rightsquigarrow$

Idee:
Konstruktion eines gemeinsamen Suffixbaumes für die Strings x_1 und x_2.

Ziel:
Konstruktion eines gemeinsamen Suffixbaumes für eine Menge $\{x_1, x_2, \ldots, x_t\}$ von Strings.

Seien $\$_1, \$_2, \ldots, \$_t$ paarweise verschiedene Symbole, die nicht in einem der Strings $x_1, x_2, \ldots, x_t$ vorkommen.

- Berechne den Suffixbaum für den String

$$x_1 \$_1 x_2 \$_2 \cdots \$_{t-1} x_t \$_t.$$

Eigenschaften:

- Wahl der Sonderzeichen $\$_1, \$_2, \ldots, \$_t$ und Konstruktionsalgorithmus

$\Rightarrow$

Der konstruierte Baum hat die gewünschte Struktur. Da die Blattmarkierungen nicht explizit erweitert werden, interpretieren wir, dass diese mit dem korrespondierenden Sonderzeichen $\$_i$ enden.

□

Zur Lösung des längsten gemeinsame Teilstring-problem konstruieren wir für die beide Eingabestrings x_1 und x_2 den gemeinsamen Suffixbaum und merken dabei uns für jeden inneren Knoten v die Anzahl der Symbole auf dem Pfad von der Wurzel zu v.

Danach traversieren wir bottom-up den Suffix-baum und entscheiden dabei für jeden inneren Knoten, ob in seinem Teilbaum bezüglich beiden Strings x_1 und x_2 ein Blatt existiert. Ein solcher mit maximaler Anzahl von Symbolen auf dem Pfad von der Wurzel zu ihm definiert einen längsten gemeinsamen Teilstring von x_1 und x_2.

Übung:

a) Arbeiten Sie den Algorithmus zur Lösung des längsten gemeinsamen Teilstringproblems aus.

b) Verallgemeinern Sie den Algorithmus, so daß ein längster gemeinsamer Teilstring von k Strings $x_1, x_2, \ldots, x_k$ in Zeit $O(k \cdot n)$ bestimmt wird, wobei $n := |x_1| + |x_2| + \ldots + |x_k|$.

Bemerkung:

Eine Vielzahl von weiteren Anwendungen von Suffixbäumen findet man in

Dan Gusfield: Algorithms on Strings, Trees, and Sequences, Kapitel 7 und 9.