Algorithmen und Datenstrukturen (Th. Ottmann und P. Widmayer)

Folien: Suffix-Bäume

Autor: Sven Schuierer

Institut für Informatik
Georges-Köhler-Allee
Albert-Ludwigs-Universität Freiburg

1 Suffix-Bäume

Suchindex für einen Text σ für Suche nach verschiedenen Mustern α

Eigenschaften:

1. Teilwortsuche in Zeit $O(|\alpha|)$.

2. Anfragen an σ selbst, z.B.:

   Längstes Teilwort von σ, das an mind. 2 Stellen auftritt

3. Präfix-Suche: Alle Stellen in σ mit Präfix α

4. Bereichs-Suche: Alle Stellen in σ im Intervall $[\alpha, \beta]$ mit $\alpha \leq_{lex} \beta$, z.B.

   $\texttt{abrakadabra}, \texttt{acacia} \in [\texttt{abc}, \texttt{acc}]$,
   $\texttt{abacus} \notin [\texttt{abc}, \texttt{acc}]$.

2 Tries

Alphabet Σ, Menge S von Schlüsseln

Schlüssel $\widehat{=}$ Zeichenkette aus Σ^*

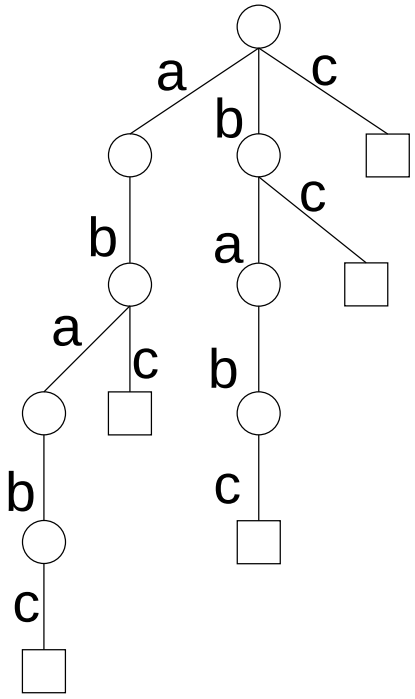
Kante eines Tries T: Beschriftung mit Zeichen aus Σ

benachbarte Kanten: verschiedene Zeichen

Blatt repräsentiert Schlüssel:
Beschriftungen der Kanten des Weges von der Wurzel zu Blatt

Engl.: retrieval Schlüssel nicht in Knoten abgespeichert.
Jeder Weg von der Wurzel zu einem Blatt entspricht einem Schlüssel

3 Beispiel

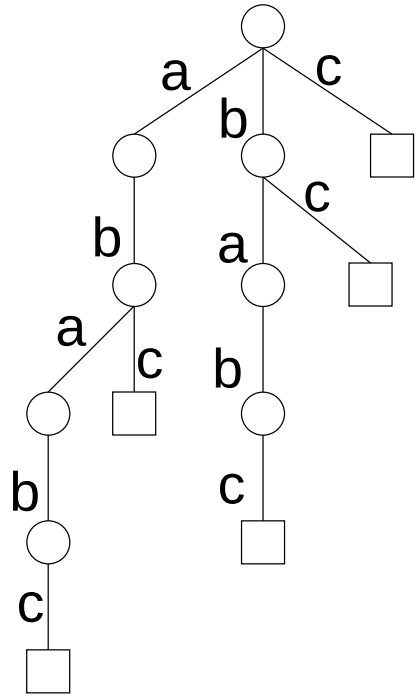


4 Suffix-Tries

Trie für alle Suffixe eines Wortes

Beispiel: $\sigma =$ ababc

Suffixe
ababc
babc
abc
bc
c



Innerer Knoten eines Suffix-Tries $=$ Teilwort von σ

$a^n b^n$: $n^2 + 2n + 1$ versch. Teilwörter $=$ innere Knoten

Bemerkungen

Suffix-Trie T erfüllt bereits alle gefordeten Eigenschaften.

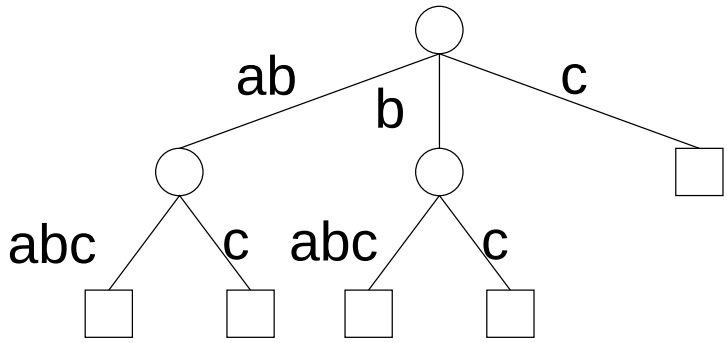
1. Zeichenkettensuche nach α: Folge dem Weg mit Kantenbeschriftung α in T in Zeit $O(|\alpha|)$.
   Blätter des Teilbaumes $\widehat{=}$ Vorkommen von α
2. Längstes, doppelt auftretendes Wort: Innerer Knoten mit größter Tiefe, der mind. zwei Söhne hat.
3. Präfix-Suche: alle Vorkommen von Zeichenketten mit Präfix α finden sich in dem Teilbaum unterhalb des inneren Knotens von α in T.

$a^i b^j$, $0 \leq i, j \leq n$

$(n+1)^2$ viele versch. Teilwörter

5 Suffix-Bäume

Kontraktion von unären Knoten



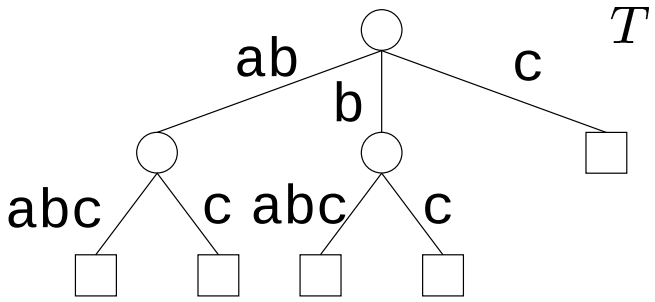
Suffix-Baum = kontraktierter Suffix-Trie

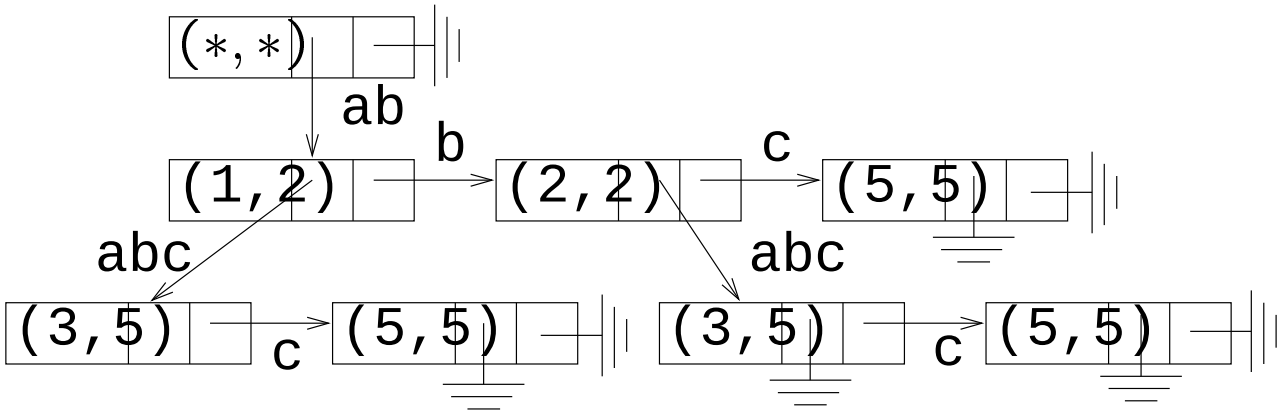
Repräsentation

Sohn/Bruder-Repräsentation

Teilworte: Zahlenpaare (i, j)

Beispiel: ababc





Knoten $v = (v.u, v.o, v.sn, v.br)$

Anwendungen

Verwendung von Suffix-Baum T:

1. Zeichenkettensuche nach α: Folge dem Weg mit Kantenbeschriftung α in T in Zeit $O(|\alpha|)$.
   Blätter des Teilbaumes $\widehat{=}$ Vorkommen von α
2. Längstes, doppelt auftretendes Wort: Ort des Wortes mit größter gewichteteter Tiefe, der innerer Knoten ist.
3. Präfix-Suche: alle Vorkommen von Zeichenketten mit Präfix α finden sich in dem Teilbaum unterhalb des „Ortes“ von α in T.

Eigenschaften

$(S1)$ Kein Suffix Präfix eines anderen Suffixes
$\Rightarrow$ Letztes Zeichen von σ: \$ $\notin \Sigma$

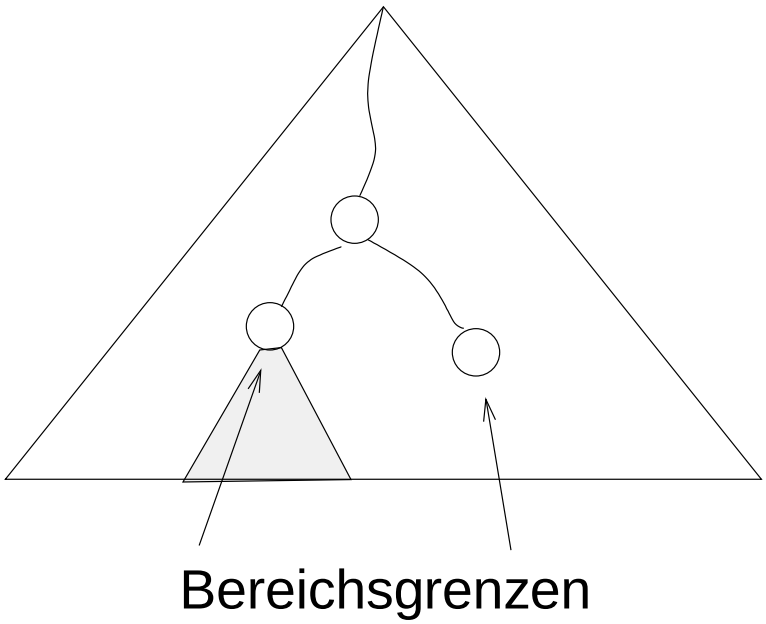
$(T1)$ Kante $\widehat{=}$ nichtleeres Teilwort von σ.
$(T2)$ Benachbarte Kanten: zugeordnete Teilworte beginnen

$(T3)$ Innerer Knoten ($\neq$ Wurzel): mind. zwei Söhne.
$(T4)$ Blatt: $\widehat{=}$ (nicht-leeres) Suffix von σ.

$n = |\sigma|$
$\overset{T4}{\Longrightarrow}$ Anzahl der Blätter: n
$\overset{T3}{\Longrightarrow}$ Anzahl der inneren Knoten: $\leq n - 1$

Anwendung

1. Bereichssuche:



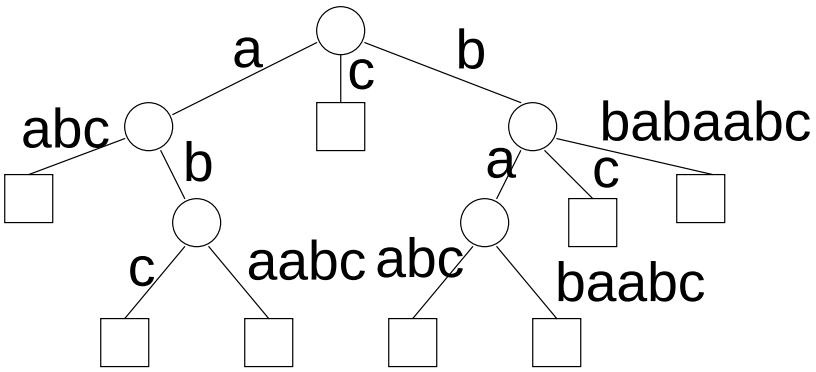
6 Konstruktion von Suffix-Bäumen

- Weg: Weg von der Wurzel zu einem Knoten von T
- Ort einer Zeichenkette α: Knoten am Ende des mit α beschrifteten part. Weges (falls er existiert). Präfix α.
- erweiterter Ort einer Zeichenkette α: Ort der kürzesten Zeichenkette mit Präfix α, deren Ort definiert ist.
- kontraktierter Ort einer Zeichenkette α: Ort des längsten Präfixes von α, dessen Ort definiert ist.

Konstruktion von Suffix-Bäumen

- suf_i: an Position i beginnendes Suffix von σ, also z.B. $suf_1 = \sigma$, $suf_n =$ \$.
- $head_i$: längstes Präfix von suf_i, das auch Präfix von suf_j für ein $j < i$ ist.

σ =bbabaabc α= baa



Betrachte suf_4 = baabc $\Rightarrow$ $head_4$ = ba

Suffix-Baum Konstruktion

Naives Verfahren:

Beginne mit dem leeren Baum T_0

Der Baum T_{i+1} entsteht aus T_i durch Einfügen des Suffixes suf_{i+1}.

Suffix-Baum

Input: Eine Zeichenkette σ

Output: Der Suffix-Baum T von σ

1 $n = |\sigma|$
2 T_0 = leerer Baum (Blatt)
3 for $i = 0$ to $n - 1$ do
4 füge suf_{i+1} in T_i ein

Bemerkungen

In T_i haben also alle Suffixe suf_j, $j < i$ bereits einen Ort. Daher gilt:

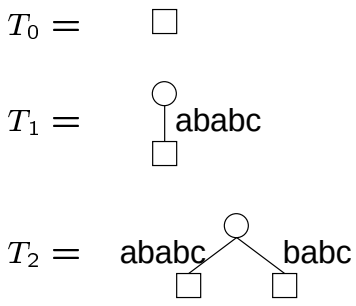
$head_i$: längstes Präfix von suf_i, dessen erweiterter Ort in T_{i-1} existiert.

Wir definieren weiter: $tail_i = suf_i - head_i$, d.h. also $suf_i = head_i tail_i$.

$(S1)$: $tail_i \neq \varepsilon$.

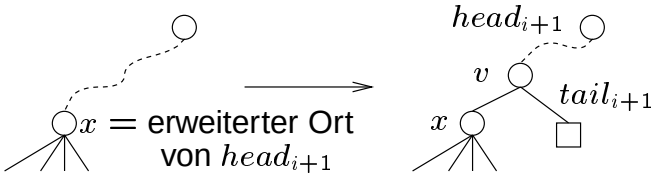
Beispiel: $\sigma =$ ababc

$$\begin{aligned} suf_3 &= \text{abc} \\ head_3 &= \text{ab} \\ tail_3 &= \text{c} \end{aligned}$$

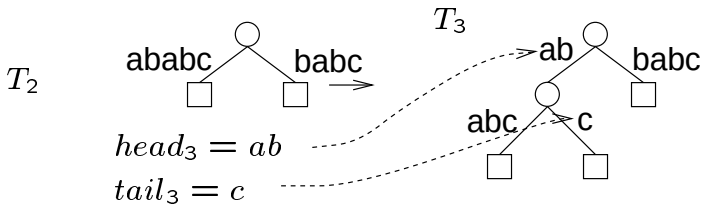


T_{i+1} kann aus T_i wie folgt konstruiert werden:

1. Man bestimme den erweiterten Ort von $head_{i+1}$ in T_i und teile die letzte zu diesem Ort führende Kante in zwei neue Kanten auf durch Einfügen eines neuen Knotens.
2. Man schaffe ein neues Blatt als Ort für suf_{i+1}.



Beispiel:



Einfügung

Suffix-Einfügen

Input: Der Baum T_i und der Suffix suf_{i+1}

Output: Der Baum T_{i+1}

1 $v :=$ Wurzel von T_i

2 $j := i$ /* Zeiger auf Zeichen von suf_{i+1} */

3 repeat

4 finde Sohn w von v mit $\sigma_{w.u} = \sigma_{j+1}$

5 if w = null then exit loop

6 $k := w.u - 1$;

7 while $k < w.o$ and $\sigma_{k+1} = \sigma_{j+1}$ do

8 $k := k + 1$; $j := j + 1$

end while;

9 if $k = w.o$ then $v := w$

10 until $k < w.o$;

11 /* v ist kontraktierter Ort von $head_{i+1}$*/

12 füge den Ort von $head_{i+1}$ und $tail_{i+1}$ in T_i unter v ein

7 Der Algorithmus M von McCreight

Erweiterte Ort von $head_{i+1}$ in T_i gefunden: Erzeugen eines neuen Knotens und Aufspalten einer Kante $O(1)$ Zeit Ziel: Erweiterten Ort von $head_{i+1}$ in konstanter amortisierter Zeit in T_i bestimmen.

Zusatzinformation erforderlich

Lemma 1 Wenn $head_i = a\gamma$ für ein Symbol a und eine (evtl. leere) Zeichenkette γ ist, dann ist γ ein Präfix von $head_{i+1}$.

Beweis: Sei $head_i = a\gamma$, dann existiert ein $j < i$, so daß $a\gamma$ Präfix von suf_i und suf_j ist nach der Definition von $head_i$. Also ist γ ein Präfix sowohl von suf_{i+1} als auch von suf_{j+1}.

8 Suffix-Zeiger

Von jedem inneren Knoten, der der Ort eines Wortes $a\gamma$ ist, gibt es einen Zeiger auf den Ort des Wortes γ.

Idee: Beginne nicht bei der Wurzel, sondern bei dem Ort von $head_i - a_i$ = Ende des Suffix-Zeigers des Ortes v von $head_i$

Probleme: Woher Suffix-Zeiger von v?
Ort von $head_i - a_i$ existiert möglicherweise gar nicht

9 Invarianten des Algorithmus M

Nach Konstruktion von T_i gilt:

$(Inv1)$: Alle inneren Knoten von T_{i-1} haben einen korrekten Suffix-Zeiger in T_i.
$(Inv2)$: Bei der Konstruktion von T_i wird der kontraktierte Ort von $head_i$ in T_{i-1} besucht.

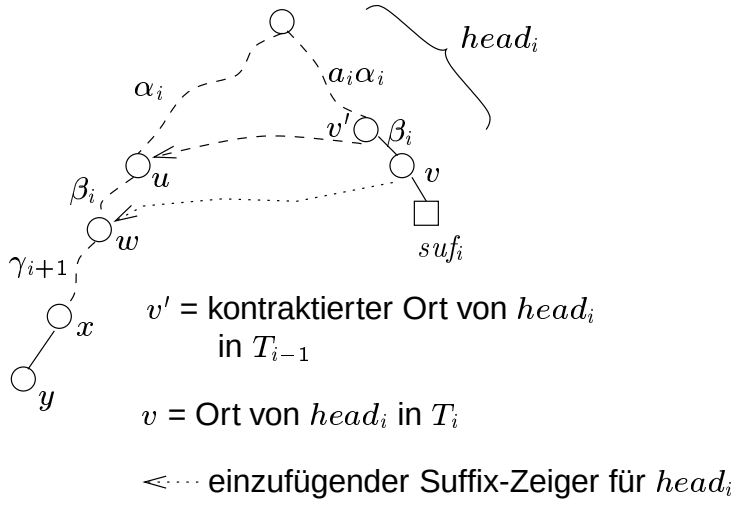
$i = 1$.

$i > 1$:

Definition: Ist $head_i \neq \varepsilon$, so bezeichnet α_i die Konkatenation der Kantenbeschriftungen des Weges zum kontraktierten Ort von $head_i$ ohne den ersten Buchstaben a_i.

Visualisierung

Ferner sei $\beta_i = head_i - a_i\alpha_i$, d.h. $head_i = a_i\alpha_i\beta_i$. Ist $head_i \neq \varepsilon$, haben wir in T_i:



Aufgrund von Lemma 1 ist

$$head_{i+1} = \alpha_i\beta_i\gamma_{i+1}.$$

$Inv2$: Konstruktion von T_{i+1} kann beim kontraktierten Ort v' von $head_i$ in T_{i-1} beginnen.

$Inv1$: Von v' gibt es bereits einen korrekten Suffix-Zeiger in T_i zu einem Knoten u nach $Inv1$.

Folge $v'.suffix\text{-}zeiger$ zu u im Gegensatz zum naiven Verfahren, das bei der Wurzel beginnt.

Algorithmus M (Schritt 1)

Schritt 1: Einfügen des Ortes von $head_{i+1}$

1 $v' =$ kontraktierter Ort von $head_i$ in T_{i-1}
$u = v'.suffix\text{-}zeiger$

2 if $\beta_i \neq \varepsilon$
then rescan β_i in T_i beginnend bei u

3 if Ort w von $\alpha_i\beta_i$ in T_i existiert
then Scan γ_{i+1} ausgehend von w
$(x, y) =$ letzte Kante
else $/* \; head_{i+1} = \alpha_i\beta_i$ (s.u.) $*/$
x = kontraktierter Ort von $\alpha_i\beta_i$
y = erweiterter Ort von $\alpha_i\beta_i$

4 Schaffe bei (x, y) einen inneren Knoten z für den Ort von $head_{i+1}$ und ein Blatt für den Ort von suf_{i+1}

Bemerkungen

Rescan bzw. Scan α ausgehend von v
folge einem Weg in T_i ausgehend von v, dessen Kantenbeschriftungen α ergeben.

rescan β_i in T_i, d.h. folge einem Weg in T_i ausgehend von u, dessen Kantenbeschriftungen β_i ergeben.

Scan γ_{i+1} ausgehend von w, d.h. folge einem Weg in T_i ausgehend von w, dessen Kantenbeschriftungen mit suf_{i+1} übereinstimmen, bis man aus dem Baum bei der Kante (x, y) herausfällt.

Algorithmus M (Schritt 2)

Schritt 2: Einfügen des Suffix-Zeigers für den Ort v von $head_i$

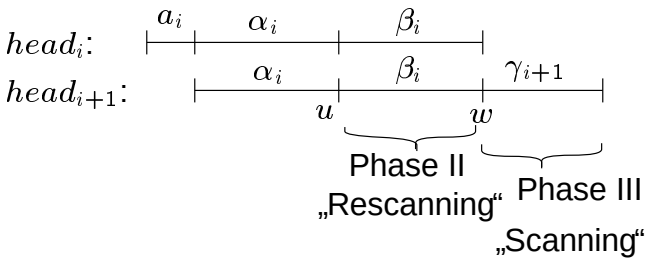
1 $u = v'.suffix\text{-}zeiger$

2 if $\beta_i \neq \varepsilon$
then rescan β_i in T_i bis zum Ort w von $\alpha_i\beta_i$
else $w = u$

3 $v =$ Ort von $head_i$
$v.suffix\text{-}zeiger = w$

Implementation: Verflechtung von Schritt 1 und 2

Lemma 2



Lemma 2 Falls der Ort von $\alpha_i\beta_i$ in T_i nicht existiert, dann ist $head_{i+1} = \alpha_i\beta_i$, d.h. $\gamma_{i+1} = \varepsilon$.

Beweis:

v = kontraktierter Ort von $\alpha_i\beta_i$
w = erweiterter Ort von $\alpha_i\beta_i$
γ = Beschriftung des Weges zu v
$\delta_1\delta_2$ = Beschriftung von (v, w),
mit $\delta_1, \delta_2 \neq \varepsilon$ und $\gamma\delta_1 = \alpha_i\beta_i$.
d.h.:
1. Suffixe mit Präfix $\alpha_i\beta_i$ sind in dem Teilbaum T_w von T mit Wurzel w und
2. alle Suffixe in T_w haben Präfix $\alpha_i\beta_i\delta_2$.

Also:
Ist $j < i + 1$ und suf_j hat Präfix $\alpha_i\beta_i \Rightarrow suf_j$ hat Präfix $\alpha_i\beta_i\delta_2$.

Wir zeigen: $suf_j = \alpha_i\beta_i a \cdots$ und $suf_{i+1} = \alpha_i\beta_i b \cdots$ mit $a \neq b$.

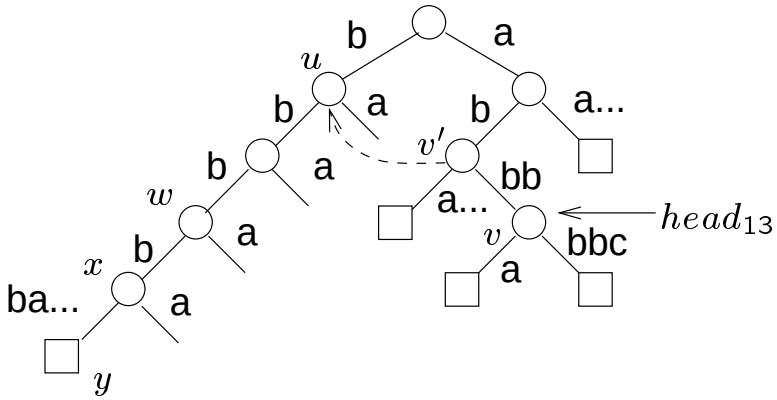
Es existiert ein $suf_{j'}$ ein Suffix mit Präfix $head_i = a_i\alpha_i\beta_i$.
$\Rightarrow$ $suf_{j'+1}$ hat Präfix $\alpha_i\beta_i\delta_2$
$\Rightarrow$ $suf_{j'}$ hat Präfix $a_i\alpha_i\beta_i\delta_2$

Da $head_i = a_i\alpha_i\beta_i$, ist der erste Buchstabe a von δ_2 von dem ersten Buchstaben b, der $a_i\alpha_i\beta_i$ in suf_i folgt, verschieden.

$\Rightarrow$ Präfix von suf_{i+1}: $\alpha_i\beta_i b$
$\Rightarrow$ Präfix von $suf_{j'}$: $\alpha_i\beta_i a$.
$\Rightarrow$ Längstes gemeinsames Präfix: $\alpha_i\beta_i$.

Suffix-Einfügen Beispiel

Beispiel: $\sigma = b^5abab^3a^2b^5c$. Konstruktion von T_{14} aus T_{13} durch einfügen von $suf_{14} =$ bbbbbc in T_{13}.



10 Analyse

Nun analysieren wir die Anzahl der Schritte, die der Algorithmus M benötigt. In jeden Schritt wird ein Suffix von σ gescannt und regescannt.

Analyse Rescannen

$\alpha_i\beta_i$ ist Präfix von $head_{i+1}$: an einer Kante muß nur festgestellt werden, um wieviele Zeichen wir in β_i vorrücken müssen.

$\Rightarrow$ konstante Zeit pro besuchte Kante
#(Schritte beim Rescannen) = $O(\#$(besuchte Kanten)).

Wir definieren

$$res_{i+1} = suf_{i+1} - \alpha_i = \beta_i\gamma_{i+1}tail_{i+1}.$$

(res steht für Zeichen, die ab Schritt $i+1$ überhaupt noch regescannt werden können). Bei jeder Kante e, um die während des Rescannens von β_i vorgerückt wird, wird α_{i+1} um die Beschriftung δ_e der Kante e länger, d.h. δ_e ist in res_{i+1}, aber nicht in res_{i+2}.

(β_{i+1} ist ja der Teil von $head_{i+1}$ zwischen dem kontraktierten und dem erweiterten Ort von $head_{i+1}$)

Da $|\delta_e| \geq 1$, folgt:

$$|res_{i+2}| \leq |res_{i+1}| - k_{i+1}$$

mit $k_i = \#$(regesannte Kanten in Schritt i). Also

$$\sum_{i=1}^{n} k_i \leq \sum_{i=1}^{n} (|res_i| - |res_{i+1}|) = res_1 - res_{n+1} \leq n$$

$\Rightarrow \#$(regesannte Kanten) $\leq n$.

Analyse Scannen

#(gescannte Zeichen in Schritt $i+1$) $= |\gamma_{i+1}|$, wobei

$$\begin{aligned} |\gamma_{i+1}| &= |head_{i+1}| - |\alpha_i \beta_i| \\ &= |head_{i+1}| - (|head_i| - 1)\,. \end{aligned}$$

wegen $head_i = a_i \alpha_i \beta_i$.

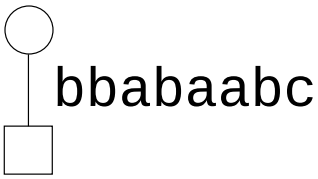
Also ist #(gescannte Zeichen) $=$

$$\begin{aligned} \sum_{i=1}^{n} |\gamma_i| &= \sum_{i=1}^{n} (|head_i| - |head_{i-1}| + 1) \\ &= n + |head_n| - |head_0| = n. \end{aligned}$$

Theorem 3 Algorithmus M liefert in Zeit $O(|\sigma|)$ einen Suffix-Baum für σ mit $|\sigma|$ Blättern und höchstens $|\sigma| - 1$ inneren Knoten.

Konstruktion Beispiel

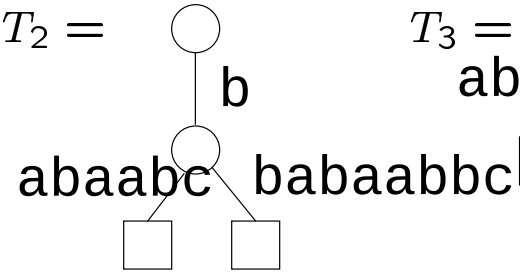
$T_0 =$ 

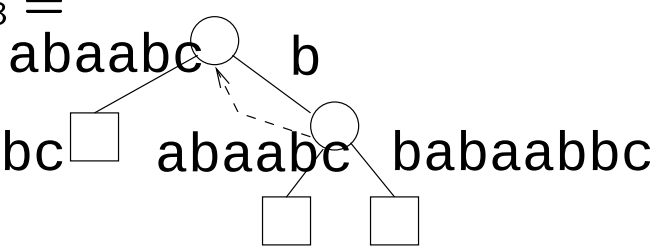
$T_1 =$ 

$suf_1 =$ bbabaabc

$suf_2 =$ babaabc

$head_2 =$ b

$T_2 =$ 

$T_3 =$ 

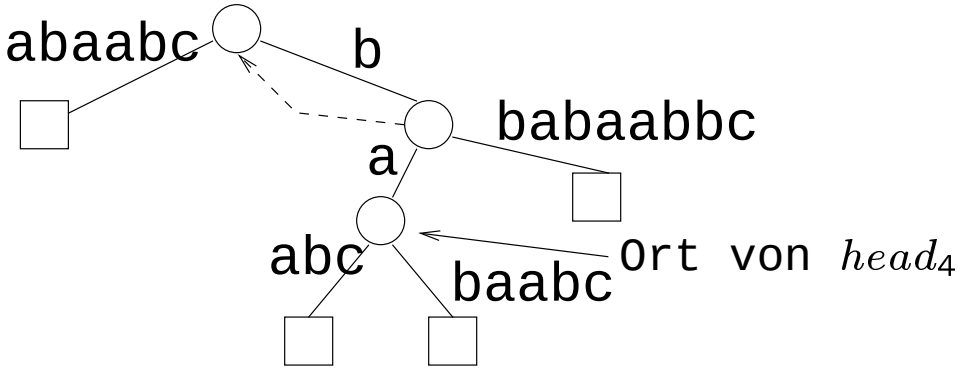
$suf_3 =$ abaabc

$head_3 = \epsilon$

$suf_4 =$ baabc

$head_4 =$ ba

$a_4 =$ b $\quad \alpha_4 = \epsilon \quad \beta_4 =$ a

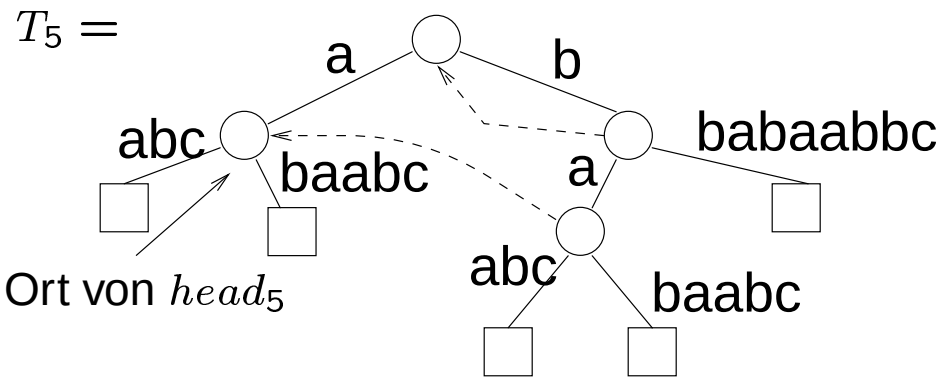
$T_4 =$ 

$suf_5 =$ aabc

$head_5 =$ a

$a_5 =$ a $\quad \alpha_5 = \epsilon \quad \beta_5 = \epsilon$

Beispiel (cont.)



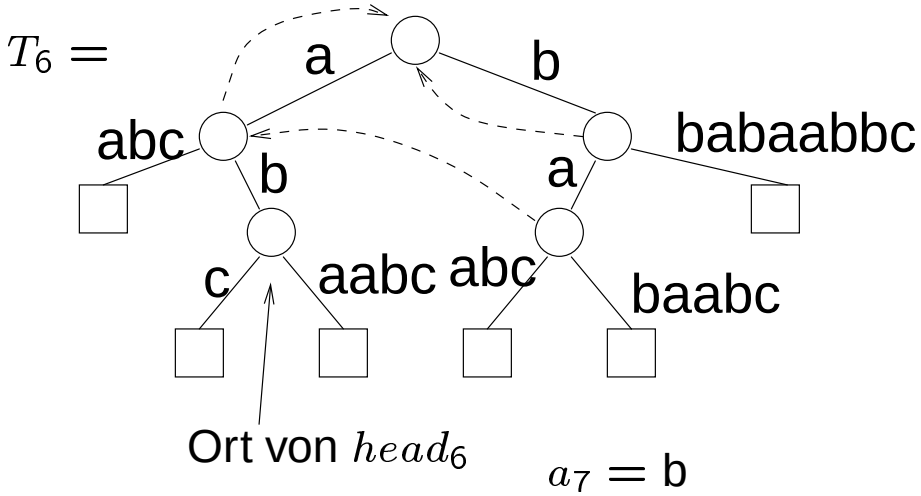
$suf_6 =$ abc

$head_6 =$ ab

$a_6 =$ a

$\alpha_6 = \epsilon$

$\beta_6 =$ b



$a_7 =$ b

$suf_7 =$ bc

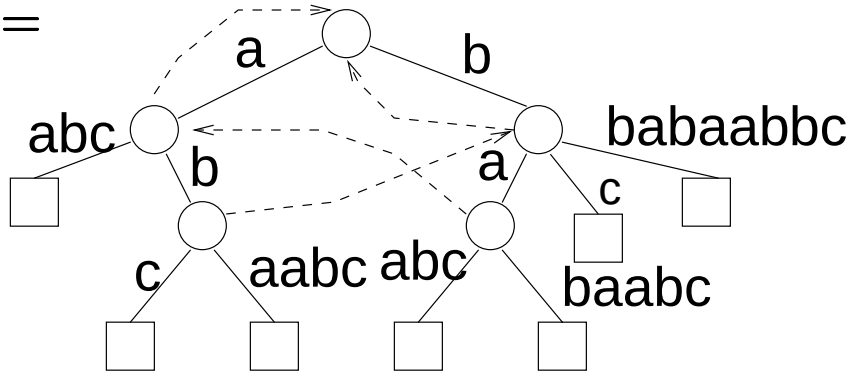
$\alpha_7 = \epsilon$

$head_7 =$ b

$\beta_7 =$ b

Beispiel (cont.)

$T_7 =$



$suf_8 =$ c

$T_8 =$