



# Suche in Texten: Suffix-Bäume

Prof. Dr. S. Albers  
Prof. Dr. Th. Ottmann

# Suche in Texten

---

Verschiedene Szenarios:

## **Dynamische Texte**

- Texteditoren
- Symbolmanipulatoren

## **Statische Texte**

- Literaturdatenbanken
- Bibliothekssysteme
- Gen-Datenbanken
- WWW-Verzeichnisse

# Eigenschaft von Suffix-Bäume

---

## Suchindex

zu einem Text  $\sigma$  für Suche nach verschiedenen Mustern  $\alpha$

## Eigenschaften:

1. **Teilwortsuche** in Zeit  $O(|\alpha|)$ .
2. **Anfragen an  $\sigma$  selbst**, z.B.:  
Längstes Teilwort von  $\sigma$ , das an mind. 2 Stellen auftritt.
3. **Präfix-Suche**: Alle Stellen in  $\sigma$  mit Präfix  $\alpha$ .

# Eigenschaft von Suffix-Bäume

---

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

abrakadabra, acacia  $\in$  [abc, acc],

abacus  $\notin$  [abc, acc] .

5. **Lineare Komplexität:**

Speicherplatzbedarf und Konstruktionszeit  $\in O(|\sigma|)$

**Trie:** Baum zur Repräsentation von Schlüsseln.

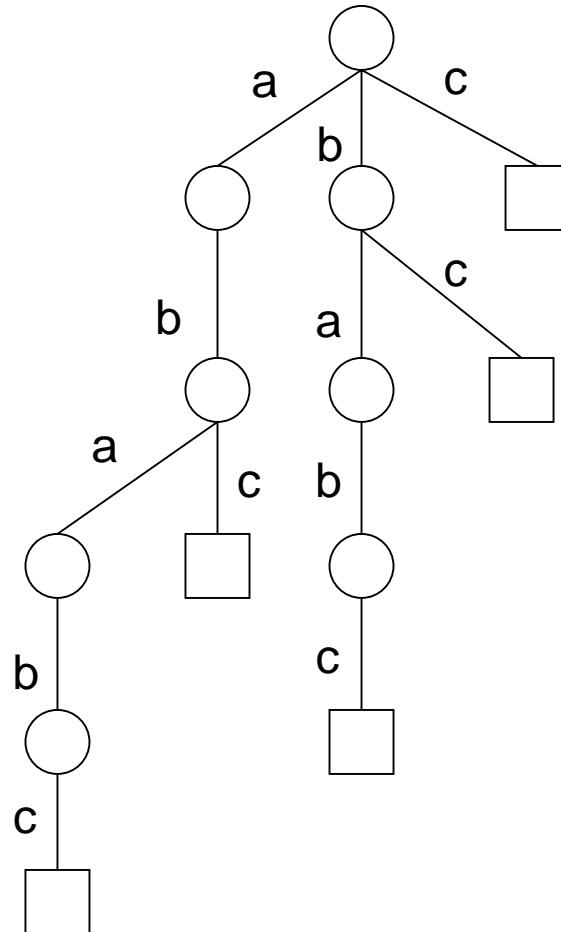
Alphabet  $\Sigma$ , Menge  $S$  von Schlüsseln,  $S \subset \Sigma^*$

**Schlüssel**  $\hat{=}$  Zeichenkette aus  $\Sigma^*$

**Kante** eines Tries  $T$ : Beschriftung mit einzelnen Zeichen aus  $\Sigma$

**benachbarte Kanten:** verschiedene Zeichen

## Beispiel:



**Blatt** repräsentiert Schlüssel:

Entspricht Beschriftung der Kanten des Weges  
von der Wurzel zu Blatt

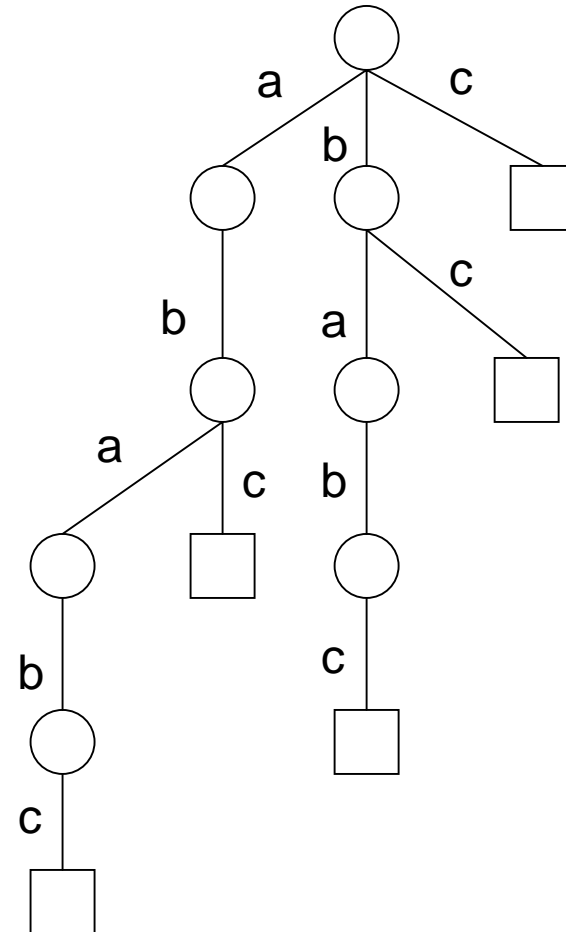
**! Schlüssel werden nicht in Knoten gespeichert !**

# Suffix-Tries

Trie für alle Suffixe eines Wortes

Beispiel:  $\sigma = ababc$

Suffixe:       $ababc = suf_1$   
                    $babc = suf_2$   
                    $abc = suf_3$   
                    $bc = suf_4$   
                    $c = suf_5$





# Suffix-Tries

---

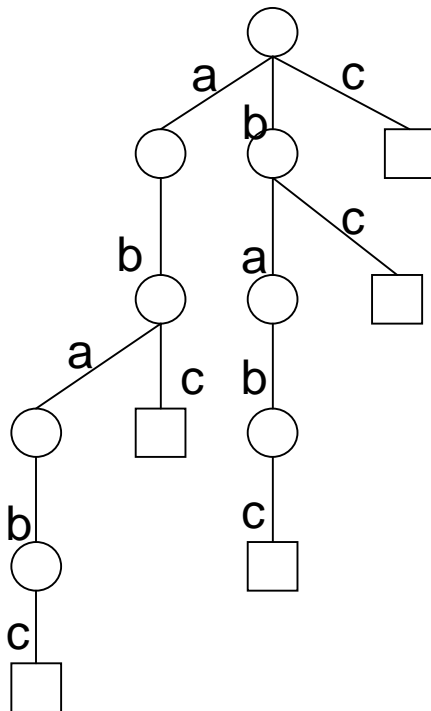
Innere Knoten eines Suffix-Tries = Teilwort von  $\sigma$ .

Jedes echte Teilwort von  $\sigma$  ist als innerer Knoten repräsentiert.

Sei  $\sigma = a^n b^n : \exists n^2 + 2n + 1$  verschied. Teilwörter = innere Knoten

$\Rightarrow$  Speicherplatzbedarf  $\in O(n^2)$ .

Ein Suffix-Trie  $T$  erfüllt bereits einige der geforderten Eigenschaften:



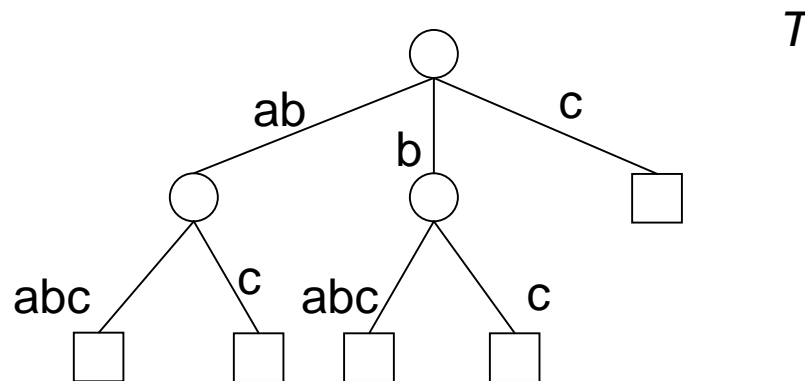
1. Zeichenkettensuche nach  $\alpha$ : Folge dem Weg mit Kantenbeschriftung  $\alpha$  in  $T$  in Zeit  $O(|\alpha|)$ . Blätter des Teilbaumes  $\hat{=}$  Vorkommen von  $\alpha$
2. Längstes, doppelt auftretendes Wort: Innerer Knoten mit größter Tiefe, der mind. zwei Söhne hat.
3. Präfix-Suche: alle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



## Sohn/Bruder-Repräsentation

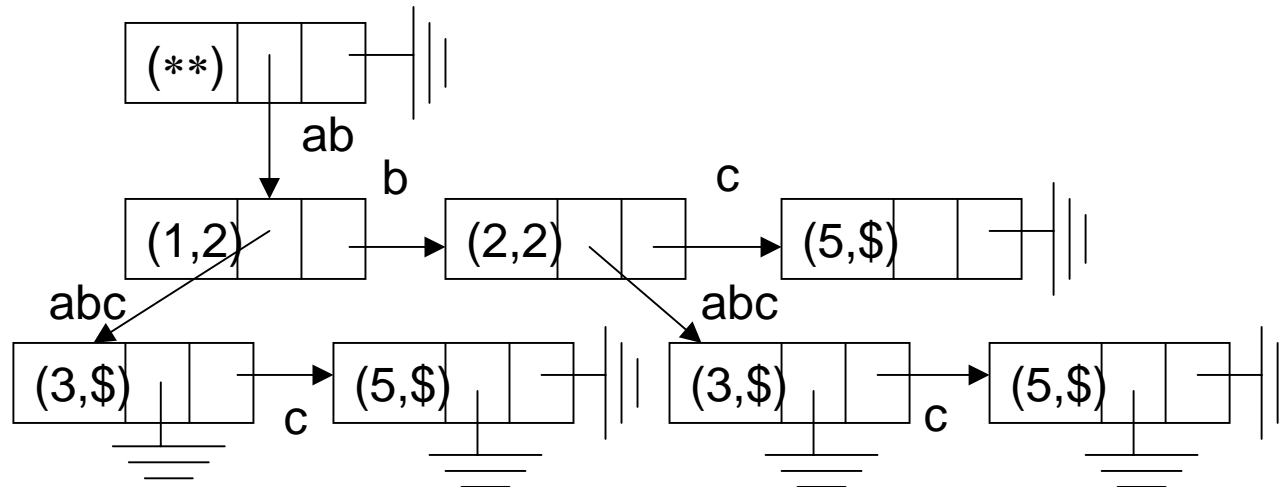
Teilwort: Zahlenpaar  $(i, j)$

Beispiel:  $\sigma = ababc$



# Interne Repräsentation von Suffix-Bäumen

Beispiel  $\sigma = ababc$



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

Weitere Zeiger (Suffix-Zeiger) kommen später hinzu

# Eigenschaften von Suffix-Bäumen

---

(S1) Kein Suffix ist Präfix eines anderen Suffixes;  
gilt, falls (letztes Zeichen von  $\sigma$ ) = \$  $\notin \Sigma$

## Suche:

(T1) Kante  $\hat{=}$  nichtleeres Teilwort von  $\sigma$ .

(T2) Benachbarte Kanten: zugeordnete Teilworte beginnen mit verschiedenen Zeichen.

# Eigenschaften von Suffix-Bäumen

---

## Größe

(T3) Innerer Knoten ( $\neq$  Wurzel): mind. zwei Söhne.

(T4) Blatt  $\triangleq$  (nicht-leeres) Suffix von  $\sigma$ .

Sei  $n = |\sigma| \neq 1$

(T4)

$\Rightarrow$  Anzahl der Blätter :  $n$

(T3)

$\Rightarrow$  Anzahl der inneren Knoten  $\leq n - 1$

$\Rightarrow$  Speicherplatz  $\in O(n)$

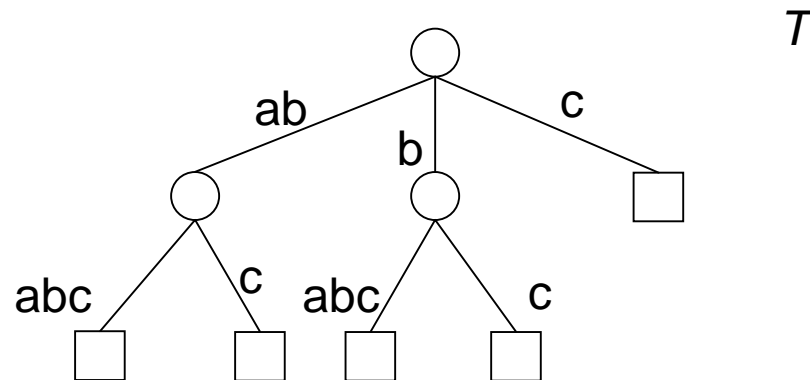
# Konstruktion von Suffix-Bäumen

## Definition:

**partieller Weg:** Weg von der Wurzel zu einem Knoten von  $T$

**Weg:** Ein partieller Weg, der bei einem Blatt endet.

**Ort** einer Zeichenkette  $\alpha$ : Knoten am Ende des mit  $\alpha$  beschrifteten partiellen Weges (falls er existiert).



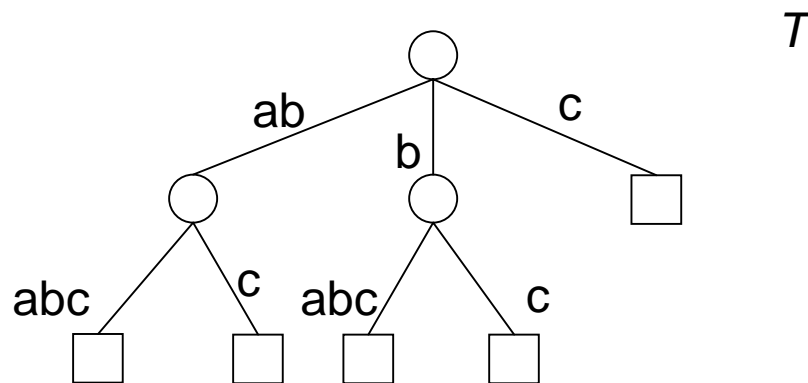


# Konstruktion von Suffix-Bäumen

**Erweiterung** einer Zeichenkette  $\alpha$ : Zeichenkette mit Präfix  $\alpha$

**erweiterter Ort** einer Zeichenkette  $\alpha$ : Ort der kürzesten Erweiterung von  $\alpha$ , deren Ort definiert ist.

**kontraktierter Ort** einer Zeichenkette  $\alpha$ : Ort des längsten Präfixes von  $\alpha$ , dessen Ort definiert ist.



# Konstruktion von Suffix-Bäumen

## Definitionen:

$suf_i$ : an Position  $i$  beginnendes Suffix von  $\sigma$ , 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.

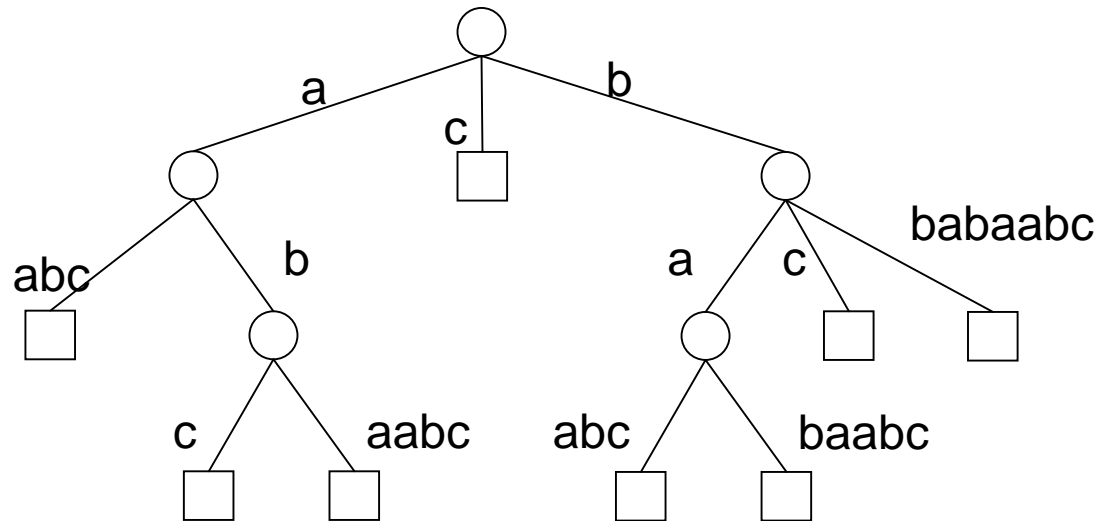
Beispiel:  $\sigma = \text{bbabaabc}$   $\alpha = \text{baa}$  (hat keinen Ort)

$suf_4 = \text{baabc}$

$head_4 = \text{ba}$

# Konstruktion von Suffix-Bäumen

$\sigma = \text{bbabaabc}$



# Naive Suffix-Baum-Konstruktion

---

Beginne mit dem leeren Baum  $T_0$

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

## **Algorithmus** Suffix-Baum

**Input:** Eine Zeichenkette  $\sigma$

**Output:** Der Suffix-Baum  $T$  von  $\sigma$

```
1  $n := |\sigma|$ ;  $T_0 := \emptyset$ ;  
2 for  $i := 0$  to  $n - 1$  do  
3   füge  $suf_{i+1}$  in  $T_i$  ein, dies sei  $T_{i+1}$  ;  
4 end for
```

# Naive Suffix-Baum-Konstruktion

---

In  $T_i$  haben alle Suffixe  $suf_j$ ,  $j < i$  bereits einen Ort.

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

## Definition:

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

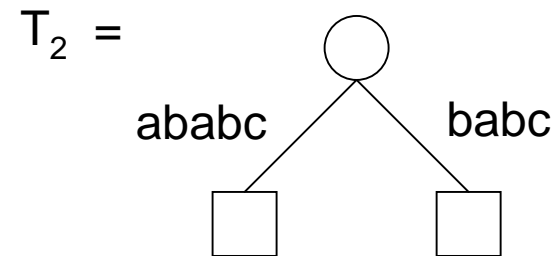
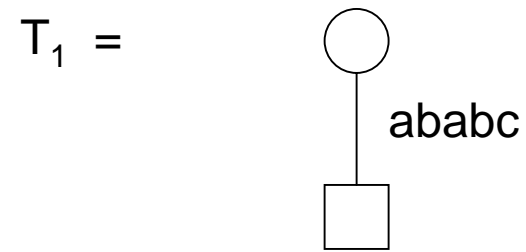
(S1)

⇒  $tail_i \neq \varepsilon$ .

# Naive Suffix-Baum-Konstruktion

Beispiel:  $\sigma = ababc$

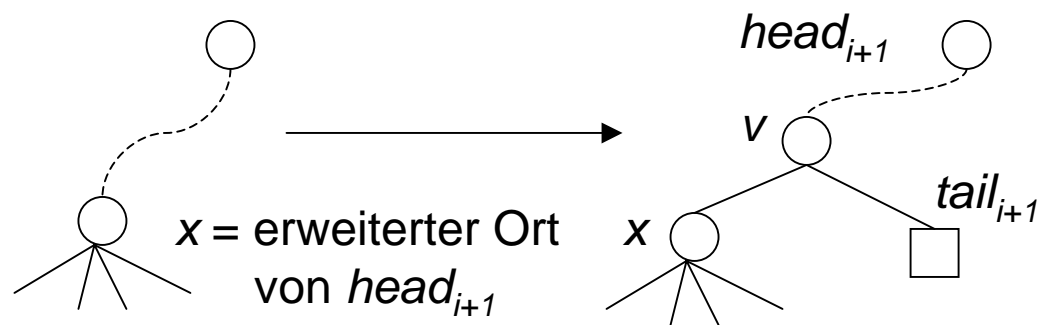
$suf_3 = abc$   
 $head_3 = ab$   
 $tail_3 = c$



# Naive Suffix-Baum-Konstruktion

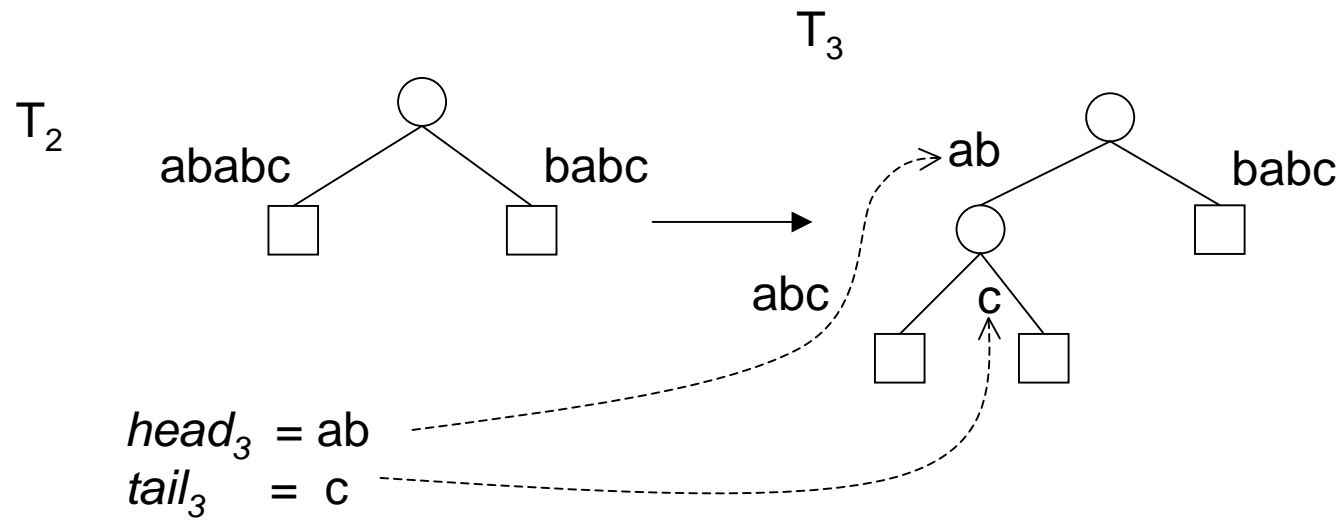
$T_{i+1}$  kann aus  $T_i$  wie folgt konstruiert werden:

1. Man bestimmt den erweiterten Ort von  $head_{i+1}$  in  $T_i$  und teilt 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  $suf_{i+1}$



# Naive Suffix-Baum-Konstruktion

Beispiel:  $\sigma = \text{ababc}$





# Naive Suffix-Baum-Konstruktion

## Algorithmus 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   $j := i$ 
3  repeat
4      finde Sohn  $w$  von  $v$  mit  $\sigma_{w.u} = \sigma_{j+1}$ 
5       $k := w.u - 1$ ;
6      while  $k < w.o$  and  $\sigma_{k+1} = \sigma_{j+1}$  do
7           $k := k + 1$ ;  $j := j + 1$ 
      end while
```

# Naive Suffix-Baum-Konstruktion

---

```
8      if  $k = w.o$  then  $v := w$   
9      until  $k < w.o$  or  $w = \text{nil}$   
10     /*  $v$  ist kontraktierter Ort von  $head_{i+1}$  */  
11     füge den Ort von  $head_{i+1}$  und  $tail_{i+1}$  in  $T_i$  unter  $v$  ein
```

Laufzeit für Suffix-Einfügen:  $O(\quad)$

Gesamtlaufzeit für naive Suffix-Baum-Konstruktion:  $O(\quad)$

# Der Algorithmus M

---

(Mc Creight, 1976)

Falls erweiterter Ort von  $head_{i+1}$  in  $T_i$  gefunden: Erzeugen eines neuen Knotens und Aufspalten einer Kante  $\in O(1)$  Zeit.+

Idee: Erweiterter Ort von  $head_{i+1}$  wird in **konstanter amortisierter** Zeit in  $T_i$  bestimmt. (Zusatzinformation erforderlich!)

# Analyse des Algorithmus M

---

## Theorem 1

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

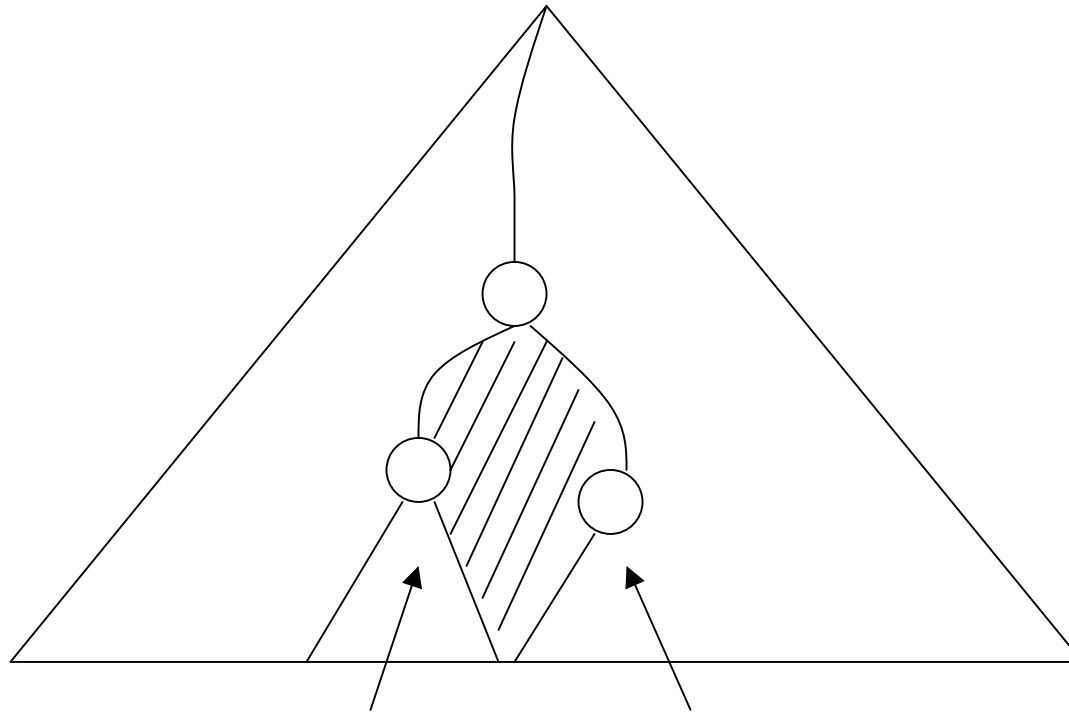
# Suffix-Baum Anwendung

---

Verwendung von Suffix-Baum  $T$ :

- 1 Suche nach Zeichenkette  $\alpha$ : Folge dem Weg mit Kantenbeschriftung  $\alpha$  in  $T$  in Zeit  $O(|\alpha|)$ .  
Blätter des Teilbaumes  $\triangleq$  Vorkommen von  $\alpha$
- 2 Suche längstes, doppelt auftretendes Wort:  
Finde Ort eines Wortes mit größter gewichteter Tiefe, der innerer Knoten ist.
- 3 Suche nach Präfix: Alle Vorkommen von Zeichenketten mit Präfix  $\alpha$  finden sich in dem Teilbaum unterhalb des „Ortes“ von  $\alpha$  in  $T$ .

## 4 Bereichssuche nach $[\alpha, \beta]$ :




Bereichsgrenzen

# Suffix-Baum Beispiel

$T_0 =$  

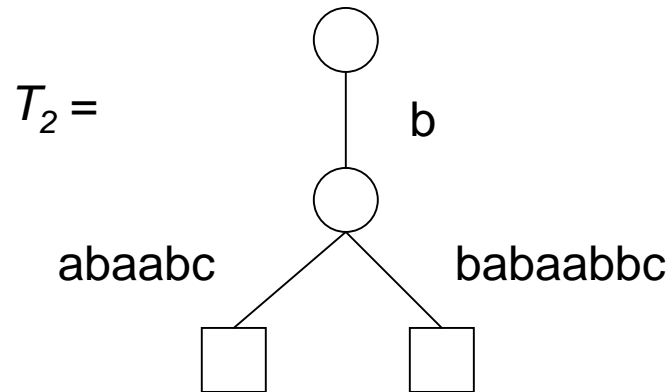
$suf_1 =$  bbabaabc

$T_1 =$   bbabaabc

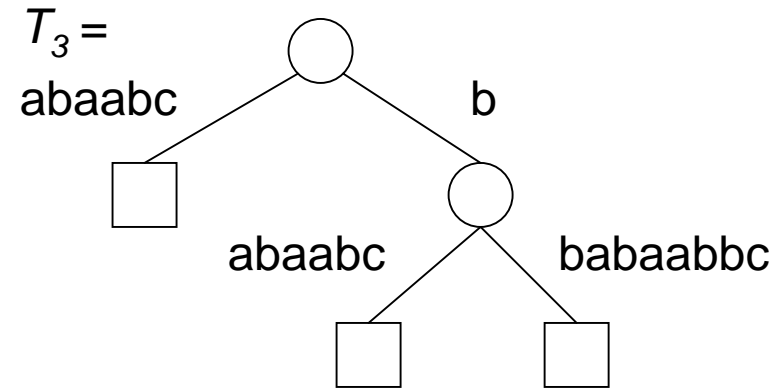
$suf_2 =$  babaabc

$head_2 =$  b

# Suffix-Baum Beispiel



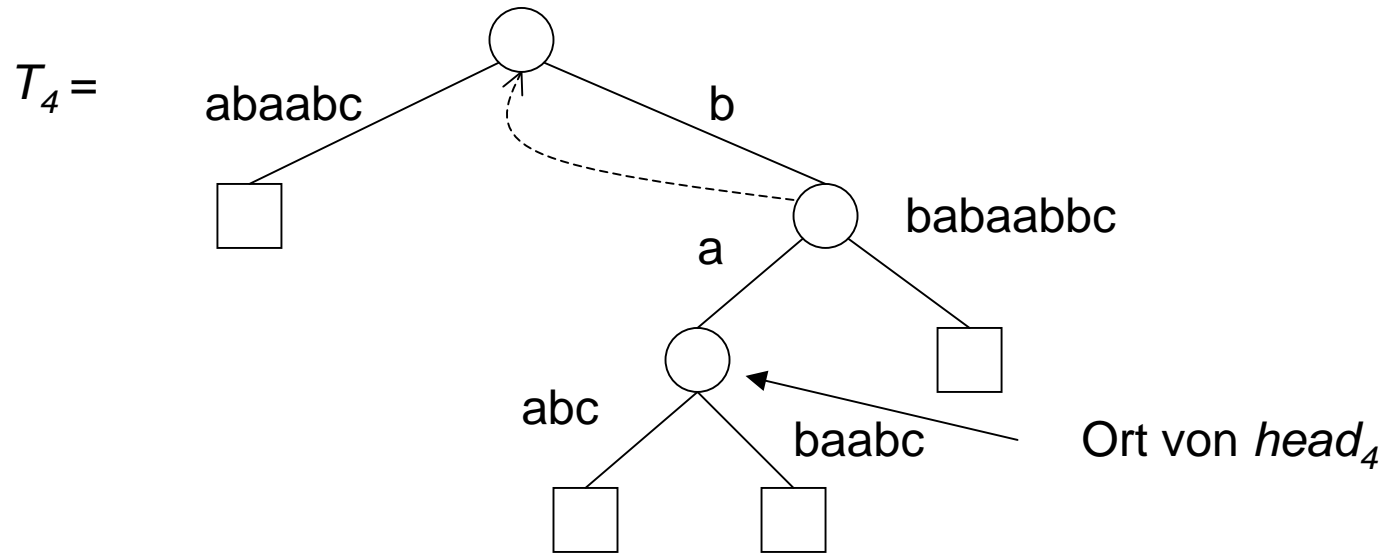
$suf_3 = abaabc$   
 $head_3 = \varepsilon$



$suf_4 = baabc$   
 $head_4 = ba$

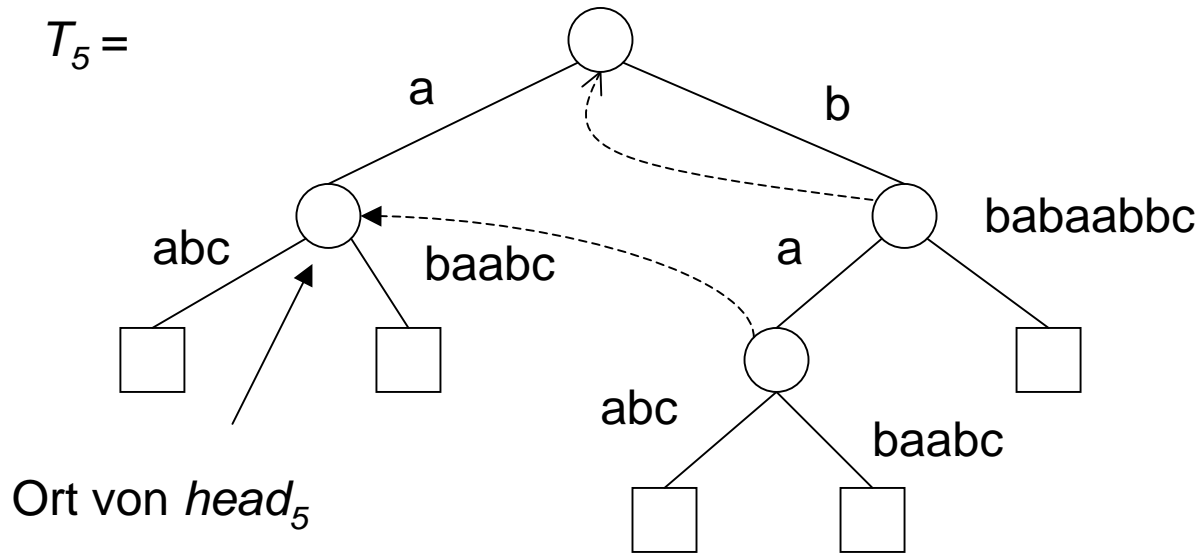


# Suffix-Baum Beispiel



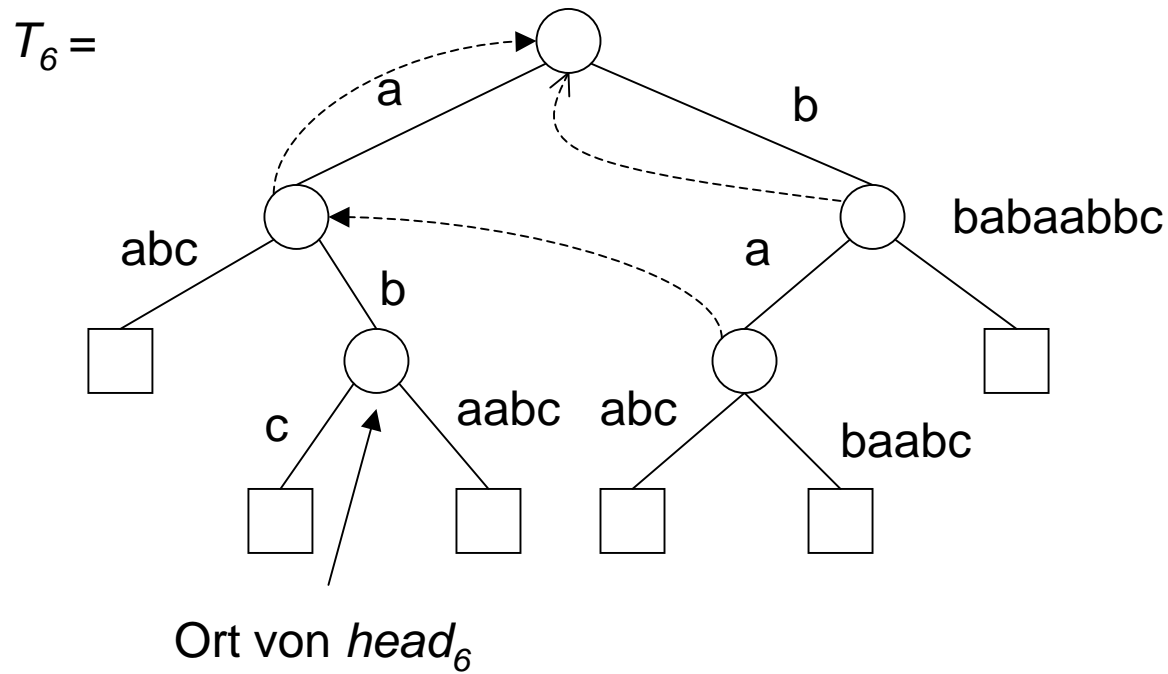
$suf_5 = aabc$   
 $head_5 = a$

# Suffix-Baum Beispiel



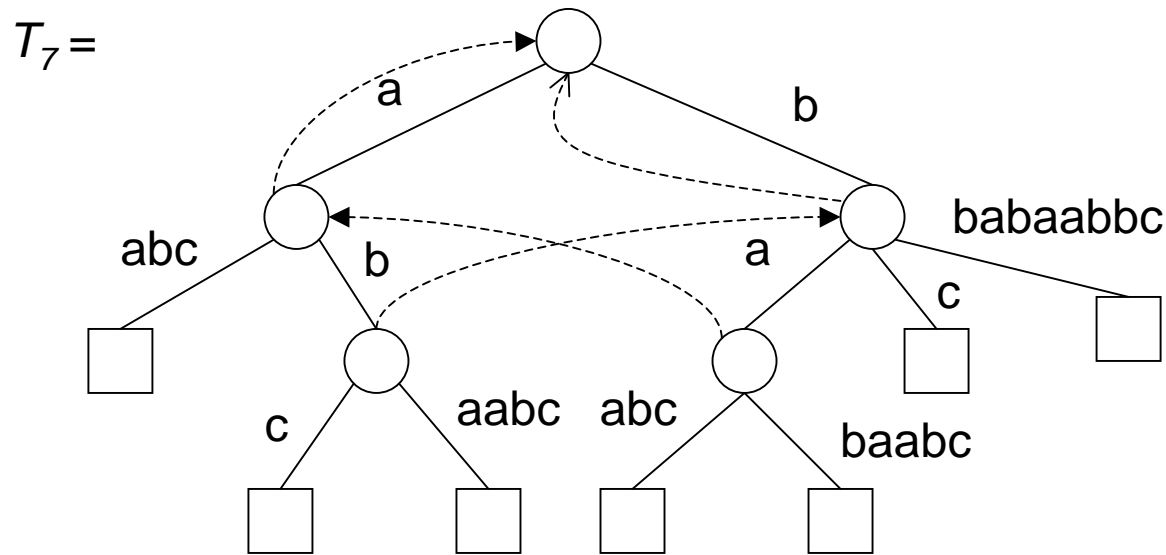
$suf_6 = abc$   
 $head_6 = ab$

# Suffix-Baum Beispiel



$suf_7 = bc$   
 $head_7 = b$

# Suffix-Baum Beispiel



$suf_8 = c$

# Suffix-Baum Beispiel

