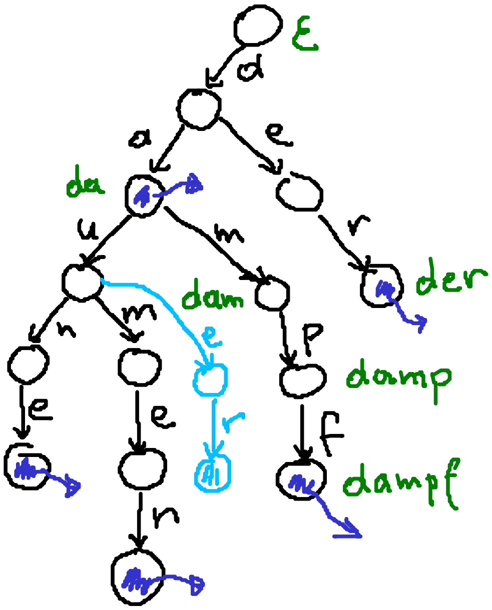
Digitale Suchbäume (tries) : Schlüssel sind Zeichen<u>ketten</u>.



- Kanten sind mit Zeichen beschriftet.
- Die Wurzel entspricht dem leeren Wort ε.
- Das Kind von x über die Kante a entspricht dem Wort xa

(Alle Präfixe jedes Wortes sind ebenfalls enthalten.)

- Die Schlüssel, die gültig sind, sind markiert. (und enthalten die entsprechenden Werte)

(auf jeden Fall alle Blätter.)

{der, da, dampf, daune, daumen}   dame?   dauer

Laufzeit: $O(|x|)$ ... linear in der Länge $|x|$ des Wortes x.
(für konstantes Σ)

Größe des Alphabets Σ ⇒

- Jeder Knoten enthält $|\Sigma|$ Verweise auf Kinder.
  viel Platz!
- Jeder Knoten enthält ein Wörterbuch mit Schlüsseln $\in \Sigma$

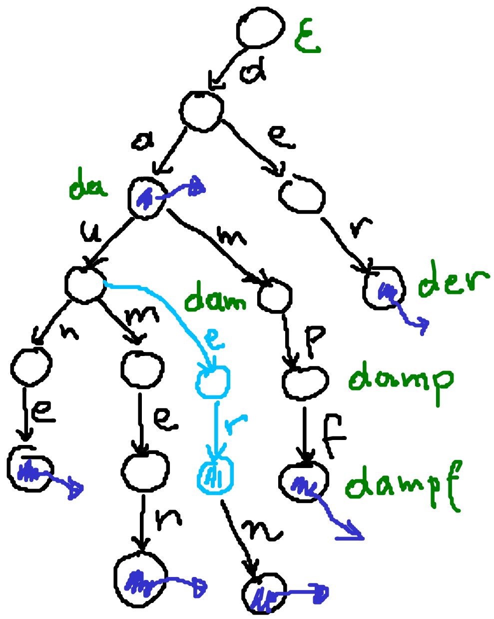
Aufwand pro Knoten $O(|\Sigma|)$, $O(\log|\Sigma|)$

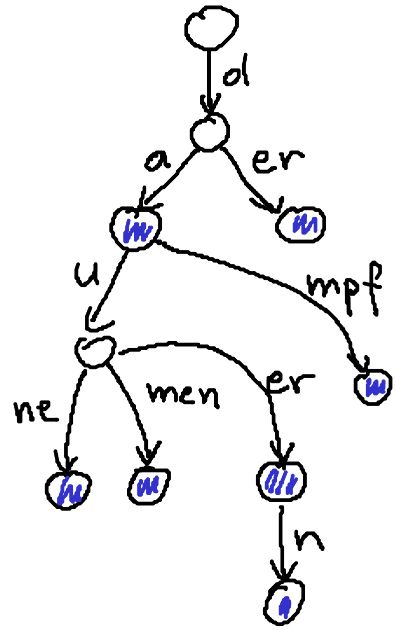
Anwendungen
- $\Sigma = \{C, G, T, A\}$ (DNA, Bioinformatik)
- Auto-Vervollständigung

Speicher: # Knoten ≤ Gesamtlänge aller Schlüsselwörter + 1

„trie": information re<u>trie</u>val   [trier (frz.) = sortieren]

Komprimierte Digitale Suchbäume





- Die Kanten sind mit Zeichen<u>folgen</u> beschriftet. (Länge ≥ 1)
- Jeder Knoten außer der Wurzel ist
  - entweder markiert
  - oder hat ≥ 2 Kinder.

SATZ.

> Ein komprimierter digitaler Suchbaum mit n Schlüsseln hat höchstens $2n$ Knoten.

Beweis:

$$\left.\begin{array}{ll} n_B & \text{Blätter} \\ n_I & \text{markierte innere Knoten} \end{array}\right\} n_B + n_I = n$$

$$\begin{array}{ll} s_I & \text{sonstige innere Knoten (außer Wurzel)} \\ +\ 1 & \text{Wurzel} \end{array} \quad \leq n$$

Summe: alle Knoten $\leq 2n$

$$\#\text{Kanten} \geq \cancel{n_I} + \cancel{2}\, s_I + 1 \qquad \text{(ausgehende Kanten)}$$

$$\#\text{Kanten} = n_B + \cancel{n_I} + \cancel{s_I} \qquad \text{(einmündende Kanten)}$$

$$s_I + 1 \leq n_B \leq n$$

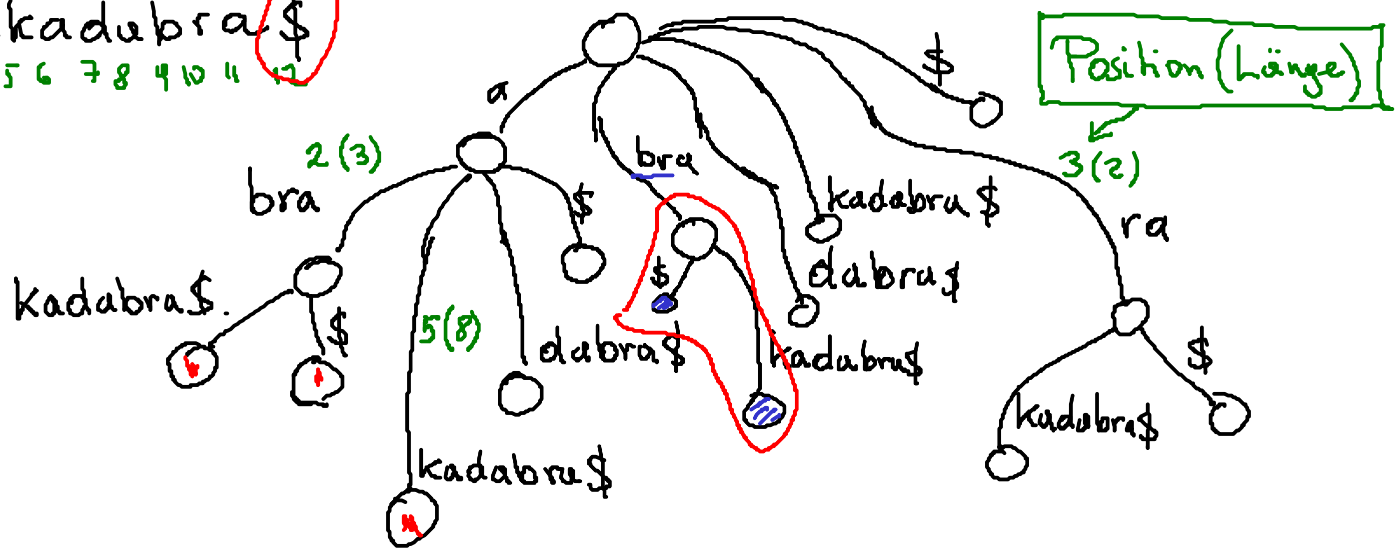
□

Suffixbäume

für eine Zeichenkette $a_1 a_2 a_3 \dots a_n$ ist der komprimierte digitale Suchbaum für alle nichtleeren Suffixe von $a_1 a_2 \underbrace{a_3 \dots a_n \$}$.

abrakadabra$

1 2 3 4 5 6 7 8 9 10 11 12



Der Suffixbaum benötigt nur $O(n)$ Speicher.

Bestimme alle k Stellen, an denen ein Teilwort w vorkommt.

= Suffixe, die mit w beginnen.

$O(k + |w|)$ Zeit.

Ein Suffixbaum kann in $O(n)$ Zeit aufgebaut werden.