Textsuche mit Suffix-Trees

Überblick:

- Motivation
- Definitionen (Suffix, Trie-Datenstruktur)
- Suche in Tries, Suffix-Tries
- Suffix-Trees
- Brute Force Algorithmus
- McCreight‘s Algorithmus

Motivation

- Pattern P, $|P| = m$, Text T, $|T| = n$
- (exakte) Suche nach verschiedenen Mustern P

  z.B. Suche nach Genen in einer Chromoson-DB
- Boyer-Moore: mit preproc. auf Muster $O(n+m)$
- Wunsch: pro Suche $O(m+k)$, k = Anz. der Hits
- Idee: prepoc. auf Text

$\rightarrow$ Suffix-Trees, mit McCreight‘s Alg. in $O(n)$

Definitionen

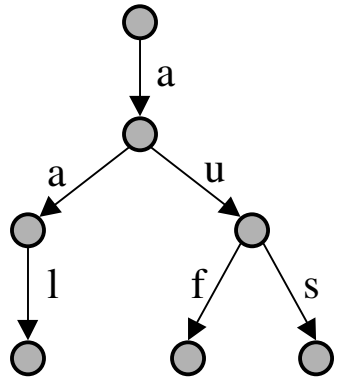
- Suffix ab Pos. i von T: $S_i = T_i\, T_{i+1} \ldots T_n$

  Bsp.: „brodt“ ist Suffix S_6 von „Scharbrodt“

- Ann.: T endet mit einem spez. Zeichen ‚$‘

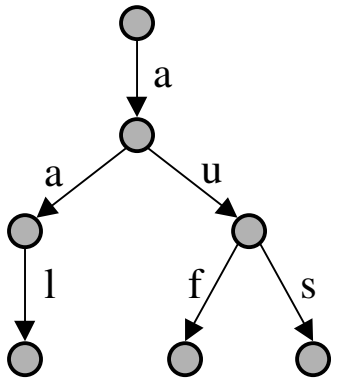
  => kein Suffix ist Anfang eines anderen Suffixes von T

- Trie: gewurzelter Baum mit Kantenbeschriftungen
  - Kinder eines Knoten mit unterschiedl. Zeichen markiert
  - Blatt entspricht Wort auf dem Pfad zum Blatt



  - „aal“ „auf“ „aus“; Achtung: „au“ + „auf“ geht nicht

Suche in Tries



- Suche nach *P*:

  P von links durchgehen, im Baum entspr. verzweigen

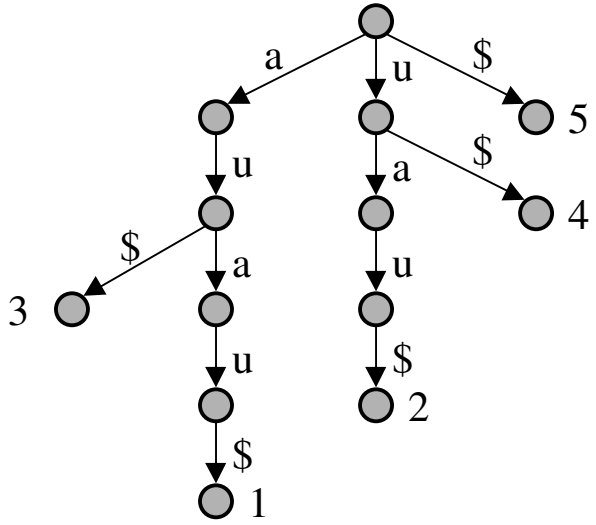
  – steckenbleiben => *P* nicht im Trie (z.B. „ab“)

  – *P* an innerem Knoten erschöpft
  => Präfix eines Wortes (z.B. „au“)

  – *P* an Blatt erschöpft => ganz im Trie (z.B. „aal“)

Suffix-Tries

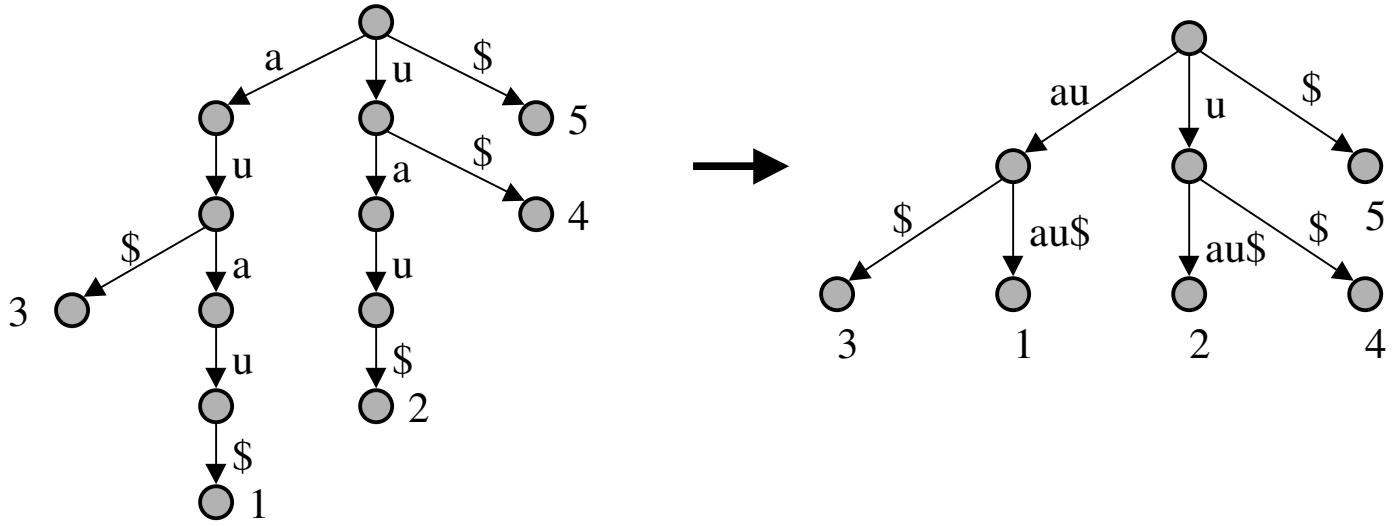
- Für die Suche nach Mustern:
  - alle Suffixe von *T* in einem Trie speichern
    (geht, da kein Suffix der Anfang eines anderen ist)
  - Suffix-Pos. an den Blättern merken
- Suche nach „au“ in „auau$“:



- Nachteil von Tries: $O(n^2)$ Speicher

Suffix-Trees

- Idee: Pfadkompression, Bsp.:



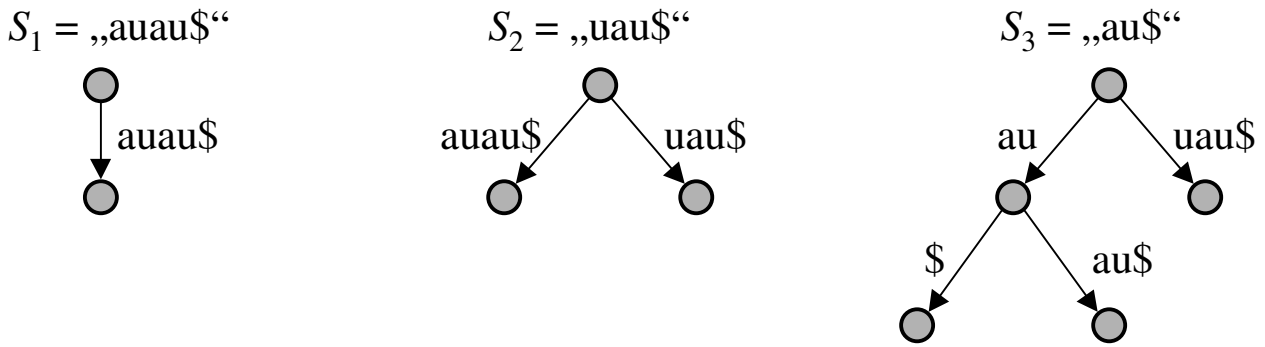
- Jeder innere Knoten hat ≥ 2 Kinder, n Blätter
  => max. $2n$ Knoten
- Beschriftung: nur Anfangs- und Endpos. merken

=> $O(n)$ Platz

- Suche bei k hits: ≤ $2k$ Knoten => $O(m + k)$

Brute Force Algorithmus

- mit Suffix S_1 beginnen
- iterativ nächsten Suffix hinzufügen:
  - Suche Stelle, ab der keine Übereinstimmung mehr
  - evtl. neuen innere Knoten
- Bsp: „auau$“

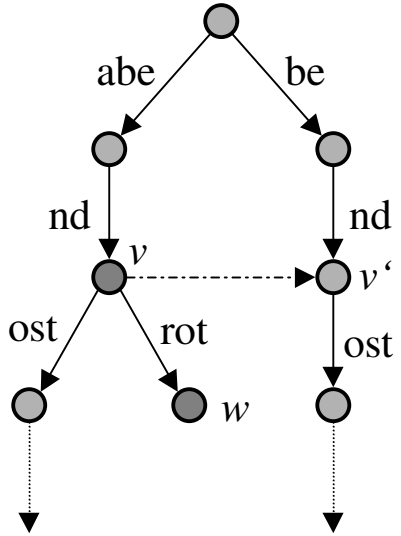


- LZ: für jedes Suffix $O(n)$ => insges. $O(n^2)$

McCreight‘s Algorithmus

- beim Einfügen von $S_{i\text{-}1}$ („abendrot“) wurde Blatt w erzeugt
  Vater von w ist v ($\neq$ Wurzel)
- X := Wort bis Knoten v („abend“)
- Beobachtungen: ($|X| = k$)
  - S_i beginnt mit $X_{2\ldots k}$ (S_i = „bendrot“)
  - $\exists\, j < i\text{-}1$ mit $S_j = X\ldots$ (S_j = „abendost…“)
    sonst wäre v woanders
  - S_{j+1} ($j+1 < i$) beginnt mit $X_{2\ldots k}$ (S_{j+1} = „bendost“)

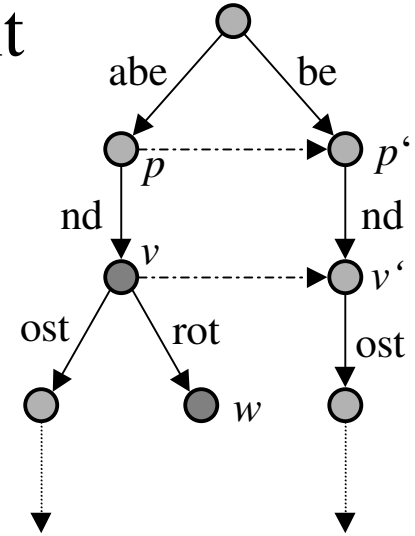
  => $X_{2\ldots k}$ ist bereits im Suffix Tree
- Überspringe $X_{2\ldots k}$ mit Hilfe eines Zeigers $v \rightarrow v‘$

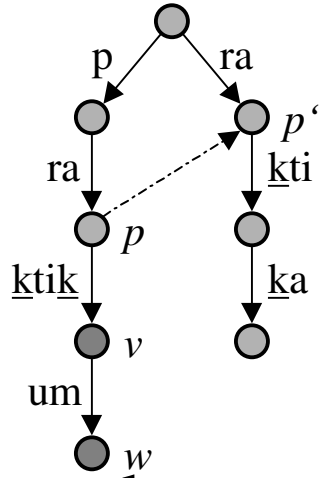


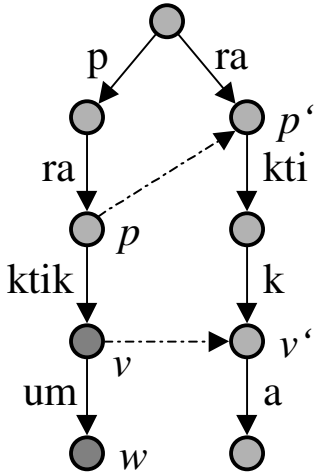
McCreight‘s Algorithmus (Forts.)

- *v* ist neu, Zeiger zu *v‘* existiert noch nicht
- *v‘* mit Hilfe von *p*→*p‘* finden

  v‘ evtl. einfügen
- „schnelle Suche“:
  - Beschr. (*p*,*v*) kommt auf jeden Fall vor
  - nur Anfangsbuchst. der Kanten nach *p‘*







- man kann zeigen: gesamt LZ $O(n)$