Prüfe, ob es sich bei einem Array A der Länge n um ein Suffix-Array handelt (von T)

1. A ist eine Permutation der Zahlen $[0, n)$ ✓ Jedes Suffix muss genau einmal im Suffix-Array vorkommen

2. Für alle i,j gilt

$$A^{-1}[i] \leq A^{-1}[j] \iff (T[i], A^{-1}[i+1]) \leq (T[j], A^{-1}[j+1])$$

Was bedeutet inverses SA? ($A^{-1}[i]$)

↳ Wo steht T^i im SA

↳ Was kann daraus geschlossen werden?

↳ Wenn $A^{-1}[i] < A^{-1}[j]$ dann $T^i <_{lex} T^j$

Beweis von 2.

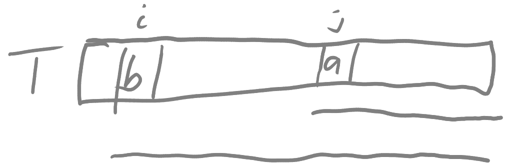
Annahme: T^i und T^j sind zwei Suffixe mit $T^i >_{lex} T^j$ ABER $A^{-1}[i] < A^{-1}[j]$

Ugs.: „Wir haben mindestens zwei falsch einsortierte Suffixe in unserem SA"

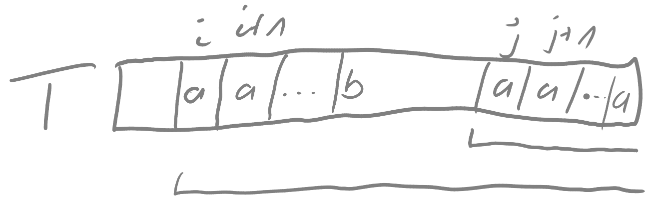
$\underbrace{\text{mit } i+j \text{ maximal}}$

falls es mehrere falsch einsortierte Suffixe gib ist das wichtig

Da $T^i >_{lex} T^j$ gilt entweder



① $T[i] > T[j] \rightarrow$ ↯ Widerspruch zur 2. Bedingung, denn $(T[i], \cdot) > (T[j], \cdot)$



② $T[i] = T[j]$ und $\boxed{T^{i+1} >_{lex} T^{j+1}}$

$\rightarrow A^{-1}[i+1] > A^{-1}[j+1]$ → hier ist die Maximalität von i+j wichtig

→ ↯ Widerspruch zur 2. Bedingung, denn $(T[i], A^{-1}[i+1]) > (\underset{\overset{\shortparallel}{T[j]}}{T[i]}, A^{-1}[j+1])$

Wir haben ein SA und wollen zeigen, dass 1. und 2. gilt.

1. Offensichtlich, da SA Permutation von $[0,n)$ ist

2. Wir haben alle Suffixe korrekt sortiert, d.h. wenn $T^i >_{lex} T^j$, dann ist $A^{-1}[i] > A^{-1}[j]$ und wenn $A^{-1}[i] > A^{-1}[j]$ daraus folgen 2 Fälle

① $T[i] > T[j]$ also ist auch $(T[i], .) > (T[j], .)$

② $T[i] = T[j]$ und $T^{i+1} >_{lex} T^{i+1}$ also ist $(T[i], A^{-1}[i+1]) > (\underset{\overset{\shortparallel}{T[i]}}{T[j]}, A^{-1}[j+1])$

→ Durch diese Eigenschaften können wir den SA-Check auf Sortieren reduzieren. Besser als naiv mit quadratischer Laufzeit

Der Code

sa_tuples : $\begin{pmatrix} 0 \\ SA[0] \end{pmatrix} \begin{pmatrix} 1 \\ SA[1] \end{pmatrix} \cdots \begin{pmatrix} n-1 \\ SA[n-1] \end{pmatrix}$

sortiere nach 2. Komponente

$\begin{pmatrix} SA^{-1}[0] \\ 0 \end{pmatrix} \begin{pmatrix} SA^{-1}[1] \\ 1 \end{pmatrix} \cdots \begin{pmatrix} SA^{-1}[n-1] \\ n-1 \end{pmatrix}$

Prüfe ob es sich um eine Permutation handelt

rts : $\begin{pmatrix} SA^{-1}[0] \\ SA^{-1}[1] \\ T[0] \end{pmatrix}, \begin{pmatrix} SA^{-1}[1] \\ SA^{-1}[2] \\ T[1] \end{pmatrix} \cdots \begin{pmatrix} SA^{-1}[n-1] \\ 0 \\ T[n-1] \end{pmatrix}$

sortiere nach der 1. Komponente

dann überprüfe Zeichen und $SA^{-1}[i+1]$

SAIS-Code

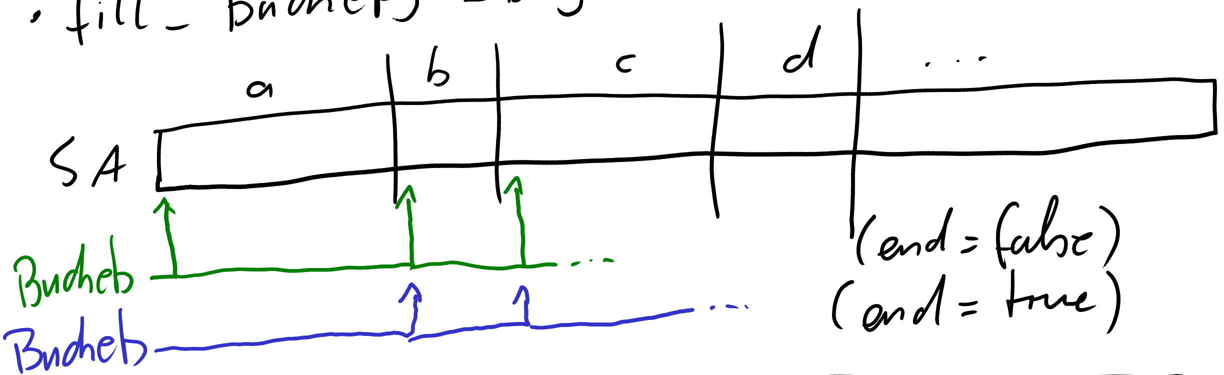
- std::vector::begin → Pointer auf den Anfang
- nutzen std::vector<bool> um S-Suffixe zu markieren
  ↳ Bitvektor (1 Bit pro Eintrag) → types

  ↳ Enthält an Position i eine 1, wenn T^i ein S-Typ ist

  ↳ Was ist mit den LMS-Suffixen (S^*)
  1. T^0 kann kein S^* sein
  2. Für jedes LMS-Suffix gilt types[i] && !types[i-1]

- nutzen uint32_t ::max als Markierung
  ↳ üblicher Trick, auch gerne mit singned Integern hier dann negative Wert

- fill_buckets → gibt Grenzen zurück



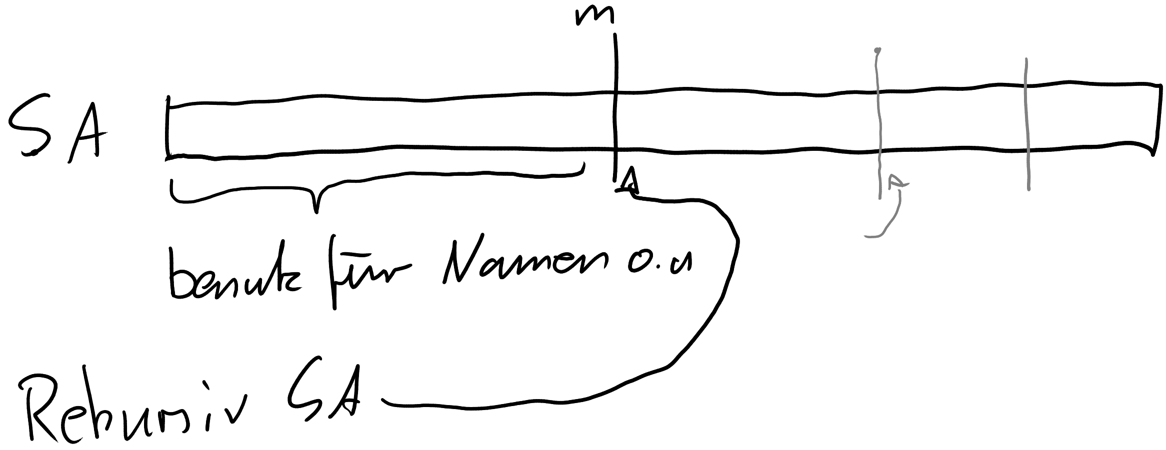
Präfixsumme A

B $\sum_{i=0}^{0} A[i] \mid \sum_{i=0}^{1} A[i]$ .... (end = true / inclusive)

B $0 \mid \sum_{i=0}^{0} A[i] \mid \sum_{i=0}^{1} A[i]$ ... (end = false / exclive)

- for-Schleifen kommen auf zwei Arten unterbrochen werden:
  - break → verlässt for-Schleife sofort
  - continue → macht sofort mit der nächsten Iteration weiter

- Verwende SA zum Speichern von Daten
  ↳ In jedem Aufruf max $n/2 = m$ LMS-Subst.



Im optimierten Code bei Klassifizierung