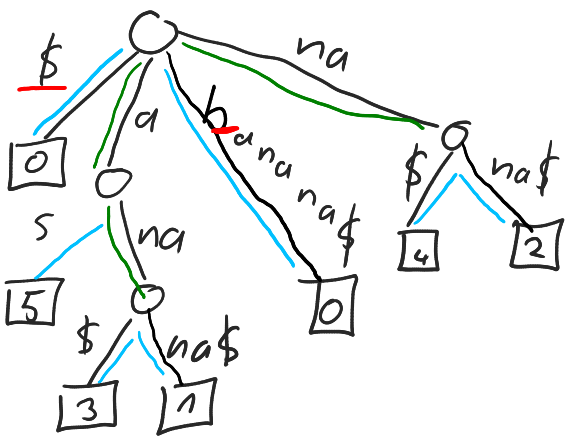
Suche von Teilwörtern

a) $T = \underset{0\,1\,2\,3\,4\,5\,6}{\text{banana\$}}$

$SA = 6531042$

$LCP = 0013002$

Suffix-Baum



Wann kommt Teilstring nur einmal vor?

↳ grüne Kanten sind gemeinsame Präfixe

↳ Nur auf Kanten zu Blättern kann ein eindeutiges Teilwort vorkommen

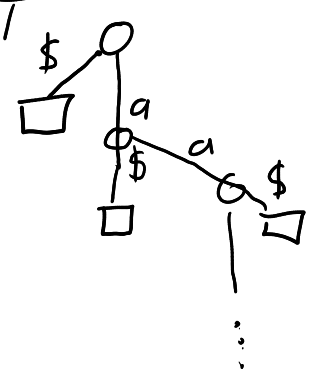
→ Suche Buchstabe mit geringster Stringtiefe auf blauen Kanten

→ In unserem Beispiel „$" und „b"

$T = aaaaaaaa\ldots\$$ aber $T = aaaaaa\ldots a$

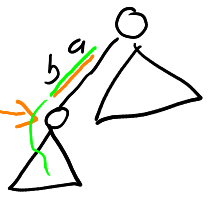
↳ aaaaa... a

↳ ST

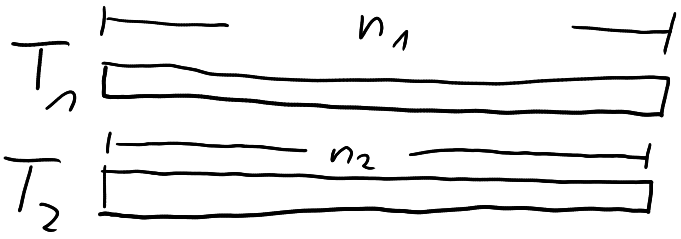


Wozu dient das $

$T = abaca\ldots ab$

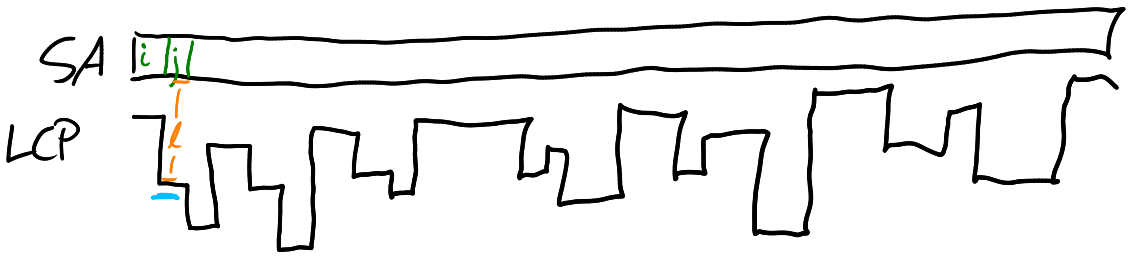


b)



$T = T_1 \$_1 T_2 \$_2$ $\$_1, \$_2$ sind unterschiedl. Sent.

↳ Konstruiere SA und LCP-Array für T



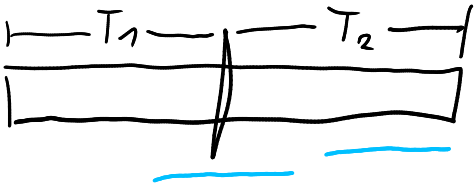
→ suche größten LCP-Wert von links nach rechts.

→ i & j haben ℓ Zeichen gemeinsam.

↳ überprüfe ob ① $i \in T_1$ und $j \in T_2$ oder
② $i \in T_2$ und $j \in T_1$.

↳ ① $i < n_1$ und $j > n_1 + 1$
② $i > n_1 + 1$ und $j < n_1$

Achtung: Ohne $\$_1$ $T_1 T_2 =$



aber $\$_2$ eigentlich nicht notwendig

Programmieraufgabe

Naiv

① Erstelle Vektor [0, n) SA

② Lambda-Funktion für std::sort

↳ Parameter: 2 Elemente a,b aus SA

↳ Rückgabe: string[a]...string[n-1] < string[b]...string[n-1]

Präfix-Doubling

↳ Idee T = a a b b $ (0 1 2 3 4)

1 1 2 2 0

1 — 2 — 4 — 3 — 0

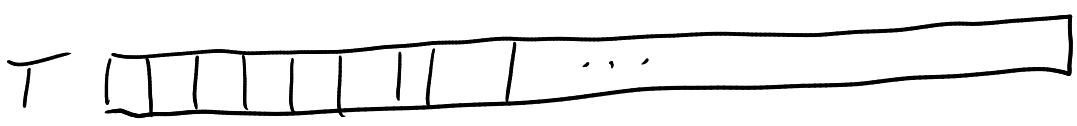
→ SA = 4 0 1 3 2

↳ Wir haben die Hilfen

Index Rank (IR)
- ↳ index
- ↳ rank

IndexRankRang (IRR)
- ↳ index
- ↳ rank1
- ↳ rank2

↳ Vor der while-Schleife



IRR(0, T[0], T[1]), IRR(1, T[1], T[2]), ... IRR(n-1, T[n-1], 0)

In der while-Schleife

① Sortiere IRRs nach Rängen

② Bestimme neue Ränge in IR

③ Abbruchbedingung → sind alle Ränge eindeutig
→ wenn ja: sind wir fertig mit der Schleife

④ Bilde neue Rang-Paare

- wir sortieren die Indizes i nach $(i \,\%\, 2^{\text{iteration}},\ i / 2^{\text{iteration}})$

Beispiel: It. 1 → Rang-Paare aus Textpos.
(0&1, 1&2, 2&3, 3&4, ...) haben wir aus dem Text

It. 2 → 0&2, 1&3, 2&4, 3&5 ...

It. 3 → 0&4, 1&5, 2&6, 3&7, 4&8 ...
$\%4=0$ $\%4=1$ $\%4=2$ $\%4=3$ $\%4=0$

Ergebnis der Sortierung (Indizes)

It 3: idx 0 4 8 12, ... , 1 5 9 13, ... , 2 6 10 14, ... 3 7, ...
rank

- Bilde IRRs aus benachbarten IRs

WENN Indizes $2^{\text{iteration}}$ weit auseinander liegen

- Nächste Iteration

⑤ Wenn alle Ränge eindeutig sind, dann gib die Indizes zurück