

The smallest grammar problem

Edgar Dorausch

05. Juli 2019

Motivation und Anwendung

- Mustererkennung
z.B. DNA-Analyse, NLP

- Mustererkennung
z.B. DNA-Analyse, NLP
- Kompression

Definitionen und Wiederholung

Kontextfreie Grammatik

Eine KFG ist ein Quadrupel $(\Sigma, \Gamma, S, \Delta)$ mit

- Σ - Terminalalphabet
- Γ - Nichtterminalalphabet
- S - Startsymbol
- Δ - Menge von Regeln der Form $T \rightarrow \alpha$
 $T \in \Gamma; \alpha \in (\Sigma \cup \Gamma)^*$

Besonderheit!:

Grammatiken sollen nur ein Wort erzeugen. Deshalb:

- Grammatik muss azyklisch sein
- Für jedes $T \in \Gamma$ existiert nur eine Regel in Δ

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle \alpha \rangle$

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle \alpha \rangle$

Expansionslänge

Anzahl Zeichen in Expansion eines Strings α

Notation: $[\alpha] = |\langle \alpha \rangle|$

Größe einer Grammatik G

Anzahl der Zeichen in den rechten Seiten der Grammatikregeln

$$\text{Notation: } m = |G| = \sum_{(T \rightarrow \alpha) \in \Delta} |\alpha|$$

Größe der kleinsten Grammatik für einen String: m^*

Beispiel

$$G: \left\{ \begin{array}{l} S \rightarrow rhaTber_TTa \\ T \rightarrow bar \end{array} \right\}$$

$$\langle S \rangle = rhabarber_barbara$$

$$[S] = 17$$

$$|G| = 11$$

Approximation Ratio

Sei G_A die Grammatik, die von einem Algorithmus A erzeugt wird.

$$a(n) = \max_{\alpha \in \Sigma^n} \frac{|G_A| \text{ für } \alpha}{m^* \text{ für } \alpha}$$

Approximation Ratio

Sei G_A die Grammatik, die von einem Algorithmus A erzeugt wird.

$$a(n) = \max_{\alpha \in \Sigma^n} \frac{|G_A| \text{ für } \alpha}{m^* \text{ für } \alpha}$$

Worstcase!

Tabelle 1: Landau Notation

	$f \in o(g)$	" $f < g$ "
(Upper bound)	$f \in \mathcal{O}(g)$	" $f \leq g$ "
	$f \in \Theta(g)$	" $f = g$ "
(Lower bound)	$f \in \Omega(g)$	" $f \geq g$ "
	$f \in \omega(g)$	" $f > g$ "

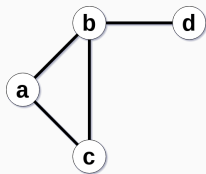
Komplexität

- Vertex Cover lässt sich auf SGP reduzieren

- Vertex Cover lässt sich auf SGP reduzieren
- Zusammenhang mit Addition Chains (nicht im Vortrag)

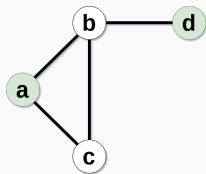
Vertex Cover

Suche (minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



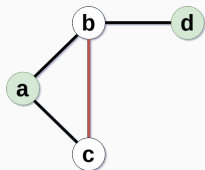
Vertex Cover

Suche (minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



Vertex Cover

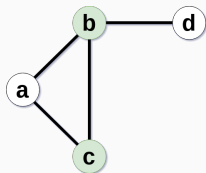
Suche (minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



(Kein Vertex Cover!)

Vertex Cover

Suche (minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



NP-härte

- Betrachte nur Graphen mit maximalen Knoten-Grad 3

NP-härte

- Betrachte nur Graphen mit maximalen Knoten-Grad 3
- Überführung von Graphen zu Wörtern

NP-härte

- Betrachte nur Graphen mit maximalen Knoten-Grad 3
- Überführung von Graphen zu Wörtern
- Zeige, dass man über die kleinste Grammatik einen Vertex Cover bestimmen kann

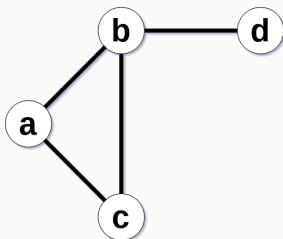
NP-härte

- Betrachte nur Graphen mit maximalen Knoten-Grad 3
- Überführung von Graphen zu Wörtern
- Zeige, dass man über die kleinste Grammatik einen Vertex Cover bestimmen kann
- Berechne Upper Bound für effiziente Approximation (außer $P = NP$)

Beispiel Graph

$$V = \{a, b, c, d\}$$

$$E = \left\{ \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\} \right\}$$



Graphen zu String überführen

$$\alpha = \prod_{v_i \in V} (\#v_i \# \dagger v_i \# \dagger)^2 \prod_{v_i \in V} (\#v_i \# \dagger) \prod_{\{v_i, v_j\} \in E} (\#v_i \# v_j \# \dagger)$$

Graphen zu String überführen

$$\alpha = \prod_{v_i \in V} (\#v_i \# \dagger v_i \# \dagger)^2 \prod_{v_i \in V} (\#v_i \# \dagger) \prod_{\{v_i, v_j\} \in E} (\#v_i \# v_j \# \dagger)$$

$$V = \{a, b, c, d\}; E = \left\{ \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\} \right\}$$

$$\alpha_{\text{Beispiel}} = (\#a \# \dagger a \# \dagger)^2 (\#b \# \dagger b \# \dagger)^2 (\#c \# \dagger c \# \dagger)^2 (\#d \# \dagger d \# \dagger)^2 \\ \#a \# \dagger \#b \# \dagger \#c \# \dagger \#d \# \dagger \\ \#a \# b \# \dagger \#a \# c \# \dagger \#b \# c \# \dagger \#b \# d \# \dagger$$

$$\alpha_{\text{Beispiel}} = (\#a \# a\#)^2 (\#b \# b\#)^2 (\#c \# c\#)^2 (\#d \# d\#)^2$$

$\#a\# \#b\# \#c\# \#d\#$
 $\#a\#b\# \#a\#c\# \#b\#c\# \#b\#d\#$

Eigenschaften der kleinsten Grammatik

- Jedes Nichtterminal expandiert zu $\#v_i$, $v_i\#$ oder $\#v_i\#$

$$\alpha_{\text{Beispiel}} = (\#a \dagger a\#\dagger)^2 (\#b \dagger b\#\dagger)^2 (\#c \dagger c\#\dagger)^2 (\#d \dagger d\#\dagger)^2$$
$$\#a\#\dagger \#b\#\dagger \#c\#\dagger \#d\#\dagger$$
$$\#a\#b\#\dagger \#a\#c\#\dagger \#b\#c\#\dagger \#b\#d\#\dagger$$

Eigenschaften der kleinsten Grammatik

- Jedes Nichtterminal expandiert zu $\#v_i$, $v_i\#$ oder $\#v_i\#$
- Enthält Regeln der Form $T_j \rightarrow \#v_i$ und $T_j \rightarrow v_i\#$

$$\begin{aligned}\alpha_{\text{Beispiel}} = & (\#a \dagger a\#\dagger)^2 (\#b \dagger b\#\dagger)^2 (\#c \dagger c\#\dagger)^2 (\#d \dagger d\#\dagger)^2 \\ & \#a\#\dagger \#b\#\dagger \#c\#\dagger \#d\#\dagger \\ & \#a\#b\#\dagger \#a\#c\#\dagger \#b\#c\#\dagger \#b\#d\#\dagger\end{aligned}$$

Eigenschaften der kleinsten Grammatik

- Jedes Nichtterminal expandiert zu $\#v_i$, $v_i\#$ oder $\#v_i\#$
- Enthält Regeln der Form $T_j \rightarrow \#v_i$ und $T_j \rightarrow v_i\#$
- $C = \{v_i \in V \mid \exists T_j \rightarrow \#v_i\# \}$ ist (minimale) Vertex Cover

Approximation Ratio

- $m^* = 15|V| + 3|E| + |C|$

Approximation Ratio

- $m^* = 15|V| + 3|E| + |C|$
- Es ist (*NP*) hart Vertex Cover kleiner als $\frac{145}{144} \cdot |C|$ zu finden
($\frac{145}{144} \approx 1,006944\dots$)

Approximation Ratio

- $m^* = 15|V| + 3|E| + |C|$
- Es ist (*NP*) hart Vertex Cover kleiner als $\frac{145}{144} \cdot |C|$ zu finden
($\frac{145}{144} \approx 1,006944\dots$)
- $\rho = \frac{15|V|+3|E|+\frac{145}{144}|C|}{15|V|+3|E|+|C|}$

Approximation Ratio

- $m^* = 15|V| + 3|E| + |C|$
- Es ist (*NP*) hart Vertex Cover kleiner als $\frac{145}{144} \cdot |C|$ zu finden
($\frac{145}{144} \approx 1,006944\dots$)
- $\rho = \frac{15|V| + 3|E| + \frac{145}{144}|C|}{15|V| + 3|E| + |C|}$
- $\rho \geq \frac{15|V| + 3 \cdot \frac{3}{2}|V| + \frac{145}{144}(\frac{1}{3}|V|)}{15|V| + 3 \cdot \frac{3}{2}|V| + \frac{1}{3}|V|} = \frac{8569}{8568} \approx 1,0001167\dots$

Algorithmen

- LZ78

- LZ78
- Bisection

- LZ78
- Bisection
- Sequential

- LZ78
- Bisection
- Sequential
- Global algorithms

- LZ78
- Bisection
- Sequential
- Global algorithms
 - Longest Match

- LZ78
- Bisection
- Sequential
- Global algorithms
 - Longest Match
 - Greedy


- LZ78
- Bisection
- Sequential
- Global algorithms
 - Longest Match
 - Greedy
 - Re-Pair

Lower bound bestimmen

- Definiere α_k ($n = |\alpha_k|$)



Lower bound bestimmen


- Definiere α_k ($n = |\alpha_k|$) 
- Bestimme lower bound von m
 $m \in \Omega(f_l(n))$

Lower bound bestimmen



- Definiere α_k ($n = |\alpha_k|$)
- Bestimme lower bound von m
 $m \in \Omega(f_l(n))$
- Bestimme upper bound von m^*
 $m^* \in \mathcal{O}(f_u(n))$

Lower bound bestimmen

- Definiere α_k ($n = |\alpha_k|$) 
- Bestimme lower bound von m
 $m \in \Omega(f_l(n))$
- Bestimme upper bound von m^*
 $m^* \in \mathcal{O}(f_u(n))$

$$\Rightarrow \boxed{a(n) \in \Omega\left(\frac{f_l(n)}{f_u(n)}\right)}$$

LZ78

- Gängiger Kompressionsalgorithmus



LZ78

- Gängiger Kompressionsalgorithmus
- Abraham Lempel und Jacob Ziv (1978)



LZ78

- Gängiger Kompressionsalgorithmus
- Abraham Lempel und Jacob Ziv (1978)
- Verwendung bei GIF und TIFF



LZ78 - Datenstrukturen

- Strings als Sequenzen von Paaren (i, c) dargestellt
 i ...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$

LZ78 - Datenstrukturen

- Strings als Sequenzen von Paaren (i, c) dargestellt
 i ...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Jedes Paar repräsentiert Substring

LZ78 - Datenstrukturen

- Strings als Sequenzen von Paaren (i, c) dargestellt
 i ...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Jedes Paar repräsentiert Substring
- Wenn i gleich 0 dann ist dieser Substring gleich c

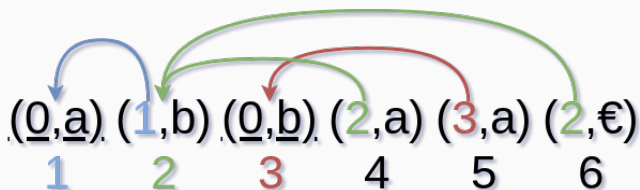
LZ78 - Datenstrukturen

- Strings als Sequenzen von Paaren (i, c) dargestellt
 i ...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Jedes Paar repräsentiert Substring
- Wenn i gleich 0 dann ist dieser Substring gleich c
- Andernfalls ist der Substring des i -ten Paares gefolgt von c

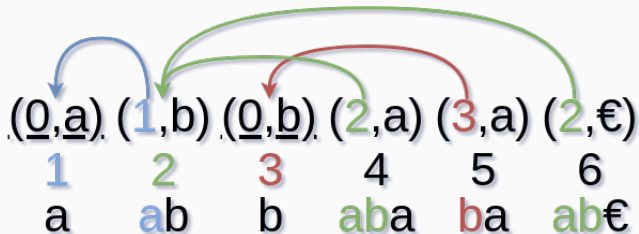
Beispiel

(0,a) (1,b) (0,b) (2,a) (3,a) (2,€)
1 2 3 4 5 6

Beispiel



Beispiel



LZ78 Grammatiken

- Paar $\hat{=}$ Nichtterminal

LZ78 Grammatiken

- Paar $\hat{=}$ Nichtterminal
- $$\begin{cases} X_j \rightarrow c, & i = 0 \\ X_j \rightarrow X_i c, & \text{sonst} \end{cases}$$

LZ78 Grammatiken

- Paar $\hat{=}$ Nichtterminal
- $$\begin{cases} X_j \rightarrow c, & i = 0 \\ X_j \rightarrow X_i c, & \text{sonst} \end{cases}$$
- $S \rightarrow X_1 \dots X_k$

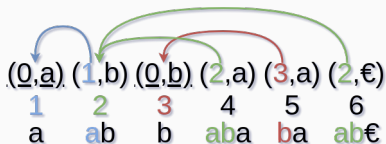
LZ78 Grammatiken

- Paar $\hat{=}$ Nichtterminal
- $$\begin{cases} X_j \rightarrow c, & i = 0 \\ X_j \rightarrow X_i c, & \text{sonst} \end{cases}$$
- $S \rightarrow X_1 \dots X_k$

$S \rightarrow X_1 X_2 X_3 X_4 X_5 X_6$

$X_1 \rightarrow a; X_2 \rightarrow X_1 b; X_3 \rightarrow b$

$X_4 \rightarrow X_2 a; X_5 \rightarrow X_3 a; X_6 \rightarrow \epsilon$



LZ78 - Algorithmus

- String wird Schrittweise in einem Durchlauf von links nach rechts in eine Sequenz von Paaren übersetzt

LZ78 - Algorithmus

- String wird Schrittweise in einem Durchlauf von links nach rechts in eine Sequenz von Paaren übersetzt
- Finde in jedem Schritt das kürzeste Präfix γ des verbleibenden Strings das nicht Expansion eines bereits erzeugten Paares ist

LZ78 - Algorithmus

- String wird Schrittweise in einem Durchlauf von links nach rechts in eine Sequenz von Paaren übersetzt
- Finde in jedem Schritt das kürzeste Präfix γ des verbleibenden Strings das nicht Expansion eines bereits erzeugten Paares ist
- Am Ende des Strings muss eventuell ein weiteres Zeichen hinzugefügt werden

LZ78 - Algorithmus

- String wird Schrittweise in einem Durchlauf von links nach rechts in eine Sequenz von Paaren übersetzt
- Finde in jedem Schritt das kürzeste Präfix γ des verbleibenden Strings das nicht Expansion eines bereits erzeugten Paares ist
- Am Ende des Strings muss eventuell ein weiteres Zeichen hinzugefügt werden
- Ein neues Paar wird an die Liste angehängen:
 1. Wenn da $\gamma = 1$ ist füge $(0, \gamma)$ hinzu

LZ78 - Algorithmus

- String wird Schrittweise in einem Durchlauf von links nach rechts in eine Sequenz von Paaren übersetzt
- Finde in jedem Schritt das kürzeste Präfix γ des verbleibenden Strings das nicht Expansion eines bereits erzeugten Paares ist
- Am Ende des Strings muss eventuell ein weiteres Zeichen hinzugefügt werden
- Ein neues Paar wird an die Liste angehängen:
 1. Wenn da $\gamma = 1$ ist füge $(0, \gamma)$ hinzu
 2. Andernfalls ist $\gamma = \alpha c$.
 α ... Expansion eines Paares mit dem Index i_α
 \Rightarrow Paar: (i, c)

Beispiel

aabbababaab€

Beispiel

aabbababaab€

$(0, a)$ abbababaab€
 $\underbrace{\hspace{1.5cm}}_a$

Beispiel

aabbababaab \in

$(0, a)$ abbababaab \in
 $\underbrace{\hspace{1.5cm}}_a$

$(0, a)$ $(1, b)$ bababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab}$

Beispiel

aabbababaab \in

$\underbrace{(0, a)}_a$ abbababaab \in

$\underbrace{(0, a)}_a \underbrace{(1, b)}_{ab}$ bababaab \in

$\underbrace{(0, a)}_a \underbrace{(1, b)}_{ab} \underbrace{(0, b)}_b$ ababaab \in

Beispiel

aabbababaab \in

$(0, a)$ abbababaab \in
 $\underbrace{\hspace{1.5cm}}_a$

$(0, a)$ $(1, b)$ bababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab}$

$(0, a)$ $(1, b)$ $(0, b)$ ababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ baab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba}$

Beispiel

aabbababaab \in

$(0, a)$ abbababaab \in
 $\underbrace{\hspace{1.5cm}}_a$

$(0, a)$ $(1, b)$ bababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab}$

$(0, a)$ $(1, b)$ $(0, b)$ ababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ baab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba}$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ $(3, a)$ ab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba} \quad \underbrace{\hspace{1.5cm}}_{ba}$

Beispiel

aabbababaab \in

$(0, a)$ abbababaab \in
 $\underbrace{\hspace{1.5cm}}_a$

$(0, a)$ $(1, b)$ bababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab}$

$(0, a)$ $(1, b)$ $(0, b)$ ababaab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ baab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba}$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ $(3, a)$ ab \in
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba} \quad \underbrace{\hspace{1.5cm}}_{ba}$

$(0, a)$ $(1, b)$ $(0, b)$ $(2, a)$ $(3, a)$ $(2, \epsilon)$
 $\underbrace{\hspace{1.5cm}}_a \quad \underbrace{\hspace{1.5cm}}_{ab} \quad \underbrace{\hspace{1.5cm}}_b \quad \underbrace{\hspace{1.5cm}}_{aba} \quad \underbrace{\hspace{1.5cm}}_{ba} \quad \underbrace{\hspace{1.5cm}}_{ab\epsilon}$

LowerBound

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $$\begin{aligned} |\alpha_k| &= k \frac{k+1}{2} + (1+k)(k+1)^2 \\ &= k^3 + \frac{7}{2}k^2 + \frac{7}{2}k + 1 \end{aligned}$$

LowerBound

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $|\alpha_k| = k \frac{k+1}{2} + (1+k)(k+1)^2$
 $= k^3 + \frac{7}{2}k^2 + \frac{7}{2}k + 1$
- $n = |\alpha_k| \in \Theta(k^3)$

UpperBound m^*

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(1 + \log(\frac{k^2+k}{2}) + \log(k+1)^2 + 1 + \log(k))$

UpperBound m^*

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(1 + \log(\frac{k^2+k}{2}) + \log(k+1)^2 + 1 + \log(k))$
- $m^* \in \mathcal{O}(\log k)$

UpperBound m^*

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(1 + \log(\frac{k^2+k}{2}) + \log(k+1)^2 + 1 + \log(k))$
- $m^* \in \mathcal{O}(\log k)$
- $m^* \in \mathcal{O}(\log n^{\frac{1}{3}}) = \mathcal{O}(\log n)$

LowerBound m

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- String wird in zwei Phasen in eine Paar-Sequenz übersetzt

LowerBound m

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- String wird in zwei Phasen in eine Paar-Sequenz übersetzt
- Erste Phase: alle Strings $a...a^k$ zu Paaren übersetzt

LowerBound m

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- String wird in zwei Phasen in eine Paar-Sequenz übersetzt
- Erste Phase: alle Strings $a \dots a^k$ zu Paaren übersetzt
- Zweite Phase: $a^i ba^j$ für alle $i, j \in [0, k]$ wird ein Paar erstellt

LowerBound m

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- String wird in zwei Phasen in eine Paar-Sequenz übersetzt
- Erste Phase: alle Strings $a...a^k$ zu Paaren übersetzt
- Zweite Phase: $a^i ba^j$ für alle $i, j \in [0, k]$ wird ein Paar erstellt
- $m \in \Omega(\sum_{z=1}^k z + (k+1)^2) = \Omega(k^2)$

LowerBound m

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- String wird in zwei Phasen in eine Paar-Sequenz übersetzt
- Erste Phase: alle Strings $a...a^k$ zu Paaren übersetzt
- Zweite Phase: $a^i ba^j$ für alle $i, j \in [0, k]$ wird ein Paar erstellt
- $m \in \Omega(\sum_{z=1}^k z + (k+1)^2) = \Omega(k^2)$
- $m \in \Omega(n^{2/3})$

LowerBound

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(\log n)$

LowerBound

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(\log n)$
- $m \in \Omega(n^{2/3})$

LowerBound

$$\alpha_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

- $m^* \in \mathcal{O}(\log n)$
- $m \in \Omega(n^{2/3})$
- $a(n) \in \Omega(\frac{n^{2/3}}{\log n})$

TODO

$$a^2 + b^2 = c^2$$

TODO

Basiert anscheinend auf $BB[\alpha] - Trees$

Zusammenfassung

- interessant für Kompression und Mustererkennung (NLP)

- interessant für Kompression und Musterkennung (NLP)
- Optimale Lösen ist NP-hart

- interessant für Kompression und Mustererkennung (NLP)
- Optimale Lösen ist NP-hart
- Mit bekannten Verfahren lassen sich Approximation generieren (zB LZ78)

- interessant für Kompression und Mustererkennung (NLP)
- Optimale Lösen ist NP-hart
- Mit bekannten Verfahren lassen sich Approximation generieren (zB LZ78)
- Approximationen können sehr gut sein ("LZ77 Variant")