The smallest grammar problem

Edgar Dorausch

05. Juli 2019

Motivation und Anwendung

Motivation und Anwendung

- Mustererkennung

Motivation und Anwendung

- Mustererkennung
- Kompression

Definitionen und Wiederholung

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 $T \to \alpha$
  $T \in \Gamma$; $\alpha \in (\Sigma \cup \Gamma)^*$

Definitionen und Wiederholung

Besonderheit!:

Die Grammatiken sollen nur ein Wort erzeugen. Deshalb:

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

Definitionen und Wiederholung

Expansion eines Strings α

Erhält man durch erschöpfendes Anwenden der Regeln in einer Grammatik bis nur noch Terminale enthalten sind.

Notation: $\langle \alpha \rangle$

Definitionen und Wiederholung

Expansionslänge

Anzahl der Zeichen in der Expansion eines Strings α

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

Definitionen und Wiederholung

Größe einer Grammatik G

Anzahl der Zeichen in den rechten Seiten der Grammatikregeln

Notation: $m = |G| = \sum_{(T \to \alpha) \in \Delta} \langle \alpha \rangle$

Größe der kleinsten Grammatik für einen String: m^*

Definitionen und Wiederholung

Beispiel

$$G: \left\{ \begin{array}{c} S \to rhaTber\_TTa \\ T \to bar \end{array} \right\}$$

$\langle S \rangle = rhabarber\_barbara$

$[S] = 17$

$|G| = 11$

Definitionen und Wiederholung

Approximation Ratio

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

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

Definitionen und Wiederholung

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!

Definitionen und Wiederholung

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

Komplexität

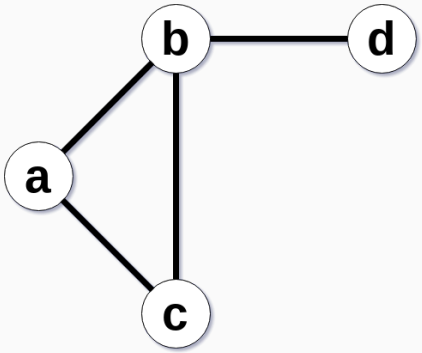
- Vertex Cover lässt sich auf SGP reduzieren

Komplexität

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

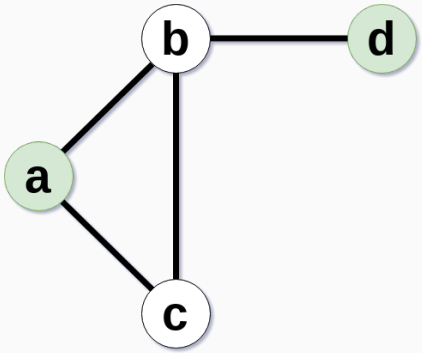
Vertex Cover

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



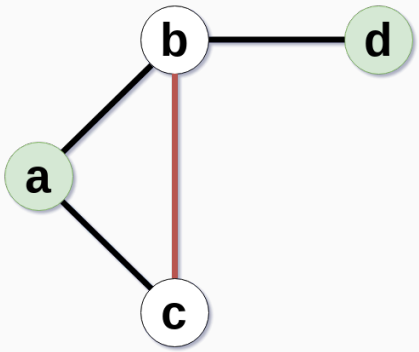
Vertex Cover

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



Vertex Cover

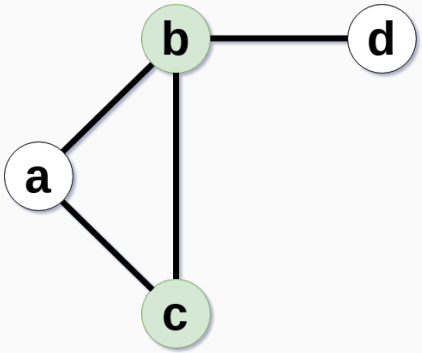
Suche (minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



(Kein Vertex Cover!)

Vertex Cover

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



Komplexität

NP-härte

- Betrachte nur Graphen mit maximalen Knoten-Grad 3

Komplexität

NP-härte

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

Komplexität

NP-härte

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

Komplexität

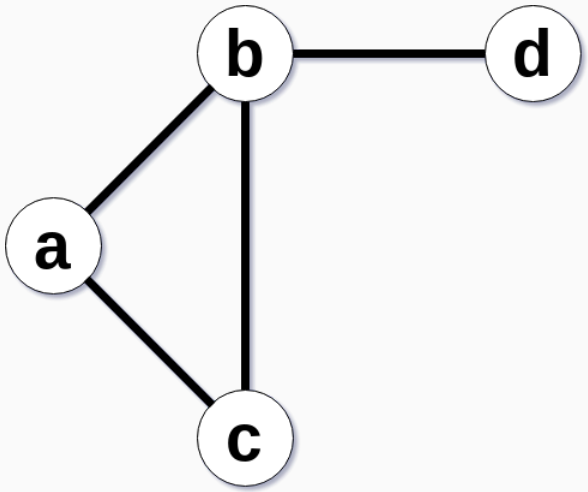
NP-härte

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

Beispiel Graph

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

$$E = \Big\{\{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}\Big\}$$



Komplexität

Graphen zu String überführen

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

Komplexität

Graphen zu String überführen

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

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

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

Komplexität

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

Eigenschaften der kleinsten Grammatik

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

Komplexität

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

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 $T_j \rightarrow v_i\#$

Komplexität

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

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 $T_j \rightarrow v_i\#$
- $C = \{v_i \in V | \exists T_j \rightarrow \#v_i\#\}$ ist (minimale) Vertex Cover

Komplexität

Approximation Ratio

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

Komplexität

Approximation Ratio

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

Komplexität

Approximation Ratio

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

Komplexität

Approximation Ratio

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

Algorithmen

Lower bound bestimmen

- Definiere α (n $= |\alpha|$)

Lower bound bestimmen

- Definiere α $(\mathrm{n} = |\alpha|)$
- Bestimme lower bound von m
  $m \in \Omega(f_l(n))$

Lower bound bestimmen

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

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

LZ78 - Datenstrukturen

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

LZ78 - Datenstrukturen

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

LZ78 - Datenstrukturen

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

LZ78 - Datenstrukturen

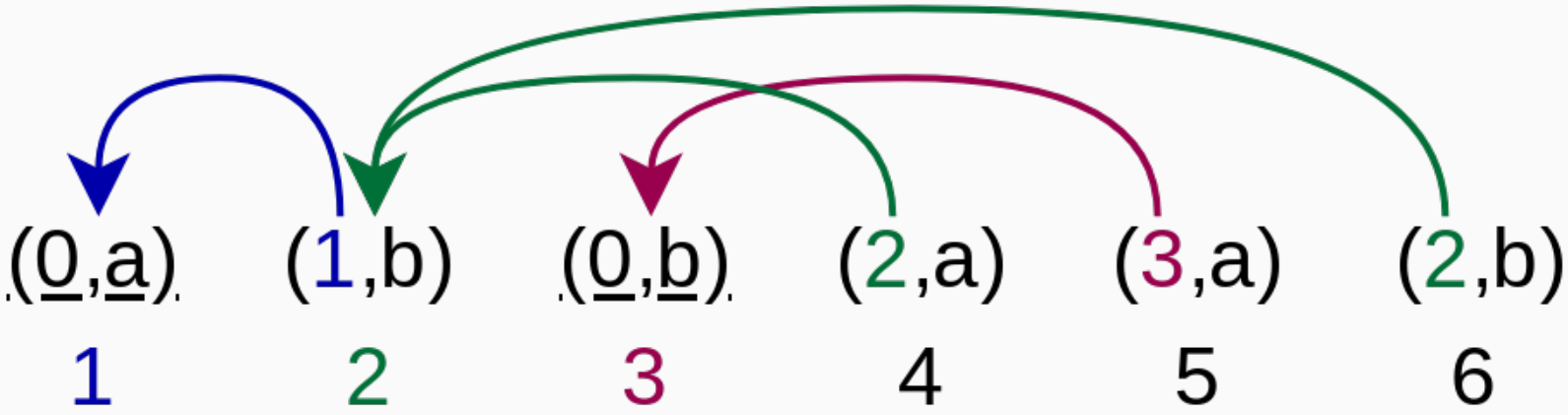
- Strings werden als Sequenzen von Paaren (i, c) dargestellt
  i...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Jedes Paar repräsentiert einen Substring
- Wenn i gleich 0 dann ist dieser Substring gleich c
- Andernfalls ist der Substring des i-ten Parres gefolgt von c

Algorithmen - LZ78

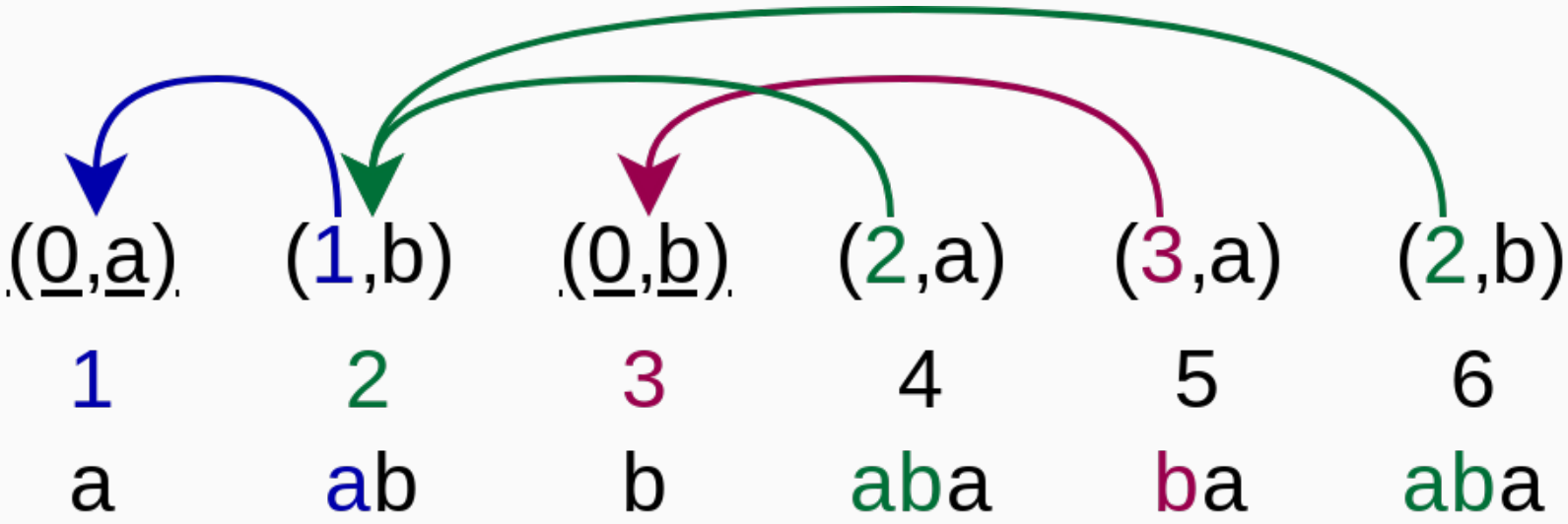
Beispiel

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

Beispiel



Beispiel



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 Paars 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 Paars 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 Paars ist
- Am Ende des Strings muss eventuell ein weiteres Zeichen hinzugefügt werden
- Ein neues Paar wird an die Liste angehangen:
  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 Paars ist
- Am Ende des Strings muss eventuell ein weiteres Zeichen hinzugefügt werden
- Ein neues Paar wird an die Liste angehangen:
  1. Wenn da $\gamma = 1$ ist füge $(0, \gamma)$ hinzu
  2. Andernfalls ist $\gamma = \alpha c$.
     α ... Expansion eines Paars mit dem Index i_α
     $\Rightarrow$ Paar: (i, c)

Algorithmen - LZ78

Beispiel

aabbababaab€

Algorithmen - LZ78

Beispiel

aabbababaab€

(0,a) abbababaab€

Algorithmen - LZ78

Beispiel

aabbababaab€

(0,a) abbababaab€

(0,a) (1,b) bababaab€

Algorithmen - LZ78

Beispiel

aabbababaab€

(0,a) abbababaab€

(0,a) (1,b) bababaab€

(0,a) (1,b) (0,b) ababaab€

Beispiel

aabbababaab€

(0,a) abbababaab€

(0,a) (1,b) bababaab€

(0,a) (1,b) (0,b) ababaab€

(0,a) (1,b) (0,b) (2,a) baab€

Beispiel

aabbababaab€

(0,a) abbababaab€

(0,a) (1,b) bababaab€

(0,a) (1,b) (0,b) ababaab€

(0,a) (1,b) (0,b) (2,a) baab€

(0,a) (1,b) (0,b) (2,a) (3,a) ab€

Beispiel

aabbababaab€

(0,a) abbababaab€

(0,a) (1,b) bababaab€

(0,a) (1,b) (0,b) ababaab€

(0,a) (1,b) (0,b) (2,a) baab€

(0,a) (1,b) (0,b) (2,a) (3,a) ab€

(0,a) (1,b) (0,b) (2,a) (3,a) (2,€)

LowerBound

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

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

LowerBound

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

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

Algorithmen - LZ78

UpperBound m^*

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

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

Algorithmen - LZ78

UpperBound m^*

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

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

Algorithmen - LZ78

UpperBound m^*

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

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

LowerBound m

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

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

LowerBound m

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

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

LowerBound m

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

- 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

LowerBound m

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

- 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

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

- 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
- $m \in \Omega(\sum_{z=1}^{k} z + (k+1)^2) = \Omega(k^2)$
- $m \in \Omega(n^{2/3})$

Algorithmen - LZ78

LowerBound

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

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

Algorithmen - LZ78

LowerBound

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

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

LowerBound

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

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

Algorithmen - global algorithms

TODO

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

Algorithmen - LZ77 variant

TODO

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