

The smallest grammar problem

Edgar Dorausch

05. Juli 2019

Motivation und Anwendung

- Kompression

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

Definitionen und Wiederholung

Kontextfreie Grammatik

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 (straight-line grammar)

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

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle \alpha \rangle$

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle \alpha \rangle$

Größe einer Grammatik G

Anzahl der Zeichen in den rechten Seiten der Grammatikregeln

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

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle \alpha \rangle$

Größe einer Grammatik G

Anzahl der Zeichen in den rechten Seiten der Grammatikregeln

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

Größe der kleinsten Grammatik G

Notation: m^*

Beispiel

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

$$\langle S \rangle = rhabarber_barbara$$

Beispiel

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

$$\langle S \rangle = rhabarber_barbara$$

$$|\langle S \rangle| = 17$$

Beispiel

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

$$\langle S \rangle = rhabarber_barbara$$

$$|\langle S \rangle| = 17$$

$$|G| = 11$$

Approximation Ratio

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

Approximation Ratio

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

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

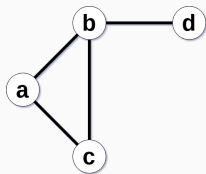
- Vertex Cover auf SGP reduzieren

- Vertex Cover auf SGP reduzieren
- Approximationsschranke für polynomielle Algorithmen

- Vertex Cover auf SGP reduzieren
- Approximationsschranke für polynomielle Algorithmen
- Zusammenhang mit Addition Chains (nicht im Vortrag)

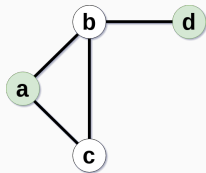
Vertex Cover

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



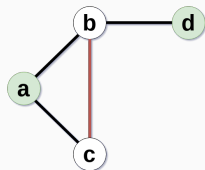
Vertex Cover

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



Vertex Cover

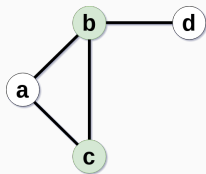
(Minimale) Menge von Knoten, sodass jede Kante mindestens einen dieser Knoten enthält.



(Kein Vertex Cover!)

Vertex Cover

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



NP-härte

- Graphen mit maximalen Knoten-Grad 3

NP-härte

- Graphen mit maximalen Knoten-Grad 3
- Graph als Wort (Unbeschränktes Alphabet)

NP-härte

- Graphen mit maximalen Knoten-Grad 3
- Graph als Wort (Unbeschränktes Alphabet)
- Kleinsete Grammatik \rightarrow Vertex Cover

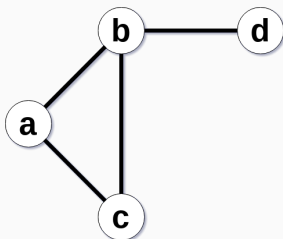
NP-härte

- Graphen mit maximalen Knoten-Grad 3
- Graph als Wort (Unbeschränktes Alphabet)
- Kleinsete Grammatik \rightarrow Vertex Cover
- 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

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



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 $T_j \rightarrow v_i\#$

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

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

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

Eigenschaften der kleinsten Grammatik

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

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



Approximation Ratio

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



Approximation Ratio

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

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



Approximation Ratio

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

$$a(n) = \max_{\alpha \in \Sigma^n} \frac{m}{15|V| + 3|E| + |C|}$$



Approximation Ratio

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

$$a(n) \geq \max_{\alpha \in \Sigma^n} \frac{15|V| + 3|E| + \frac{145}{144}|C|}{15|V| + 3|E| + |C|}$$



Approximation Ratio

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

$$a(n) \geq \max_{\alpha \in \Sigma^n} \frac{15|V| + 3 \cdot \frac{3}{2}|V| + \frac{145}{144}(\frac{1}{3}|V|)}{15|V| + 3 \cdot \frac{3}{2}|V| + \frac{1}{3}|V|}$$



Approximation Ratio

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

$$a(n) \geq \max_{\alpha \in \Sigma^n} \frac{8569}{8568}$$



Approximation Ratio

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



$$a(n) \geq \frac{8569}{8568} \approx 1,0001167\dots$$

Approximationsalgorithmen

- 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

LZ78

- Gängiger Kompressionsalgorithmus



LZ78

- Gänginger Kompressionsalgorithmus
- Abraham Lempel & Jacob Ziv (1978)



LZ78

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



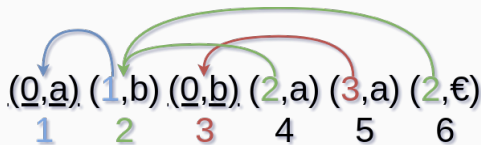
LZ78 - Datenstrukturen

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

$(\underline{0}, \underline{a})$ $(1, b)$ $(\underline{0}, \underline{b})$ $(2, a)$ $(3, a)$ $(2, \epsilon)$
1 2 3 4 5 6

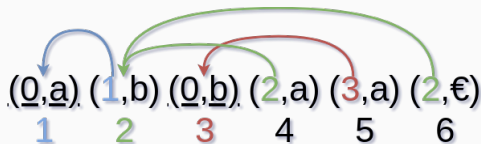
LZ78 - Datenstrukturen

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



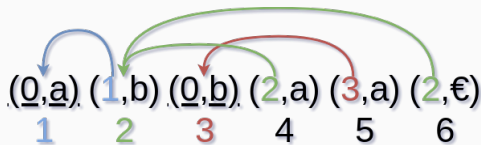
LZ78 - Datenstrukturen

- Strings \rightarrow Sequenzen von Paaren (i, c)
 $i \dots$ Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\hat{=}$ Substring



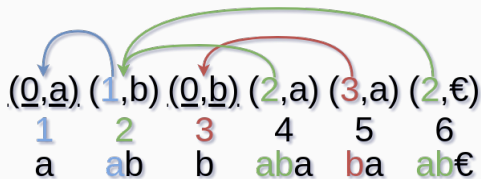
LZ78 - Datenstrukturen

- Strings \rightarrow Sequenzen von Paaren (i, c)
 $i \dots$ Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\hat{=}$ Substring
- Wenn i gleich 0 dann ist dieser Substring gleich c
- Andernfalls ist der Substring des i -ten Paares gefolgt von c



LZ78 - Datenstrukturen

- Strings \rightarrow Sequenzen von Paaren (i, c)
 $i \dots$ Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\hat{=}$ Substring
- Wenn i gleich 0 dann ist dieser Substring gleich c
- Andernfalls ist der Substring des i -ten Paares gefolgt von c



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 sequenziell (von links nach rechts) übersetzt

LZ78 - Algorithmus

- String sequenziell (von links nach rechts) übersetzt
- Iteration: finde kürzestes Präfix γ des in Strings das nicht Expansion eines bereits erzeugten Paars ist

LZ78 - Algorithmus

- String sequenziell (von links nach rechts) übersetzt
- Iteration: finde kürzestes Präfix γ des in Strings das nicht Expansion eines bereits erzeugten Paares ist
- Übersetzungsvorschrift:

LZ78 - Algorithmus

- String sequenziell (von links nach rechts) übersetzt
- Iteration: finde kürzestes Präfix γ des in Strings das nicht Expansion eines bereits erzeugten Paars ist
- Übersetzungsvorschrift:
 1. Wenn $|\gamma| = 1 \Rightarrow (0, \gamma)$

LZ78 - Algorithmus

- String sequenziell (von links nach rechts) übersetzt
- Iteration: finde kürzestes Präfix γ des in Strings das nicht Expansion eines bereits erzeugten Paares ist
- Übersetzungsvorschrift:
 1. Wenn $|\gamma| = 1 \Rightarrow (0, \gamma)$
 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

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

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

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

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

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

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


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

Lower Bound


- Definiere α_k ($n = |\alpha_k|$)




Lower Bound

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

Lower Bound

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

Lower Bound

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

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

Lower Bound (1/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

Lower Bound (1/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

Lower Bound (1/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (1/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (1/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (2/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- Es existiert Grammatik mit Größe: (Lemma 1...3)

$$\underbrace{\mathcal{O}\left(1 + \log\left(\frac{k^2 + k}{2}\right)\right)}_{a^{k(k+1)/2}} + \underbrace{\log((k+1)^2) + 1 + 1 + \log(k)}_{(ba^k)^{(k+1)^2}}$$

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

Lower Bound (3/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (3/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (3/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

- 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

Lower Bound (3/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (3/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

Lower Bound (4/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

$$m \in \Omega(n^{2/3})$$

Lower Bound (4/4)

$$\alpha_k = a^1 a^2 \dots a^k \underbrace{ba^k \dots ba^k}_{(k+1)^2}$$

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

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

$$m \in \Omega(n^{2/3})$$

$$a(n) \in \Omega\left(\frac{n^{2/3}}{\log n}\right)$$

Upper bound

- m proportional zu Anzahl der Nichtterminale

Upper bound

- m proportional zu Anzahl der Nichtterminale
- Nichtterminale in Gruppen zerlegt

Upper bound

- m proportional zu Anzahl der Nichtterminale
- Nichtterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichtterminal und Anzahl Gruppen

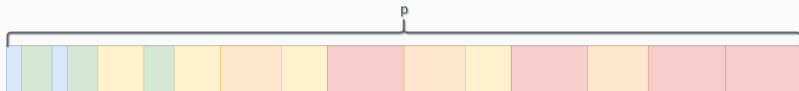
Upper bound

- m proportional zu Anzahl der Nichtterminale
- Nichtterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichtterminal und Anzahl Gruppen
- Abschätzung der Gruppenanzahl

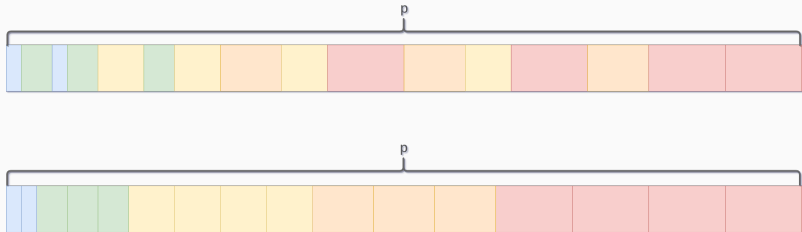
Upper bound

- m proportional zu Anzahl der Nichtterminale
- Nichtterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichtterminal und Anzahl Gruppen
- Abschätzung der Gruppenanzahl
- m bestimmen

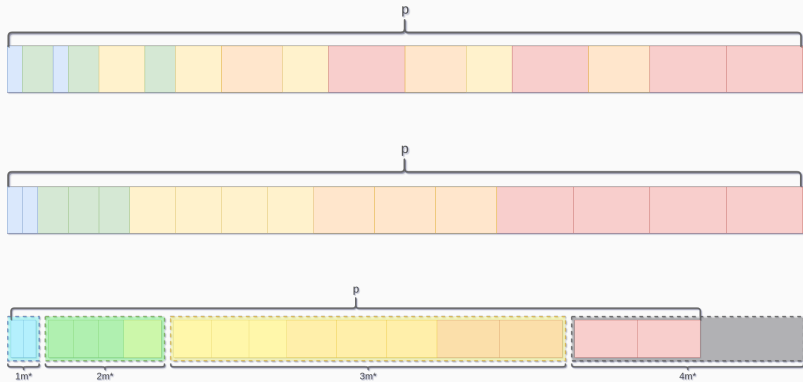
Upper bound (1/4)



Upper bound (1/4)



Upper bound (1/4)



Global Algorithms

- Klasse von Algorithmen

Global Algorithms

- Klasse von Algorithmen
- Haben alle ein upper bound von $\mathcal{O}\left(\left(\frac{n}{\log(n)}\right)^{\frac{2}{3}}\right)$

Global Algorithms

- Klasse von Algorithmen
- Haben alle ein upper bound von $\mathcal{O}\left(\left(\frac{n}{\log(n)}\right)^{\frac{2}{3}}\right)$
- Lower bounds sind sehr schlecht

Global Algorithms - Verfahren

- Grammatik schrittweise verbessert

Global Algorithms - Verfahren

- Grammatik schrittweise verbessert
- Initialisiere Grammatik mit $S \rightarrow \alpha$

Global Algorithms - Verfahren

- Grammatik schrittweise verbessert
- Initialisiere Grammatik mit $S \rightarrow \alpha$
- Wähle einen String γ

Global Algorithms - Verfahren

- Grammatik schrittweise verbessert
- Initialisiere Grammatik mit $S \rightarrow \alpha$
- Wähle einen String γ
- Füge $T \rightarrow \gamma$ (T ... neues Nichtterminal)

Global Algorithms - Verfahren

- Grammatik schrittweise verbessert
- Initialisiere Grammatik mit $S \rightarrow \alpha$
- Wähle einen String γ
- Füge $T \rightarrow \gamma$ (T ... neues Nichtterminal)
- Traversiere alle anderen Regeln von links nach rechts und ersetze vorkommen von γ durch T

Auswahl von γ

- $|\gamma| \geq 2$

Auswahl von γ

- $|\gamma| \geq 2$
- γ kommt mind. zwei mal in Grammatik vor
(ohne Überschneidung)

Auswahl von γ

- $|\gamma| \geq 2$
- γ kommt mind. zwei mal in Grammatik vor
(ohne Überschneidung)
- Alle Strings länger als γ kommen seltener vor

Upper Bound

- Ähnlich upper bound von LZ78

Upper Bound

- Ähnlich upper bound von LZ78
- Auflistung von Substrings der Länge 2

Upper Bound

- Ähnlich upper bound von LZ78
- Auflistung von Substrings der Länge 2
- Einordnung in Gruppen

Upper Bound

- Ähnlich upper bound von LZ78
- Auflistung von Substrings der Länge 2
- Einordnung in Gruppen
- Abschätzen der Gesamt-Expansionslänge der Gruppen

Upper Bound (1/4)

- Wähle $\frac{2}{9}m$ Substrings der Länge 2
(ohne Überschneidung)

Upper Bound (1/4)

- Wähle $\frac{2}{9}m$ Substrings der Länge 2
(ohne Überschneidung)
- ist immer möglich

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu
- Füge die nächsten $3m^*$ Substrings der zweiten Gruppe hinzu

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu
- Füge die nächsten $3m^*$ Substrings der zweiten Gruppe hinzu
- usw. ...(bis zur Gruppe mit gm^* Elementen)

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu
- Füge die nächsten $3m^*$ Substrings der zweiten Gruppe hinzu
- usw. ...(bis zur Gruppe mit gm^* Elementen)
- $2m^* + 3m^* + \dots + gm^*(g + 1)m^* > \frac{2}{9}m$

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu
- Füge die nächsten $3m^*$ Substrings der zweiten Gruppe hinzu
- usw. ... (bis zur Gruppe mit gm^* Elementen)
- $2m^* + 3m^* + \dots + gm^*(g+1)m^* > \frac{2}{9}m$
- $m^* \sum_{k=2}^{g+1} k = m^* \left(\frac{g^2}{2} + \frac{3g}{2} \right) > \frac{2}{9}m$

Upper Bound (2/4)

- Sortiere Substrings aufsteigend nach deren Expansionslänge
- Füge die ersten $2m^*$ Substrings der ersten Gruppe hinzu
- Füge die nächsten $3m^*$ Substrings der zweiten Gruppe hinzu
- usw. ... (bis zur Gruppe mit gm^* Elementen)
- $2m^* + 3m^* + \dots + gm^*(g+1)m^* > \frac{2}{9}m$
- $m^* \sum_{k=2}^{g+1} k = m^* \left(\frac{g^2}{2} + \frac{3g}{2} \right) > \frac{2}{9}m$
- $m \in \mathcal{O}(g^2 m^*)$

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq \sigma$

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq \sigma$
- $\sigma \leq 2n$

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq \sigma$
- $\sigma \leq 2n$
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq 2n$

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq \sigma$
- $\sigma \leq 2n$
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq 2n$
- $m^* \sum_{k=2}^g k^2 = m^* \left(\frac{g^3}{3} + \frac{g^2}{2} + \frac{g}{6} - 1 \right) \leq 2n$

Upper Bound (3/4)

- Sei $\sigma =$ "Gesamt-Expansionslänge"
- Für jedes α in i -ten Gruppen gilt: $|\langle \alpha \rangle| \geq i + 1$ (mk-Lemma)
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq \sigma$
- $\sigma \leq 2n$
- $2^2 m^* + 3^2 m^* + \dots + g^2 m^* \leq 2n$
- $m^* \sum_{k=2}^g k^2 = m^* \left(\frac{g^3}{3} + \frac{g^2}{2} + \frac{g}{6} - 1 \right) \leq 2n$
- $g^3 \in \mathcal{O}\left(\frac{n}{m^*}\right) \Rightarrow g \in \mathcal{O}\left(\left(\frac{n}{m^*}\right)^{\frac{1}{3}}\right)$

Upper Bound (4/4)

- $m \in \mathcal{O}(g^2 m^*)$ und $g \in \mathcal{O}((\frac{n}{m^*})^{\frac{1}{3}})$

Upper Bound (4/4)

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

Upper Bound (4/4)

- $m \in \mathcal{O}(g^2 m^*)$ und $g \in \mathcal{O}((\frac{n}{m^*})^{\frac{1}{3}})$
- $m \in \mathcal{O}((\frac{n}{m^*})^{\frac{2}{3}} m^*) = \mathcal{O}((\frac{n}{\log(n)})^{\frac{2}{3}} m^*)$
- $a(n) = \max_{\alpha \in \Sigma^n} \frac{m}{m^*} \in \mathcal{O}((\frac{n}{\log(n)})^{\frac{2}{3}})$ \square

Greedy

In jeder Iteration wird das γ gewählt welches die Größe der Grammatik am meisten senkt.

Zusammenfassung

- interessant für Kompression und Mustererkennung (NLP)

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

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

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