The smallest grammar problem

Edgar Dorausch

05. Juli 2019

Motivation und Anwendung

Motivation und Anwendung

- Kompression

Motivation und Anwendung

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

Besonderheit:

Grammatiken sollen nur ein Wort erzeugen (straight-line grammar)

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

Definitionen und Wiederholung

Expansion eines Strings α

Erschöpfendes Anwenden der Regeln

Notation: $\langle\alpha\rangle$

Definitionen und Wiederholung

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

Definitionen und Wiederholung

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

Größe der kleinsten Grammatik G

Notation: m^*

Definitionen und Wiederholung

Beispiel

$$G: \begin{Bmatrix} S \rightarrow rhaTber\text{␣}TTa \\ T \rightarrow bar \end{Bmatrix}$$

$\langle S \rangle = rhabarber\text{␣}barbara$

Beispiel

$$G:\ \left\{\begin{matrix} S \to rhaTber\text{␣}TTa \\ T \to bar \end{matrix}\right\}$$

$\langle S \rangle = rhabarber\text{␣}barbara$

$|\langle S \rangle| = 17$

Beispiel

$$G: \left\{ \begin{matrix} S \to rhaTber\textvisiblespace TTa \\ T \to bar \end{matrix} \right\}$$

$\langle S \rangle = rhabarber\textvisiblespace barbara$

$|\langle S \rangle| = 17$

$|G| = 11$

Approximation Ratio

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

Definitionen und Wiederholung

Approximation Ratio

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

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)	$f \in \Omega(g)$	"$f \geq g$"
	$f \in \omega(g)$	"$f > g$"

Lemmata

Lemmata

Lemma 1

$$\boxed{m^* \in \Omega(log(n))}$$

(Ohne Beweis)

Lemmata

Lemma 2a

$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + |G_\beta| + 2 \text{ für } \alpha\beta.}$

Lemmata

Lemma 2a

$$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + |G_\beta| + 2 \text{ für } \alpha\beta.}$$

- $|G_\alpha|$ und $|G_\beta|$ vereinigen

Lemma 2a

$$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + |G_\beta| + 2 \text{ für } \alpha\beta.}$$

- $|G_\alpha|$ und $|G_\beta|$ vereinigen
- Neue Startregel: $S \to S_\alpha S_\beta$

Lemmata

Lemma 2b

$$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + \mathcal{O}(log(k)) \text{ für } \alpha^k.}$$

Lemma 2b

$$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + \mathcal{O}(log(k)) \text{ für } \alpha^k.}$$

- Neue Regeln baumartig strukturieren

Lemma 2b

$\boxed{\text{Es gibt Grammatik der Größe } |G_\alpha| + \mathcal{O}(log(k)) \text{ für } \alpha^k.}$

- Neue Regeln baumartig strukturieren
- Baum hat nur logarithmische Höhe

Lemmata

Lemma 3 (mk-Lemma)

$\boxed{\text{In einem String gibt es maximal } mk \text{ \underline{unterschiedliche} Substrings der Länge } k}$

(Ohne Beweis)

Komplexität

Komplexität

- Vertex Cover auf SGP reduzieren

Komplexität

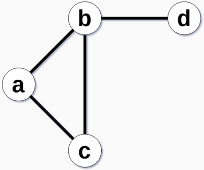
- Vertex Cover auf SGP reduzieren
- Approximationsschranke für polynomielle Algorithmen

Komplexität

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

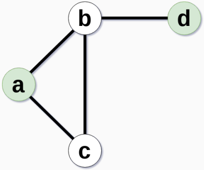
Vertex Cover

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



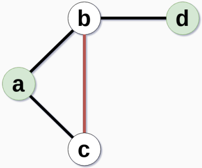
Vertex Cover

(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.

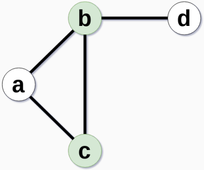


(Kein Vertex Cover!)

Komplexität

Vertex Cover

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



Komplexität

NP-härte

- Graphen mit maximalen Knoten-Grad 3

NP-härte

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

Komplexität

NP-härte

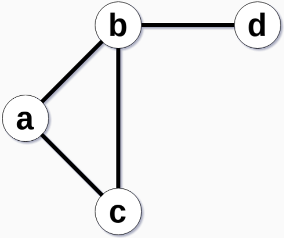
- Graphen mit maximalen Knoten-Grad 3
- Graph als Wort (Unbeschränktes Alphabet)
- Kleinsete Grammatik $\rightarrow$ Vertex Cover

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

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



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

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

Eigenschaften der kleinsten Grammatik

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

$\square$

Approximation Ratio

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

Komplexität

Approximation Ratio

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

Komplexität

Approximation Ratio

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

Komplexität

Approximation Ratio

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

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

Approximation Ratio

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

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

Komplexität

Approximation Ratio

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

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

Komplexität

Approximation Ratio

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

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

Approximation Ratio

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

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

Komplexität

Approximation Ratio

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

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

Approximationsalgorithmen

Approximationsalgorithmen

- LZ78

Approximationsalgorithmen

- LZ78
- Bisection

Approximationsalgorithmen

- LZ78
- Bisection
- Sequential

Approximationsalgorithmen

- LZ78
- Bisection
- Sequential
- Global algorithms

Approximationsalgorithmen

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

Approximationsalgorithmen

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

Approximationsalgorithmen

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

Approximationsalgorithmen - LZ78

LZ78

- Gänginger Kompressionsalgorithmus

LZ78

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

LZ78

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

Approximationsalgorithmen - LZ78

LZ78 - Datenstrukturen

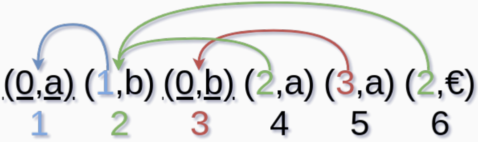
- Strings → Sequenzen von Paaren (i, c)
  i...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$

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

Approximationsalgorithmen - LZ78

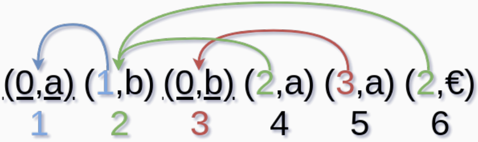
LZ78 - Datenstrukturen

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



LZ78 - Datenstrukturen

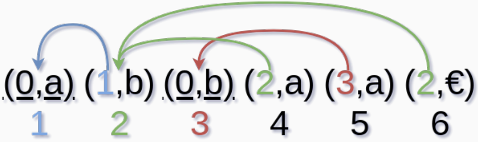
- Strings $\rightarrow$ Sequenzen von Paaren (i, c)
  i...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\hat{=}$ Substring



Approximationsalgorithmen - LZ78

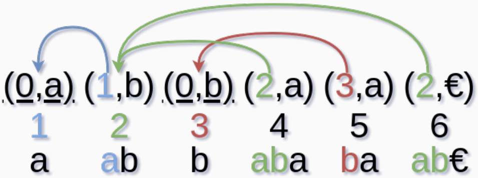
LZ78 - Datenstrukturen

- Strings → Sequenzen von Paaren (i, c)
  i...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\widehat{=}$ 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 → Sequenzen von Paaren (i, c)
  i...Index eines Vorgänger-Paares oder 0; $c \in \Sigma$
- Paar $\triangleq$ 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 $\hat{=}$ Nichterminal

LZ78 - Grammatiken

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

Approximationsalgorithmen - LZ78

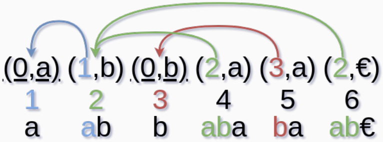
LZ78 - Grammatiken

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

LZ78 - Grammatiken

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

$S \rightarrow X_1 X_2 X_3 X_4 X_5 X_6$
$X_1 \rightarrow a; X_2 \rightarrow X_1 b; X_3 \rightarrow b$
$X_4 \rightarrow X_2 a; X_5 \rightarrow X_3 a; X_4 \rightarrow X_6 €$



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 Paars 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 Paars ist
- Übersetzungsvorschrift:
  1. Wenn $|\gamma| = 1 \Rightarrow (0, \gamma)$
  2. Andernfalls ist $\gamma = \alpha c$.
     α ... Expansion eines Paars mit dem Index i_α
     $\Rightarrow$ Paar: (i, c)

Beispiel

aabbababaab€

Beispiel

aabbababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

Beispiel

aabbababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

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

Beispiel

aabbababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

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

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}$ ababaab€

Beispiel

aabbabababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

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

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}$ ababaab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}$ baab€

Beispiel

aabbababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

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

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}$ ababaab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}$ baab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}\ \underbrace{(3, a)}_{ba}$ ab€

Beispiel

aabbababaab€

$\underbrace{(0, a)}_{a}$ abbababaab€

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

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}$ ababaab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}$ baab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}\ \underbrace{(3, a)}_{ba}$ ab€

$\underbrace{(0, a)}_{a}\ \underbrace{(1, b)}_{ab}\ \underbrace{(0, b)}_{b}\ \underbrace{(2, a)}_{aba}\ \underbrace{(3, a)}_{ba}\ \underbrace{(2, \text{€})}_{ab\text{€}}$

Approximationsalgorithmen - LZ78

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
  $m \in \Omega(f_l(n))$

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

Lower Bound (1/4)

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

Lower Bound (1/4)

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

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

Approximationsalgorithmen - LZ78

Lower Bound (1/4)

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

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

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

Lower Bound (1/4)

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

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

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

Lower Bound (1/4)

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

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

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

Lower Bound (2/4)

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

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

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

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

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
  $\mathcal{O}(\underbrace{1 + log(\frac{k^2+k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$
- $m^* \in \mathcal{O}(1 + log(\frac{k^2+k}{2}) + log((k+1)^2) + 1 + 1 + log(k))$

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
$$\mathcal{O}(\underbrace{1 + log(\frac{k^2 + k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$$
- $m^* \in \mathcal{O}(log(\frac{k^2+k}{2}) + log((k+1)^2) + log(k))$

Approximationsalgorithmen - LZ78

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
  $\mathcal{O}(\underbrace{1 + log(\frac{k^2+k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$
- $m^* \in \mathcal{O}(2log(k) + 2log(k) + log(k))$

Approximationsalgorithmen - LZ78

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
  $\mathcal{O}(\underbrace{1 + log(\frac{k^2+k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$
- $m^* \in \mathcal{O}(log(k))$

Approximationsalgorithmen - LZ78

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
  $\mathcal{O}(\underbrace{1 + log(\frac{k^2+k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$
- $m^* \in \mathcal{O}(log(k))$
- $m^* \in \mathcal{O}(log(n^{\frac{1}{3}}))$ $n \in \Theta(k^3)$

Approximationsalgorithmen - LZ78

Lower Bound (2/4)

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

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

- Es existiert Grammatik mit Größe: (Lemmma 1...3)
  $\mathcal{O}(\underbrace{1 + log(\frac{k^2+k}{2})}_{a^{k(k+1)/2}} + \underbrace{log((k+1)^2) + 1 + 1 + log(k)}_{(ba^k)^{(k+1)^2}})$
- $m^* \in \mathcal{O}(log(k))$
- $m^* \in \mathcal{O}(log(n^{\frac{1}{3}}))$ $n \in \Theta(k^3)$
- $m^* \in \mathcal{O}(log(n))$

Lower Bound (3/4)

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

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

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

Lower Bound (3/4)

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

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

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

Lower Bound (3/4)

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

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

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

Approximationsalgorithmen - LZ78

Lower Bound (3/4)

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

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

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

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

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

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

Approximationsalgorithmen - LZ78

Lower Bound (4/4)

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

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

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

Lower Bound (4/4)

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

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

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

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

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

Approximationsalgorithmen - LZ78

Upper bound

- m proportional zu zu Anzahl der Nichterminale

Approximationsalgorithmen - LZ78

Upper bound

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

Upper bound

- m proportional zu zu Anzahl der Nichterminale
- Nichterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichterminal und Anzahl Gruppen

Upper bound

- m proportional zu zu Anzahl der Nichterminale
- Nichterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichterminal und Anzahl Gruppen
- Abschätzung der Gruppenanzahl

Upper bound

- m proportional zu zu Anzahl der Nichterminale
- Nichterminale in Gruppen zerlegt
- Abhängigkeit Anzahl Nichterminal und Anzahl Gruppen
- Abschätzung der Gruppenanzahl
- m bestimmen

Approximationsalgorithmen - LZ78

Upper bound (1/4)

$$S \rightarrow X_1 X_2 X_3 X_4 X_5 X_6$$

$$X_1 \rightarrow a; X_2 \rightarrow X_1 b; X_3 \rightarrow b\ X_4 \rightarrow X_2 a; X_5 \rightarrow X_3 a; X_4 \rightarrow X_6 €$$

- Jede Nicht-Start-Regel enthält max. zwei Symbole

Upper bound (1/4)

$$S \rightarrow X_1X_2X_3X_4X_5X_6$$

$$X_1 \rightarrow a; X_2 \rightarrow X_1 b; X_3 \rightarrow b\ X_4 \rightarrow X_2 a; X_5 \rightarrow X_3 a; X_4 \rightarrow X_6 \in$$

- Jede Nicht-Start-Regel enthält max. zwei Symbole
- p...Anzahl der Nicht-Start-regeln

Approximationsalgorithmen - LZ78

Upper bound (1/4)

$$S \to X_1X_2X_3X_4X_5X_6$$

$$X_1 \to a; X_2 \to X_1 b; X_3 \to b\ X_4 \to X_2 a; X_5 \to X_3 a; X_4 \to X_6 \in$$

- Jede Nicht-Start-Regel enthält max. zwei Symbole
- p...Anzahl der Nicht-Start-regeln
- $m \leq 3p$

Upper bound (1/4)

$$S \rightarrow X_1 X_2 X_3 X_4 X_5 X_6$$

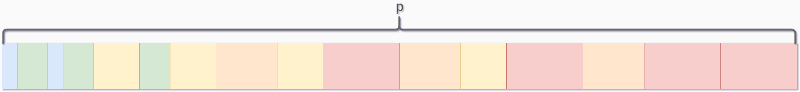
$$X_1 \rightarrow a; X_2 \rightarrow X_1 b; X_3 \rightarrow b\ X_4 \rightarrow X_2 a; X_5 \rightarrow X_3 a; X_4 \rightarrow X_6 €$$

- Jede Nicht-Start-Regel enthält max. zwei Symbole
- p...Anzahl der Nicht-Start-regeln
- $m \leq 3p$

$\Rightarrow$ $m \in \mathcal{O}(p)$

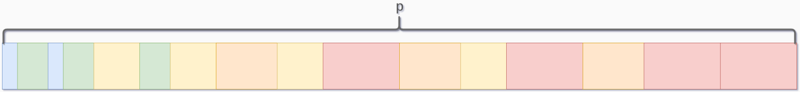
Approximationsalgorithmen - LZ78

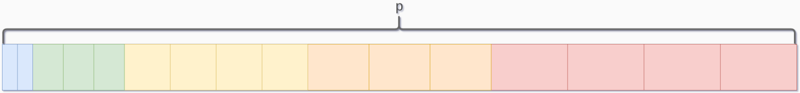
Upper bound (2/4)



Approximationsalgorithmen - LZ78

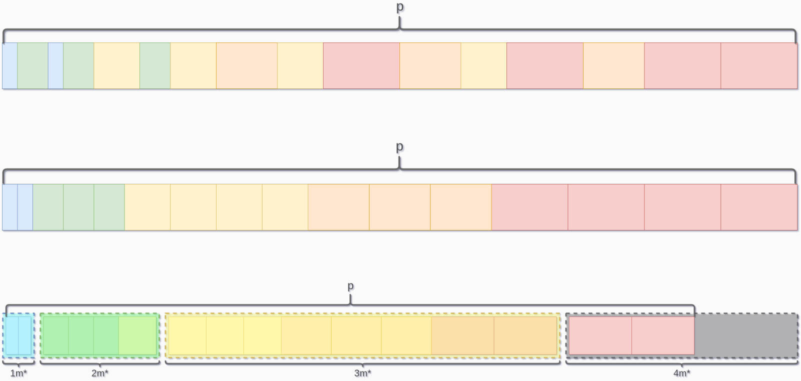
Upper bound (2/4)



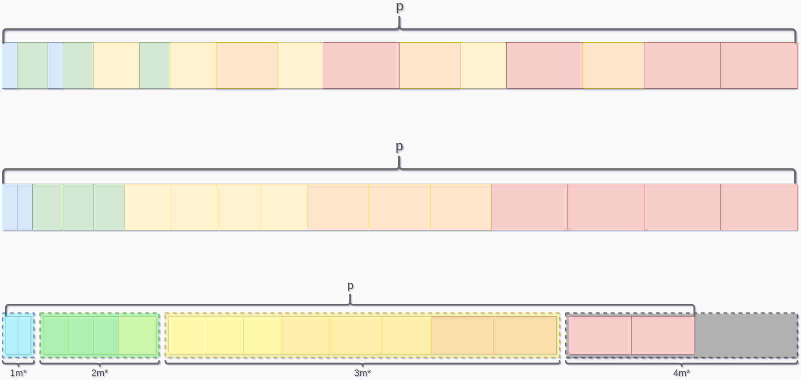


Approximationsalgorithmen - LZ78

Upper bound (2/4)



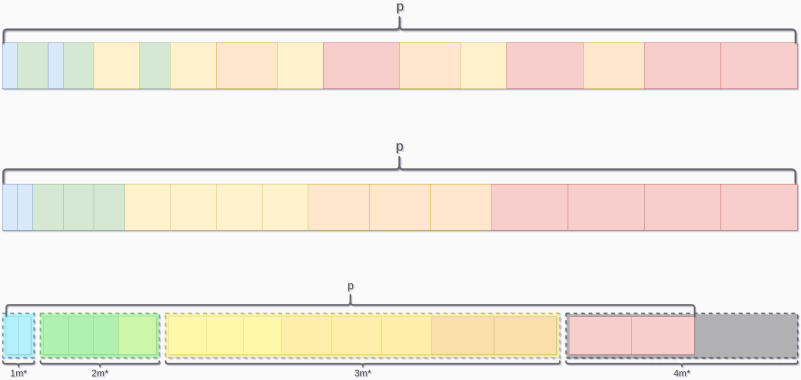
Upper bound (2/4)



$$1m^* + 2m^* + \ldots + gm^* + (g+1)m^* > p$$

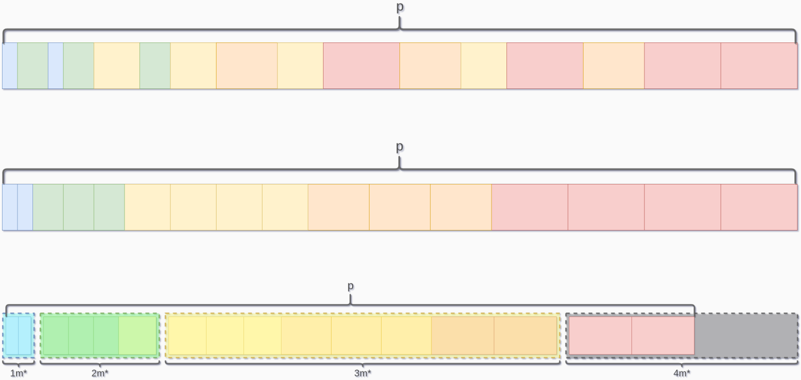
Approximationsalgorithmen - LZ78

Upper bound (2/4)



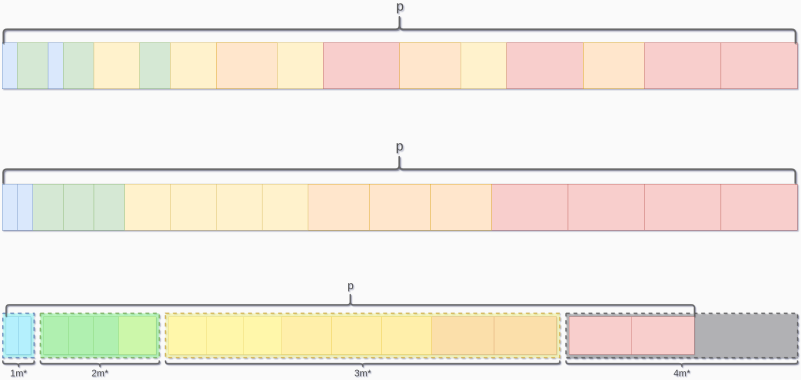
$m^* \sum_{k=1}^{g+1} k > p$

Upper bound (2/4)



$m^*(\frac{g^2}{2} + \frac{3g}{2} + 1) > p$

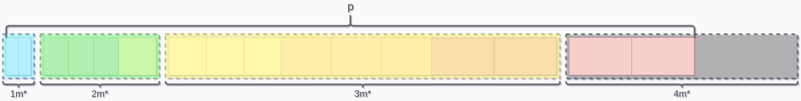
Upper bound (2/4)



$m^*(\frac{g^2}{2} + \frac{3g}{2} + 1) > p$

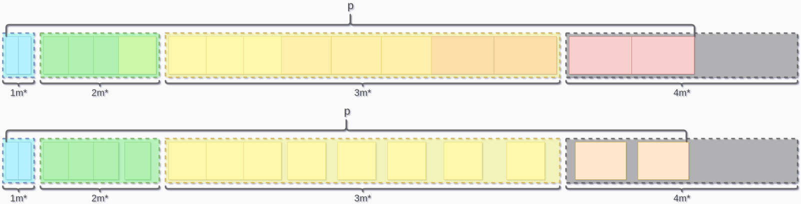
$\Rightarrow p \in \mathcal{O}(m^* g^2)$

Upper bound (3/4)

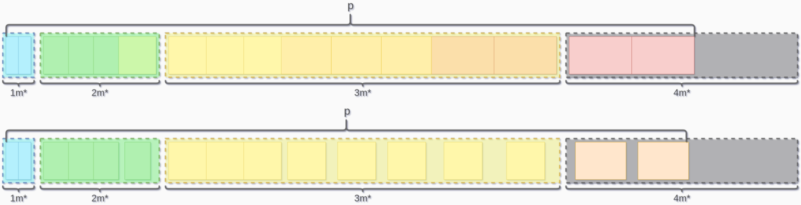


Approximationsalgorithmen - LZ78

Upper bound (3/4)

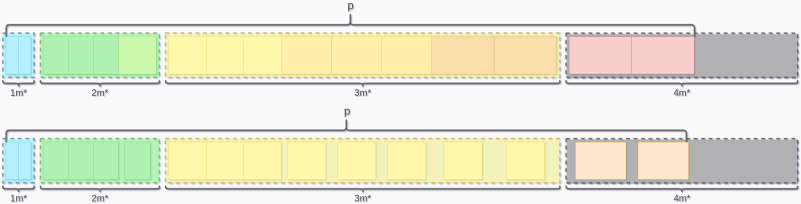


Upper bound (3/4)



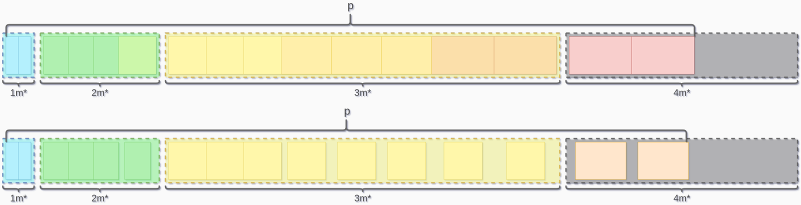
$1^2 m^* + 2^2 m^* + ... + g^2 m^* \leq n$

Upper bound (3/4)



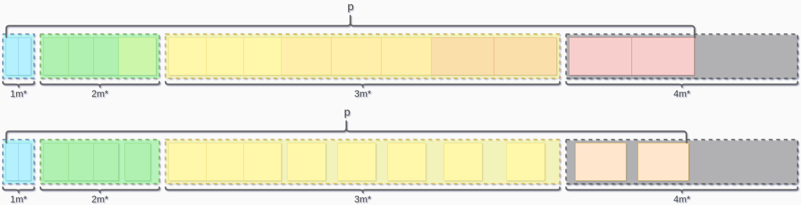
$m^* \sum_{k=1}^{g} k^2 \leq n$

Upper bound (3/4)



$m^*(\frac{g^3}{3} + \frac{g^2}{2} + \frac{g}{6}) \leq n$

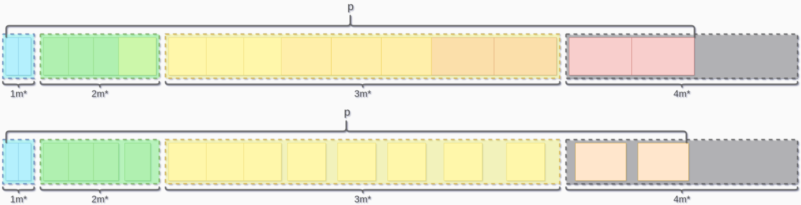
Upper bound (3/4)



$m^*(\frac{g^3}{3} + \frac{g^2}{2} + \frac{g}{6}) \leq n$

$\Rightarrow g^3 \in \mathcal{O}(\frac{n}{m^*})$

Upper bound (3/4)



$$m^*(\frac{g^3}{3} + \frac{g^2}{2} + \frac{g}{6}) \leq n$$
$$\Rightarrow g^3 \in \mathcal{O}(\frac{n}{m^*})$$
$$\Rightarrow g \in \mathcal{O}((\frac{n}{m^*})^{\frac{1}{3}})$$

Upper bound (4/4)

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

Upper bound (4/4)

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

$\Rightarrow p \in \mathcal{O}(m^*(\frac{n}{m^*})^{\frac{2}{3}})$

Upper bound (4/4)

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

$\Rightarrow p \in \mathcal{O}(m^*(\frac{n}{m^*})^{\frac{2}{3}})$
$\Rightarrow p \in \mathcal{O}(m^*(\frac{n}{log(n)})^{\frac{2}{3}})$ (mk-Lemma)

Upper bound (4/4)

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

$\Rightarrow p \in \mathcal{O}(m^* (\frac{n}{m^*})^{\frac{2}{3}})$
$\Rightarrow p \in \mathcal{O}(m^* (\frac{n}{log(n)})^{\frac{2}{3}})$ (mk-Lemma)
$\Rightarrow m \in \mathcal{O}(m^* (\frac{n}{log(n)})^{\frac{2}{3}})$

Upper bound (4/4)

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

$\Rightarrow p \in \mathcal{O}(m^*(\frac{n}{m^*})^{\frac{2}{3}})$
$\Rightarrow p \in \mathcal{O}(m^*(\frac{n}{log(n)})^{\frac{2}{3}})$ (mk-Lemma)
$\Rightarrow m \in \mathcal{O}(m^*(\frac{n}{log(n)})^{\frac{2}{3}})$
$\Rightarrow a(n) = \max_{\alpha \in \Sigma^n} \frac{m}{m^*} \in \mathcal{O}((\frac{n}{log(n)})^{\frac{2}{3}}) \quad \square$

Global Algorithms

- Klasse von Algorithmen

Global Algorithms

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

Global Algorithms

- Klasse von Algorithmen
- Haben alle ein upper bound von $\mathcal{O}((\frac{n}{log(n)})^{\frac{2}{3}})$
- 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 $S \to \alpha$
- Wähle einen String γ
- Füge $T \to \gamma$ (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

- $|\gamma| \geq 2$

Approximationsalgorithmen - Global Algorithms

Auswahl von γ

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

Auswahl von γ

- $|\gamma| \geq 2$
- γ kommst 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)
- $2m^* + 3m^* + ... + gm^*(g+1)m^* > \frac{2}{9}m$

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)
- $2m^* + 3m^* + ... + gm^*(g+1)m^* > \frac{2}{9}m$
- $m^* \sum_{k=2}^{g+1} k = m^*(\frac{g^2}{2} + \frac{3g}{2}) > \frac{2}{9}m$

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)
- $2m^* + 3m^* + ... + gm^*(g+1)m^* > \frac{2}{9}m$
- $m^* \sum_{k=2}^{g+1} k = m^*(\frac{g^2}{2} + \frac{3g}{2}) > \frac{2}{9}m$
- $m \in \mathcal{O}(g^2 m^*)$

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)
- $2^2 m^* + 3^2 m^* + ... + g^2 m^* \leq \sigma$

Upper Bound (3/4)

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

Upper Bound (3/4)

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

Upper Bound (3/4)

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

Upper Bound (3/4)

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

Upper Bound (4/4)

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

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

Approximationsalgorithmen - Global Algorithms

Greedy

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

Lower bound

Zusammenfassung

Zusammenfassung

- interessant für Kompression und Musterkennung (NLP)

Zusammenfassung

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

Zusammenfassung

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

Zusammenfassung

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