

Kompressionsverfahren: Übersicht

- Klassifikationen:
 - Universell vs. speziell (für bestimmte Informationstypen)
 - Verlustfrei vs. verlustbehaftet
 - In diesem Kapitel: nur universelle & verlustfreie Verfahren
- Im folgenden vorgestellte Verfahren:
 - Statistische Verfahren:
 - » Huffman-Codierung
 - » Arithmetische Codierung
 - Zeichenorientierte Verfahren:
 - » Lauflängencodierung (RLE Run Length Encoding)
 - » LZW-Codierung 

Wörterbuch-Kompressionen

- Grundidee:
 - Suche nach dem „Vokabular“ des Dokuments, d.h. nach sich wiederholenden Teilsequenzen
 - Erstelle Tabelle: Index <--> Teilsequenz („Wort“)
 - Tabelle wird dynamisch während der Kodierung aufgebaut
 - Codiere Original als Folge von Indizes
- Praktische Algorithmen:
 - Abraham Lempel, Jacob Ziv (Israel), Ende 70er-Jahre
 - » LZ77- und LZ78-Algorithmen
 - Verbessert 1984 von A. Welch = „LZW“-Algorithmus (Lempel/Ziv/Welch)
 - Basis vieler semantikunabhängiger Kompressionsverfahren (z.B. UNIX „compress“, Zip, gzip, V42.bis)
 - Verwendet in vielen Multimedia-Datenformaten (z.B. GIF)

“Wörterbuch” für LZW

- Abbildung von Zeichenreihen auf Zahlen (Code)
- Annahme:
 - Vorbesetzung der Tabelle mit fest vereinbarten Codes für Einzelzeichen (muß nicht explizit gespeichert und übertragen werden)

Startzustand:

Zeichenreihe	Code
“a”	97
“b”	98
“c”	99
...	
“z”	122

Während Berechnung:

Zeichenreihe	Code
“a”	97
...	
“z”	122
“ba”	256
“an”	257
...	

Für “neue” Codes Bereich verwendet, der von den vorbesetzten Codes verschieden ist (z.B. > 255)

Prinzip der LZW-Codierung

- Nicht alle Teilworte ins Wörterbuch, sondern nur eine "Kette" von Teilworten, die sich um je ein Zeichen überschneiden.
- Sequentieller Aufbau:
Neu einzutragendes Teilwort =
Kürzestes ("erstes") noch nicht eingetragenes Teilwort
- Beispiel:

b a n a n e n a n b a u

ba	an	na	ane	en	nan	nb	bau
----	----	----	-----	----	-----	----	-----

Aufbau des LZW-Wörterbuchs

Neu einzutragendes Teilwort =
Kürzestes ("erstes")
noch nicht
eingetragenes Teilwort

Gelesenes Zeichen	Bereits in Wörterbuch?	Neueinträge mit Code
b	"b": Ja	
a	"b": Ja "ba": Nein	"ba" - neuer Code (256)
n	"a": Ja "an": Nein	"an" - neuer Code (257)
a	"n": Ja "na": Nein	"na" - neuer Code (258)
n	"a": Ja "an": Ja	
e	... "ane": Nein	"ane" - neuer Code (259)

b a n a n e . . .

ba an na ane

LZW-Codierung (1)

Prinzipieller Ablauf:

SeqChar $p = \langle \text{NächstesEingabezeichen} \rangle$;

Char $k = \text{NächstesEingabezeichen}$;

Wiederhole:

Falls $p \ \& \ \langle k \rangle$ in Tabelle enthalten

dann $p = p \ \& \ \langle k \rangle$

sonst trage $p \ \& \ \langle k \rangle$ neu in Tabelle ein

(und erzeuge neuen Index dafür);

Schreibe Tabellenindex von p auf Ausgabe;

$p = \langle k \rangle$;

Ende Fallunterscheidung;

$k = \text{NächstesEingabezeichen}$;

solange bis Eingabeende

Schreibe Tabellenindex von p auf Ausgabe;

Achtung:
“alter” Puffer p ,
nicht $p \ \& \ \langle k \rangle$

Algorithmus-Beschreibung (“Pseudo-Code”)

- Variablen (ähnlich zu C/Java-Syntax):
 - Datentyp fett geschrieben, gefolgt vom Namen der Variablen
 - Zuweisung an Variable mit “=”
- Datentypen:
 - **int**: Ganze Zahlen
 - **Char**: Zeichen (Buchstaben, Zahlen, Sonderzeichen)
 - **SeqChar**: Zeichenreihen (Sequenzen von Zeichen)
 - » Einelementige Zeichenreihe aus einem Zeichen: < x >
 - » Aneinanderreihung (Konkatenation) mit &
- NächstesEingabezeichen:
 - Liefert nächstes Zeichen der Eingabe und schaltet Leseposition im Eingabepuffer um ein Zeichen weiter

LZW-Codierung (3)

Beispieltext: "bananenanbau"

Ablauf:

Wiederhole:

Falls $p \& \langle k \rangle$ in Tabelle enthalten

dann $p = p \& \langle k \rangle$

sonst trage $p \& \langle k \rangle$ neu in Tabelle ein
(und erzeuge neuen Index dafür);
Schreibe Tabellenindex von p auf Ausgabe;

$p = \langle k \rangle$;

Ende Fallunterscheidung;

$k =$ Nächstes Eingabezeichen;

solange bis Eingabeende

Lesen (k)	Codetabelle schreiben ($p \& \langle k \rangle$)	Ausgabe	Puffer füllen (p)
			$\langle b \rangle$
a	$(\langle ba \rangle, 256)$	98	$\langle a \rangle$
n	$(\langle an \rangle, 257)$	97	$\langle n \rangle$
a	$(\langle na \rangle, 258)$	110	$\langle a \rangle$
n			$\langle an \rangle$
e	$(\langle ane \rangle, 259)$	257	$\langle e \rangle$
n	$(\langle en \rangle, 260)$	101	$\langle n \rangle$
a			$\langle na \rangle$
n	$(\langle nan \rangle, 261)$	258	$\langle n \rangle$
b	$(\langle nb \rangle, 262)$	110	$\langle b \rangle$
a			$\langle ba \rangle$
u	$(\langle bau \rangle, 263)$	256	$\langle u \rangle$
EOF		117	

LZW-Decodierung bei bekannter Tabelle

Wiederhole solange Eingabe nicht leer:

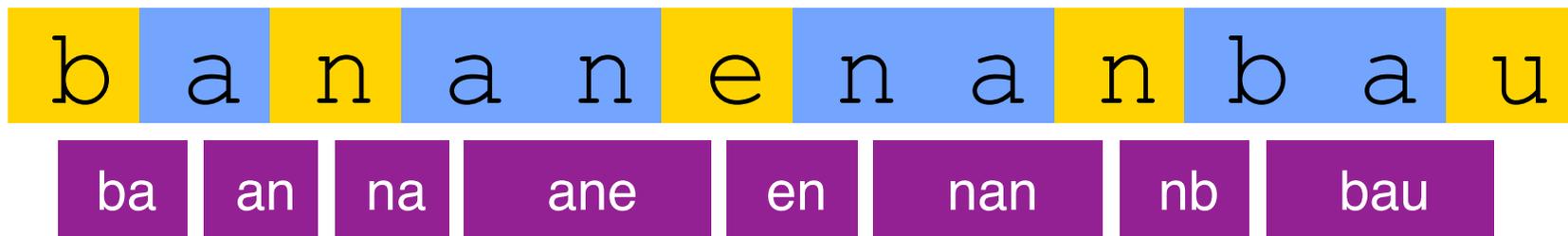
k = NächsteEingabezahl;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

Ende Wiederholung;

LZW-Decodierung (1)

- Grundidee („symmetrische Codierung“):
 - Das aufgebaute Wörterbuch muß *nicht* zum Empfänger übertragen werden.
 - Das Wörterbuch wird nach dem gleichen Prinzip wie bei der Codierung bei der Decodierung dynamisch aufgebaut.
 - Das funktioniert, weil bei der Codierung immer *zuerst* der neue Eintrag für das Wörterbuch nach bekannten Regeln aus dem schon gelesenen Text aufgebaut wird, bevor der neue Eintrag in der Ausgabe verwendet wird.
- Algorithmusidee:
 - Neu einzutragendes Teilwort = letztes Teilwort plus erstes Zeichen des aktuellen Teilworts



LZW-Decodierung (2)

Neu einzutragendes Teilwort =
letztes Teilwort plus erstes Zeichen
des aktuellen Teilworts

Prinzipieller Algorithmus:

SeqChar $p := \diamond$;

int $k = \text{NächsteEingabezahl}$;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

int $old = k$;

Wiederhole solange Eingabe nicht leer:

$k = \text{NächsteEingabezahl}$;

SeqChar $akt = \text{Zeichenreihe mit Tabellenindex } k$;

Schreibe Zeichenreihe akt auf Ausgabe;

$p = \text{Zeichenreihe mit Tabellenindex } old \text{ (letztes Teilwort)}$;

Char $q = \text{erstes Zeichen von } akt$;

Trage $p \ \& \ \langle q \rangle$ in Tabelle ein
(und erzeuge neuen Index dafür);

Wiederhole solange Eingabe nicht leer:

k = NächsteEingabezahl;

SeqChar akt = Zeichenreihe mit Tabellenindex k ;

Schreibe Zeichenreihe akt auf Ausgabe;

p = Zeichenreihe mit Tabellenindex old
(letztes Teilwort);

Char q = erstes Zeichen von akt ;

Trage p & $\langle q \rangle$ in Tabelle ein
(und erzeuge neuen Index dafür);

$old = k$;

Ende Wiederholung;

LZW-Decodierung (3)

Beispielzeichenreihe:

"98-97-110-257-101-
258-110-256-117"

Lesen (k)	Ausgabe (q ist jeweils unterstrichen)	Puffer füllen (p)	Codetabelle schreiben (p & $\langle q \rangle$)	Merken (old)
98	b			98
97	<u>a</u>	b	($\langle ba \rangle$, 256)	97
110	<u>n</u>	a	($\langle an \rangle$, 257)	110
257	<u>an</u>	n	($\langle na \rangle$, 258)	257
101	<u>e</u>	an	($\langle ane \rangle$, 259)	101
258	<u>na</u>	e	($\langle en \rangle$, 260)	258
110	<u>n</u>	na	($\langle nan \rangle$, 261)	110
256	<u>ba</u>	n	($\langle nb \rangle$, 262)	256
117	<u>u</u>	ba	($\langle bau \rangle$, 263)	117
EOF				

LZW-Decodierung (4)

- Beispielzeichenreihe: "abababa...", Beispielcode: "97-98-256-258"
- Ablauf:

Lesen (k)	Ausgabe (q ist jeweils unterstrichen)	Puffer füllen (p)	Codetabelle schreiben (p & <q>)	Merken (old)
97	a			97
98	<u>b</u>	a	(<ab>, 256)	98
256	<u>ab</u>	b	(<ba>, 257)	256
258	???			



Decodierung ist so noch nicht korrekt!

QUIZ!

- Welche der folgenden Aussagen ist richtig?
 - Es gibt Zeichenreihen, die sich nicht mit dem LZW-Verfahren codieren und decodieren lassen.
 - Man muss den Codierungs-Algorithmus von LZW, den wir bisher kennen, erweitern, um alle Zeichenreihen codieren und decodieren zu können.
 - Man muss den Decodierungs-Algorithmus von LZW, den wir bisher kennen, erweitern, um alle Zeichenreihen codieren und decodieren zu können.
 - Auch nach einer geschickten Erweiterung, um den beschriebenen Spezialfall abzudecken, könnte es noch andere Spezialfälle geben, an denen unser LZW-Algorithmus scheitert.

LZW-Decodierung, vollständige Fassung

SeqChar $p := \langle \rangle$;

int $k = \text{NächsteEingabezahl}$;

Schreibe Zeichenreihe mit Tabellenindex k auf Ausgabe;

int $old = k$;

Wiederhole solange Eingabe nicht leer:

$k = \text{NächsteEingabezahl}$;

SeqChar $akt = \text{Zeichenreihe mit Tabellenindex } k$;

$p = \text{Zeichenreihe mit Tabellenindex } old \text{ (letztes Teilwort)}$;

Falls Index k in Tabelle enthalten

dann **Char** $q = \text{erstes Zeichen von } akt$;

Schreibe Zeichenreihe akt auf Ausgabe;

sonst **Char** $q = \text{erstes Zeichen von } p$;

Schreibe Zeichenreihe $p \ \& \ \langle q \rangle$ auf Ausgabe;

Ende Fallunterscheidung;

Trage $p \ \& \ \langle q \rangle$ in Tabelle ein
(und erzeuge neuen Index dafür);

$old = k$;

Ende Wiederholung;