

# Kapitel 1      Grundlagen digitaler Medien

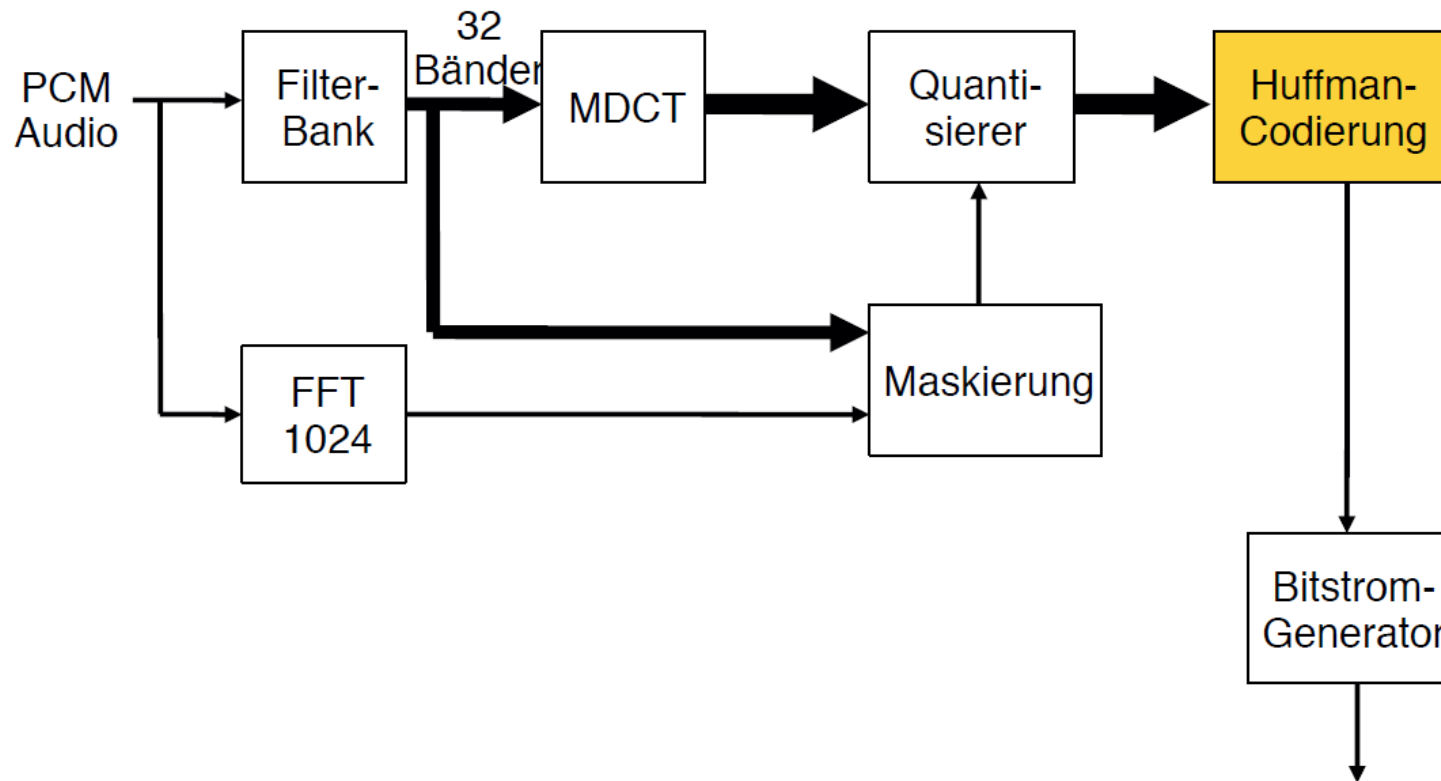
- 1.1      Medium, Medieninformatik, Multimedia
- 1.2      Digitalisierung
- 1.3      Informationstheoretische Grundlagen**
  - 1.3.1 Abtasttheorem
  - 1.3.2 Stochastische Nachrichtenquelle, Entropie, Redundanz**
- 1.4      Verlustfreie universelle Kompression

Basis:

- Andreas Butz, Heinrich Hußmann und Rainer Malaka:  
Medieninformatik: Eine Einführung. Pearson Studium, ISBN-10:  
3827373530, 2009. – Kapitel 2
- Digitale Medien (Prof. Dr. Andreas Butz, LMU München, WiSe 2011)
- Digitale Medien (Prof. Dr. Hendrik Lensch, Uni Ulm, SoSe 2011)

## Einschub: Motivation für Informationstheorie

Aufbau eines MPEG-Layer III (MP3) Encoders, Details siehe später!



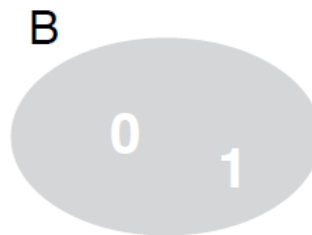
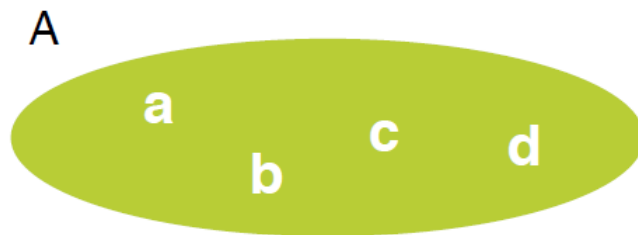
## Stochastische Informationstheorie: Zeichenvorrat und Codierung

- Ein *Zeichenvorrat* ist eine endliche Menge von *Zeichen*.
- Eine Nachricht (im Zeichenvorrat A) ist eine Sequenz von Zeichen aus A
- Seien A und B Zeichenvorräte. Eine *Codierung*  $c$  ist eine Abbildung von Nachrichten in A auf Nachrichten in B.

$$c: A \rightarrow B^*$$

( $B^*$  : Zeichenreihen über B)

- Wir beschränken uns meist auf *binäre* Codierungen, d.h.  $B = \{ 0, 1 \}$



Beispiel:

abca  $\rightarrow$  00011000

ddc  $\rightarrow$  111110

- *Informationstheorie* (nach *Shannon*) betrachtet die Häufigkeit des Auftretens bestimmter Zeichen(folgen) in den Nachrichten einer Nachrichtenquelle.

## Entropie (1)

- Annahme *Stochastische Nachrichtenquelle*: Wir kennen die Häufigkeitsverteilung der Zeichen in den Nachrichten.
- *Entscheidungsgehalt (Entropie)* der Nachrichtenquelle:
  - Wie viele Ja/Nein-Entscheidungen ( $x_a$ ) entsprechen dem Auftreten eines Einzelzeichens ( $a$ )?
  - Eine Ja/Nein-Entscheidung = 1 „bit“
- Beispiele:

Quelle 1	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	1	0	0	0
	$x_a$	0	-	-	-

Quelle 2	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	0.25	0.25	0.25	0.25
	$x_a$	2	2	2	2

$p_a$  = Häufigkeit

$x_a$  = Zahl der Entscheidungen

$$2^{x_a} = 1/p_a$$

$$x_a = \text{ld}(1/p_a)$$

(ld = Logarithmus zur Basis 2)

## Entropie (2)

Durchschnittlicher Entscheidungsgehalt je Zeichen: *Entropie*  $H$

$$H = \sum_{a \in A} p_a \lg\left(\frac{1}{p_a}\right)$$

mit  $x_a = \lg(1/p_a)$ :  $H = \sum_{a \in A} p_a x_a$

Quelle 1	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	1	0	0	0
	$x_a$	0	-	-	-

$$H = 0$$

Quelle 2	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	0.25	0.25	0.25	0.25
	$x_a$	2	2	2	2

$$H = 2$$

Quelle 3	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	0.5	0.25	0.125	0.125
	$x_a$	1	2	3	3

$$H = 1.75$$

Entropie ist Maß für „Unordnung“, „Zufälligkeit“

## Wortlängen und Redundanz

- Eine (Binär-)Codierung der Nachrichten einer stochastischen Nachrichtenquelle ergibt eine *durchschnittliche Wortlänge*  $L$ .

$$L = \sum_{a \in A} p_a |c(a)|$$

Quelle 2	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	0.25	0.25	0.25	0.25
	Code $c(a)$	00	01	10	11

$$H = 2$$

$$L = 2$$

Quelle 3	Zeichen $a$	A	B	C	D
	Häufigk. $p_a$	0.5	0.25	0.125	0.125
	Code $c(a)$	00	01	10	11

$$H = 1.75$$

$$L = 2$$

- $\text{Redundanz} = L - H$
- Redundanz ist ein Maß für die Güte der Codierung: möglichst klein!

## Optimale Codierung

- Eine Codierung ist *optimal*, wenn die Redundanz 0 ist.
- Durch geeignete Codierung sollte man die Redundanz beliebig niedrig wählen.
- Redundanz ermöglicht andererseits die Rekonstruktion fehlender Nachrichtenteile!
  - B ispi I: Natürlich Sprach
  - Beispiel: Fehlererkennende und -korrigierende Codes (z.B. Paritätsbits)

Quelle 3	Zeichen $a$	A	B	C	D	$H = 1.75$ $L = 2$
	Häufigk. $p_a$	0.5	0.25	0.125	0.125	
	Code $c(a)$	00	01	10	11	
Quelle 3	Zeichen $a$	A	B	C	D	$H = 1.75$ $L = 1.75$
	Häufigk. $p_a$	0.5	0.25	0.125	0.125	
	Code $c'(a)$	0	10	110	111	

# Kapitel 1      Grundlagen digitaler Medien

- 1.1      Medium, Medieninformatik, Multimedia
- 1.2      Digitalisierung
- 1.3      Informationstheoretische Grundlagen
  - 1.3.1 Abtasttheorem
  - 1.3.2 Stochastische Nachrichtenquelle, Entropie, Redundanz
- 1.4      Verlustfreie universelle Kompression**

Basis:

- Andreas Butz, Heinrich Hußmann und Rainer Malaka:  
Medieninformatik: Eine Einführung. Pearson Studium, ISBN-10:  
3827373530, 2009. – Kapitel 2
- Digitale Medien (Prof. Dr. Andreas Butz, LMU München, WiSe 2011)
- Digitale Medien (Prof. Dr. Hendrik Lensch, Uni Ulm, SoSe 2011)



## Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell
    - Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
  - 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

## Grundidee zur Huffman-Codierung

- Zeichen größerer Häufigkeit werden durch kürzere Codes repräsentiert  
– vgl. Morse-Code

T –	M – –	O – – –	CH – – – –
		G – – .	Ö – – – .
		K – . –	Q – – . –
	N – .	D – . .	Z – – . .
		W . – –	Y – . – –
		R . – .	C – . . .
E .	A . –	U . . –	X – . . –
		S . . .	B – . . .
			J . – – –
	I . .		P . – – .
			Ä . – . –
			L . – . .
			Ü . . – –
			F . . – .
			V . . . –
			H . . . .

<http://de.wikipedia.org/wiki/Morsecode>

- Das führt zu einem *Code variabler Wortlänge*:  
– Kein Codewort darf Anfang eines anderen sein (*Fano-Bedingung*)

# Huffman-Codierung (1)

- Gegeben: Zeichenvorrat und Häufigkeitsverteilung
- Ergebnis: Codierung (optimal, wenn alle Häufigkeiten Kehrwerte von Zweierpotenzen sind)
- Wiederholte Anwendung dieses Schritts auf die Häufigkeitstabelle:
  - Ersetze die beiden Einträge niedrigster Häufigkeit durch einen Codebaum mit zwei Ästen „0“ und „1“ und trage die Summe der Häufigkeiten als Häufigkeit dafür ein.


Zeichen	A	B	C	D
Häufigkeit	0.5	0.25	0.125	0.125

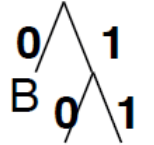
  

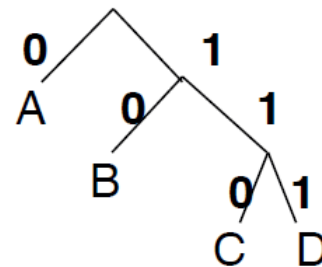
			0	1
			↙	↘
Zeichen	A	B	C	D
Häufigkeit	0.5	0.25	0.25	

## Huffman-Codierung (2)

Ersetze die beiden Einträge niedrigster Häufigkeit durch einen Codebaum mit zwei Ästen „0“ und „1“ und trage die Summe der Häufigkeiten als Häufigkeit dafür ein.

Zeichen	A	B	
Häufigkeit	0.5	0.25	0.25

Zeichen	A	
Häufigkeit	0.5	0.5



Resultierender  
Codebaum

## Huffman-Codierung (3)

- Eine Nachricht, die sich an die gegebene Häufigkeitsverteilung hält:  
ababacadaabacdba (Länge = 16 Zeichen)
- Codierung mit festen Wortlängen  
(z.B. a = 00, b = 01, c = 10, d = 11)  
Länge 32 bit
- Huffman-Codierung  
(a = 0, b = 10, c = 110, d = 111)  
0100100110011100100110111100  
  
Länge 28 bit (d.h. ca. 12.5% Reduktion)

## Experiment: Huffman-Kompression von Bildern

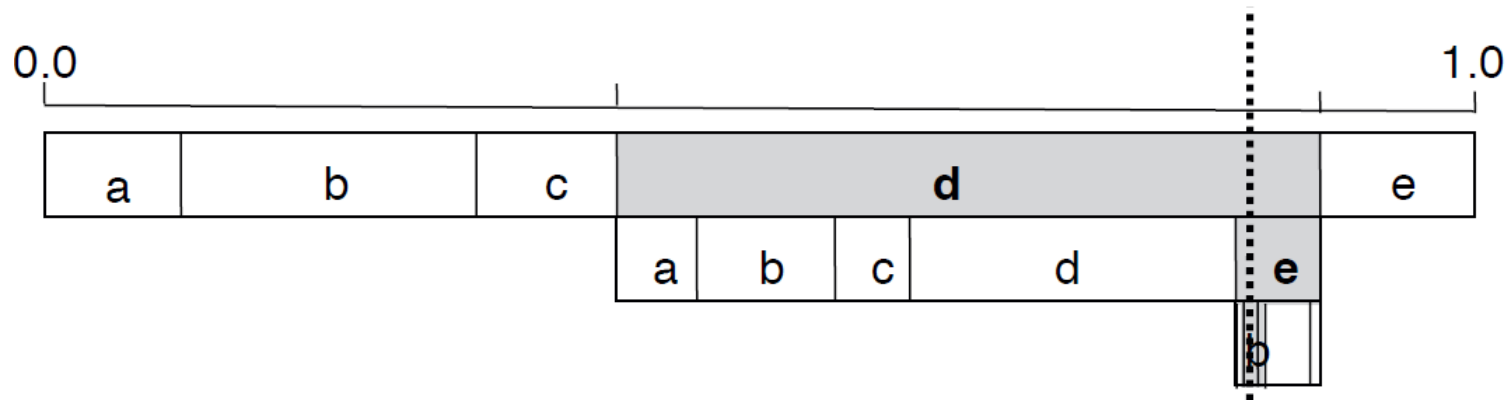
- Grautonbild, 256 x 256 Pixel, 8 bit (d.h. 256 Graustufen)
- Unkomprimiert: 65.536 Bytes
- Mit Huffman kodiert: 40.543 Bytes (ca. 38% Reduktion)
- Einfacher "Zusatztrick":
  - *Differenz* zwischen benachbarten Pixeln speichern und Huffman dann anwenden  
33.880 Bytes (ca. 51% Reduktion)
  - Keine universelle Kompression mehr, sondern speziell für Pixelbilder
  - Solche "semantischen Kodierungen" siehe später!

## Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell
    - Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
  - 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

## Arithmetische Codierung (1)

- Gegeben: Zeichenvorrat und Häufigkeitsverteilung
- Ziel: Bessere Eignung für Häufigkeiten, die keine Kehrwerte von Zweierpotenzen sind
- Patentiertes Verfahren; (war nur mit Lizenz verwendbar, Patente ausgelaufen)
- Grundidee:
  - Code = Gleitkommazahl berechnet aus den Zeichenhäufigkeiten
  - Jedes Eingabezeichen bestimmt ein Teilintervall





## Arithmetische Codierung (2)

Beispiel:

Zeichenindex i	1=Leerz.	2=l	3=M	4=S	5=W
Häufigkeit $p_i$	0.1	0.2	0.1	0.5	0.1
linker Rand $L_i$	0.0	0.1	0.3	0.4	0.9
rechter Rand $R_i$	0.1	0.3	0.4	0.9	1.0

Allgemein:

$$L_i = \sum_{j=1}^{i-1} p_j \quad R_i = \sum_{j=1}^i p_j$$

Algorithmus:

**real** L = 0.0; **real** R = 1.0;

**Solange** Zeichen vorhanden **wiederhole**

Lies Zeichen und bestimme Zeichenindex i;

**real** B = (R-L);

R = L + B \*  $R_i$ ;

L = L + B \*  $L_i$ ;

**Ende Wiederholung;**

*Code des Textes ist Zahl im Intervall (L, R]*

Algorithmus in  
"Pseudocode":

"**real**" Datentyp  
(Gleitkommazahl)

"=" Zuweisung an  
Variable

# Arithmetische Codierung (3)

## Beispieltext-Codierung ("SWISS\_MISS")

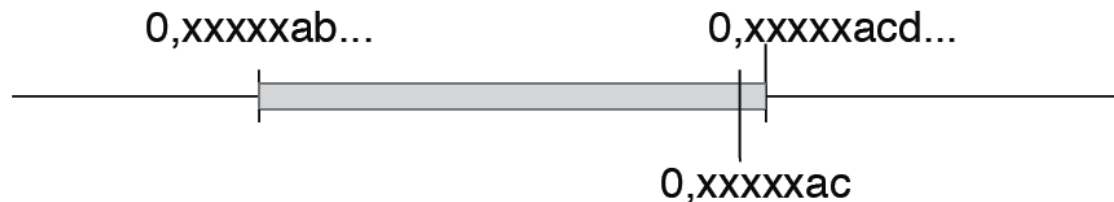
Z	Index	Li	Ri	B	L	R
					0	1
S	4	0,4	0,9	1	0,4	0,9
W	5	0,9	1	0,5	0,85	0,9
I	2	0,1	0,3	0,05	0,855	0,865
S	4	0,4	0,9	0,01	0,859	0,864
S	4	0,4	0,9	0,005	0,861	0,8635
_	1	0	0,1	0,0025	0,861	0,86125
M	3	0,3	0,4	0,0002500	0,86107500	0,86110000
I	2	0,1	0,3	0,0000250	0,86107750	0,86108250
S	4	0,4	0,9	0,0000050	0,86107950	0,86108200
S	4	0,4	0,9	0,0000025	0,86108050	0,86108175

```
real L = 0.0; real R = 1.0;
Solange Zeichen vorhanden wiederhole
    Lies Zeichen und bestimme Zeichenindex i;
    real B = (R-L);
    R = L + B*Ri;
    L = L + B*Li;
Ende Wiederholung;
```

Zeichenindex i	1=Leerz.	2=I	3=M	4=S	5=W
Häufigkeit p <sub>i</sub>	0.1	0.2	0.1	0.5	0.1
linker Rand L <sub>i</sub>	0.0	0.1	0.3	0.4	0.9
rechter Rand R <sub>i</sub>	0.1	0.3	0.4	0.9	1.0

## Arithmetische Kodierung (4)

- Welcher Binärcode:
  - Ober- und Untergrenze binär codieren
  - Code = Oberer Wert, abgebrochen nach der ersten Stelle, die verschieden vom unteren Wert ist



- Komprimierung ("SWISS\_MISS")
  - "861081" = 20 Bit (Arithmetische Kodierung)
  - 10 Zeichen \* 3 Bit = 30 Bit

## Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell
    - Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
  - 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

## Lauf längencodierung

- Unkomprimierte Repräsentationen von Information enthalten häufig Wiederholungen desselben Zeichens (z.B. lange Folgen von x00- oder xFF-Bytes)
- Idee: Ersetzen einer Folge gleicher Zeichen durch 1 Zeichen + Zähler
- Eingesetzt z.B. in Fax-Standards
- Beispiel:  
aaaabcdeeeffggggghiabttttiikkddde (31 Zeichen)  
ersetzt durch  
#a4bcd#e3f#g4hiab#t3#i2#k3#d3e (30 Zeichen)
- Probleme:
  - Bei geringer Häufigkeit von Wiederholungen ineffektiv (verschlechternd)
  - Syntaktische Trennung von Wiederholungsindikatoren und unverändertem Code

## Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell
    - Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
  - 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**

## LZW-Decodierung

- Grundidee („symmetrische Codierung“):
  - 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
  - Tabelle mit wird beim Empfänger aufgebaut (nicht übertragen)
    - Vorbesetzung der Tabelle mit fest vereinbarten Codes für Einzelzeichen (muss nicht explizit gespeichert und übertragen werden)
- Grundkonzept
  - 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.

## Wörterbuch-Kompressionen

- 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)



## 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
----	----	----	-----	----	-----	----	-----

- Codierung:

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

Neu ins Wörterbuch einzutragen, codiert nach altem Wb.-Zustand

# LZW-Codierung (Algorithmus, Pseudo Code)

**SeqChar**  $p$  = NächstesEingabezeichen; **Char**  $k$  = NächstesEingabezeichen;

**Wiederhole:**

**Falls**  $p$  &  $k$  in Tabelle enthalten

**dann**  $p = p \& k$

**sonst**     trage  $p$  &  $k$  neu in Tabelle ein & erzeuge neuen Index dafür;  
                Schreibe Tabellenindex von  $p$  auf Ausgabe;  
                 $p = k$ ;

**Ende Fallunterscheidung;**

$k$  = NächstesEingabezeichen;

**solange bis** Eingabeende

Schreibe Tabellenindex von  $p$  auf Ausgabe;

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

## LZW Codierung

- Vorbesetzte Tabelle (z.B. mit ASCII-Codes):

[(**<a>**, 97), (**<b>**, 98), (**<c>**, 99), (**<d>**, 100), (**<e>**, 101), (**<f>**, 102), (**<g>**, 103), (**<h>**, 104), (**<i>**, 105), (**<j>**, 106), (**<k>**, 107), (**<l>**, 108), (**<m>**, 109), (**<n>**, 110), (**<o>**, 111), (**<p>**, 112), (**<q>**, 113), (**<r>**, 114), (**<s>**, 115), (**<t>**, 116), (**<u>**, 117), (**<v>**, 118), (**<w>**, 119), (**<x>**, 120), (**<y>**, 121), (**<z>**, 122)]

- Für neue Einträge z.B. Nummern von 256 aufwärts verwendet.

# Beispieltext: "bananenanbau"

## LZW-Codierung

**SeqChar**  $p$  = < NächstesEingabezeichen >; **Char**  $k$  = NächstesEingabezeichen;

**Wiederhole:**

**Falls**  $p$  &  $k$  in Tabelle enthalten

**dann**  $p = p \& k$

**sonst** trage  $p$  &  $k$  neu in Tabelle ein & erzeuge neuen Index dafür;

Schreibe Tabellenindex von  $p$  auf Ausgabe;

$p = k$ ;

**Ende Fallunterscheidung;**

$k$  = NächstesEingabezeichen;

**solange bis** Eingabeende

Schreibe Tabellenindex von  $p$  auf Ausgabe;

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	

Ausschnitt aus  
ACII Tabelle

$[(\langle a \rangle, 97),$   
 $(\langle b \rangle, 98), \dots,$   
 $(\langle e \rangle, 101), \dots,$   
 $(\langle n \rangle, 110), \dots,$   
 $(\langle u \rangle, 117), \dots]$

## Komprimierung

Bananenanbau = 12 Zeichen ASCII Code (256 Einträge, 8 Bit) = 96 Bit

LZW: 98, 97, 110, 257, 101, 258, 110, 256, 117  
= 9 Zeichen a 9 Bit = 81 Bit (16% Reduktion)

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

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;

$old = k$ ;

**Ende Wiederholung**;

Nachricht:

“98-97-110-257-101-  
258-110-256-117”

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

Ausschnitt aus  
ACII Tabelle

[(<a>, 97),  
(<b>, 98), .. ,  
(<e>, 101), ...,  
(<n>, 110), ...,  
(<u>, 117), ...]

# Beispieltext: "abababa"

## LZW-Codierung

**SeqChar**  $p$  = < NächstesEingabezeichen >; **Char**  $k$  = NächstesEingabezeichen;

**Wiederhole:**

**Falls**  $p$  &  $k$  in Tabelle enthalten

**dann**  $p = p \& k$

**sonst** trage  $p$  &  $k$  neu in Tabelle ein & erzeuge neuen Index dafür;

Schreibe Tabellenindex von  $p$  auf Ausgabe;

$p = k$ ;

**Ende Fallunterscheidung;**

$k$  = NächstesEingabezeichen;

**solange bis** Eingabeende

Schreibe Tabellenindex von  $p$  auf Ausgabe;

Lesen ( $k$ )	Codetabelle schreiben ( $p \& \langle k \rangle$ )	Ausgabe	Puffer füllen ( $p$ )
			$\langle a \rangle$
b	( $\langle ab \rangle$ , 256)	97	$\langle b \rangle$
a	( $\langle ba \rangle$ , 257)	98	$\langle a \rangle$
b			$\langle ab \rangle$
a	( $\langle aba \rangle$ , 258)	256	$\langle a \rangle$
b			$\langle ab \rangle$
a			$\langle aba \rangle$
EOF		258	

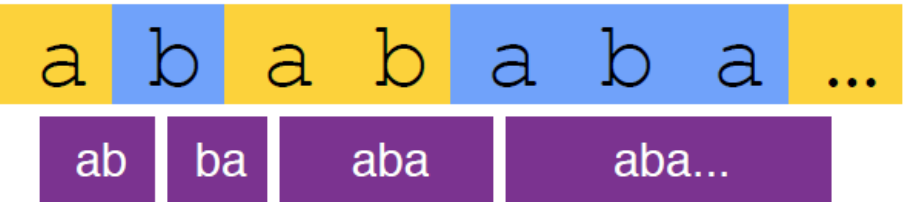
Ausschnitt aus  
ACII Tabelle

$[(\langle a \rangle, 97),$   
 $(\langle b \rangle, 98)]$

# LZW-Decodierung

- 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>a</u> b	b	(<ba>, 257)	256
258	???			



Decodierung  
ist so noch nicht  
korrekt!



## 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  $p \ \& \ q$  auf Ausgabe;  
    Ende Fallunterscheidung;  
    Trage  $p \ \& \ q$  in Tabelle ein und erzeuge neuen Index dafür;  
     $old = k$ ;  
Ende Wiederholung;
```

## Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell
    - Speziell für bestimmte technische Medien (Bild, Ton, Bewegtbild)
  - 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