2. Digitale Codierung und Übertragung

2.1 Informationstheoretische Grundlagen
2.2 Speicherbedarf und Kompression
2.3 Digitalisierung

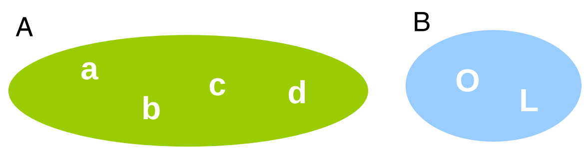
Speicherbedarf multimedialer Information

- Bsp. Schrift
  - Laufschrift (8 bit/Zeichen, 40 Zeichen/s): 320 Bit/s
- Bsp. Audio-Signale
  - Sprachsignal niedriger Qualität (Mono, 8 bit, 11 kHz): 88 kBit/s
  - CD-Qualität (Stereo, 16 bit, 44,1 kHz): 1,4 MBit/s
- Bsp. Bilder (9x13cm = 1062x1536 Pixel)
  - Schwarz/weiß (1 bit Farbtiefe): 200 kByte
  - TrueColor (24 bit Farbtiefe): 4,9 MByte
- Bsp. Video (ohne Ton)
  - 720 x 525 Pixel, 25 Bilder/s, 16 bit Farbtiefe: 151,2 MBit/s
  - 1280 x 720 Pixel, 60 Bilder/s, 24 bit Farbtiefe: 1,32 GBit/s
- ***Kompression*** der Information ist extrem wichtig!

Pixel= Bildpunkt

Zeichenvorräte 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* ist eine Abbildung von Nachrichten in A auf Nachrichten in B.
- Wir beschränken uns meist auf *binäre* Codierungen, d.h. B = { 0, L }
- Die Informationstheorie (nach *Shannon*) befaßt sich mit Nachrichtenquellen aus rein *stochastischer* Sicht
  – Die Interpretation der Nachrichten ist hier kein Thema!



Beispiel:

abca $\rightarrow$ 000LL000

ddc $\rightarrow$ LLLLLO

Entropie (1)

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

Quelle 1

Zeichen	a	b	c	d
Häufigkeit	1	0	0	0
Entschdg.	0	-	-	-

Quelle 2

Zeichen	a	b	c	d
Häufigkeit	0.25	0.25	0.25	0.25
Entschdg.	2	2	2	2

p = Häufigkeit

x = Zahl der Entscheidungen

$2^x = 1/p$

$x = \mathrm{ld}\,(1/p)$

(Logarithmus zur Basis 2)

Entropie (2)

- *Durchschnittlicher* Entscheidungsgehalt je Zeichen: *Entropie H*

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

Quelle 1

Zeichen	a	b	c	d
Häufigkeit	1	0	0	0
Entschdg.	0	-	-	-

H = 0

Quelle 2

Zeichen	a	b	c	d
Häufigkeit	0.25	0.25	0.25	0.25
Entschdg.	2	2	2	2

H = 2

Quelle 3

Zeichen	a	b	c	d
Häufigkeit	0.5	0.25	0.125	0.125
Entschdg.	1	2	3	3

H = 1.75

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	b	c	d
Häufigkeit	0.25	0.25	0.25	0.25
Code	00	0L	L0	LL

H = 2
L = 2

Quelle 3

Zeichen	a	b	c	d
Häufigkeit	0.5	0.25	0.125	0.125
Code	00	0L	L0	LL

H = 1.75
L = 2

- *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 (z.B. Wortcodierung statt Einzelzeichencodierung) kann man die Redundanz beliebig niedrig wählen.
- Redundanz ermöglicht andererseits die Rekonstruktion fehlender Nachrichtenteile!
  - B ispi l: Natürlich Sprach
  - Beispiel: Fehlererkennende und -korrigierende Codes (z.B. Paritätsbits)

Quelle 3

Zeichen	a	b	c	d
Häufigkeit	0.5	0.25	0.125	0.125
Code 3A	00	0L	L0	LL

H = 1.75
L = 2

Quelle 3

Zeichen	a	b	c	d
Häufigkeit	0.5	0.25	0.125	0.125
Code 3B	0	L0	LL0	LLL

H = 1.75
L = 1.75

2. Digitale Codierung und Übertragung

Weiterführende Literatur zum Thema Datenkompression:

Khalid Sayood: Introduction to Data Compression, 2nd. ed., Morgan Kaufmann 2000

Kompressionsverfahren: Übersicht

- Klassifikationen:
  - Universell vs. speziell (für bestimmte Informationstypen)
  - Nicht verlustbehaftet vs. verlustbehaftet
  - In diesem Kapitel: nur universelle & nicht verlustbehaftete 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
- Das führt zu einem Code variabler Wortlänge:
  - Kein Codewort darf Anfang eines anderen sein *(Fano-Bedingung)*
- In optimalem Code müssen die beiden Symbole der niedrigsten Häufigkeit mit gleicher Länge codiert sein.

  "Beweis"-Skizze:
  - Wären die Längen verschieden, könnte man das längere Wort bei der Länge des kürzeren abschneiden
    - » Dann sind die beiden Codes verschieden (sonst wäre Fano-Bedingung vorher verletzt gewesen)
    - » Kein anderes Codewort kann länger sein (da Zeichen niedrigster Wahrscheinlichkeit), also kann die Kürzung nicht die Fano-Bedingung verletzen
  - Dann hätten wir einen neuen Code mit kleinerer durchschnittlicher Wortlänge!

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 „L“ und trage die Summe der Häufigkeiten als Häufigkeit dafür ein.

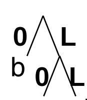
Zeichen	a	b	c	d
Häufigkeit	0.5	0.25	0.125	0.125

Zeichen	a	b	c d
Häufigkeit	0.5	0.25	0.25

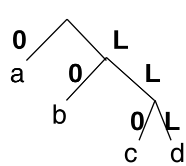
David Huffman 1951

Huffman-Codierung (2)

Zeichen	a	b	c d
Häufigkeit	0.5	0.25	0.25



Zeichen	a	c d
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 = 0L, c = L0, d = LL)
  Länge 32 bit

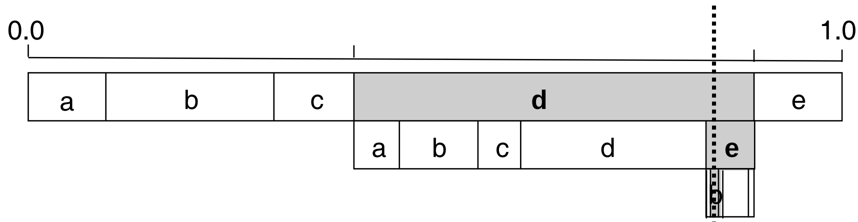
- Huffman-Codierung
  (a = 0, b = L0, c = LL0, d = LLL)
  0L00L00LL00LLL00b0LL0LLLL00
  Länge 27 bit (d.h. ca. 16% Reduktion)

Experiment: Huffman-Kompression von Bildern

- Grautonbild, 256 x 256 Pixel, 8 bit (d.h. 265 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
    - » Solche "semantischen Kodierungen" siehe später!

Arithmetische Codierung (1)

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



Arithmetische Codierung (2)

- Beispiel:

Zeichen i	Leerzeichen	I	M	S	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: $L_i = \sum_{j=0}^{i} p_j \quad R_i = \sum_{j=0}^{i+1} p_j$

- Algorithmus:

  real L = 0.0; **real** R = 1.0;
  while Zeichen vorhanden **do**
  Lies Zeichen und bestimme Zeichenindex i;
  real B = (R–L);
  R := L + B*R_i;
  L := L + B*L_i;
  enddo;
  Code des Textes ist Zahl im Intervall [L, R)

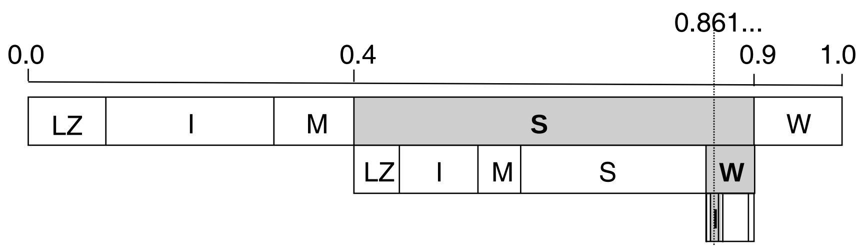
Arithmetische Codierung (3)

- Beispieltext-Codierung ("SWISS_MISS"):

Zeichen	L	R
S	0,4	0,9
W	0,85	0,9
I	0,855	0,865
S	0,859	0,864
S	0,861	0,8635
Leerz.	0,861	0,86125
M	0,861075	0,86110
I	0,8610775	0,8610782
S	0,86107778	0,86107813
S	0,86107792	0,861078095

Arithmetische Kodierung (4)

- Problem Gleitkomma-Arithmetik:
  - Konversion in Ganzzahl-Bereich durch "Skalieren"
- Welcher Binärcode:
  - Ober- und Untergrenze binär codieren
  - Code = Unterer Wert, abgebrochen an der ersten Stelle, die verschieden vom oberen Wert ist
- Veranschaulichung:



Lauflängencodierung

- 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:
  aaaabcdeeefggghiabtttiikkkddde
  ersetzt durch
  #a4bcd#e3f#g4hiab#t3#i2#k3#d3e

- Probleme:
  – Bei geringer Häufigkeit von Wiederholungen ineffektiv (verschlechternd)
  – Syntaktische Trennung von Wiederholungsindikatoren und unverändertem Code

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)

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 (1)

- Tabelle (tab) mit Abbildung Zeichenreihe -> Indizes
  - Vorbesetzung z.B. ASCII-Zeichen -> ASCII-Code
    (muß nicht explizit gespeichert und übertragen werden)
- Prinzipieller Ablauf:

```
SeqChar p := < input.getChar() >;
repeat
        Char k := input.getChar();
        if tab.contains(p & < k >)
                then  p := p & < k >
                else  int c = tab.newCode(p & <k>);
                      tab.put(p & < k >, c);
                      output.write(tab.get(p))
                      p := < k >;
        endif
until k = EOF
```

LZW-Codierung (2)

- Vorbesetzte Tabelle (z.B.):
  [(<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 255 aufwärts verwendet.

LZW-Codierung (3)

- Beispieltext: “bananenanbau"
- Ablauf:

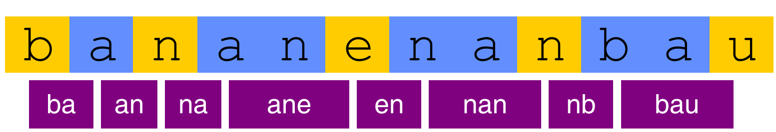
p	k	c	neues Paar für tab	output
<b>	a	256	(<ba>, 256)	98
<a>	n	257	(<an>, 257)	97
<n>	a	258	(<na>, 258)	110
<a>	n			
<an>	e	259	(<ane>, 259)	257
<e>	n	260	(<en>, 260)	101
<n>	a			
<na>	n	261	(<nan>, 261)	258
<n>	b	262	(<nb>, 262)	110
<b>	a			
<ba>	u	263	(<bau>, 263)	256
<u>	**EOF**	264	(<u**EOF**>, 264)	117
<**EOF**>				

Kompression durch LZW

- Am Beispiel:
  - 9 (16-Bit-)Worte statt 12 (16-Bit-)Worte, d.h. 25%
- In realen Situationen werden oft ca. 50% erreicht.
- Verfeinerungen des Algorithmus (z.B. Unix "compress"):
  - Obergrenze für Tabellengröße, dann statisch
  - Laufendes Beobachten der Kompressionsrate und ggf. Neustart

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)

- Prinzipieller Algorithmus:

```
k := input.getCode();
SeqChar old := tab.get(k);
output.write(old);
SeqChar p := <>;
k := input.getCode();
while k ≠ EOF do
    p := tab.get(k);
    output.write(p);
    SeqChar q := old & < firstChar(p) >;
    int c = tab.newCode(q); tab.put(q, c);
    old := p;
    k := input.getCode();
enddo;
```

Hinweis: Hier ist ein Spezialfall nicht berücksichtigt, in dem der Eintrag nicht in der Tabelle vorhanden sein kann. Vollständiger Algorithmus sh. z.B. Henning-Buch.

LZW-Decodierung (3)

- Beispielcode: “98-97-110-257-101-258-110-256-117"
- Ablauf:

k	old	p	q	neues Paar für tab	output
98	<b>	<>			b
97	<b>	<a>	<ba>	(<ba>, 256)	a
110	<a>	<n>	<an>	(<an>, 257)	n
257	<n>	<an>	<na>	(<na>, 258)	an
101	<an>	<e>	<ane>	(<ane>, 259)	e
258	<e>	<na>	<en>	(<en>, 260)	na
110	<na>	<n>	<nan>	(<nan>, 261)	n
256	<n>	<ba>	<nb>	(<nb>, 262)	ba
117	<ba>	<u>	<bau>	(<bau>, 263)	u
EOF	<u>				