Digitale Medien

Übung

Übungsbetrieb

- Informationen zu den Übungen: http://www.medien.ifi.lmu.de/dm

- Zwei Stunden pro Woche
- Praktische Anwendungen des theoretischen Vorlesungsstoffs
- Wichtige Voraussetzung für die Klausur

Überblick

1. Kompression
2. LZW-Kompression, Digitalisierung
3. Einführung in HTML
4. HTML/CSS
5. Signalverarbeitung
6. Audio, MIDI
7. Rasterbilder, verlustfreie Komprimierung
8. Rasterbilder, verlustbehaftete K.
9. XML
10. Javascript/DOM
11. SVG
12. Flash

1 Bit		*1 / 0, Strom an / Strom aus, ja / nein*
1 Byte	= 8 Bit	*Ein Buchstabe*
1 Kilobyte	= 1.000 Byte	*1.000 Buchstaben, 3 Paragraphen Text*
1 Megabyte	= 1.000 Kilobyte	*Ein Foto, eine Minute Audio (MP3)*
1 Gigabyte	= 1.000 Megabyte	*2 Stunden komprimierter Film*
1 Terabyte	= 1.000 Gigabyte	*3 Monate komprimierter Film*
1 Petabyte	= 1.000 Terabyte	*230 Jahre komprimierter Film*

Umrechnung:

Bit -> **Byte**: durch 8 teilen
Byte -> **Bit**: mit 8 multiplizieren

Byte -> **Kilobyte**: durch 1.000 teilen
Kilobyte -> **Byte**: mit 1.000 multiplizieren

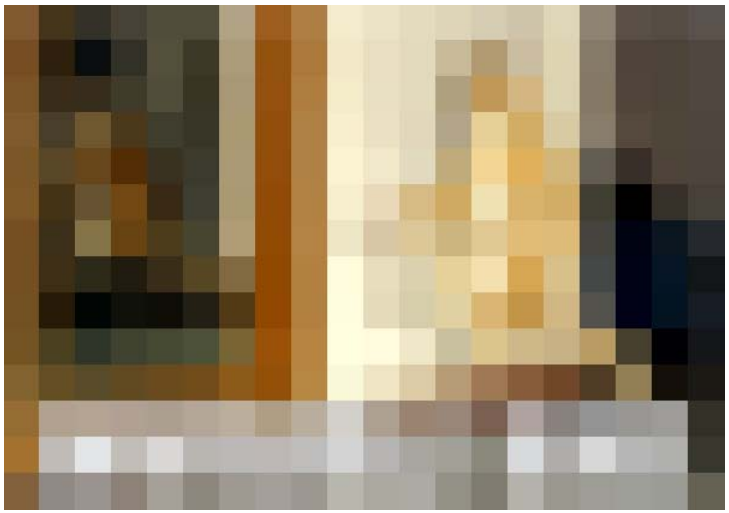
ASCII Code 1 Byte / Zeichen

Nonsense

Dezimal: 78 111 110 115 101 110 115 101
N o n s e n s e

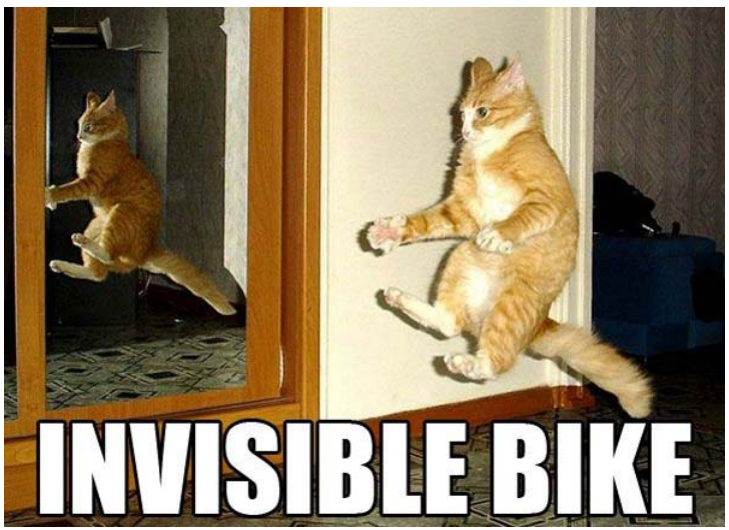
Binär:
01001110	N
01101111	o
01101110	n
01110011	s
01100101	e
01101110	n
01110011	s
01100101	e

8 Byte



20 x 14 Pixel (= Bildpunkte)
16 Bit Farben (= 65.536 verschiedene Farben)

560 Byte

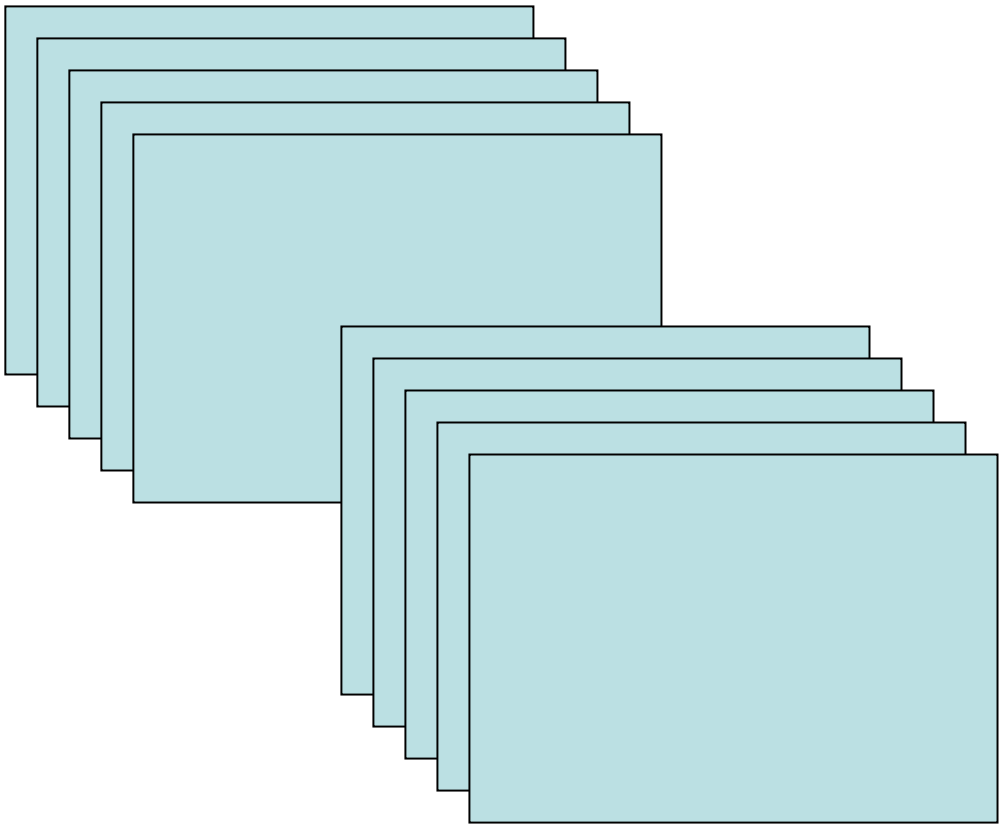


600 x 432 Pixel
16 Bit Farben

518 Kilobyte

6 Megapixel Camera:
2848 x 2136
32 Bit

24 Megabyte

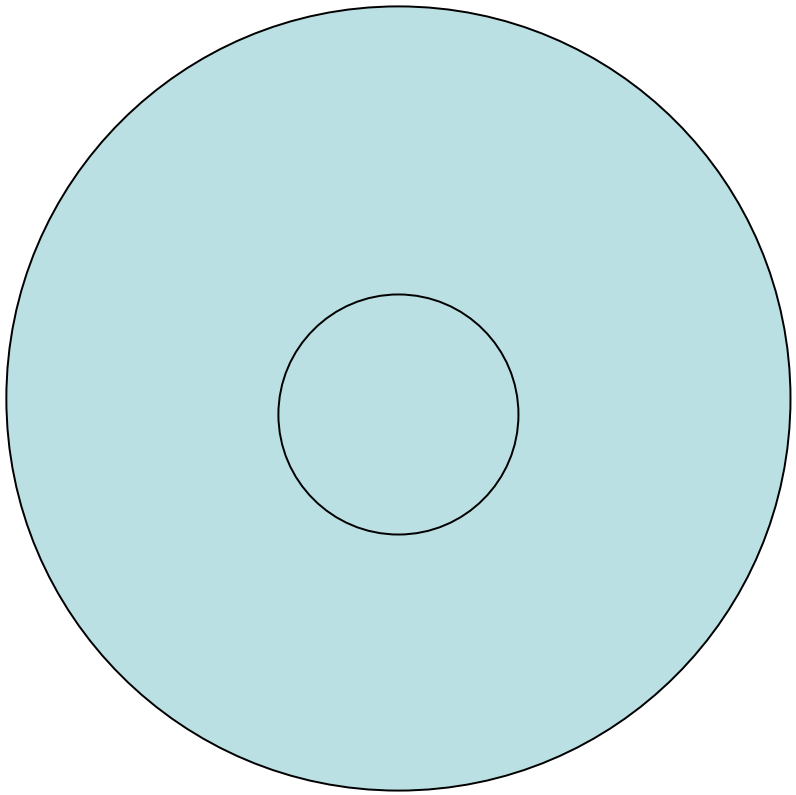


PAL Video

768 x 576 Pixel
24 Bit Farbe

25 Bilder / Sekunde
3 Minuten Film

6 Gigabyte



DVD Spielfilm (ohne Ton!)

720 x 576 Pixel
24 Bit Farbe

25 Bilder / Sekunde
100 Minuten Film

187 Gigabyte

Lauflängencodierung (RLE)

AAAAAAAAAAAAAAAAAAAA
20 Zeichen

#A20
4 Zeichen

Dies ist ein Beispieltext.
26 Zeichen

Dies ist ein Beispieltext.
26 Zeichen

$\Rightarrow$ Funktioniert besser mit Bildern als mit Text

aaaaaabbbcde

ASCII:
aaaaaabbbcde — 96 Bit

Morse-Code (2 Bit / Symbol):
.- .- .- .- .- .- -... -... -... -.-. -.. . — 86 Bit

Lauflängenkodierung:
#a6#b3cde — 72 Bit

Huffmankodierung:
111111010101001000100000 — 23 Bit

Arithmetische Kodierung:
000000101010110100111 — 21 Bit

Jede Nachricht hat einen Informationsgehalt,
die *Entropie*.

Generell: Die Entropie gibt an, wie „überraschend“ es ist, in der Nachricht ein bestimmtes Zeichen anzutreffen.

AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA

p(A) = 1

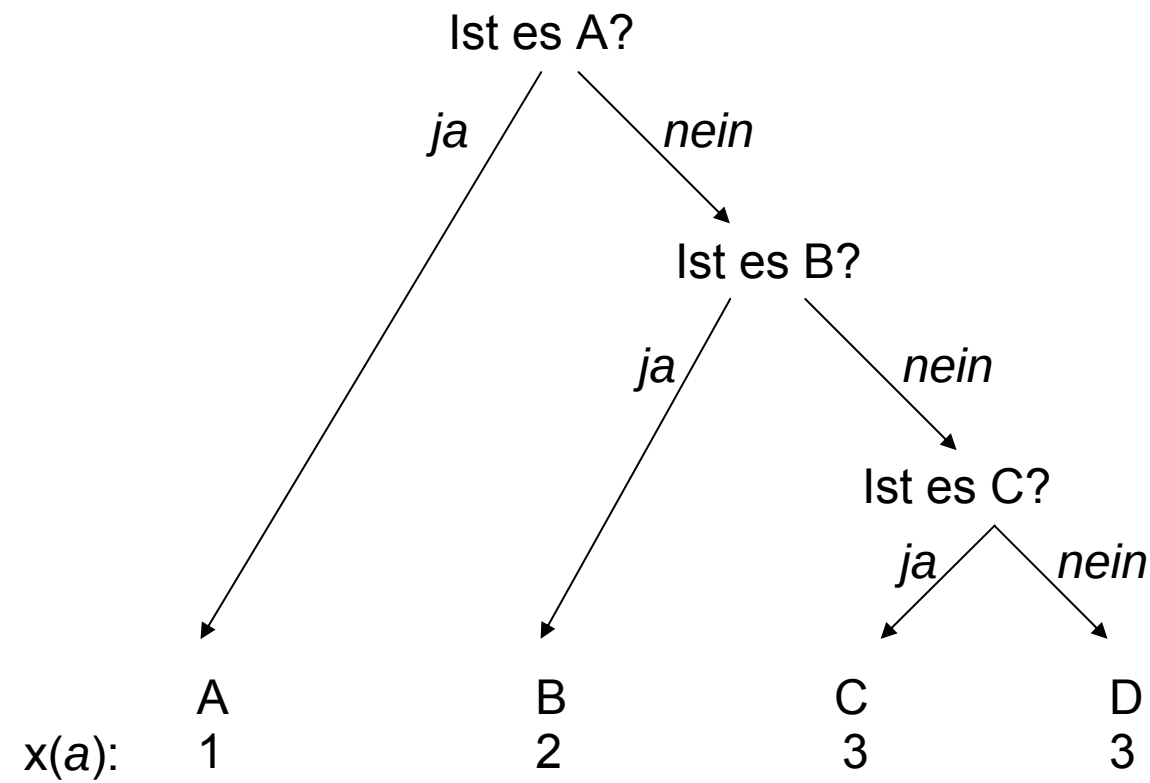
Entropie H = 0

p(*a*): Wahrscheinlichkeit, dass *a* auftritt
x(*a*): Anzahl der Entscheidungen für *a*

$$x(a) = \mathrm{ld}\left(1 / p(a) \right)$$

AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
BBBBBBBBBBBBB
BBBBBBBBBBBBB
CCCCCCCCCCCCC
DDDDDDDDDDDDD

p(A) = 0,5
p(B) = 0,25
p(C) = 0,125
p(D) = 0,125



Berechnung des Zweier-Logarithmus ld aus dem natürlichen Logarithmus ln

$a = b^x$

$x = \log_b a$

$\log_b x = \log_y x / \log_y b$

$\log_2 x = \ln x / \ln 2$
$\log_2 x = \lg x / \lg 2$
…

Beispiele:

$256 = 2^x$
$x = \log_2 256$
$x = 8$

$1.000.000 = 10^x$
$x = \log_{10} 1.000.000$
$x = 6$

$\log_2 256$ im Taschenrechner:

1. „ 256 “ eintippen
2. „ *log* “ drücken
3. „ / “ drücken
4. „ 2 “ eintippen
5. „ *log* “ drücken
6. „ = “ drücken

$\log_e$ heißt *ln* (natürlich Log.)
$\log_2$ heißt *ld* (oder *lg* im Englischen)
$\log_{10}$ heißt *lg* (*log* auf Taschenrechnern)

Online „Scientific Calculator“:
http://www.calculator.com
http://www.google.com

	p	x
A	0,5	1
B	0,25	2
C	0,125	3
D	0,125	3

$$\text{Entropie } H = \sum p(a)\, x(a)$$

Durchschnittlicher Entscheidungsgehalt pro Zeichen

$H = 1{,}75$

$$L = \sum p(a)\, |c(a)|$$

Durchschnittlicher Wortlänge pro Zeichen

Beispielcode:

	c	|c|
A	00	2
B	01	2
C	10	2
D	11	2

$$R = L - H$$

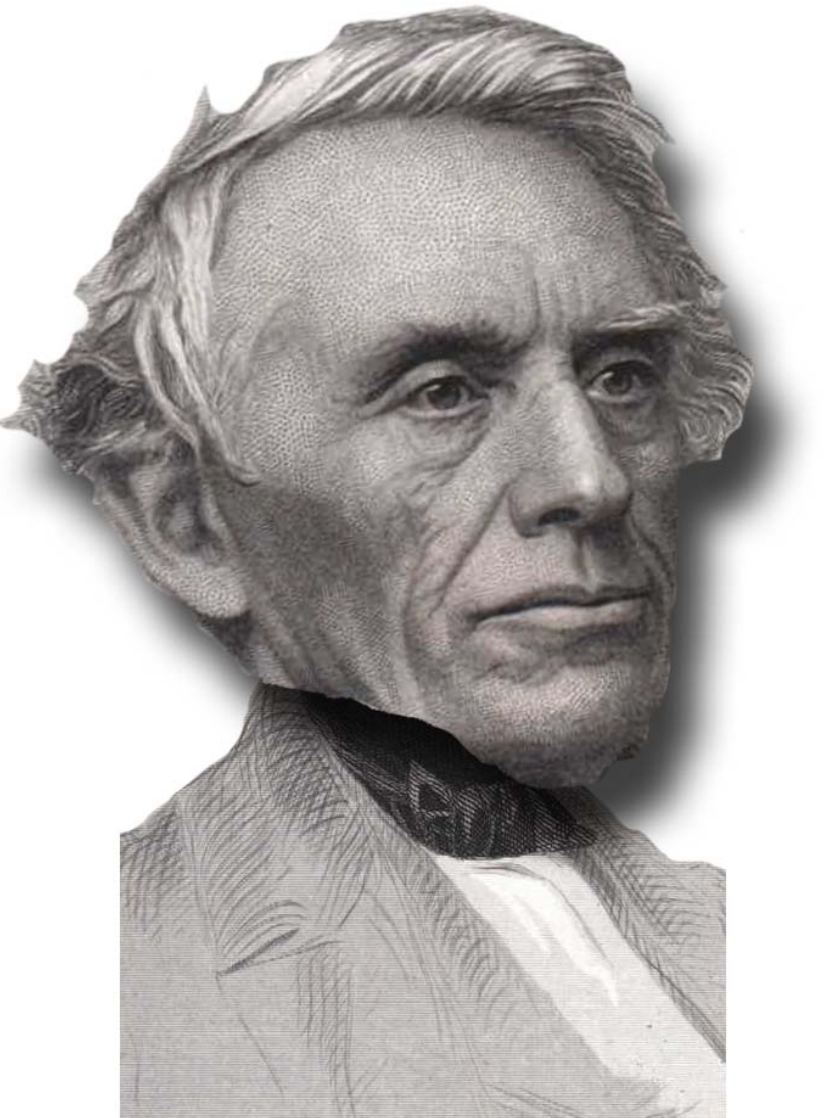
Redundanz eines Codes

$L = 2$

$R = 0{,}25$

Je kleiner, desto besser!

Samuel Morse



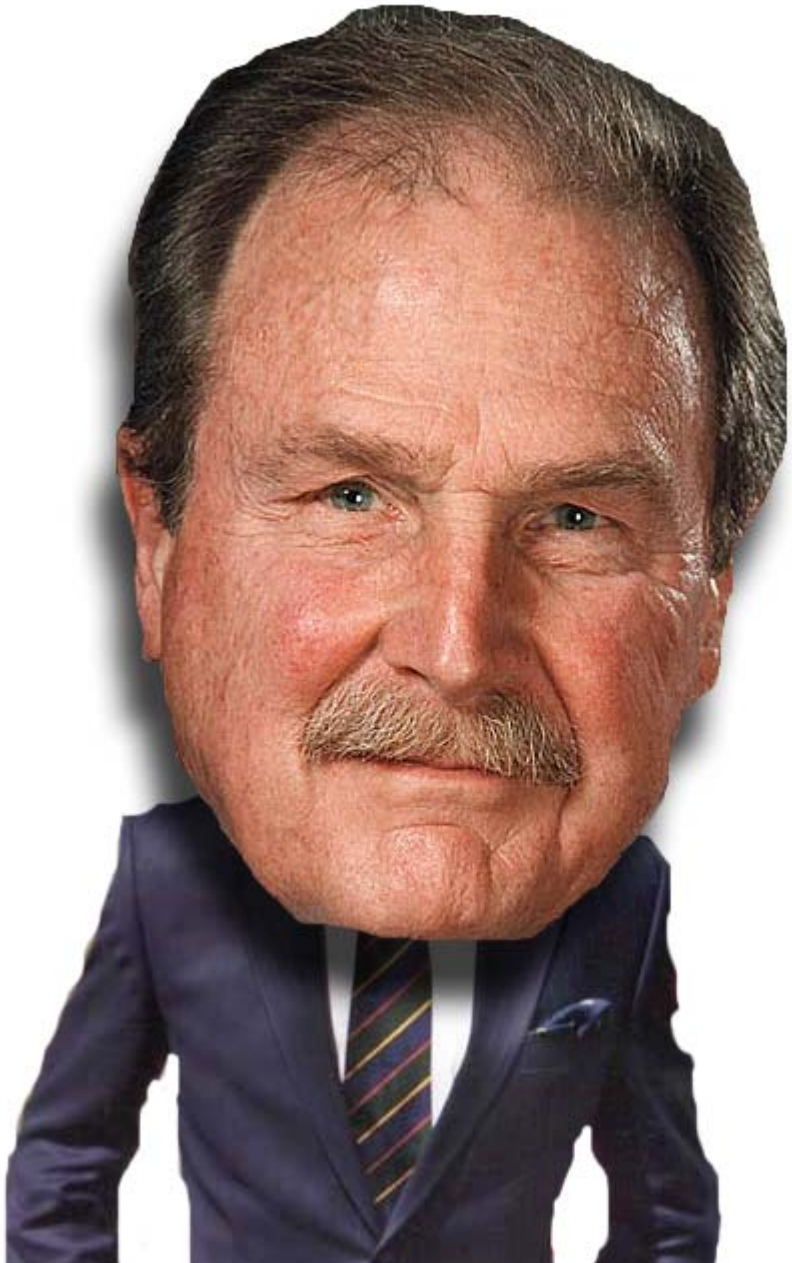
```
. . . .  . .  /  . .  . - - - -  - -  /  . . .
. -  - -  . . -  .  . - . .  /  - -  - - -  . - .
. . .  .  . - . - . -  /  . .  /  - . .  - - -  /
- . - .  - - -  - . .  .  . . .  . - . - . -
```

Idee:

Je häufiger ein Zeichen vorkommt, desto kürzer das kodierte Symbol
=>
kürzere Nachrichten mit gleichem Inhalt!

Allerdings: Kein binärer Code (kurz . , lang - und Pause /)!
Häufigkeiten falsch eingeschätzt!

David A. Huffman:

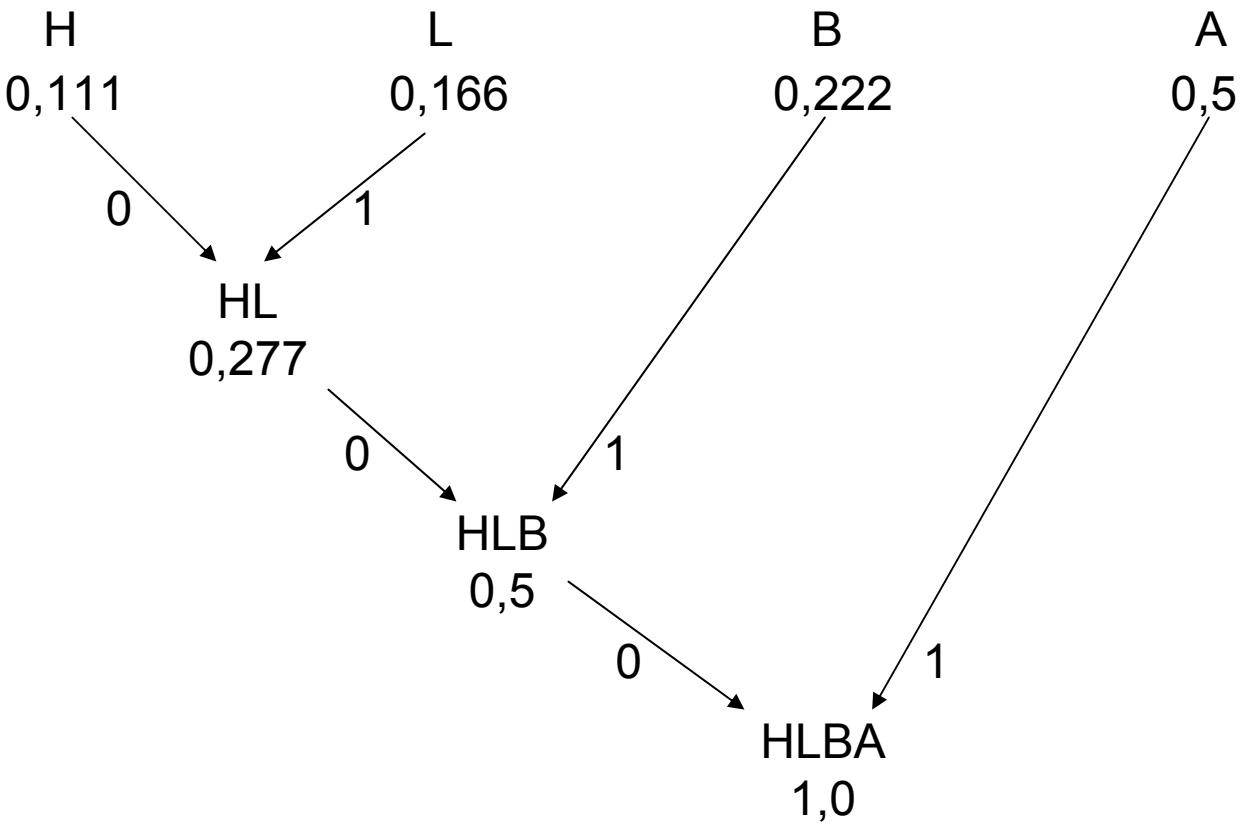


In optimalem Code müssen die beiden Symbole der niedrigsten Häufigkeit mit gleicher Länge codiert sein.

AAAAHHABBLLABBLAAA

A	9/18	0,5
B	4/18	0,222
L	3/18	0,166
H	2/18	0,111

1. Ermittlung der Häufigkeiten
2. Aufbau des Codebaums
3. Code



A	1
B	01
L	001
H	000

aaaaaabbbcde

1. Ermittlung der Häufigkeiten
2. Aufbau des Codebaums
3. Code

Ergebnis:
111111010101001000100000

Warum nicht dieser kürzere Code?

a	1
b	01
c	10
d	11
e	100

Nicht dekodierbar! Fano-Bedingung verletzt!

a	1
b	01
c	001
d	0001
e	0000

Ist der Code optimal?

	p	x
a	0,5	1
b	0,25	2
c	0,083	3,58
d	0,083	3,58
e	0,083	3,58

H = 1,896

	c	|c|
a	1	1
b	01	2
c	001	3
d	0001	4
e	0000	4

L = 1,917

R = 0,021

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

$H = \sum p(a)\ x(a)$

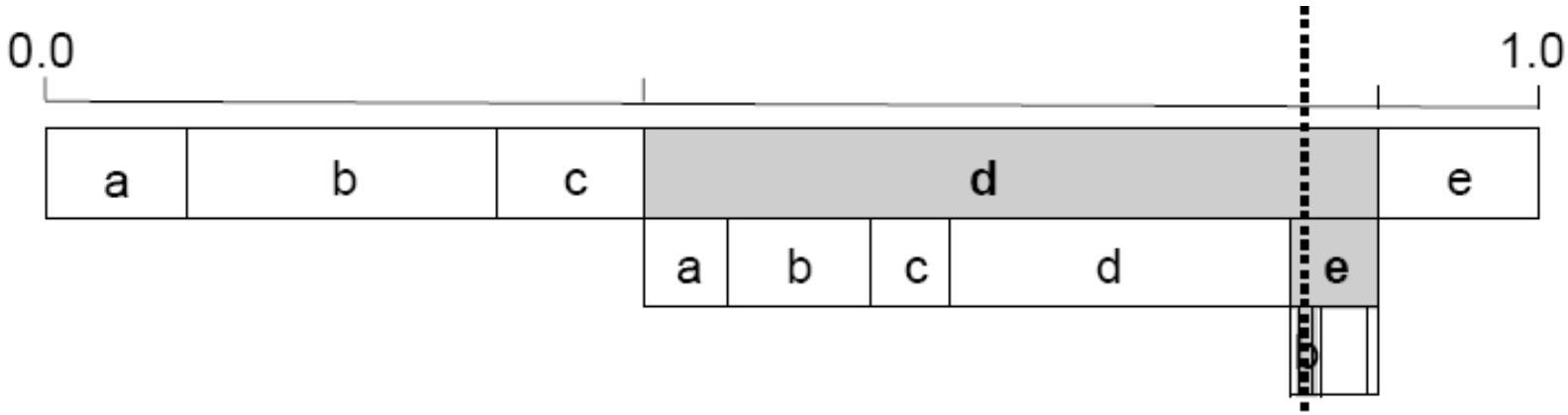
$L = \sum p(a)\ |c(a)|$

$R = L - H$

Generell: Huffman-Code optimal,
falls Häufigkeiten negative Zweierpotenzen
sind, also 0,5 , 0,25 , 0,125 , ...

Idee:

Codieren nicht zeichenweise sondern der kompletten Nachricht in einem Zahlenintervall von 0 bis 1. Jedes Zeichen erhält ein Teilintervall je nach Häufigkeit.



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*Ri;
L = L + B*Li;
Ende Wiederholung;

Code des Textes ist Zahl im Intervall [L, R]

L_i und R_i sind Ränder eines Zeichens, definiert durch seine Auftrittswahrscheinlichkeit

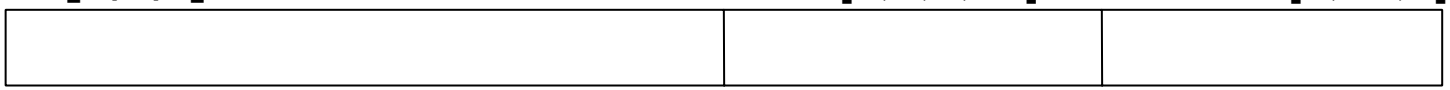
Ausführliche Einleitung mit Java-Applet:
Seite von Oliver Schmid

ABCA

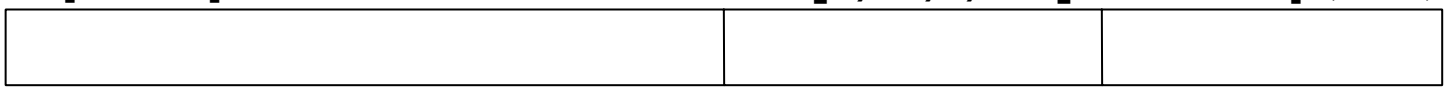
A	0,5
B	0,25
C	0,25

A:	$L_0 = 0$	$R_0 = 0,5$
B:	$L_1 = 0,5$	$R_1 = 0,75$
C:	$L_2 = 0,75$	$R_2 = 1,0$

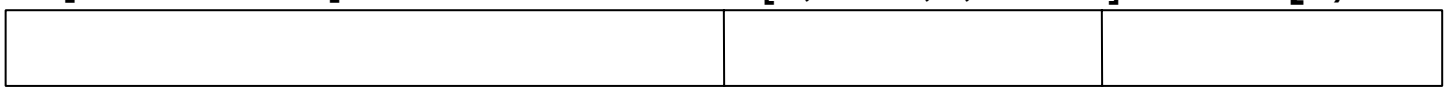
A [0;0,5] B [0,5;0,75] C [0,75;1]



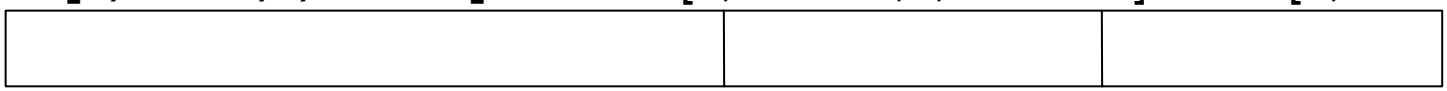
A [0;0,25] **B [0,25;0,375]** C [0,375;0,5]



A [0,25;0,3125] B [0,3125;0,34375] **C [0,34375;0,375]**



A [0,34375;0,359375] B [0,359375;0,3671875] C [0,3671875;0,375]



Konvertierung zwischen ganzen Binär- und Dezimalzahlen

*Ganze **Binär**zahlen nach **Dezimal***

Arbeite die Ziffern der Zahl von rechts nach links durch.
Falls eine Ziffer an der Position *z* gleich 1 ist (Achtung: Die rechteste Position ist 0 und nicht 1!), rechne 2 hoch *z* und addiere die Lösung zum Gesamtergebnis.

*Ganze **Dezimal**zahlen nach **Binär***

Teile die Zahl durch 2.
Der verbleibende Rest ist die nächste Ziffer (fülle von rechts nach links auf!).
Höre auf, sobald das Ergebnis 0 wird.

Beispiel:
1001011 nach Dezimal

$1 \times 2^0 + 1 \times 2^1 + 1 \times 2^3 + 1 \times 2^6$
$= 1 + 2 + 8 + 64$
$= \mathbf{75}$

Beispiel:
75 nach Binär

75 / 2 = 37	Rest: **1**
37 / 2 = 18	Rest: **1**
18 / 2 = 9	Rest: **0**
9 / 2 = 4	Rest: **1**
4 / 2 = 2	Rest: **0**
2 / 2 = 1	Rest: **0**
1 / 2 = 0	Rest: **1**

Mehr Infos: http://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm

Konvertierung zwischen Binär- und Dezimalzahlen zwischen 0 und 1

***Binär** nach **Dezimal** (Komma)*

Arbeite die Ziffern der Zahl hinter dem Komma von **links nach rechts** durch. Falls eine Ziffer an der Position *z* gleich 1 ist (Achtung: Die linkeste Position ist diesmal 1!), rechne 2 hoch -*z* und addiere die Lösung zum Gesamtergebnis.

Beispiel:
0,0101 nach Dezimal

$1 \times 2^{-2} + 1 \times 2^{-4}$
$= 0{,}25 + 0{,}0625$
$= \mathbf{0{,}3125}$

Hinweis:
$2^{-x} = 1 / 2^{x}$

***Dezimal** nach **Binär** (Komma)*

Multipliziere die Zahl mit 2.
Die Zahl vor dem Komma ist die nächste Zahl des Ergebnisses.
Entferne die Zahl vor dem Komma.
Wiederhole das Verfahren, bis nichts mehr rechts vom Komma steht oder sich die Ergebnisse wiederholen.

Beispiel:
0,3125 nach Binär

0,3125 x 2 = **0**,625
0,625 x 2 = **1**,25
0,25 x 2 = **0**,5
0,5 x 2 = **1**

=> **0,0101**

Mehr Infos: http://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm

ABCA

Ergebnisintervall: [0,34375;0,359375]

Untere Grenze in binär:
0,34375

0,34375 x 2 = **0**,6875
0,6875 x 2 = **1**,375
0,375 x 2 = **0**,75
0,75 x 2 = **1**,5
0,5 x 2 = **1**

0,01011

Obere Grenze in binär:
0,359375

0,359375 x 2 = **0**,71875
0,71875 x 2 = **1**,4375
0,4375 x 2 = **0**,875
0,875 x 2 = **1**,75
0,75 x 2 = **1**,5
0,5 x 2 = **1**

0,010111

Code endet mit der ersten Ziffer, in der sich Ober- und Untergrenze in binär unterscheiden.
Das führende „0,“ wird weggelassen.

Code: 010111