

Digitale Medien

Übung

Übungsbetrieb

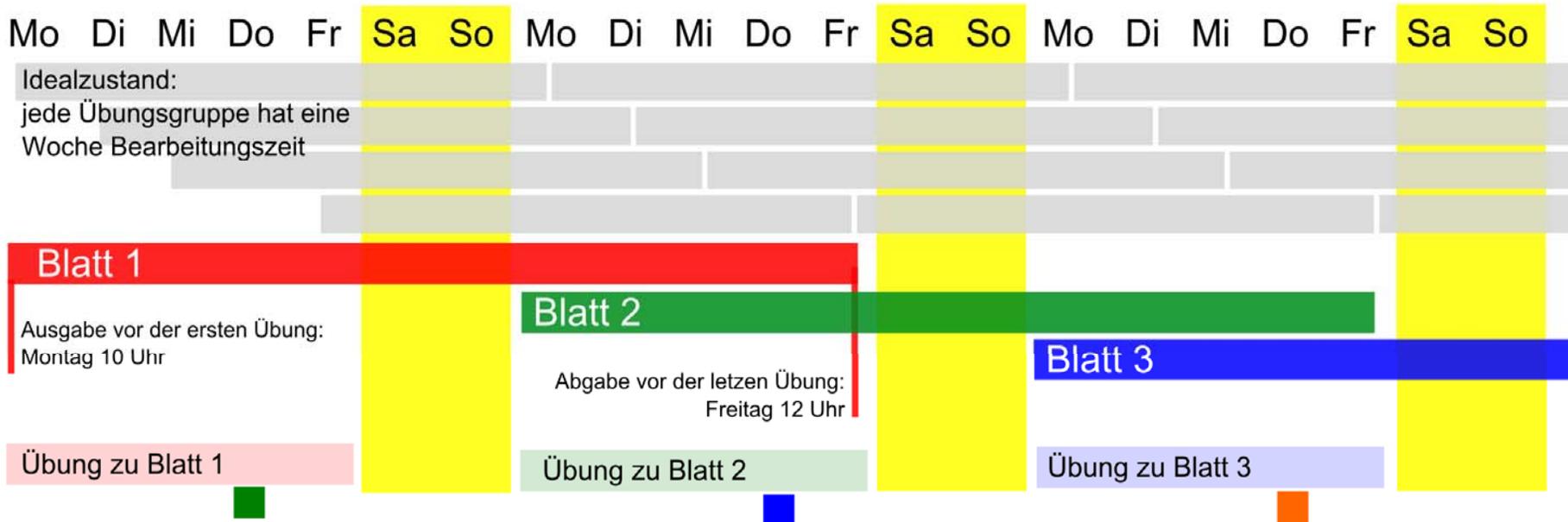
- Informationen zu den Übungen:
<http://www.medien.ifi.lmu.de/dm>
<http://www.die-informatiker.net>
- Zwei Stunden pro Woche
- Praktische Anwendungen des theoretischen Vorlesungsstoffs
- Wichtige Voraussetzung für die Klausur
- Bonuspunkte für die Klausur

Ausgabe der Übungsblätter:

- jeden Montag ab 10 Uhr
- auf <http://www.medien.ifi.lmu.de/lehre/ws0910/dm/>

Abgabe der Übungsblätter:

- jeweils spätestens übernächster Freitag 12 Uhr
- über <https://www.pst.informatik.uni-muenchen.de/uniworx/>



Ausgabe Blatt 1:

ab Mo, 26.10. 10:00

DM-Homepage

Abgabe Blatt 1:

bis Fr, 06.11. 12:00

UniWorx

Oktober	November	Dezember
1 Do	1 So <small>1. Advent</small>	1 Di
2 Fr	2 Mo <small>45</small>	2 Mi
3 Sa <small>Tag 4. Di. Erntedankfest</small>	3 Di	3 Do
4 So <small>Erntedankfest</small>	4 Mi	4 Fr
5 Mo <small>41</small>	5 Do VL1	5 Sa
6 Di	6 Fr	6 So <small>Mittwoch 2. Advent</small>
7 Mi	7 Sa	7 Mo <small>50</small>
8 Do	8 So	8 Di
9 Fr	9 Mo <small>46</small>	9 Mi
10 Sa	10 Di	10 Do
11 So	11 Mi	11 Fr
12 Mo <small>42</small>	12 Do VL1	12 Sa
13 Di	13 Fr	13 So <small>3. Advent</small>
14 Mi	14 Sa	14 Mo <small>51</small>
15 Do	15 So <small>Reformationstag</small>	15 Di
16 Fr	16 Mo <small>47</small>	16 Mi
17 Sa	17 Di	17 Do
18 So	18 Mi <small>Ball- und Feiertag</small>	18 Fr
19 Mo <small>43</small>	19 Do	19 Sa
20 Di	20 Fr	20 So <small>4. Advent</small>
21 Mi	21 Sa	21 Mo <small>52</small>
22 Do VL1	22 So <small>Tageworttag</small>	22 Di
23 Fr	23 Mo <small>48</small>	23 Mi
24 Sa	24 Di	24 Do <small>5. Advent</small>
25 So	25 Mi	25 Fr <small>1. Weihnachtstag</small>
26 Mo <small>44</small>	26 Do	26 Sa <small>2. Weihnachtstag</small>
27 Di	27 Fr	27 So
28 Mi	28 Sa	28 Mo <small>53</small>
29 Do VL1	29 So <small>1. Advent</small>	29 Di
30 Fr	30 Mo <small>49</small>	30 Mi
31 Sa <small>Reformationstag</small>		31 Do <small>Stifter</small>

Heute

- Speicherung unkomprimierter Datentypen
- Lauflängencodierung
- Entropie und optimale Codes
- Huffman Codierung
- Arithmetische Codierung

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

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

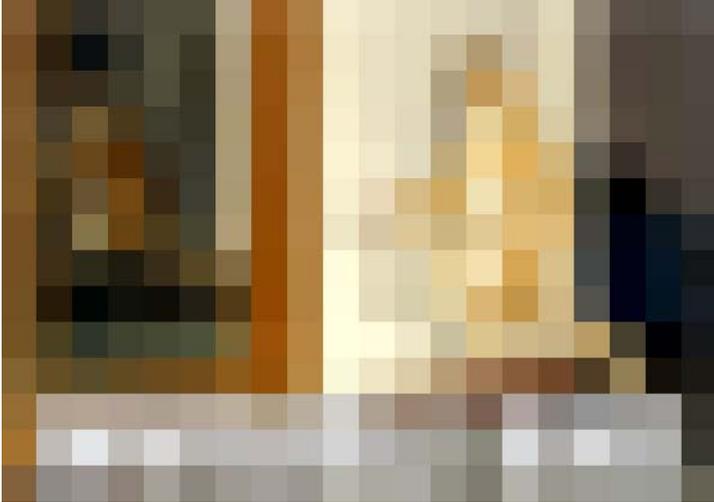
1 Byte / Zeichen

Nonsense

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

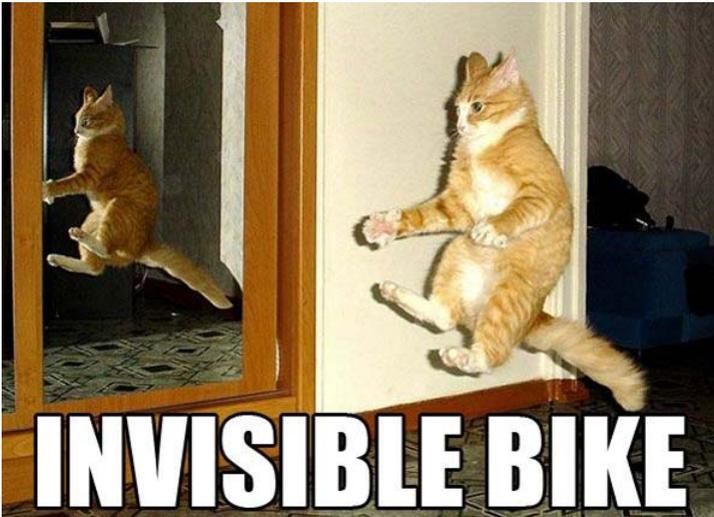
Binär: 01001110 N
01101111 o
01101110 n
01110011 s
01100101 e
01101110 n
01110011 s
01100101 e

8 Byte



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

560 Byte



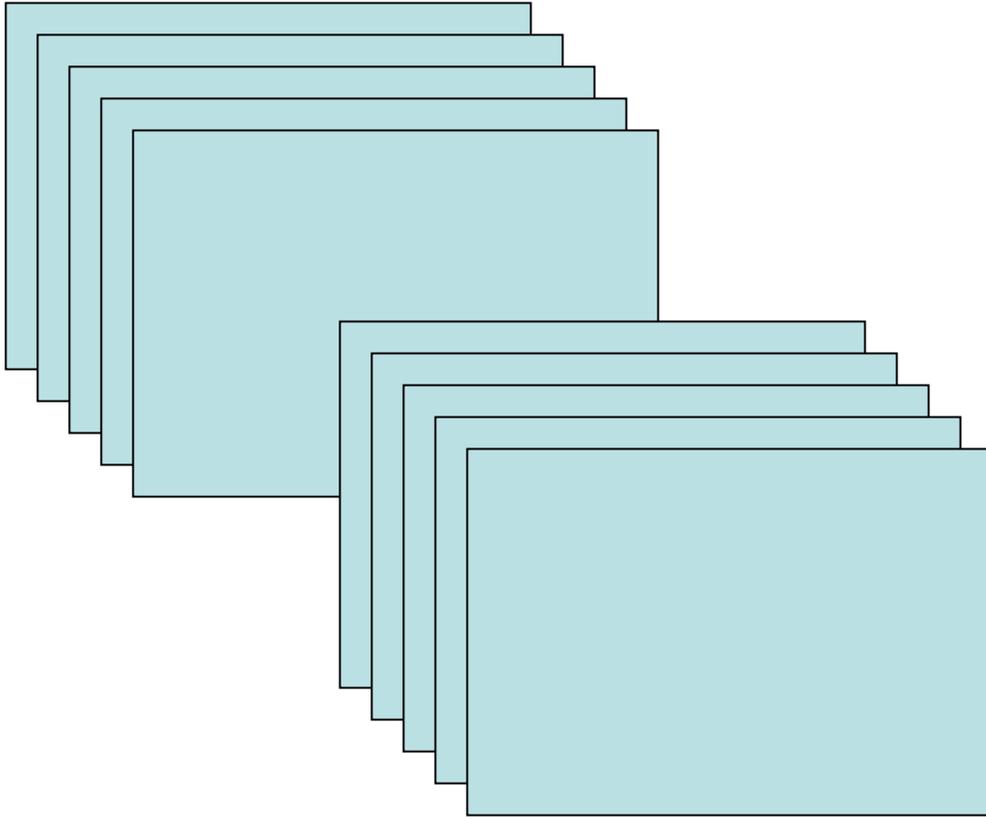
600 x 432 Pixel
16 Bit Farben

518 Kilobyte



6 Megapixel Kamera:
2848 x 2136
32 Bit

24 Megabyte



PAL Video

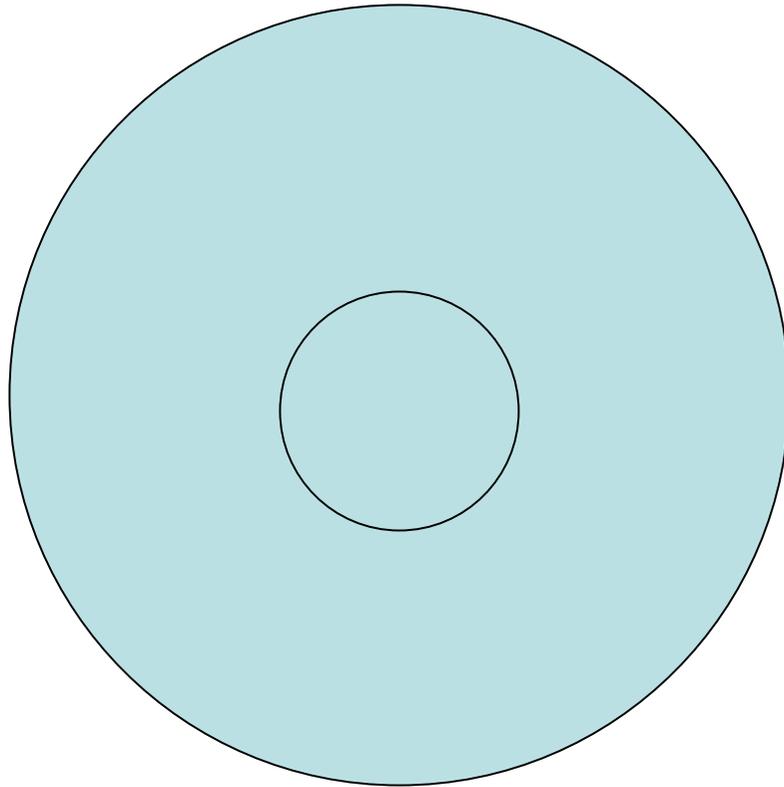
768 x 576 Pixel

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

Laufängencodierung (RLE)



AAAAAAAAAAAAAAAAAAAAAA

20 Zeichen

#A20

4 Zeichen

Dies ist ein Beispieltext.

26 Zeichen

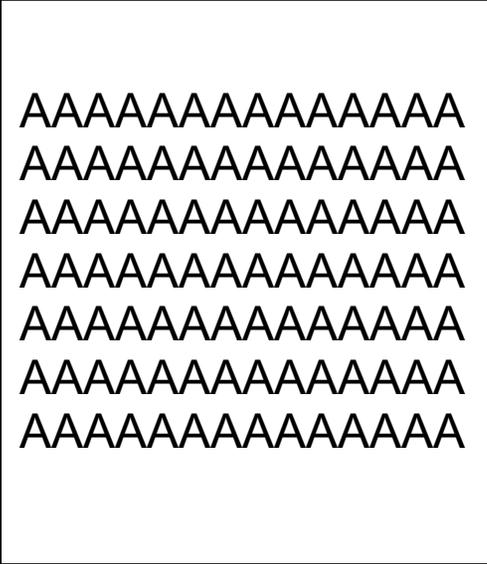
Dies ist ein Beispieltext.

26 Zeichen

⇒ Funktioniert besser mit Bildern als mit Text

Jede Nachricht hat einen Informationsgehalt,
die *Entropie*.

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



AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA

$$p(A) = 1$$

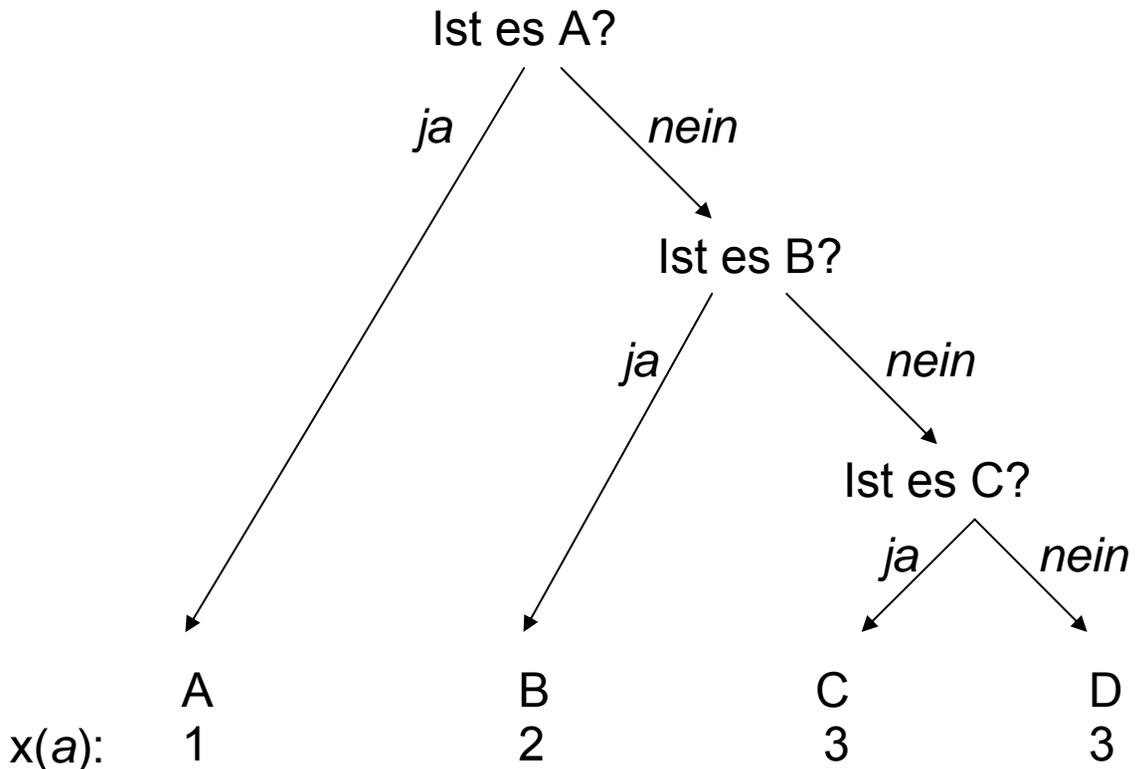
$$\text{Entropie } H = 0$$

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

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

AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCC
DDDDDDDDDDDDDD

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



Berechnung des Zweier-Logarithmus \log_2 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

$$H = 1,75$$

Beispielcode:

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

$$L = 2$$

$$R = 0,25$$

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

Durchschnittlicher Entscheidungsgehalt
eines Zeichen

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

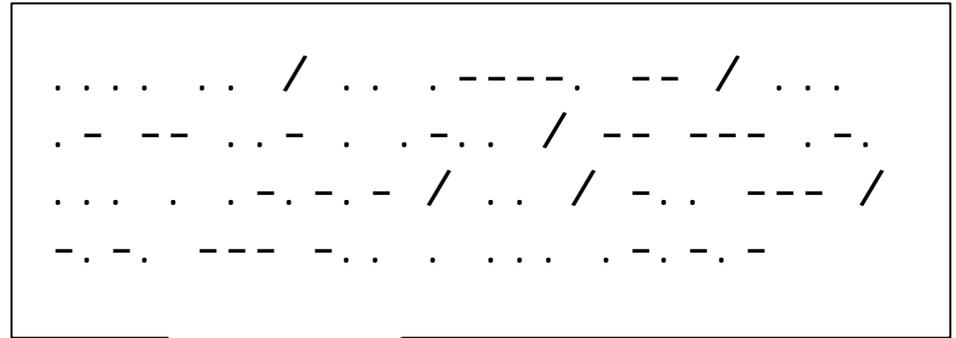
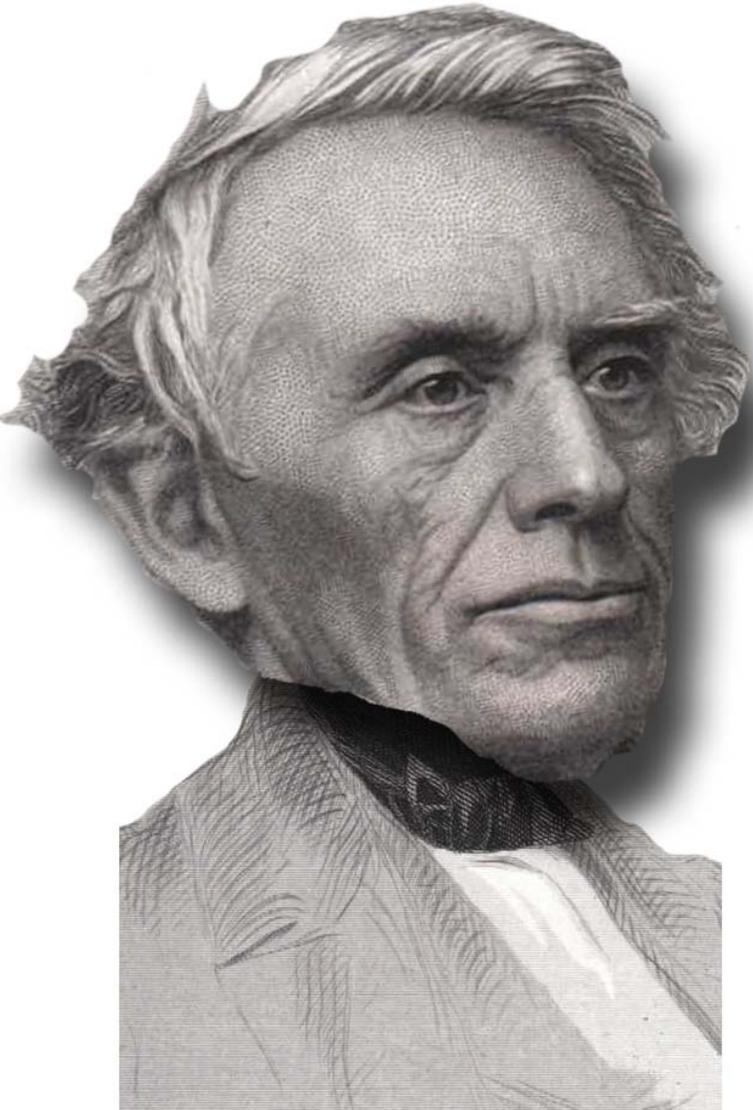
Durchschnittlicher Wortlänge
pro Zeichen

$$R = L - H$$

Redundanz eines Codes

Je kleiner, desto besser!

Samuel Morse



Idee:

Je häufiger ein Zeichen vorkommt,
desto kürzer das kodierte Symbol

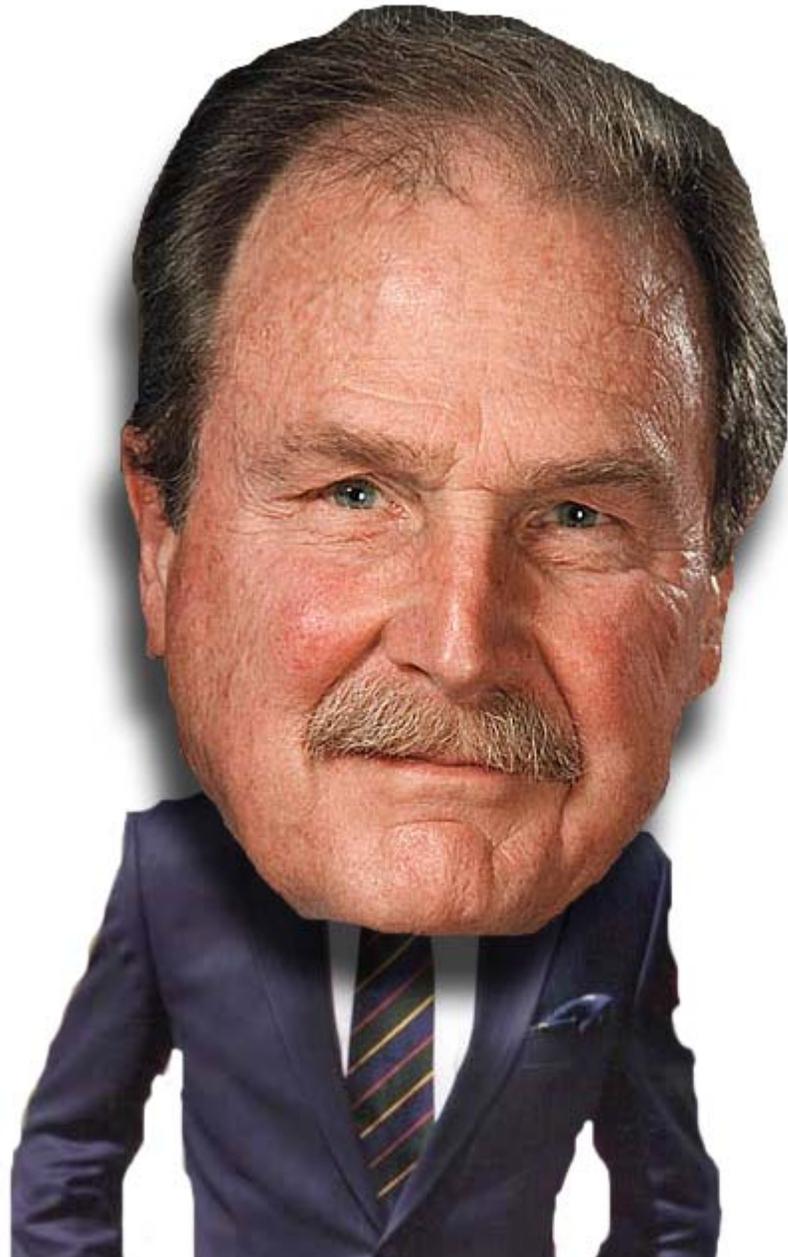
=>

kürzere Nachrichten mit gleichem Inhalt!

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

Häufigkeiten falsch eingeschätzt!

David A. Huffman:

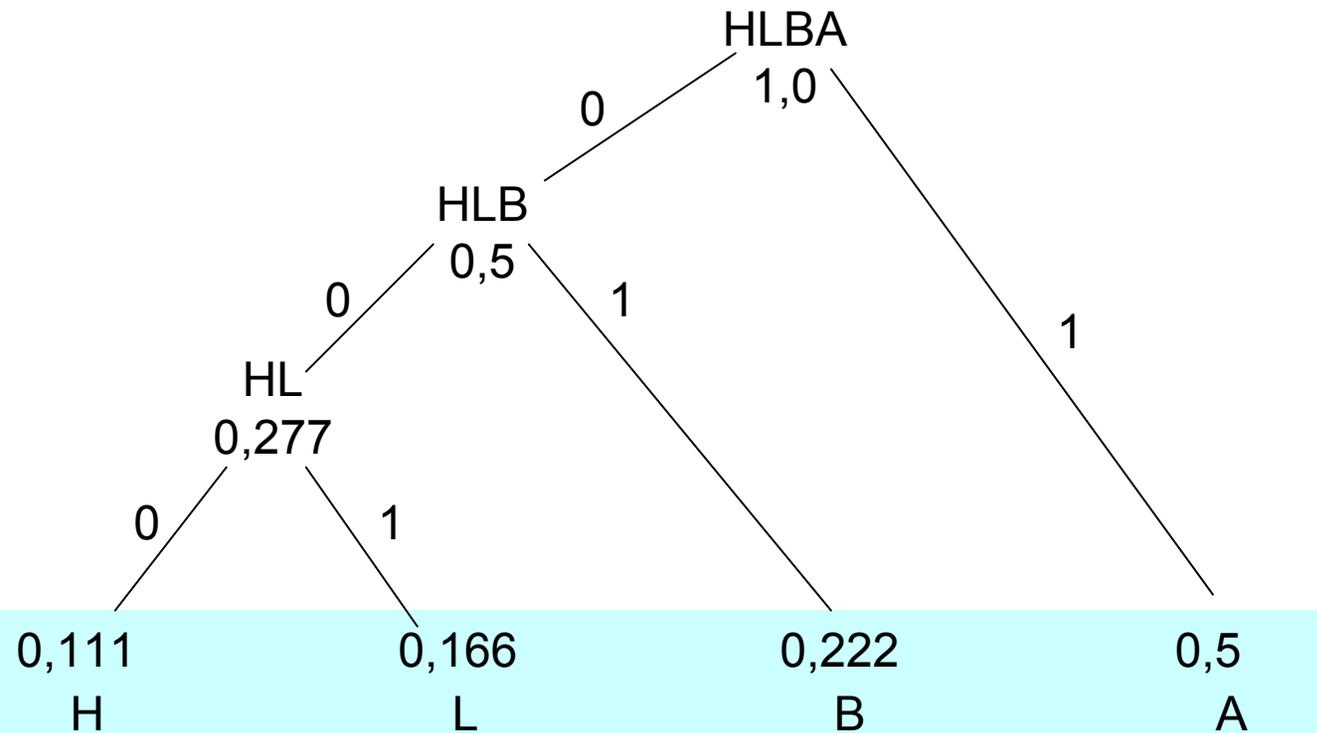


In optimalem Code
müssen die beiden
Symbole der niedrigsten
Häufigkeit mit gleicher
Länge codiert sein.

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

Ergebnis:

11111101010100100010000

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

Warum nicht dieser kürzere Code?

a	1
b	01
c	10
d	11
e	100

a	1
b	01
c	001
d	0001
e	0000

Nicht dekodierbar! Fano-Bedingung verletzt!

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

$$L = 1,917$$

$$R = 0,021$$

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

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

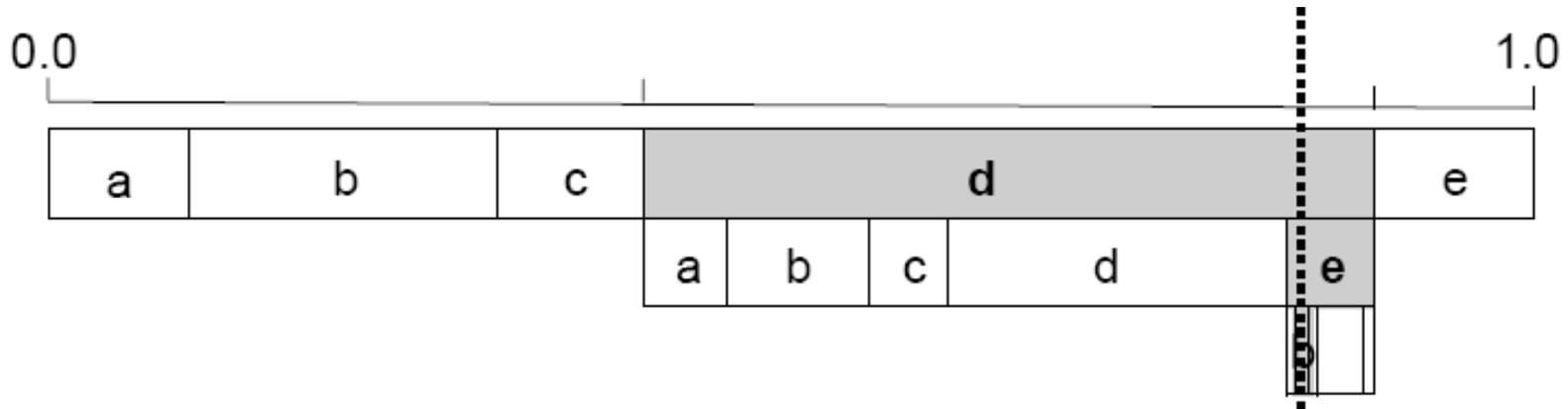
$$L = \sum p(a) |c(a)|$$

$$R = L - H$$

Generell: Huffman-Code optimal,
falls Häufigkeiten negative/Kehrwerte von
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); (Intervallbreite)
  R = L + B*Ri; (Obergrenze)
  L = L + B*Li; (Untergrenze)
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	A:	$L_0 = 0$	$R_0 = 0,5$
B	0,25	B:	$L_1 = 0,5$	$R_1 = 0,75$
C	0,25	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 **Binärzahlen** 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.

Beispiel:

1001011 nach Dezimal

$$\begin{aligned} & 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^3 + 1 \times 2^6 \\ = & \quad 1 + \quad 2 + \quad 8 + \quad 64 \\ = & \mathbf{75} \end{aligned}$$

Ganze **Dezimalzahlen** 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:

75 nach Binär

$$\begin{array}{ll} 75 / 2 = 37 & \text{Rest: } \mathbf{1} \\ 37 / 2 = 18 & \text{Rest: } \mathbf{1} \\ 18 / 2 = 9 & \text{Rest: } \mathbf{0} \\ 9 / 2 = 4 & \text{Rest: } \mathbf{1} \\ 4 / 2 = 2 & \text{Rest: } \mathbf{0} \\ 2 / 2 = 1 & \text{Rest: } \mathbf{0} \\ 1 / 2 = 0 & \text{Rest: } \mathbf{1} \end{array}$$

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 $2^{\text{hoch } -z}$ und addiere die Lösung zum Gesamtergebnis.

Beispiel:

0,0101 nach Dezimal

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

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

$$\begin{aligned} 0,3125 \times 2 &= \mathbf{0,625} \\ 0,625 \times 2 &= \mathbf{1,25} \\ 0,25 \times 2 &= \mathbf{0,5} \\ 0,5 \times 2 &= \mathbf{1} \end{aligned}$$

=> **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

Nächste Woche

- LZW Codierung und Decodierung
- Digitalisierung von Analogen Signalen