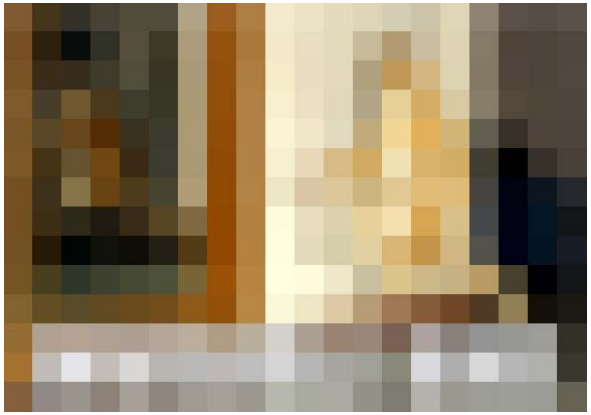
Übung zur Vorlesung

Digitale Medien

Vorlesung: Heinrich Hußmann
Übung: Renate Häuslschmid, Hanna Schneider

Ludwig-Maximilians-Universität München
Wintersemester 2015/16

Bilder



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

560 Byte

600 x 432 Pixel
16 Bit Farben

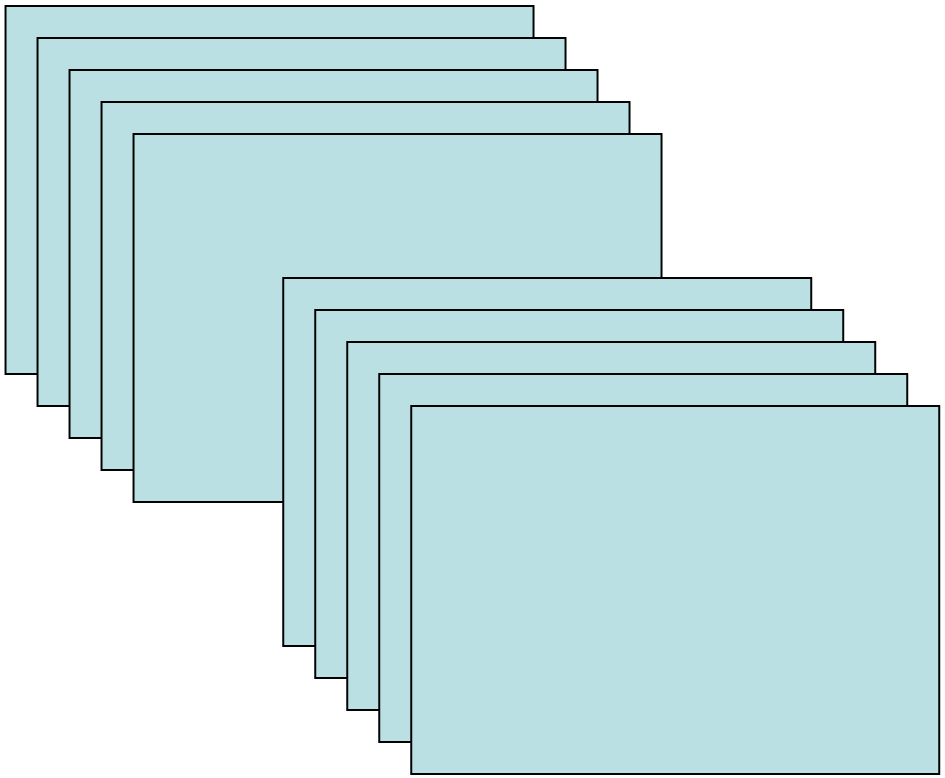
518 Kilobyte

Bilder

6 Megapixel Camera:
2848 x 2136
32 Bit

24 Megabyte

Video



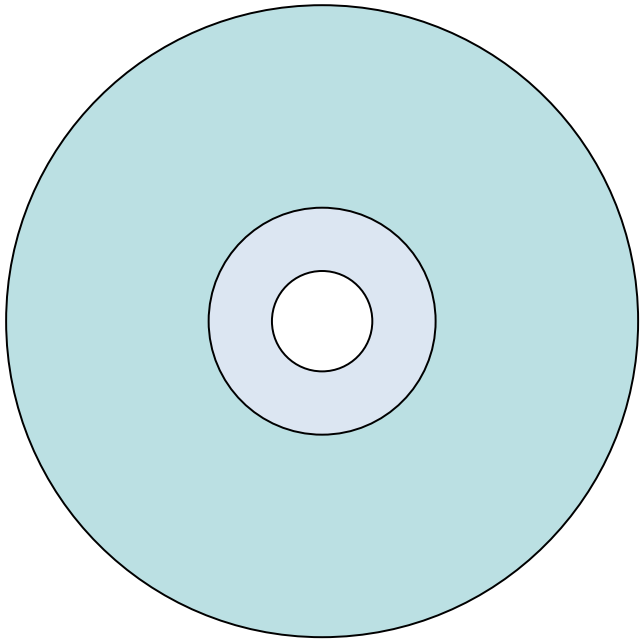
PAL Video

768 x 576 Pixel
24 Bit Farbe

25 Bilder / Sekunde
3 Minuten Film

6 Gigabyte

Video



DVD Spielfilm (ohne Ton!)

720 x 576 Pixel
24 Bit Farbe

25 Bilder / Sekunde
100 Minuten Film

187 Gigabyte

Codierung

aaaaaabbbcde

ASCII:
aaaaaabbbcde — 96 Bit

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

Lauflängenkodierung:
#6a#3bcde — 72 Bit

Huffmankodierung:
111111010101001000100000 — 23 Bit

Arithmetische Kodierung:
00000010101011010011 1 — 21 Bit

LAUFLÄNGENCODIERUNG

Lauflängencodierung

Idee: Ersetzen einer Folge gleicher Zeichen durch 1 Zeichen + Zähler

Aber: Funktioniert besser mit Bildern als mit Text

AAAAAAAAAAAAAAAAAAAA → 20 Zeichen

Lauflängenkodierung:
#20A → 4 Zeichen

Dies ist ein Beispieltext. → 26 Zeichen

Lauflängenkodierung:
Dies ist ein Beispieltext. → 26 Zeichen

Entropie

AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA

Jede Nachricht hat einen Informations-gehalt, die *Entropie*.

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

p(A) = 1

Entropie H = 0

Entropie

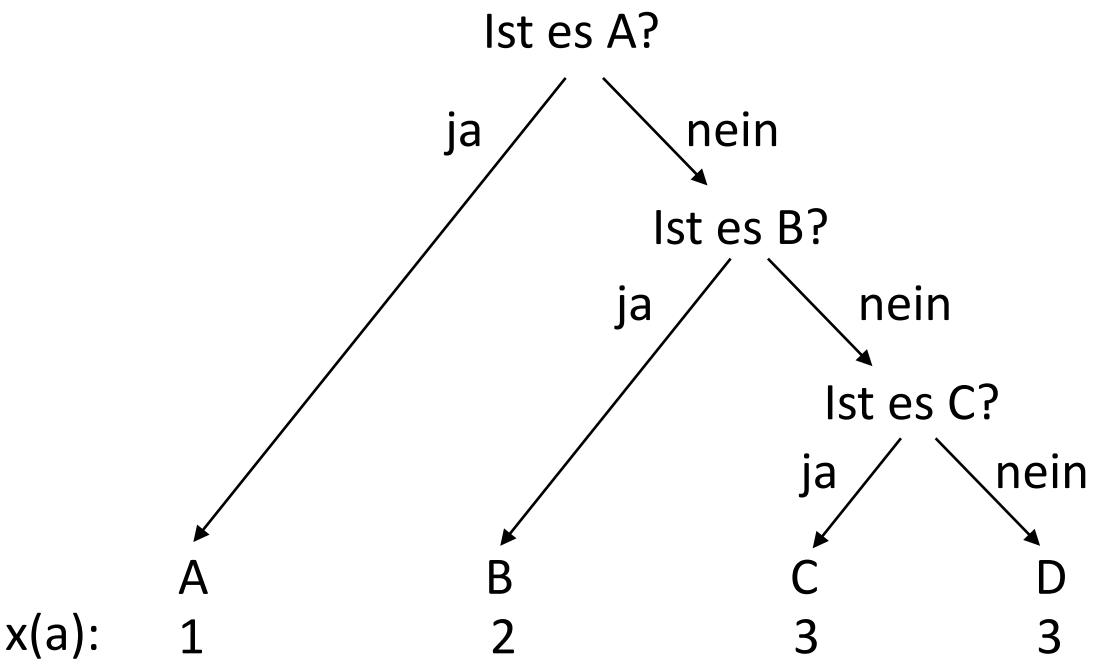
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
AAAAAAAAAAAAAA
BBBBBBBBBBBBB
BBBBBBBBBBBBB
CCCCCCCCCCCCC
DDDDDDDDDDDDD

Lauflängen-Codierung:
#56A #28B #14C #14D

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

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

$x(a) = \log_2 (1 / p(a))$



Einschub: Zweierlogarithmus

$a = b^x$

$x = \log_b a$

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

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

Beispiele:

$256 = 2^x$

$x = \log_2 256$

$x = 8$

$1.000.000 = 10^x$

$x = \log_{10} 1.000.000$

$x = 6$

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

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

Entropie

Durchschnittlicher Entscheidungsgehalt eines Zeichen:

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

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

Durchschnittliche Wortlänge pro Zeichen:

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

Redundanz eines Codes: Je kleiner, desto besser!

$$R = L - H$$

Beispiel:

	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

Aufgabe 1

a) Komprimieren Sie die Nachricht mit Hilfe der Lauflängencodierung:

DDBB BB00 00CC CCCC B141 1122 222G GGGC

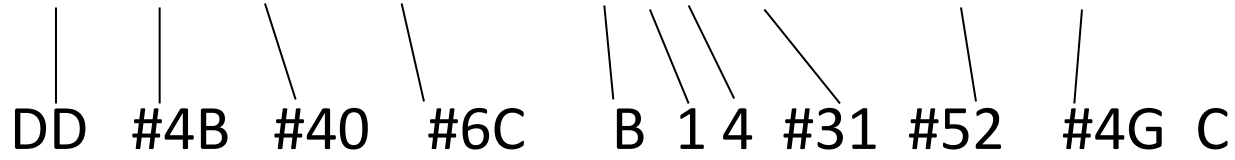
Die Leerzeichen in der Nachricht dienen der Lesbarkeit und sollen *nicht* mitcodiert werden.

a) Wie viel Prozent an Zeichen werden dadurch eingespart?

b) Erzeugen sie eine Nachricht (12 Zeichen), für welche die Lauflängen-codierung ein schlechtes Ergebnis liefert (Komprimierung < 5%). Erläutern sie kurz, welche Eigenschaften eine Nachricht haben muss, damit die Lauflängencodierung gute Ergebnisse liefern kann.

Lösung zu Aufgabe 1

a) DDBB BB00 00CC CCCC B141 1122 222G GGGC



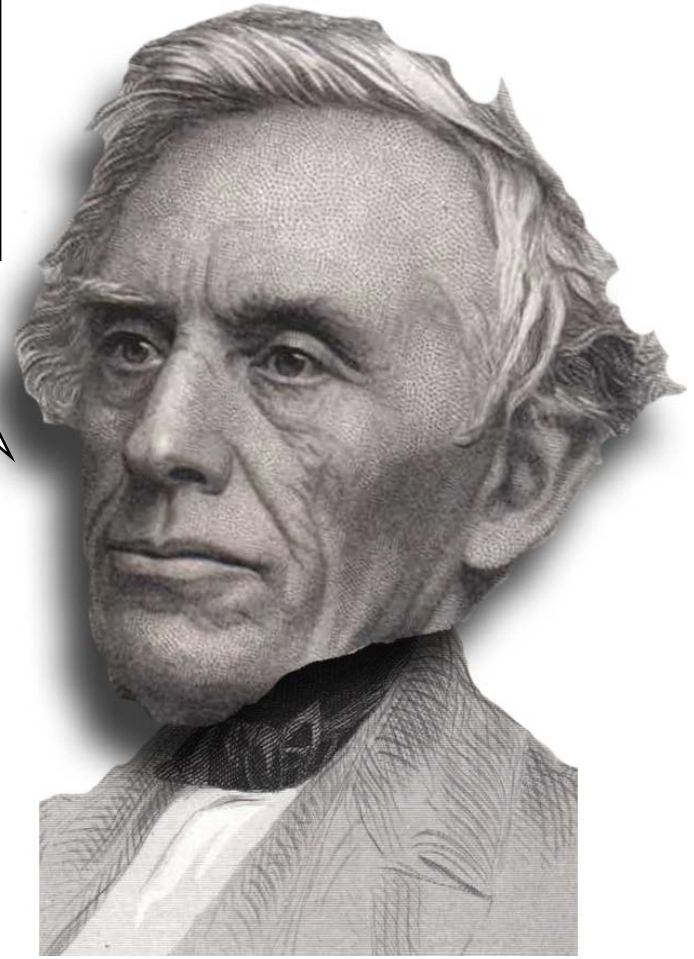
b) 32 - 24 = 8 Zeichen
8 / 32 = 25 % werden eingespart

c) 0123 4567 89AB CDEF
Je mehr Wiederholungen eine Nachricht hat, desto besser ist die Komprimierung.

HUFFMAN-CODIERUNG

Samuel Morse

.... .. / .. .----. -- / ...
.- -- ..- . .-.. / -- --- .-.
... . .-.-.- / .. / -.. --- /
-.-. --- -.. . ... .-.-.-



Idee:

Je häufiger ein Zeichen vorkommt,

desto kürzer das kodierte Symbol

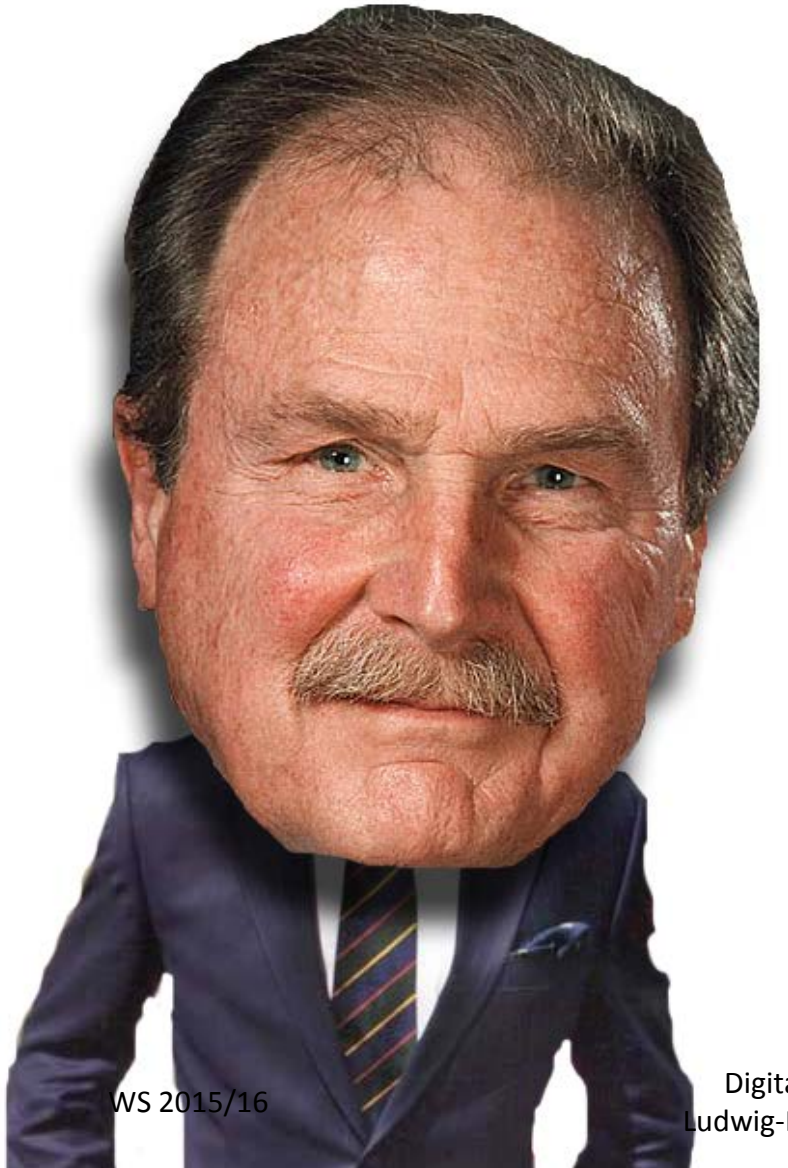
➔kürzere Nachrichten mit gleichem Inhalt!

Problem:

Kein binärer Code: kurz . , lang - und Pause /

Häufigkeiten falsch eingeschätzt!

David Huffman



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

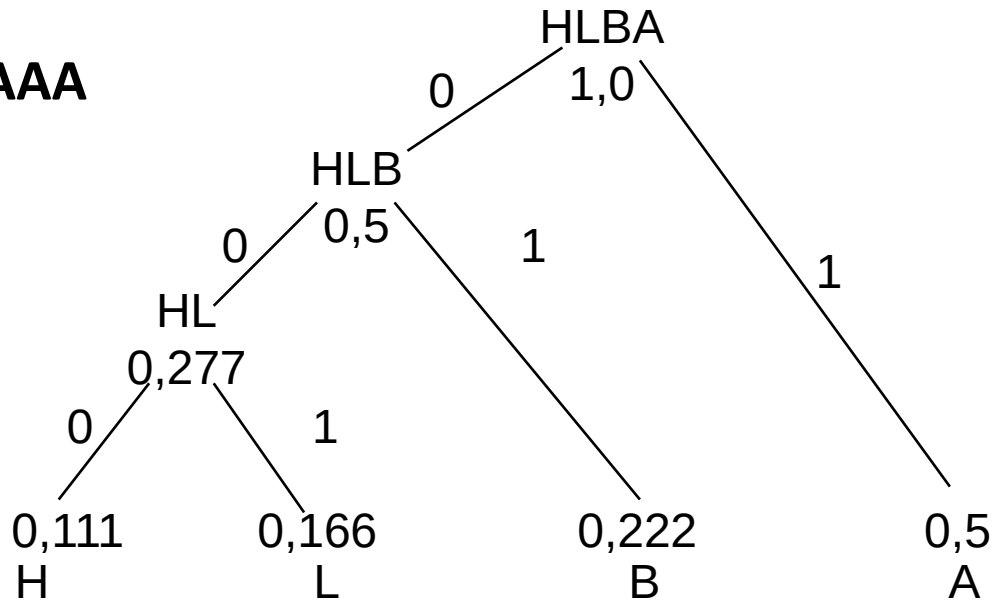
Huffman Codierung

Vorgehen:

1. Ermittlung der Häufigkeiten
2. Aufbau des Codebaums (von unten)
3. Code

Beispiel: **AAAAHHABBLLABBBLAAA**

	Häufigkeiten		***Code***
A	9/18	0,5	1
B	4/18	0,222	01
L	3/18	0,166	001
H	2/18	0,111	000



Huffman Codierung

Beispiel: **aaaaaabbbcde**

Code:

a	1
b	01
c	001
d	0001
e	0000

→ **11111101010100100010000**

Warum nicht dieser kürzere Code?

a	1
b	01
c	10
d	11
e	100

Nicht dekodierbar!
Fano-Bedingung verletzt:
1110 = aac oder dc?

Ist der Code optimal?

Huffman-Code optimal, wenn Häufigkeiten negative Kehrwerte von Zweierpotenzen sind, also 0,5 , 0,25 , 0,125

	p	x
a	0,5	1
b	0,25	2
c	0,083	3,585
d	0,083	3,585
e	0,083	3,585

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

H = 1,893

L = 1,913

R = 0,02

$x(a) = ld (1 / p(a))$

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

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

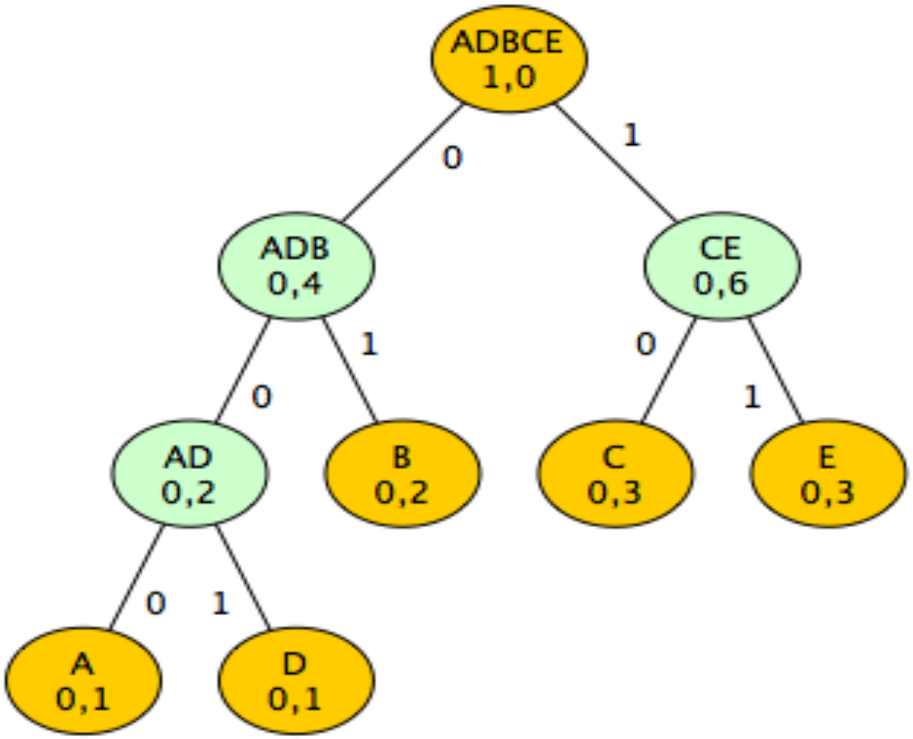
$R = L - H$

Anderes Beispiel

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

Beispiel: **CECEDBCABE**

	Häufigkeiten		**Code**
A	1/10	0,1	000
B	2/10	0,2	01
C	3/10	0,3	10
D	1/10	0,1	001
E	3/10	0,3	11



Aufgabe 2

Für einen Zeichenvorrat sind folgende Auftrittswahrscheinlichkeiten gegeben:

Zeichen	S	E	R	K	I	L
Häufigkeit	0,08	0,07	0,4	0,12	0,1	0,23

a) Leiten Sie die zugehörige Huffman Codierung her.

b) Berechnen Sie die Redundanz des Codes.

c) Codieren Sie die Zeichenfolgen
   a) KREIS
   b) KIESEL

d) Ist der Code optimal? Begründen Sie Ihre Entscheidung.

Lösung zu Aufgabe 2a)

Leiten Sie die zugehörige Huffman Codierung her.

Sortieren nach Auftrittswahrscheinlichkeit:

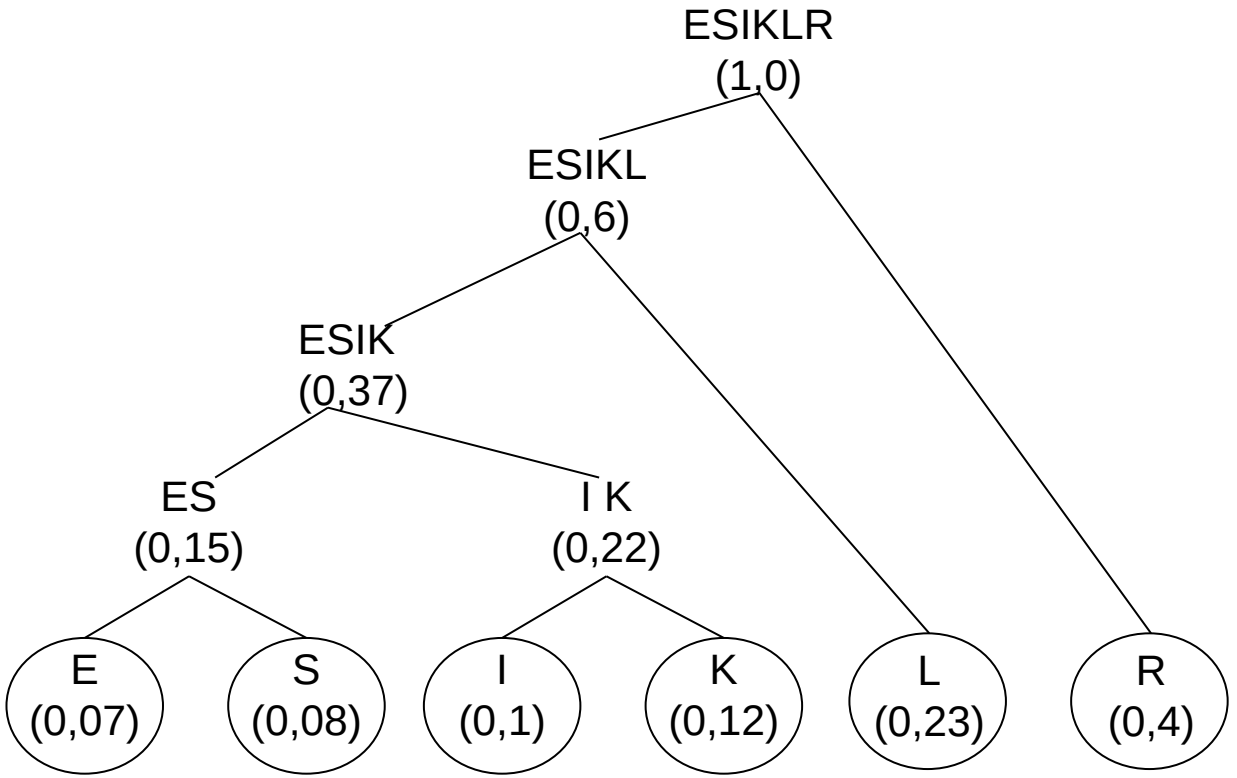
Zeichen	E	S	I	K	L	R
Häufigkeit	0,07	0,08	0,1	0,12	0,23	0,4

Wiederholte Anwendung folgenden Schritts auf die Häufigkeitstabelle:

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

Lösung zu Aufgabe 2a)

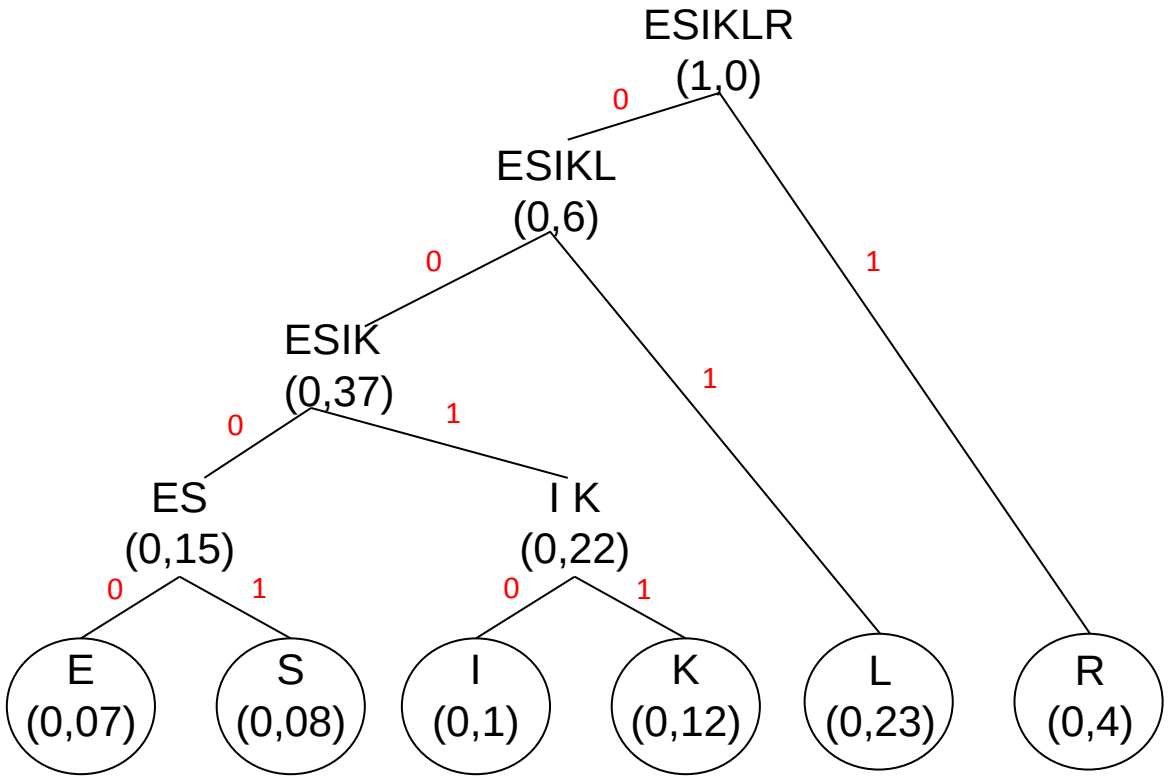
Leiten Sie die zugehörige Huffman Codierung her.



Lösung zu Aufgabe 2a)

Leiten Sie die zugehörige Huffman Codierung her.

E	0000
S	0001
I	0010
K	0011
L	01
R	1



Lösung zu Aufgabe 2b)

Berechnen Sie die Redundanz des Codes.

L = 0,4 * 1 (R)
+ 0,23 * 2 (L)
+ 0,12 * 4 (K)
+ 0,1 * 4 (I)
+ 0,08 * 4 (S)
+ 0,07 * 4 (E)
= 0,4 + 0,46 + 0,48 + 0,4 + 0,32 + 0,28
= **2,34**

H = 0,4 * ld (1/0,4) (R)
+ 0,23 * ld (1/0,23) (L)
+ 0,12 * ld (1/0,12) (K)
+ 0,1 * ld (1/0,1) (I)
+ 0,08 * ld (1/0,08) (S)
+ 0,07 * ld (1/0,07) (E)
= **2,276**

R = L – H = 2,34 – 2,275 = 0,064

Lösung zu Aufgabe 2c)

Codieren Sie die Zeichenfolgen

a) KREIS

b) KIESEL

K	R	E	I	S	
0011	1	0000	0010	0001	

K	I	E	S	E	L
0011	0010	0000	0001	0000	01

Lösung zu Aufgabe 2d)

Ist der Code optimal? Begründen Sie Ihre Entscheidung.

Nein, denn die Redundanz ist nicht 0!

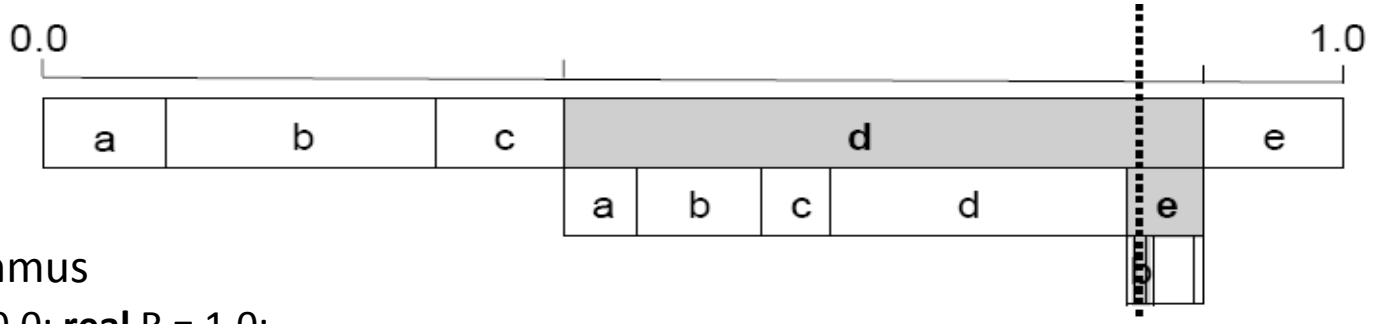
Der Code wird nur optimal, wenn die Auftrittswahrscheinlichkeiten umgekehrte 2er-Potenzen sind.

z.B. 0.5, 0.25, 0.125 usw.

ARITHMETISCHE CODIERUNG

Arithmetische Codierung

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



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

Arithmetische Codierung

Beispiel **ABCA**:

A	0,5	$L_0 = 0$	$R_0 = 0,5$
B	0,25	$L_1 = 0,5$	$R_1 = 0,75$
C	0,25	$L_2 = 0,75$	$R_2 = 1,0$

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

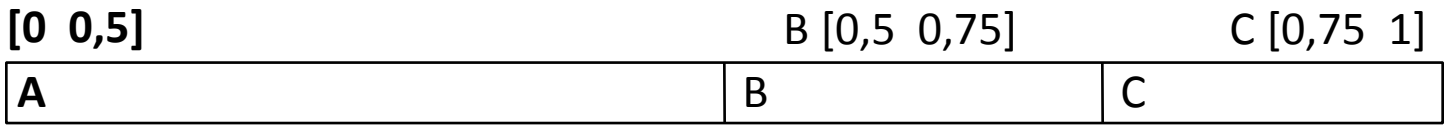
real L = 0.0; **real** R = 1.0;
Solange Zeichen vorhanden **wiederhole**
Lies Zeichen und bestimme Zeichenindex i;
real B = (R–L); (Intervallbreite)
R = L + B*Ri; (Obergrenze)
L = L + B*Li; (Untergrenze)
Ende Wiederholung;

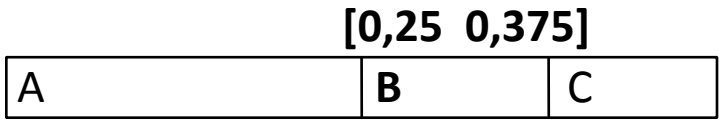
Code des Textes ist Zahl im Intervall [L, R]

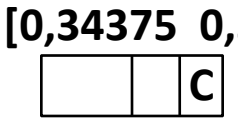
Arithmetische Codierung

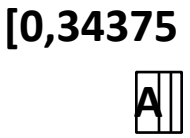
Beispiel **ABCA**:

A	0,5	$L_0 = 0$	$R_0 = 0,5$
B	0,25	$L_1 = 0,5$	$R_1 = 0,75$
C	0,25	$L_2 = 0,75$	$R_2 = 1,0$









Arithmetische Codierung

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: 010111

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

Aufgabe 3

Es sei folgende Nachricht gegeben: **EGHEGFGGEF**

a) Veranschaulichen Sie die arithmetische Codierung, indem Sie die Aufteilung in Intervalle **graphisch** darstellen. Ordnen Sie die Intervalle dabei **alphabetisch** an!

b) Codieren Sie die gesamte Nachricht mittels des Algorithmus der arithmetischen Codierung. Geben Sie für jeden Rechenschritt die obere und untere Grenze sowie die Breite des Intervalls an. Verwenden Sie dafür die Tabelle.

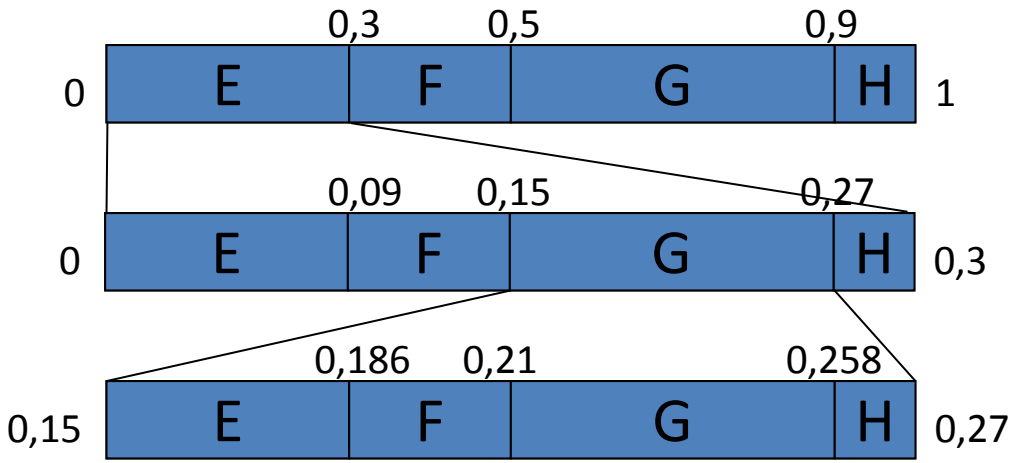
	B	L	R
E			
G			
H			
E			
G			
F			
G			
G			
E			
F			

c) Geben Sie die obere und untere Grenze des Ergebnisintervalls in dezimal und binär sowie den letztendlichen Code in binär an.

Aufgabe 3a)

Beispiel: **EGHEGFGGEF**

Veranschaulichen Sie die arithmetische Codierung, indem Sie die Aufteilung in Intervalle **graphisch** darstellen. Ordnen Sie die Intervalle dabei **alphabetisch** an!



E	3
F	2
G	4
H	1

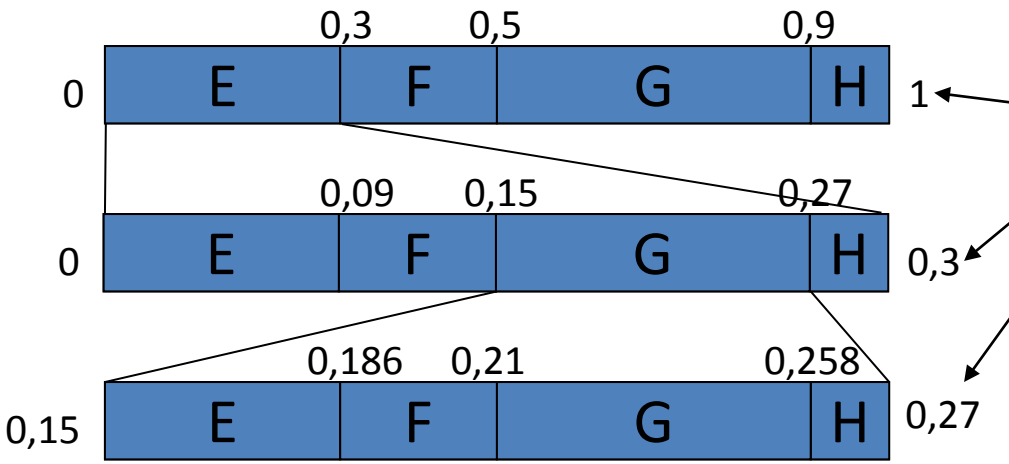
Aufgabe 3b)

Beispiel: **EGHEGFGGEF**

Codieren Sie die gesamte Nachricht mittels des Algorithmus der arithmetischen Codierung. Geben Sie für jeden Rechenschritt die obere und untere Grenze sowie die Breite des Intervalls an. Verwenden Sie dafür die Tabelle.

E	3
F	2
G	4
H	1

W	3
X	2
Y	4
Z	1



	B	L	R
E	1,0	0,0	0,3
G	0,3	0,15	0,27
H	0,12	0,258	0,27
E	0,012	0,258	0,2616
G	0,0036	0,2598	0,26124
F	0,00144	0,260232	0,26052
G	0,000288	0,260376	0,2604912
G	0,0001152	0,2604336	0.26047968
E	0.00004608	0.2604336	0.260447424
F	0.000013824	0.2604377472	0.260440512

Aufgabe 3b)

Geben Sie die obere und untere Grenze des Ergebnisintervalls in dezimal und binär so-wie den letztendlichen Code in binär an.

[0.2604377472; 0.260440512]

Untere Grenze in binär:
0.2604377472

0.2604377472 x 2 = **0**.5208754944
0.5208754944 x 2 = **1**.0417509888
0.0417509888 x 2 = **0**.0835019776
0.0835019776 x 2 = **0**.1670039552
0.1670039552 x 2 = **0**.3340079104
0.3340079104 x 2 = **0**.6680158208
0.6680158208 x 2 = **1**.3360316416
0.3360316416 x 2 = **0**.6720632832
0.6720632832 x 2 = **1**.3441265664
0.3441265664 x 2 = **0**.6882531328
0.6882531328 x 2 = **1**.3765062656
0.3765062656 x 2 = **0**.7530125312
0.7530125312 x 2 = **1**.5060250624
0.5060250624 x 2 = **1**.0120501248
0.0120501248 x 2 = **0**.0241002496
0.0241002496 x 2 = **0**.0482004992
0.0482004992 x 2 = **0**.0964009984
0.0964009984 x 2 = **0**.1928019968
0.1928019968 x 2 = **0**.3856039936

Obere Grenze in binär:
0.260440512

0.260440512 x 2 = **0**.520881024
0.520881024 x 2 = **1**.041762048
0.041762048 x 2 = **0**.083524096
0.083524096 x 2 = **0**.167048192
0.167048192 x 2 = **0**.334096384
0.334096384 x 2 = **0**.668192768
0.668192768 x 2 = **1**.336385536
0.336385536 x 2 = **0**.672771072
0.672771072 x 2 = **1**.345542144
0.345542144 x 2 = **0**.691084288
0.691084288 x 2 = **1**.382168576
0.382168576 x 2 = **0**.764337152
0.764337152 x 2 = **1**.528674304
0.528674304 x 2 = **1**.057348608
0.057348608 x 2 = **0**.114697216
0.114697216 x 2 = **0**.229394432
0.229394432 x 2 = **0**.458788864
0.458788864 x 2 = **0**.917577728
0.917577728 x 2 = **1**.835155456

→ 01000010101011000001

Übungsblatt 2

- Übungsblatt 2:
  https://www.medien.ifi.lmu.de/lehre/ws1516/dm/

- Abgabe bis Freitag den 06.11.2015, 09:00 Uhr morgens in UniWorX