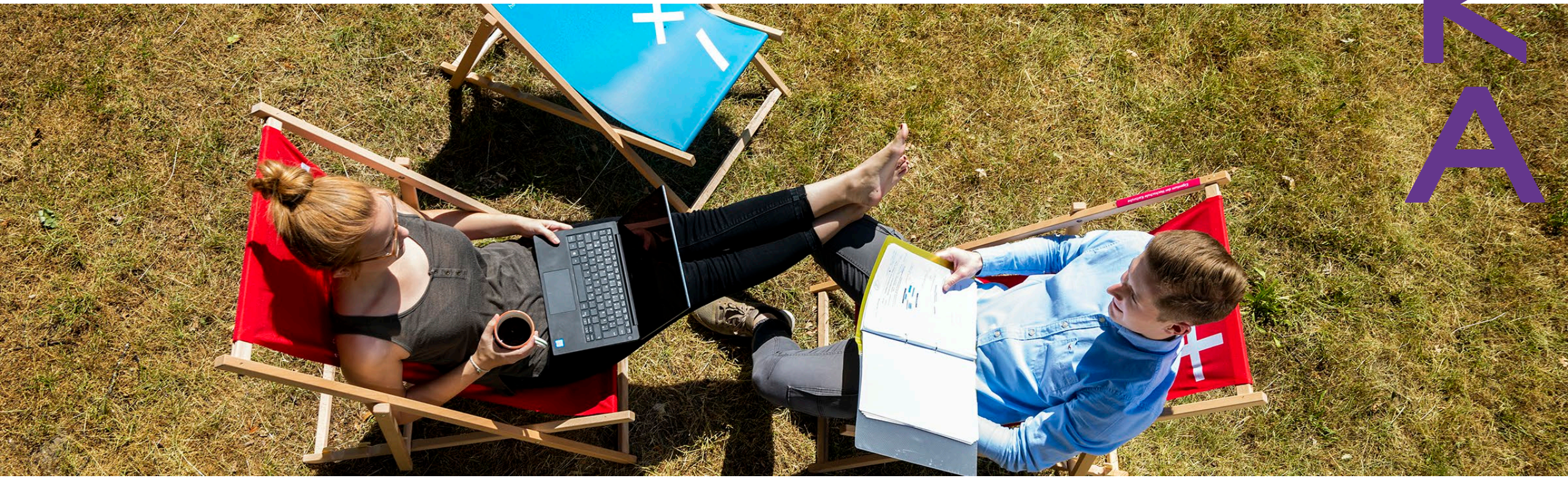
Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik

DSCB130 | 1.5 Information und Codierung



Matthias Mruzek-Vering M.Sc.

Foto: Quirin Leppert

Themenblock 1 Informationsverarbeitung

Lehrinhalte

1.5 Information und Codierung

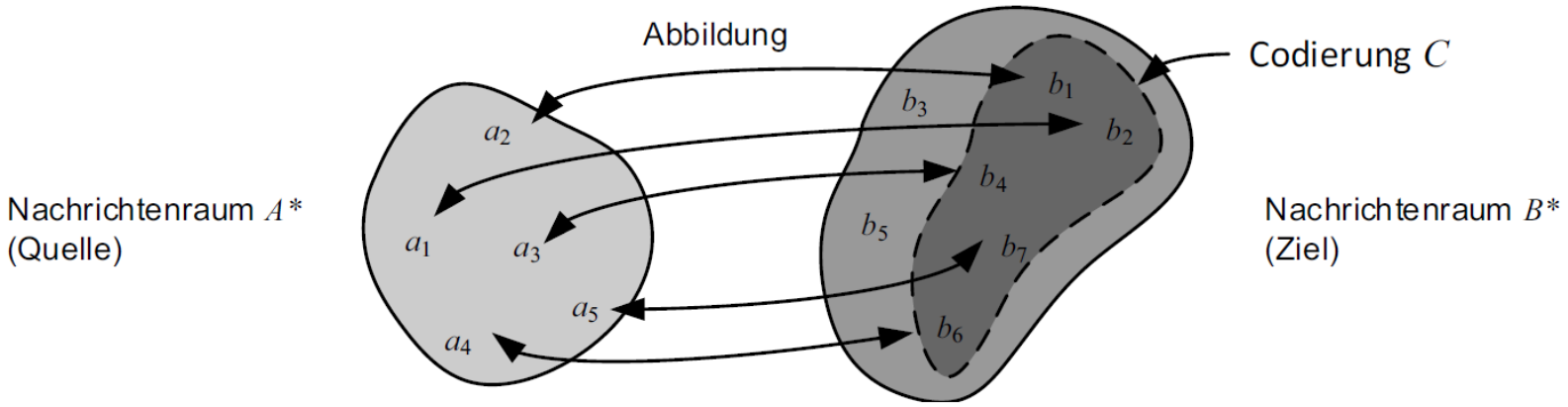
- Begriffsklärung
- Ziele und Verfahren
- Nachrichtenübertragung
- Beispiele für Codes
- Quellencodierung und Datenkompression
- Codebäume
- Der Huffman-Algorithmus
- Decodieren – Die Fano-Bedingung
- Codesicherung und Kanalcodierung
- Stellendistanz und Hamming-Distanz
- Sicherung nicht-binärer Codes
- Optische Zeichenerkennung - 2D-Barcodes oder Matrix-Codes
- Reed-Solomon Codes

1.5 Information und Codierung

Begriffsklärung

Codierung

+ Gegeben seien ein Nachrichtenraum A* (*Quelle*) über einem Alphabet $A = \{a_1, a_2, \ldots, a_n\}$ und ein Nachrichtenraum B* (*Ziel*) über einem Alphabet $B = \{b_1, b_2, \ldots, b_n\}$.
+ Eine umkehrbar eindeutige Abbildung von A^* in B^* heißt Codierung C.
  Dabei braucht nicht die gesamte Menge B^* erfasst werden.
+ Es gilt $C \subseteq B^*$, dass also C eine Teilmenge von B^* ist.



Vgl.[4] S. 69

Codierung: Eine dem Problem angepasste Darstellung einer Nachricht für dessen Speicherung und Übertragung.

1.5 Information und Codierung

Begriffsklärung

Binärcodierung

+ Zielmenge umfasst den Nachrichtenraum B^* über dem Alphabet $\{0, 1\}$
+ Verwendung aus technischen Gründen, zumindest auf unterster Ebene der Übertragungsschicht
+ Auch die Ergebnisse von Verfahren, die nicht-binäre Codes erzeugen, wie z. B. die *Reed-Solomon Codierung*, siehe [4] S. 127 ff., werden letztendlich in Binärdarstellung gespeichert.
+ Die Codierung betrifft Zeichenfolgen bzw. Wörter, aber auch Einzelzeichen.

Abgrenzung der Codierung zu Verschlüsselung

+ Ziel der Verschlüsselung ist es einem Angreifer das Lesen der Daten möglichst schwer zu machen.
+ Verschlüsselung ist nicht Thema dieser Vorlesung. Hierzu gibt es Wahlpflichtfächer.

1.5 Information und Codierung

Ziel und Verfahren

Ziele

+ Erlaubt die Darstellung zu sendender Daten mit möglichst wenigen Zeichen (*Quellencodierung*)
+ Möglichst unempfindlich gegen Störungen (*Kanalcodierung*)
+ In einem Computer leicht zu verarbeiten

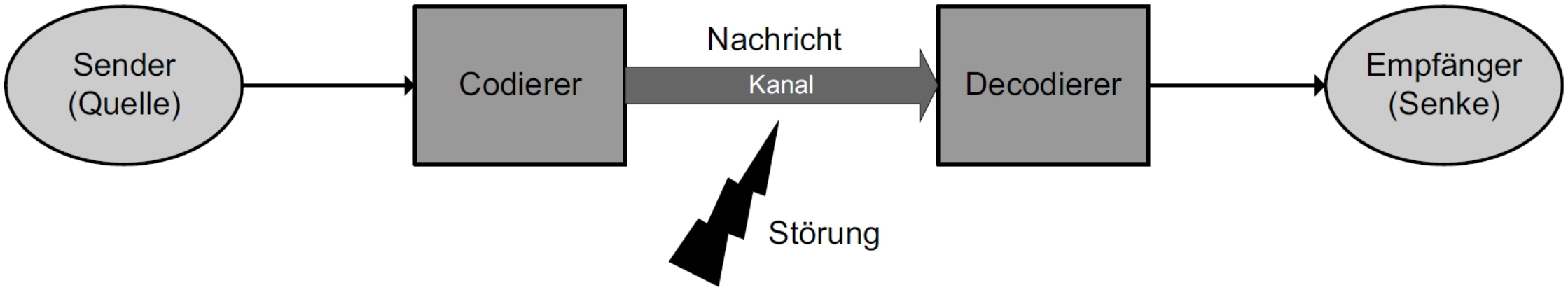
Verfahren

1. Zur Übertragung bzw. Speicherung von Daten wird üblicherweise zunächst mit einem Verfahren der **Quellencodierung** die Redundanz verkleinert.
2. Anschließend wird der entstandene Code mit Hilfe einer **Kanalcodierung** gegen Fehler gesichert, wodurch sich die Redundanz gezielt erhöht. Diese trägt nicht zur eigentlichen Information bei, erlaubt aber Fehlererkennung oder sogar Fehlerkorrektur.

1.5 Information und Codierung

Nachrichtenübertragung

Schematische Darstellung der Nachrichtenübertragung unter Einbeziehung der Codierung und des dazu inversen Vorgangs der Decodierung.



Vgl. [4] S. 37

1.5 Information und Codierung

Begriffsklärung

Mittlere Wortlänge

+ Ein wesentliches Charakteristikum eines Codes ist seine mittlere**Wortlänge L**.

$$L = \sum_{i=1}^{n} p_i l_i$$

+ *Wortlänge* l_i des i-ten Zeichens bzw. Wortes im Zielcode und p_i die zugehörige *Auftrittswahrscheinlichkeit*
+ Die Summe läuft über alle n codierten Zeichen. Dies entspricht genau dem gewichteten arithmetischen Mittel.
+ Die Gewichte sind hier die Wahrscheinlichkeiten, da diese sich zu eins summieren, erreicht man automatisch eine Normierung.

1.5 Information und Codierung

Begriffsklärung

Entropie

+ Die **Entropie $\boldsymbol{H}$** ist maximal, wenn alle Zeichen mit gleicher Häufigkeit auftreten.
+ Daraus folgt die als *Shannonsches Codierungstheorem* bekannte Beziehung:

$$H \leq L$$

+ Das Gleichheitszeichen gilt genau dann, wenn alle Wahrscheinlichkeiten p_i gleich sind.
+ H ist demnach die untere Grenze der, bei einer im Sinne der Wortlängenreduktion optimalen Codierung, erzielbaren mittleren Wortlänge.
+ Es gibt Codierungen, so dass $L - H$ beliebig klein wird, wenn man sich nicht auf die Codierung von einzelnen Zeichen beschränkt, sondern Gruppen von Zeichen zusammenfasst, die möglichst übereinstimmende *Auftrittswahrscheinlichkeiten* haben.
+ Im Allgemeinen wird jedoch $L > H$ sein.

1.5 Information und Codierung

Begriffsklärung

Redundanz

+ Die **Redundanz R** ist als Differenz aus mittlerer Codewortlänge und Entropie definiert:

$$R = L - H \text{ [Bit/Zeichen]}$$

+ R gibt an, wie groß der Anteil einer Nachricht ist, der im statistischen Sinne keine Information trägt.
+ Schnelle Nachrichtenübertragung und Platz sparende Speicherung benötigt Codes mit einer geringen Redundanz
+ Andererseits kann die Redundanz auch zur **Erhöhung der Störsicherheit** beitragen, da auf Grund der Redundanz aus einer gestörten Nachricht innerhalb gewisser Grenzen die ungestörte Nachricht rekonstruierbar ist.
+ Die mittlere Wortlänge L eines Codes lässt sich verringern unter Verwendung einer **Codierung mit variabler Wortlänge**, wobei häufig auftretende Zeichen einen kurzen und selten auftretende Zeichen einen langen Code erhalten.

1.5 Information und Codierung

Begriffsklärung

Quellen-Redundanz

Die Redundanz R_Q ist als Differenz aus der maximal möglichen Entropie H_0 der Quelle und tatsächlicher Entropie H definiert:

$$R_Q = H_0 - H \text{ [Bit/Zeichen]}$$

+ Sie gibt an, wie stark der mittlere Informationsgehalt der betrachteten Quelle vom maximal möglichen bei gleichem Alphabet aber anderer Auftrittswahrscheinlichkeiten der Zeichen abweicht.
+ **Quellen-Redundanz ist praktisch gesehen von geringerer Bedeutung als die Code-Redundanz.**
+ Maximal mögliche Entropie H_0 der Quelle, wenn alle Zeichen des Alphabets gleich wahrscheinlich auftreten
+ **Die Quellen-Redundanz hängt nur von der Anzahl Zeichen des Alphabets ab.**
+ Sie ist unabhängig vom tatsächlich verwendeten Code

1.5 Information und Codierung

Begriffsklärung

Morse-Code

+ Gilt als erster technischer Code mit variabler Zeichenlänge.
+ Wurde früher zur Übertragung telegraphischer Botschaften verwendet.
+ Streng genommen kein Binärcode, da neben den beiden Zeichen Punkt . und Strich – noch die *Pause* als drittes Zeichen hinzukommt.
+ Codes mit variabler Wortlänge haben aber technische Nachteile, beispielsweise bei Speicherung und Zugriff oder bei byteweiser paralleler Übertragung.

Buchstaben								Ziffern	
a	.–	i	..	r	.–.			1	.–––––
ä	.–.–	j	.–––	s	...			2	..–––
b	–...	k	–.–	t	–			3	...––
c	–.–.	l	.–..	u	..–			4	....–
ch	––––	m	––	ü	..––			5	.....
d	–..	n	–.	v	...–			6	–....
e	.	o	–––	w	.––			7	––...
f	..–.	ö	–––.	x	–..–			8	–––..
g	––.	p	.––.	y	–.––			9	––––.
h	....	q	––.–	z	––..			0	–––––

Vgl. [4] S. 7

1.5 Information und Codierung

Beispiele für Codes

BCD-Codierung

+ Für die Codierung von Zahlen verwendet man in der Regel die hexadezimale bzw. binäre oder– vor allem im kaufmännischen Bereich – die BCD-Codierung (*Binary Coded Decimal*).
+ Die Umwandlung einer Zahl vom Dezimalsystem in einen BCD-Code funktioniert ähnlich wie die Konvertierung zwischen Hexadezimal- und Binärsystem
  + Aus einer Dezimalziffer werden vier binäre
  + die Dezimalziffern werden separat in ihren zugehörigen Binärcode umgewandelt.

Beispiel

+ Darstellung von 439 als BCD-Zahl:
  `0100 0011 1001`
+ Darstellung von 52,11 als BCD-Zahl:
+ `0101 0010,0001 0001`

Dez.	Hex.	Binär	BCD	Stibitz	Dez.	Hex.	Binär	BCD	Stibitz
0	0	0000	0000 0000	0000 0011	8	8	1000	0000 1000	0000 1011
1	1	0001	0000 0001	0000 0100	9	9	1001	0000 1001	0000 1100
2	2	0010	0000 0010	0000 0101	10	A	1010	0001 0000	0100 0011
3	3	0011	0000 0011	0000 0110	11	B	1011	0001 0001	0100 0100
4	4	0100	0000 0100	0000 0111	12	C	1100	0001 0010	0100 0101
5	5	0101	0000 0101	0000 1000	13	D	1101	0001 0011	0100 0110
6	6	0110	0000 0110	0000 1001	14	E	1110	0001 0100	0100 0111
7	7	0111	0000 0111	0000 1010	15	F	1111	0001 0101	0100 1000

Vgl. [4] S. 72

1.5 Information und Codierung

Beispiele für Codes

ASCII-Code

+ ***A****merican* ***S****tandard* ***C****ode for* ***I****nformation* ***I****nterchange*
+ Häufige Verwendung zur Codierung von Buchstaben, Ziffern und Sonderzeichen
+ 1968 unter ANSI X3.4 sowie ISO 8859/1.2 standardisiert und seither um viele nationale Zeichensätze erweitert
+ ASCII und ISO 8859-1 (*LATIN-1, Westeuropäisch*) verwenden jeweils 1 Byte.
+ **7-Bit-Zeichenkodierung: 1 Zeichen benötigt 1 Byte, Zeichenvorrat = 128**
+ Zeichen 0-32 sind nicht druckbar (Sonderzeichen) die folgenden 95 Zeichen sind druckbar
+ Die wichtigsten Sonderzeichen sind:
  - Rückschritt (8, *BACK SPACE*)
  - Horizontaler Tabulator (9, *HORIZONTAL TAB*)
  - Zeilenvorschub (10, *LINE FEED*)
  - Wagenrücklauf (13, *CARRIAGE RETURN*)
  - Leerschritt (*ugs. Leerzeichen*) (32, *SPACE*)

1.5 Information und Codierung

Beispiele für Codes

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	**!**
34	22	100010	42	**"**
35	23	100011	43	**#**
36	24	100100	44	**$**
37	25	100101	45	**%**
38	26	100110	46	**&**
39	27	100111	47	**'**
40	28	101000	50	**(**
41	29	101001	51	**)**
42	2A	101010	52	*****
43	2B	101011	53	**+**
44	2C	101100	54	**,**
45	2D	101101	55	**-**
46	2E	101110	56	**.**
47	2F	101111	57	**/**
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	**:**
59	3B	111011	73	**;**
60	3C	111100	74	**<**
61	3D	111101	75	**=**
62	3E	111110	76	**>**
63	3F	111111	77	**?**
64	40	1000000	100	**@**
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	**[**
92	5C	1011100	134	****
93	5D	1011101	135	**]**
94	5E	1011110	136	**^**
95	5F	1011111	137	**_**
96	60	1100000	140	**`**
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	**{**
124	7C	1111100	174	**|**
125	7D	1111101	175	**}**
126	7E	1111110	176	**~**
127	7F	1111111	177	*[DEL]*

A list of all the useful characters in the ASCII table. Goes up to 0x7F. Subject to change any time. von ZZT32 auf Wikimedia Commons

1.5 Information und Codierung

Beispiele für Codes

Unicode

+ Erweiterung des ASCII-Codes auf 16 bzw. 32 Bit
+ Darstellung sämtlicher Zeichensysteme in den bekannten Schriftkulturen, derzeit in v14.0 (Stand September 2021) 159 Schriftkulturen mit insgesamt 144.697 Zeichen
+ Moderne Betriebssysteme basieren intern auf dem Unicode-System.
+ Zum Unicode gehören auch zahlreiche Sonderzeichen, mathematische Formelzeichen, fernöstliche Silbensymbole sowie Angaben über die Schreibrichtung und Stellung der Zeichen.
+ Zeichen lassen sich dynamisch kombinieren, so kann das deutsche „ä“, für das es im Unicode natürlich ein eigenes Zeichen gibt, ebenfalls durch ein „a“ und zwei darüber angeordnete Punkte dargestellt werden.
+ Die Version 2.0 des Unicode-Systems ist konform zur internationalen Norm ISO/IEC 10646, die auch den standardisierten Universal Character Set (*UCS*) umfasst.

1.5 Information und Codierung

Beispiele für Codes

Unicode

- Die Umsetzung des Zeichensatzes in konkrete Codierungen erfolgt mit dem Unicode Transformation Format (*UTF*).
- Der Code existiert in verschiedenen Ausprägungen, die sich insbesondere in der benötigten Byte-Anzahl für ein einzelnes Zeichen unterscheiden.
- **UTF-32** verwendet beispielsweise immer genau **4 Byte** (4x8 = 32 Bit). Dabei wurde häufig viel **Speicherplatz verschwendet** (ASCII 1 Byte), weshalb die **flexibleren Varianten** UTF-16 (MS Windows, macOS) und UTF-8 (WWW, E-Mail, Unix) weiter verbreitet sind.
- **UTF-16** benötigt pro Zeichen 2 Byte für die „gebräuchlichen“ Symbole oder 4 Byte für die „exotischeren“ Symbole.
- **UTF-8** verwendet zur Codierung zwischen einem und vier Byte. Vorteil: Die 1-Byte Codierung stimmt exakt mit dem 127-Bit ASCII Code überein. Sie ist daher bei Verwendung von lateinischen Zeichen (*LATIN_1*) sehr effizient, im Vergleich zu 2 Byte bei UTF-16. Für andere Schriften benötigt UTF-8 dagegen typischerweise 3 Byte.
- Beispiele:

1.5 Information und Codierung

Quellencodierung und Datenkompression

Blockcodes und Huffman-Codierung

+ Ziel: Erzeugung von Codes mit möglichst kleiner Redundanz, so dass zu übertragende bzw. zu speichernde Daten möglichst klein werden.
+ Dieser Schritt wird auch häufig als **Entropiecodierung** bezeichnet.
+ Am Häufigsten werden dabei Blockcodes z. B. ASCII verwendet, die **Codewörter** haben also eine **feste Wortlänge**.
+ In der Praxis erfolgt die Codierung durch Analog/Digital-Convertern (*ADCs*), die analoge Signale in binäre Daten mit fester Wortlänge umwandeln.
+ In technischen Anwendungen hat sich dafür der Begriff Pulse-Code-Modulation (*PCM*) eingebürgert.
+ *Blockcodes* weisen eine vergleichsweise hohe Redundanz auf, die sich durch den Einsatz von Codes mit variabler Wortlänge reduzieren lässt. Solche Codes kann man beispielsweise mithilfe des *Huffman-Verfahrens* generieren.
+ Aus dieser **Redundanzminimierung** ergibt sich in vielen Fällen bereits eine beachtliche Datenkompression im Vergleich zu Blockcodes.

1.5 Information und Codierung

Quellencodierung und Datenkompression

Blockcodes und Huffman-Codierung

+ Ein Maß für die Effizienz der Datenkompression, d. h. der Kompressionsfaktor, folgt dann einfach aus einem Vergleich der mittleren Wortlänge des Huffman-Codes mit der konstanten Wortlänge des entsprechenden Blockcodes.
+ Oft wird die **sehr schnell und preiswert** durchführbare Huffman-Codierung als **letzter Schritt in mehrstufigen Kompressionsverfahren** eingesetzt z. B. in ***JPEG***.
+ Da die Kompression bei der Huffman-Codierung und ähnlichen Methoden auf einem rein statistischen Verfahren beruht, spricht man von einer **statistischen Datenkompression**.
+ Neben der Datenkompression durch Huffman-Codes stehen noch zahlreiche andere Methoden zur Verfügung.
+ Bei Auswahl oder Entwicklung eines Datenkompressionsverfahrens muss man sich jedoch auch darüber im Klaren sein, dass mit komprimierenden Codes der Aspekt der Korrigierbarkeit von Übertragungsfehlern in Konkurrenz steht.

1.5 Information und Codierung

Quellencodierung und Datenkompression

Verlustfreie Kompression

+ Ziel dieser Art von Codierung ist eine Redundanzminimierung möglichst auf null, um die zu speichernde bzw. zu übertragende Datenmenge möglichst gering zu halten und so **Übertragungszeit bzw. Speicherplatz** zu **sparen**.
+ Wesentliche Forderung ist hier, dass die in den Daten enthaltene Information ohne Änderung erhalten bleibt.
+ Beispiele für verlustfreie Speicherung: Texte, Programmcode, Rastergrafikformat *PNG*, etc.

1.5 Information und Codierung

Quellencodierung und Datenkompression

Verlustbehaftete Kompression

+ Ziel der Codierung ist eine über die verlustfreie Datenkompression hinausgehende Verringerung der Datenmenge, die Information bleibt im Wesentlichen erhalten, aber ein **gewisser Informationsverlust wird in Kauf genommen**.
+ Datenreduktion oder verlustbehafteten Datenkompression
+ Nicht in jedem Fall anwendbar. Vorteile ergeben sich bei der Verarbeitung von Messwerten, da diese immer durch **Rauschen** überlagert sind, das **keine sinnvolle Information** trägt.
+ **Bilddaten**, bei denen es meist nicht auf eine bitgenaue Darstellung ankommt, sondern nur darauf, dass der visuelle Eindruck des komprimierten Bildes sich nicht erkennbar von dem des Originalbildes unterscheidet. Bspw. *JPEG*
+ **Musik**
  - Wahlweise verlustfrei: *Apple Lossless Audio Codec (ALAC), DTS-HD, Free Lossless Audio Codec (FLAC)*
  - Verlustbehaftet: *AC-3 Dolby Digital, DTS, MP3*
+ Ein wesentlicher Vorteil verlustbehafteter Kompression ist, dass die Kompressionsraten erheblich größer sind als bei verlustfreier Kompression, und der Anwender typischerweise abwägen kann, ob höhere oder niedrigere Kompressionsraten gewünscht sind, und damit auch schlechtere bzw. bessere Qualität.

1.5 Information und Codierung

Codebäume

Erzeugung

+ Einfachste Möglichkeit, aus einem gegebenen Alphabet von Zeichen
+ mit bekannter Auftrittswahrscheinlichkeit
+ einen Code mit variabler Wortlänge zu erzeugen,
+ ist die **Bestimmung der Wortlänge aus den ganzzahlig aufgerundeten Informationsgehalten**
+ und die **Anordnung der Codewörter als Endknoten (Blätter) eines Codebaums**

Wichtig

+ Die Entropie ist allein von den Eigenschaften der Nachrichtenquelle abhängig,
+ aber nicht von der konkreten Codierung.
+ Die mittlere Wortlänge und damit auch die Redundanz hängen dagegen von der Codierung ab.

1.5 Information und Codierung

Codebäume

Einführungsbeispiel

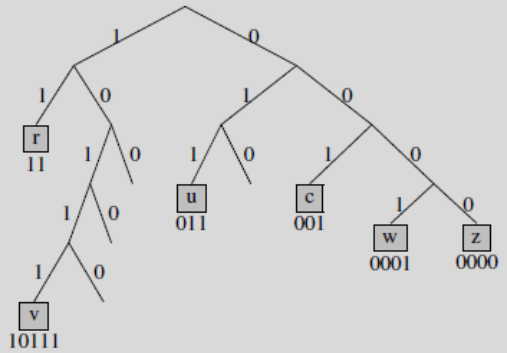
+ Auswahl von sechs Buchstaben des lateinischen Alphabets $\{c, v, w, u, r, z\}$ mit möglichst geringer Redundanz binär codieren.

Tabelle 3.3 Beispiel zur Codierung einer Auswahl von sechs Buchstaben des Alphabets (vgl. Bsp. 3.1)

x_i	$P(x_i)$	$I(x_i)$	$l(x_i)$	Codebeispiel
c	0,1638	2,6100	3	001
v	0,0454	4,4612	5	10111
w	0,0871	3,5212	4	0001
u	0,1957	2,3533	3	011
r	0,4209	1,2485	2	11
z	0,0871	3,5212	4	0000

Vgl. [4] S. 77 f.

Die Auftrittswahrscheinlichkeiten für die Buchstaben dieses Alphabets {c, v, w, u, r, z} entnimmt man Tabelle 2.3. Da man nur dieses verkürzte Alphabet betrachtet, müssen die Auftrittswahrscheinlichkeiten $P(x_i)$ noch so normiert werden, dass ihre Summe 1 ergibt. Das Ergebnis sowie die berechneten Informationsgehalte $I(x_i)$ sind in Tabelle 3.3 gezeigt. Für den in der Tabelle gezeigten Code wurden als Wortlängen die zur nächsthöheren ganzen Zahl aufgerundeten Informationsgehalte der Zeichen benutzt. Die Codewörter mit der gewünschten Wortlänge wurden durch Probieren „erfunden". Man kann diese mithilfe eines Codebaumes veranschaulichen:



Für die Entropie dieser Nachrichtenquelle berechnet man:

$$H = P(\mathrm{c}) \cdot I(\mathrm{c}) + P(\mathrm{v}) \cdot I(\mathrm{v}) + P(\mathrm{w}) \cdot I(\mathrm{w}) + P(\mathrm{u}) \cdot I(\mathrm{u}) + P(\mathrm{r}) \cdot I(\mathrm{r}) + P(\mathrm{z}) \cdot I(\mathrm{z})$$
$$\approx 0{,}4275 + 0{,}2025 + 0{,}3067 + 0{,}4605 + 0{,}5255 + 0{,}3067 = 2{,}2295 \text{ Bit/Zeichen.}$$

Die mittlere Wortlänge L dieses Codes ist:

$$L = P(\mathrm{c}) \cdot l(\mathrm{c}) + P(\mathrm{v}) \cdot l(\mathrm{v}) + P(\mathrm{w}) \cdot l(\mathrm{w}) + P(\mathrm{u}) \cdot l(\mathrm{u}) + P(\mathrm{r}) \cdot l(\mathrm{r}) + P(\mathrm{z}) \cdot l(\mathrm{z})$$
$$\approx 0{,}4914 + 0{,}2270 + 0{,}3484 + 0{,}5871 + 0{,}8418 + 0{,}3484 = 2{,}8441 \text{ Bit/Zeichen.}$$

Für die Redundanz folgt damit: $R = L - H \approx 2{,}8441 - 2{,}2295 = 0{,}5491$ Bit/Zeichen.
Es wird bei dieser Codierung also pro Zeichen im Mittel etwa ein halbes Bit verschwendet.

1.5 Information und Codierung

Der Huffman-Algorithmus

Merkmale

+ David A. Huffman (1925–1999), amerikanischer Mathematiker
+ Erfand 1952 ein Verfahren zu Optimierung eines Codes mit variabler Wortlänge hinsichtlich der Redundanz

Vorgehensweise

1. Alle Zeichen nach ihren Auftrittswahrscheinlichkeiten ordnen
2. Zusammenfassen der beiden Zeichen mit den geringsten Wahrscheinlichkeiten p_1 und p_2 zu einem Knoten
3. Diesem Knoten wird dann die Wahrscheinlichkeit $p_1 + p_2$ zugeordnet
4. Damit erhält man eine neue Folge von Wahrscheinlichkeiten, die auch den neu gebildeten Knoten mit einschließt, die Wahrscheinlichkeiten der soeben bearbeiteten Zeichen jedoch nicht mehr enthält.
5. Zusammenfassen der zu den beiden kleinsten Wahrscheinlichkeiten gehörenden Elemente, das können nun Zeichen oder Knoten sein, zu einem neuen Knoten.
6. Fortsetzen bis alle Zeichen einen Platz im so entstandenen Huffman-Baum gefunden haben.

1.5 Information und Codierung

Der Huffman-Algorithmus

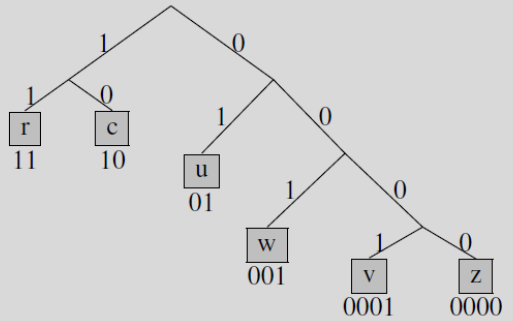
Verbesserter Code-Baum des Einführungsbeispiels

Tabelle 3.4 Verbesserte Codierung des Alphabets {c, v, w, u, r, z} (vgl. Bsp. 3.2)

x_i	$l(x_i)$	Codebeispiel
c	2	10
v	4	0001
w	3	001
u	2	01
r	2	11
z	4	0000

Bsp. 3.2 Verbesserter Codebaum

Der Codebaum aus Bsp. 3.1 weist offenbar vier Äste auf, die keine durch Codewörter besetzte Blätter tragen, wobei diese aber näher an der Wurzel liegen, als bereits besetzte. Das im Text beschriebene Optimierungsverfahren ist also anwendbar. Die neue Codetabelle ist in Tabelle 3.4 gezeigt, der Codebaum sieht wie folgt aus:



Die mittlere Wortlänge L für diesen verbesserten Code ist nun:

$$L \approx (0{,}1638 + 0{,}1957 + 0{,}4209) \cdot 2 + 0{,}0871 \cdot 3 + (0{,}0454 + 0{,}0871) \cdot 4 = 2{,}3521 \text{ Bit/Zeichen.}$$

und die zugehörige Redundanz: $R \approx 2{,}3521 - 2{,}2295 = 0{,}1226$ Bit/Zeichen.
Die Redundanz wurde also von 0,5491 auf nur noch 0.1226 Bit/Zeichen reduziert.

Vgl. [4] S. 78

1.5 Information und Codierung

Der Huffman-Algorithmus

Anwendungsbeispiel

+ Wendet man das Verfahren auf das Eingangsbeispiel mit den Wahrscheinlichkeiten nach Tabelle 3.3 an, so ergibt sich der in Abb. 3.3 dargestellte Codebaum mit dem zugehörigen Huffman-Code.
+ Wegen der von den Blättern ausgehenden, rekursiven Konstruktion des Codes, entsteht der Baum im Vergleich mit den beiden zuvor betrachteten Codebäumen **in umgekehrter Anordnung**.
+ Die mittlere Wortlänge für diesen Code ist nun:

$$L \approx 0{,}4209 + (0{,}1957 + 0{,}1638 + 0{,}0871) \cdot 3 + (0{,}0871 + 0{,}0454) \cdot 4 = 2{,}2907\ Bit/Zeichen$$

+ Die zugehörige Redundanz:

$$R \approx 2{,}2907 - 2{,}2295 = 0{,}0612\ Bit/Zeichen$$

Vgl. [4] S. 79

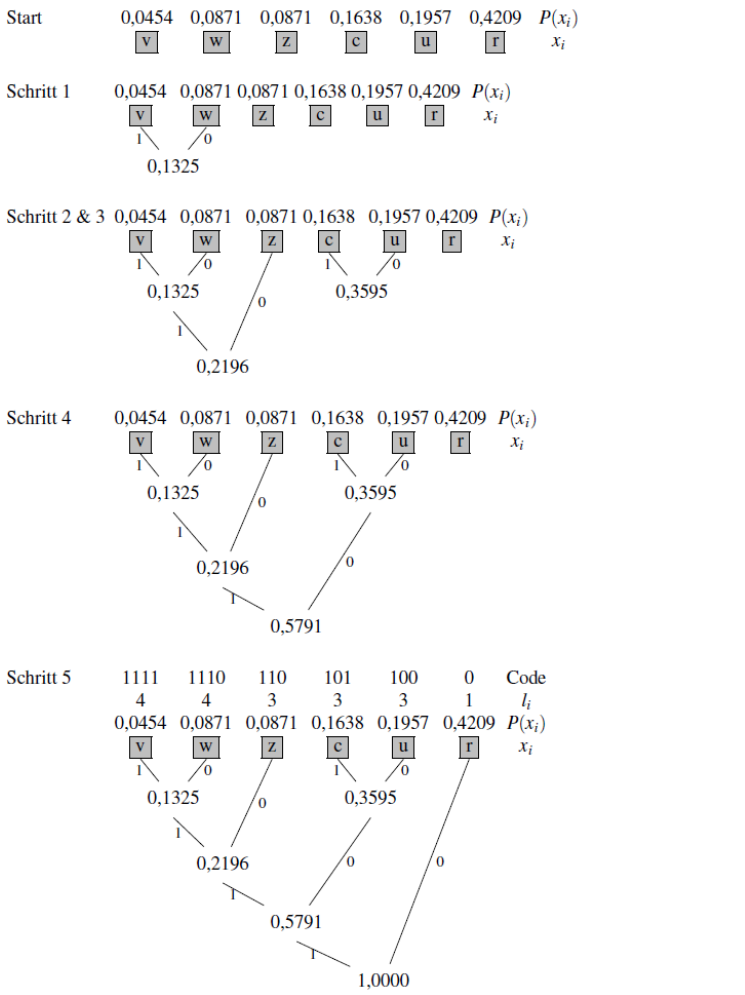


Abb. 3.3 Schrittweiser Aufbau des Huffman-Baum und Huffman-Code für das Alphabet {c, v, w, u, r, z}

1.5 Information und Codierung

Decodieren – Die Fano-Bedingung

Merkmale

+ Wesentliche Forderung an einen Code ist, dass er in eindeutiger Weisedecodierbar sein muss.
+ Bei Codes mit konstanter Wortlänge (Blockcodes) stellt dies kein Problem dar, da sich ein Wortende einfach durch Abzählen der Zeichen ermitteln lässt.
+ Die eindeutige Decodierbarkeit von Codes mit variabler Wortlänge lässt sich in folgende, von Robert Fano (*1917) formulierten, als Fano-Bedingung bekannte Forderung fassen:
  - Ein Code mit variabler Wortlänge muss so generiert werden,
  - dass kein Codewort eines Zeichens mit dem Anfang (*Präfix*) des Codewortes irgendeines anderen Zeichens übereinstimmt.
  - Mit dieser Formulierung ist gleichbedeutend, dass bei Darstellung eines Codes als Codebaum die Codewörter nur die Blätter (*Endknoten*) des Baumes besetzen dürfen, nicht aber die Verzweigungsstellen (*innere Knoten*).
  - Ein der Fano-Bedingung genügender Code, oder allgemeiner eine Sprache, heißt präfixfrei oder **Präfixcode**.

1.5 Information und Codierung

Decodieren – Die Fano-Bedingung

Merkmale

+ Bei Codes mit variabler Wortlänge, welche die Fano-Bedingung erfüllen, lässt sich eine gegebene Zeichenkette eindeutig decodieren.
+ Für die Decodierung werden die zu interpretierenden Zeichen Bit für Bit in einem Puffer gesammelt und laufend mit den tabellierten Codes verglichen.
+ Sobald der Pufferinhalt mit einem tabellierten Codewort übereinstimmt, ist das entsprechende Zeichen decodiert.
+ Der Puffer wird dann gelöscht und der Decodiervorgang beginnt von neuem für das nächste Zeichen.

Beispiel

+ Ausgehend vom Huffman-Code gemäß [4] Abb. 3.3 kann man die Zeichenfolge `10110010110001001l0` als den Text **cucuruz** identifizieren.

+ Vgl. [4] S. 81

1.5 Information und Codierung

Codesicherung und Kanalcodierung

Merkmale

+ Die Kanalcodierung erfolgt im Anschluss an die Quellencodierung
+ Ziel: Störsicherheit erhöhen
+ Oft wählt man absichtlich eine redundante Codierung, so dass sich die Codewörter zweier Zeichen (Nutzwörter) durch möglichst viele binäre Stellen von allen anderen Nutzwörtern unterscheiden.
+ Zwischen den Nutzwörtern sind also eine Anzahl von Wörtern eingeschoben, die kein Zeichen repräsentieren und demnach nur infolge einer Störung entstehen können. Dementsprechend werden sie als Fehlerwörter bezeichnet.

+ Vgl. [4] S. 102

1.5 Information und Codierung

Stellendistanz und Hamming-Distanz

Merkmale

+ Ein Maß für die Störsicherheit, also für die Fehlererkennung und Fehlerkorrektur eines Codes
+ Von Richard W. Hamming (1915–1998) eingeführte Hamming-Distanz, auch Hamming-Abstand oder Hamming-Gewicht h, die als die **minimale paarweise Stellendistanz eines Codes** definiert ist
+ Als Stellendistanz $d(x, y)$ wird dabei die Anzahl der Stellen bezeichnet, in denen sich zwei gleich lange Binärwörter x und y unterscheiden. Zur Berechnung bildet man xXORy und zählt die resultierenden Einsen.
+ Für unterschiedlich lange Codewörter ist die Stellendistanz nicht definiert.

1.5 Information und Codierung

Stellendistanz und Hamming-Distanz

Merkmale

+ Die Stellendistanz ist auch ein Maß für die bei einer Übertragung eines Wortes entstandenen Fehler.
+ Wird beispielsweise ein binäres Wort x gesendet und y empfangen, so gibt $d(x, y)$ die Anzahl der fehlerhaften Binärstellen von y an; bei korrekter Übertragung ist $x = y$ und daher $d(x, y) = 0$.
+ Die Stellendistanz erfüllt alle drei Axiome, die eine Distanz (Metrik) in einem linearen Raum definieren, nämlich:
  - **Axiom 1:** $d(x, x) = 0$,
  - **Axiom 2:** $d(x, y) = d(y, x)$ (Symmetrie)
  - **Axiom 3:** $d(x, z) \leq d(x, y) + d(y, z)$ (Dreiecksungleichung)

+ Es besteht damit eine Analogie zu anderen Distanzen, z. B. zu der in der Geometrie üblicherweise verwendeten euklidischen Distanz zwischen zwei Punkten *A* und *B* im Raum.

1.5 Information und Codierung

Stellendistanz und Hamming-Distanz

Beispiel

Gegeben seien die Ziffern 1 bis 4 in ihrer binären Codierung: $1 = 001$, $2 = 010$, $3 = 011$, $4 = 100$. Man erhält daraus folgende Stellendistanzen d von je zwei Codewörtern:

$$d(010,001) = 2$$
$$d(011,001) = 1 \quad d(011,010) = 1$$
$$d(100,001) = 2 \quad d(100,010) = 2 \quad d(100,011) = 3$$

Die Berechnung der Stellendistanzen lässt sich durch folgendes Matrix-Schema erleichtern und formalisieren:

	001	010	011	100
001	–	–	–	–
010	2	–	–	–
011	1	1	–	–
100	2	2	3	–

Die Hamming-Distanz als kleinste Stellendistanz ist in diesem Beispiel $h = 1$. Fehler lassen sich hier nicht in jedem Fall erkennen, da es offenbar Nutzwörter gibt, zwischen denen keine Fehlerwörter liegen.

+ Vgl. [4] S. 103

1.5 Information und Codierung

Sicherung nicht-binärer Codes

Berechnung einer ISBN-Prüfziffer

Für die ISBN-10-Nummer 3-528-25717-2 berechnet man zur Ermittlung der Prüfziffer $p = 2$ im ersten Schritt die gewichtete Quersumme

$10 \cdot 3 + 9 \cdot 5 + 8 \cdot 2 + 7 \cdot 8 + 6 \cdot 2 + 5 \cdot 5 + 4 \cdot 7 + 3 \cdot 1 + 2 \cdot 7 = 229.$

Anschließend bestimmt man die kleinste durch 11 teilbare Zahl, die größer als 229 ist, sie lautet offenbar 231. Man muss also 2 zu 229 addieren, um 231 zu erhalten. Die gesuchte Prüfziffer lautet also in der Tat $p = 2$.

+ Vgl. [4] S. 124

1.5 Information und Codierung

Sicherung nicht-binärer Codes

Berechnung und Validierung einer IBAN

+ In Deutschland hat sie **22 Stellen** und den Aufbau DE *p1p2 b1b2b3b4b5b6b7b8 k1k2k3k4k5k6k7k8k9k10*
+ wobei *k1k2..k10* und *b1b2..b8* die ehemalige Kontonummer bzw. Bankleitzahl ist und *p1p2* Prüfziffern
+ Zur Berechnung der Prüfziffern werden zunächst Länderkürzel und die auf Nullen gesetzten Prüfziffern ganz nach rechts gestellt und anschließend die Buchstaben der Länderkennung durch ihre Position im Alphabet plus 9 ersetzt: A=10, B=11, etc.
+ Für die so entstandene Zahl wird der Rest bei Division durch 97 berechnet und die Ziffern dann so festgelegt, dass sich bei der Prüfung der korrekten IBAN eins ergibt.
+ Beispiel:

Für die deutsche Bankleitzahl 711 500 00 und Kontonummer 215 632 soll die IBAN berechnet werden. Kombination der beiden Zahlen mit rechts angehängtem Länderkürzel und Prüfziffern Null lautet: 711 500 00 0000 215 632 DE 00.
Nach dem Ersetzen des Kürzels DE mit den Positionen im Alphabet plus 9 (D=13, E=14) erhält man: 711 500 00 0000 215 632 131400.
Der Rest bei Division durch 97 liefert: $711\,500\,00\,0000\,215\,632\,131\,400 \bmod 97 = 49$.
Die gesuchten Prüfziffern lauten also $98 - 49 = 49$, die gesamte IBAN ergibt sich zu DE 49 711 500 00 0000 215 632.

Eine Validierung der IBAN liefert $711\,500\,00\,0000\,215\,632\,131\,449 \bmod 97 = 1$, diese ist also korrekt.

+ Vgl. [4] S. 125

1.5 Information und Codierung

Optische Zeichenerkennung

Merkmale

+ Konstruktion genormter, durch optische Zeichenerkennung lesbarer Balkenschriften sind ebenfalls redundante und fehlertolerante Codierungen gebräuchlich
+ Je zwei Zeichen unterscheiden sich durch wesentliche geometrische Details
+ Beispiele: **OCR-Normschriften** (*Optical Character Recognition*)
+ Auch bei nur teilweise sichtbaren Zeichen, bspw. in abgenutzten oder verschmutzten Autokennzeichen, ist mit **Methoden der digitalen Bildverarbeitung** und **Mustererkennung** noch eine robuste Erkennbarkeit gewährleistet.
+ Beispiele:

(b) Zeichensatz der 1973 gemäß ISO 1073-2 genormten maschinenlesbaren Schrift OCR-B (Bild gemeinfrei)

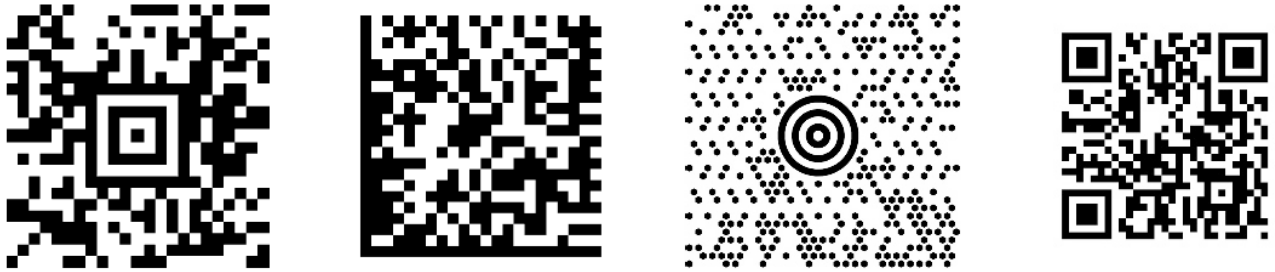
+ Vgl. [4] S. 126, https://de.wikipedia.org/wiki/Consolas, abgerufen am 19.10.2022

1.5 Information und Codierung

2D-Barcodes oder Matrix-Codes

Merkmale

+ 2D-Barcodes oder Matrix-Codes arbeiten vom Grundprinzip her ähnlich wie Strichcodes, aber auf zwei Dimensionen erweitert.
+ Typisch ist die Verwendung unterschiedlich breiter Punkte und Striche mit Lücken dazwischen, die **einen hohen Kontrast beim Auslesen** mit Laserscanner oder Kamera liefern und mit Methoden der digitalen Bildverarbeitung analysiert werden.
+ Es gibt verschiedene Varianten.
+ Beispiele:



(a) Aztec-Code **(b)** DataMatrix-Code **(c)** MaxiCode **(d)** QR-Code

Abb. 3.16 Beispiele für 2D-Barcodes (Matrix-Codes) (Bilder gemeinfrei)

+ Vgl. [4] S. 126 f.

1.5 Information und Codierung

Reed-Solomon Codes

Merkmale

+ Verwendung zur Fehlerkorrektur
+ Irving Reed und Gustave Solomon veröffentlichten 1960 einen Artikel, in dem die nach Ihnen benannten Codes Codes beschrieben wurden
+ Mittlerweile sind diese Codes weit verbreitet und werden nicht nur bei vielen Matrix-Codes eingesetzt,

Konstruktion eines Reed-Solomon Codes $RS(q, m, n)$

- Wähle einen endlichen Körper $\mathbb{F}_q$ mit $q = p^k$ Elementen als Alphabet, mit p prim, $k \in \mathbb{N}$.
- Fasse die Nachricht (Block aus m Zahlen) $\boldsymbol{a} = (a_0, a_1, \ldots, a_{m-1})$ als Polynom $p(x)$ über $\mathbb{F}_q$ auf:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} \quad .$$

- Wähle n paarweise verschiedene Elemente $(n \geq m)$ $u_0, \ldots, u_{n-1} \in \mathbb{F}_q$. Da es im Körper $\mathbb{F}_q$ nur q Elemente gibt, gilt $n \leq q$.

+ [4] S. 127 ff.

1.5 Information und Codierung

Reed-Solomon Codes

Beispiele

+ Audio-CD
  - Zwei hintereinander geschaltete Reed-Solomon Codes + räumliche Verteilung der einzelnen Codewörter, sog. Cross-Interleaved Reed-Solomon Coding (CIRC).
  - Die beiden Codes sind RS(33, 28, 32) und RS(29, 24, 28).
  - Lesefehler auf der CD werden als Ausfälle behandelt. Hiermit sind Bündelfehler von bis zu ca. 4.000 Bit, entspricht etwa einem kreisförmigen Kratzer der Länge 2,5 mm, exakt korrigierbar.
+ DVD
  - Verfahren ähnlich wie bei Audio-CD, allerdings werden wegen der höheren Datendichte größere Codes eingesetzt, nämlich RS(209, 192, 208) und RS(183, 172, 182).
+ Blu-Ray Disc
  - Ähnlich wie bei DVD mit noch längeren Codes.
  - Diese Codes sind eine Untermenge der BCH-Codes, also zyklische, nicht-binäre Codes, die die Erkennung und Korrektur von zufälligen Mehrfachfehlern und Bündelfehlern sowie die Rekonstruktion von fehlenden Daten (Auslöschungen) erlauben.

1.5 Information und Codierung

Reed-Solomon Codes

Bsp. 3.25 Reed-Solomon: Codierung mit RS(5, 3, 5) im Körper $\mathbb{F}_5$

Betrachtet wird nun der Code RS(5, 3, 5), also im Körper $\mathbb{F}_5$, der durch Rechnung modulo 5 entsteht. Die Nachricht hat Länge drei, und das Polynom wird an fünf Stellen ausgewertet (die maximal mögliche Anzahl). Zu senden sei die Nachricht $\boldsymbol{a} = (1, 1, 2)$. Diese entspricht dem Polynom $p(x) = 1 + x + 2x^2$. Zur Codierung wird $p(x)$ an $n = 5$ Stellen ausgewertet (man beachte die Rechnung modulo 5):

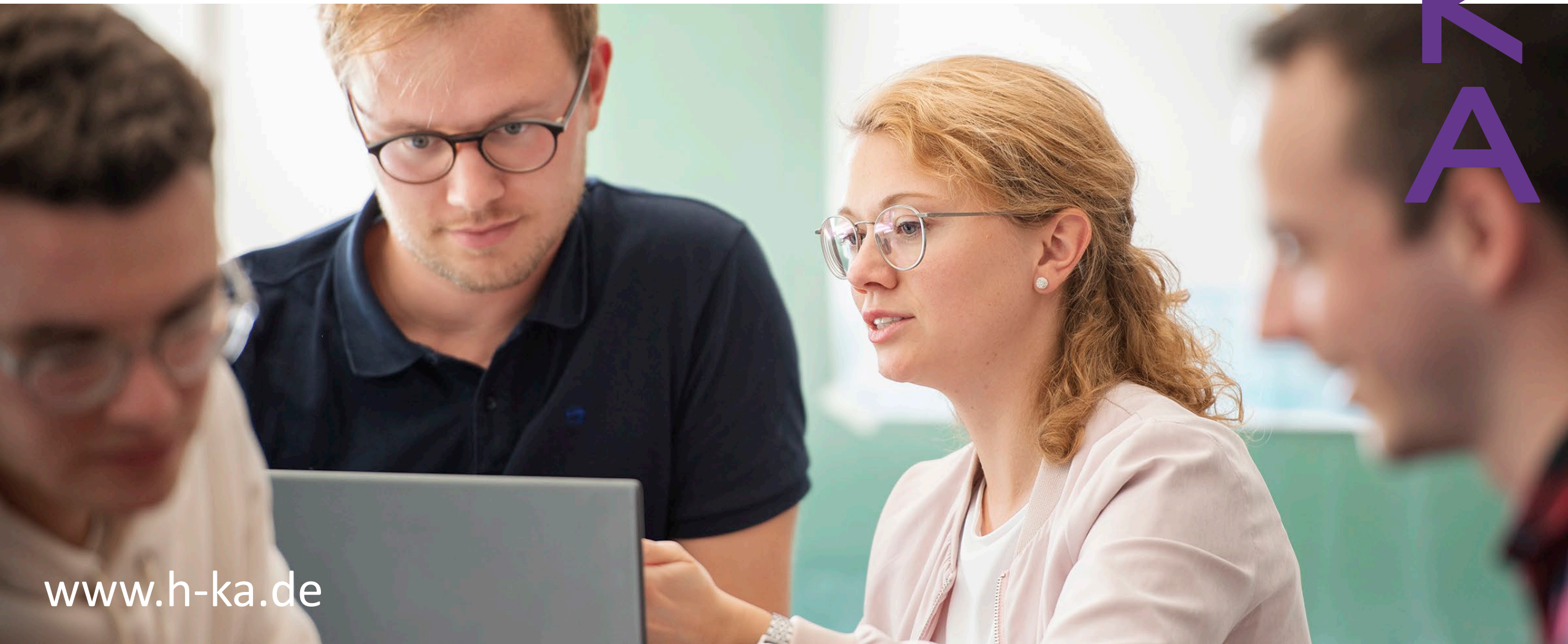
$$
\begin{aligned}
p(0) &= 1 + 0 + 0 &&= 1 \\
p(1) &= 1 + 1 + 2 &&= 4 \\
p(2) &= 1 + 2 + 8 = 11 &&= 1 \\
p(3) &= 1 + 3 + 18 = 22 &&= 2 \\
p(4) &= 1 + 4 + 32 = 37 &&= 2
\end{aligned}
$$

Das zu sendende Codewort lautet damit $\boldsymbol{c} = (1, 4, 1, 2, 2)$.

+ Reed-Solomon Decodierung, Ausfälle und Fehlerkorrektur werde in der VL nicht behandelt.
  Hierzu siehe Literatur [4] S. 129 f.

Hochschule Karlsruhe
University of
Applied Sciences

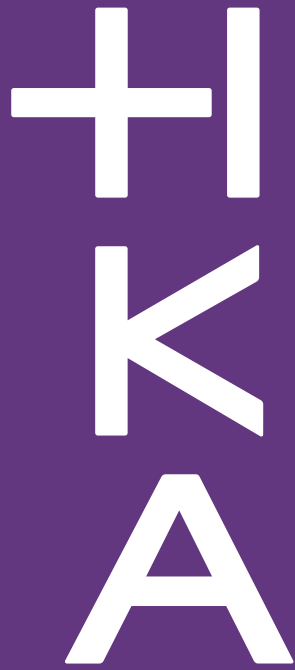
Fakultät für
Informatik und
Wirtschaftsinformatik



www.h-ka.de

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik



www.h-ka.de