

Kapitel 2 – Kodierung

1. Kodierung von Zeichen

2. Kodierung von Zahlen

3. Anwendung: ReTI

Albert-Ludwigs-Universität Freiburg

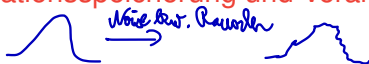
Prof. Dr. Christoph Scholl

Institut für Informatik

WS 2015/16

Motivation

- Ein Rechner speichert, verarbeitet und produziert Informationen.
- Alle Ergebnisse müssen als Funktion der Anfangswerte exakt reproduzierbar sein.
- Informationsspeicherung und Verarbeitung müssen exakt sein.



- Probleme: Noise, Crosstalk, Abschwächung
- Es gibt keine exakte Datenübertragung oder Datenspeicherung.

- Ziel: Quantisierung der Informationsspeicherung mit Signal groß gegenüber maximaler Störung



- Binär-Codierung (nur zwei Zustände) ist die einfachste (und sicherste) Signal-Quantisierung.
- BIT (0, 1) als grundlegende Informationseinheit

$5V$ --- $\}$ interpretiert als logische 1
 V_H --- $\}$

 V_L --- $\}$ interpretiert als logische 0
 $0V$ $\underline{\curvearrowright}$ $\}$
} Binärfizierung

$5V$ --- $\}$ $\hat{=}$ 9

 \vdots

 --- $\}$ $\hat{=}$ 2
 --- $\}$ $\hat{=}$ 1
 --- $\}$ $\hat{=}$ 0
 $0V$ ---

- Ein Rechner kann üblicherweise
 - Zeichen verarbeiten (Textverarbeitung)
 - mit Zahlen rechnen
 - Bilder, Audio- und Videoinformationen verarbeiten und darstellen ...
 - Ein Algorithmus kann zwar prinzipiell mit abstrakten Objekten verschiedener Art operieren, aber diese müssen im Rechner letztendlich als Folgen von Bits repräsentiert werden.
- **Kodierung!**

- Wie werden im Rechner Zeichen dargestellt ?
- Codes fester Länge
- “Längenoptimale Kodierungen” von Zeichen:
Häufigkeitscodes (Bsp.: Huffman-Code)

Definition

Eine nichtleere Menge $A = \{a_1, \dots, a_m\}$ heißt (endliches) **Alphabet** der Größe m .

a_1, \dots, a_m heißen **Zeichen** des Alphabets.

- $A^* = \{w \mid w = b_1 \dots b_n \text{ mit } n \in \mathbb{N}, \forall i \text{ mit } 1 \leq i \leq n : b_i \in A\}$ ist die **Menge aller Wörter** über dem Alphabet A .
- $|b_1 \dots b_n| := n$ heißt **Länge** des Wortes $b_1 \dots b_n$.
- Das Wort der Länge 0 wird mit ε bezeichnet.

Beispiel:

Sei $A = \{a, b, c, d\}$. $A^* = \{\varepsilon, a, b, c, d, aa, ab, ac, \dots\}$

Dann ist $bcada$ ein Wort der Länge 5 über A .

↑
(leeres Wort)

Sei $A = \{a_1, \dots, a_m\}$ ein endliches Alphabet der Größe m .

- Eine Abbildung $c: A \rightarrow \{0, 1\}^*$ oder $c: A \rightarrow \{0, 1\}^n$ heißt **Code**, falls c injektiv ist.

Werte über $\{0, 1\}$ der Länge n

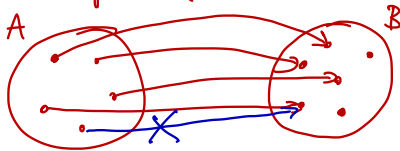
- Die Menge $c(A) := \{w \in \{0, 1\}^* \mid \exists a \in A: c(a) = w\}$ heißt Menge der **Codewörter**.

- Ein Code $c: A \rightarrow \{0, 1\}^n$ heißt **Code fester Länge**.

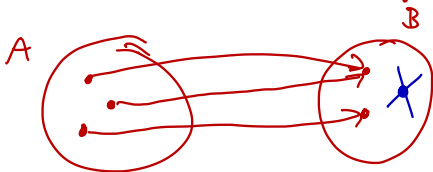
- Für einen Code $c: A \rightarrow \{0, 1\}^n$ fester Länge n gilt:
 $n \geq \lceil \log_2 m \rceil$.

- Ist $n = \lceil \log_2 m \rceil + r$ mit $r > 0$, so können die r zusätzlichen Bits zum Test auf **Übertragungsfehler** verwendet werden (siehe Kap. 6).

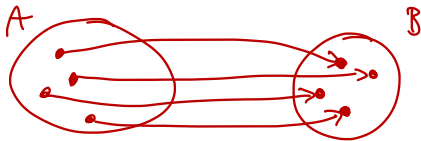
- $\kappa: A \rightarrow B$ ist injektiv, falls $\forall a_1, a_2 \in A$ gilt: $\kappa(a_1) = \kappa(a_2) \Rightarrow a_1 = a_2$



- $\kappa: A \rightarrow B$ ist surjektiv, falls $\forall b \in B$ gilt: $\exists a \in A$ mit $\kappa(a) = b$

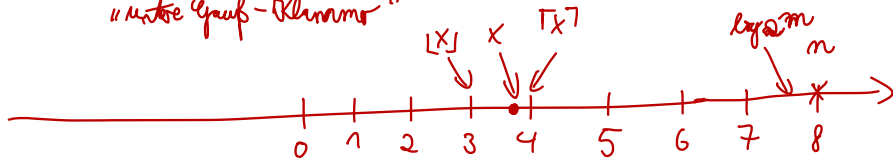


- $\kappa: A \rightarrow B$ ist bijektiv, falls κ injektiv und surjektiv ist.



Für $x \in \mathbb{R}$ ist $\lceil x \rceil$ ist die kleinste ganze Zahl, die größer gleich x ist.
 „obere Gauß-Klammer“

Für $x \in \mathbb{R}$ ist $\lfloor x \rfloor$ „ „ größte „ „ , die kleiner gleich x ist.
 „untere Gauß-Klammer“



Bezuhr: Bei Code $c: A \rightarrow \{0,1\}^n$ muss gelten: $n \geq \lceil \log_2 m \rceil$

$$m \leq 2^n \quad (\text{wegen } c \text{ injektiv!})$$

$$\log_2 m \leq n$$

$\lceil \log_2 m \rceil \leq n$, da n eine natürliche Zahl ist.



m Elemente



2^n Elemente

- Die Kodierung eines jeden Zeichens besteht aus n Bits.
 - ASCII (American Standard Code for Information Interchange): 7 Bits (es gibt Erweiterungen mit 8 Bits)
 - EBCDIC: 8 Bits
 - Unicode: 16 Bits
- Diese Kodierungen sind recht einfach zu behandeln. Unter Umständen wird für sie aber mehr Speicherplatz gebraucht als unbedingt nötig.

Beispiel: ASCII-Tabelle

		erste 3 Bits							
		0	0	0	0	1	1	1	
		0	0	1	1	0	1	1	
		0	1	0	1	0	1	1	
letzte 4 Bits	0000	nul	dle		0	@	P	p	
	0001	soh	dc1	!	1	A	Q	q	
	0010	sfx	dc2	"	2	B	R	r	
	0011	etx	dc3	#	3	C	S	s	
	0100	eot	dc4	\$	4	D	T	t	
	0101	enq	nak	%	5	E	U	u	
	0110	ack	syn	&	6	F	V	v	
	0111	bel	etb	'	7	G	W	w	
	1000	bs	can	(8	H	X	x	
	1001	ht	em)	9	I	Y	y	
	1010	lf	sub	*	:	J	Z	z	
	1011	vt	esc	+	;	K	[{	
	1100	ff	fs	,	<	L	\		
	1101	cr	qs	-	=	M]	}	
	1110	so	rs	.	>	N	^	*	
	1111	si	us	/	?	O	_	del	
		Steuerzeichen				Schriftzeichen			

$$c(S) = 10100111$$

$$c(R) = 0111000$$

Häufigkeitsabhängige Codes

- **Ziel:** Reduktion der Länge einer Nachricht durch Wahl **verschieden langer Codewörter** für die verschiedenen Zeichen eines Alphabets (also **kein** Code fester Länge!)
- **Idee:** Häufiges Zeichen → kurzer Code
Seltenes Zeichen → langer Code
- **Voraussetzungen:**
 - Häufigkeitsverteilung ist bekannt → **statische Kompression**
 - Häufigkeitsverteilung ist nicht bekannt → **dynamische Kompression**

*Bsp.: Deutsche Sprache : c(e) = 01
c(x) = 101111101*

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

- Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

← alphabet

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

9

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

11

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

13 10 9

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

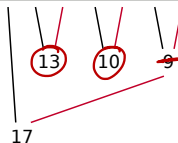
Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

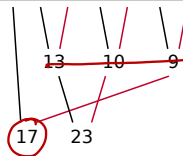


Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

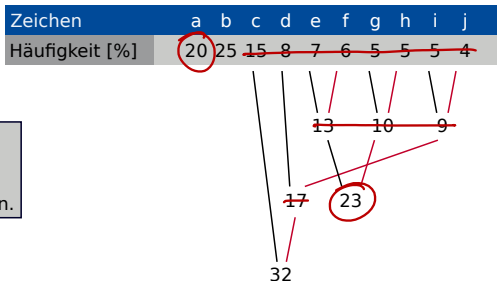


Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:



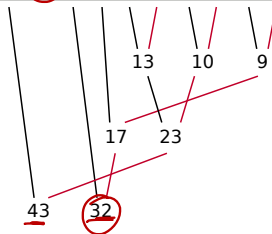
Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4



Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

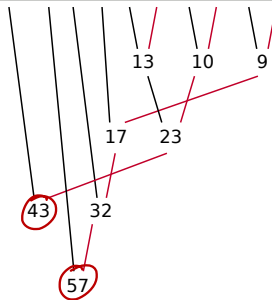
Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.



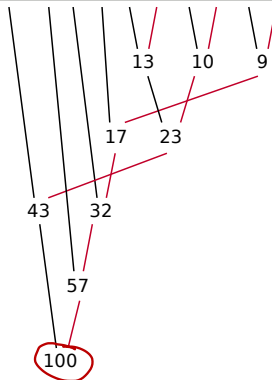
Huffman-Code

- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.



Huffman-Code

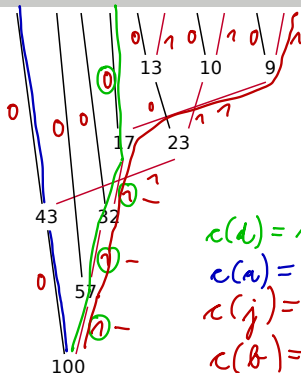
- Der **Huffman-Code** ist der bekannteste häufigkeitsabhängige Code.
- Kommt als Teilschritt z.B. in MP3 oder JPEG vor.

■ Beispiel:

Zeichen	a	b	c	d	e	f	g	h	i	j
Häufigkeit [%]	20	25	15	8	7	6	5	5	5	4

Baue binären Baum, indem die beiden kleinsten Häufigkeiten jeweils zu einem neuen Knoten addiert werden.

Markiere nun die linken Kanten mit 0 und die rechten Kanten mit 1, fertig ist der Huffman-Code!



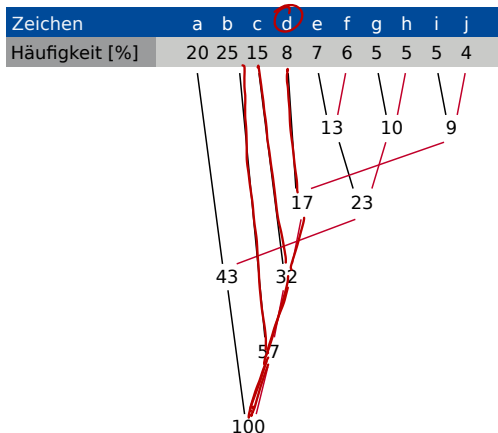
$c(d) = 1110$

$c(a) = 00$

$c(j) = 11111$

$c(b) = 10$

Erzeugte Huffman-Kodierung



Erzeugte Kodierung:

a	b	c	d	e	f	g	h	i	j
00	10	110	1110	0100	0101	0110	0111	11110	11111


Handwritten note: $a, c, d \rightarrow 101101110$

Huffman-Code: Dekodierung

Erzeugte Kodierung:

a	b	c	d	e	f	g	h	i	j
00	10	110	1110	0100	0101	0110	0111	11110	11111

- 1 Lesen des Bitstromes bis Symbol erkannt wurde.
- 2 Erkanntes Symbol ausgeben und weiter mit 1.

$$c(a) = \underline{01} \quad c(b) = \underline{01000} \quad c(d) = 000$$


Definition

Sei A ein Alphabet der Größe m .

- $a_1 \dots a_p \in A^*$ heißt Präfix von $b_1 \dots b_l \in A^*$, falls $p \leq l$ und $a_i = b_i \forall i, 1 \leq i \leq p$.
- Ein Code $c : A \rightarrow \{0, 1\}^*$ heißt Präfixcode, falls es kein Paar $i, j \in \{1, \dots, m\}$ gibt, so dass $c(a_i)$ Präfix von $c(a_j)$.
 - Der Huffman-Code ist ein Präfixcode.
 - Bei Präfixcodes können Wörter über $\{0, 1\}$ eindeutig dekodiert werden. (Sie entsprechen Binärbäumen mit Codewörtern an den Blättern.)
 - Huffman-Code ist ein bzgl. mittlerer Codelänge optimaler Präfixcode (unter Voraussetzung einer bekannten Häufigkeitsverteilung) - ohne Beweis.

Optimalität:

Sei $A = \{a_1, \dots, a_m\}$

Häufigkeitsverteilung p mit $\sum_{i=1}^m p(a_i) = 1$

Mittlere Codelänge eines Codes $c: A \rightarrow \{0,1\}^*$:

$$\sum_{i=1}^m p(a_i) \cdot |c(a_i)|$$

Beispiele: Präfixcodes

Frage: Welche dieser Codes sind Präfixcodes

- a. $c('A') = \underline{01}$, $c('B') = 110$, $c('C') = \underline{011}$ → *kein Präfixcode*
- b. $c('A') = 01$, $c('B') = 110$, $c('C') = 111$ → *Präfixcode*
- c. $c('1') = xz$, $c('2') = xy$, $c('3') = yz$ → *überhaupt kein Code!*
- ~~d. Keiner der Obigen.~~

- Es gibt zahlreiche Ansätze zur Datenkompression.

(Beispiel: Lempel-Ziv-Welch.)

↑ Folgen von Zeichen werden mit Codewörtern kodiert.

- In Programmtexten gibt es häufig viele Leerzeichen, gleiche Schlüsselwörter und so weiter.

→ Kodiere Folgen von Leerzeichen bzw. Schlüsselwörter durch kurze Codes.

- Das wird z.B. bei GIF und TIFF genutzt.
- Das soll auch funktionieren, wenn man noch nicht weiß, welche Zeichenketten häufig vorkommen.