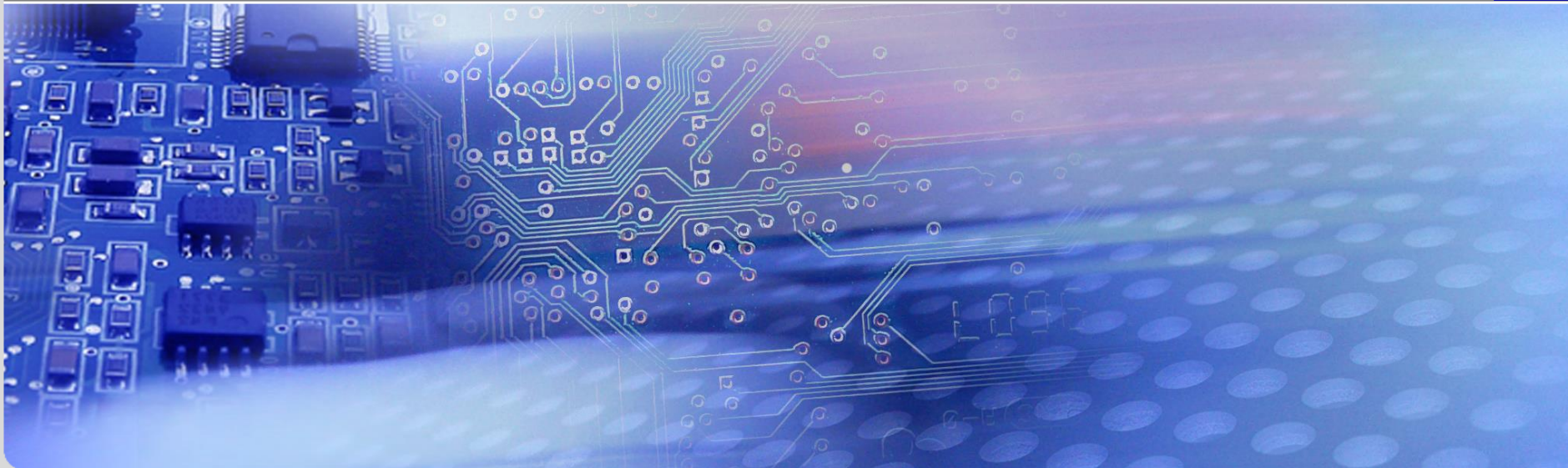


Digitaltechnik Tutorium 2

Institut für Technik der Informationsverarbeitung (ITIV)

ITIV



Nachtrag Fehlererkennung & -korrektur

- $HD_{min} = 3$
kann man [2 bit-Fehler erkennen] ODER [1 bit-Fehler korrigieren]
nicht beides!
- Bei $HD_{min} = 4$
kann man [2 bit-Fehler erkennen] UND ein 1 bit-Fehler korrigieren]
ODER [3 bit-Fehler erkennen]
nicht beides!

Beispiel

- $HD = 3$
- z.B.: gültige Codewort (gCW): $000 \rightarrow 111$
- 000 (gCW) $\rightarrow 001$ (oder $010, 100$) (uCW) $\rightarrow 011$ (oder $110, 101$) (uCW) $\rightarrow 111$ (gCW)
- Festlegen: 1 Bit Fehler korrigieren
 - $\rightarrow 001$ (oder $010, 100$) zu 000 korrigiert
 - $\rightarrow 011$ (oder $110, 101$) zu 111 korrigiert

ODER

- Festlegen 2 Bit Fehler erkennen
man sieht $001, 010, 100, 011, 110, 101$ sind uCW
kann aber nicht sehen, ob 2 oder nur 1 Fehler passiert sind
und kann deshalb nicht auf das gültige Codewort rückschließen

Beispiel 2

- $HD = 4$ ODER wird zu UND
- z.B.: gCW: 0000 \rightarrow 1111
- 0000 (gCW) \rightarrow 0001 (oder 0010, 0100, 1000) (uCW) \rightarrow 0011 (...)
(uCW) \rightarrow 0111 (...) (uCW) \rightarrow 1111 (gCW)
- 1 Bit Fehler korrigieren (0001 \rightarrow 0000; 0111 \rightarrow 1111)
UND
- 2 Bit Fehler erkennen, aber nicht sagen was richtige CW ist
- 3 Bit Fehler dürfte es dabei nicht geben, da sie falsch korrigiert werden würden (wegen 1 Bit Fehler korrigieren)
ODER
- 3 Bit Fehler erkennen, dann kann man z.B. 0001 erkennen, aber nicht sagen ob es 1 Fehler von 0000 ist oder 3 Fehler von 1111 und somit keine Fehler korrigieren.

Optimale Codes

- Optimierung von Codes durch weniger Speicherbelegung
→ z.B. Shannon-Fano oder Huffman

- Bisher:

z.B. 6 Codewörter (a,b,c,d,e,f)

wie viele bits? → $\lceil \ln 6 \rceil = 3$

mögliche Zuordnung:

a 000

b 001

c 010 →

d 011

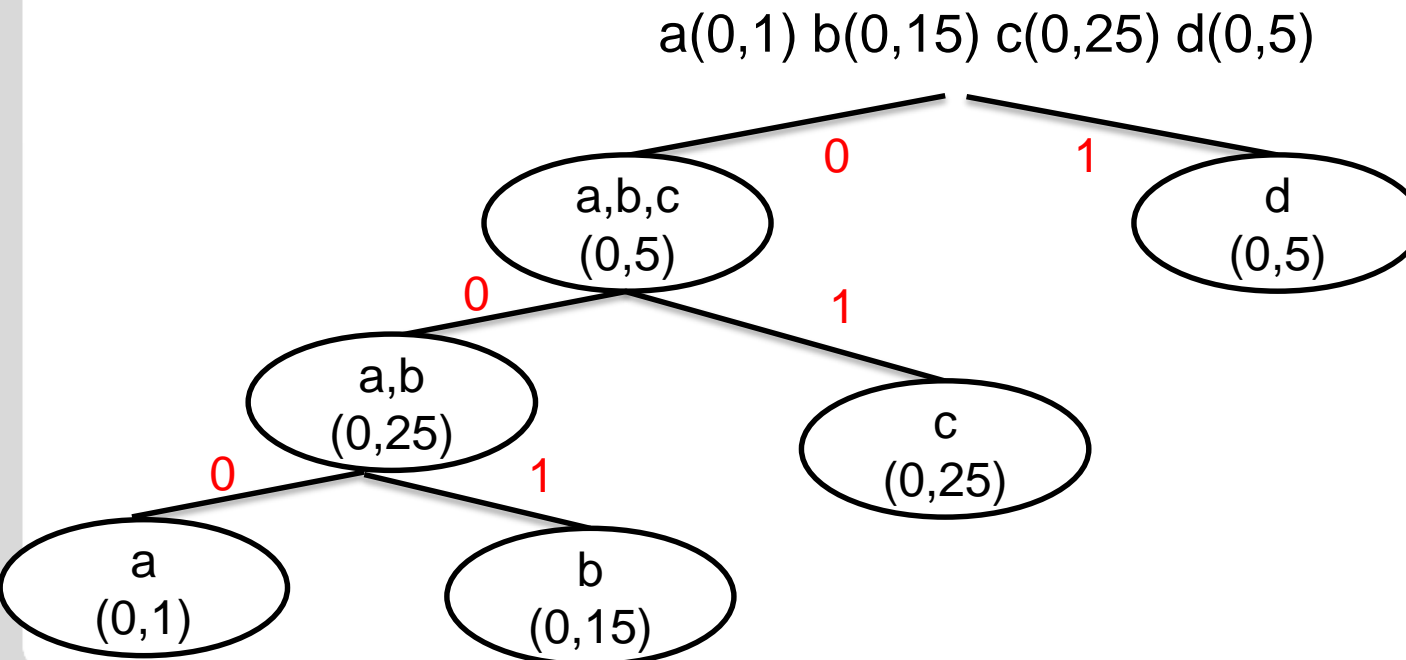
e 100

f 101

durchschnittliche Anzahl
an Bits: 3

Shannon-Fano-Codierung

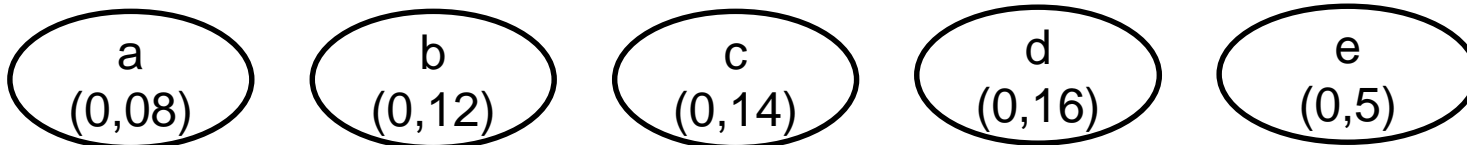
- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
Reihenfolge bleibt stets bestehen
- möglichst nach gleichen Wahrscheinlichkeiten aufteilen
- Links: 0 rechts: 1
- Beispiel:



a: 000
b: 001
c: 01
d: 1

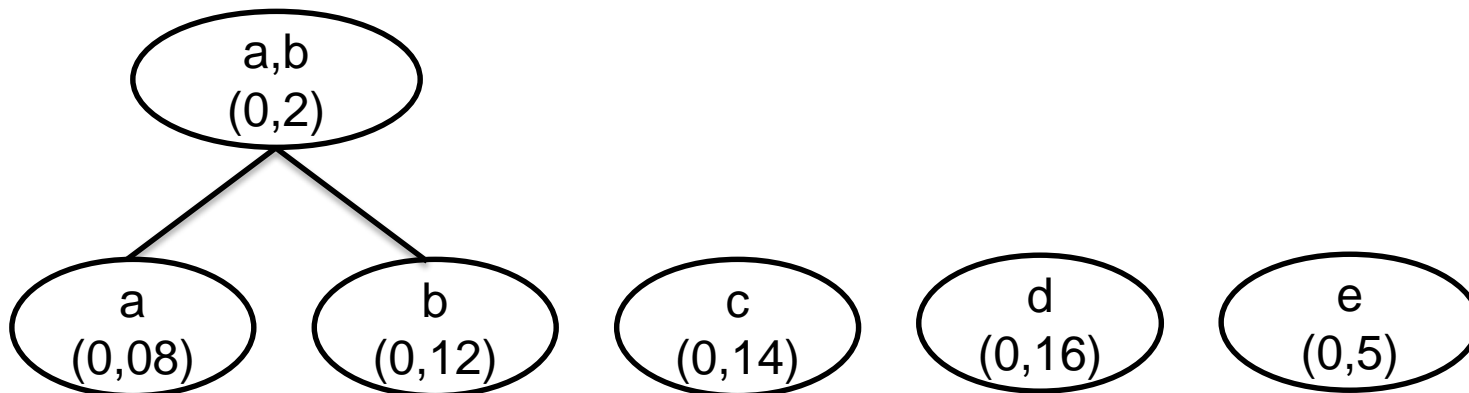
Huffman-Codierung

- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
- Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
- Links: 0 Rechts: 1
- Beispiel:



Huffman-Codierung

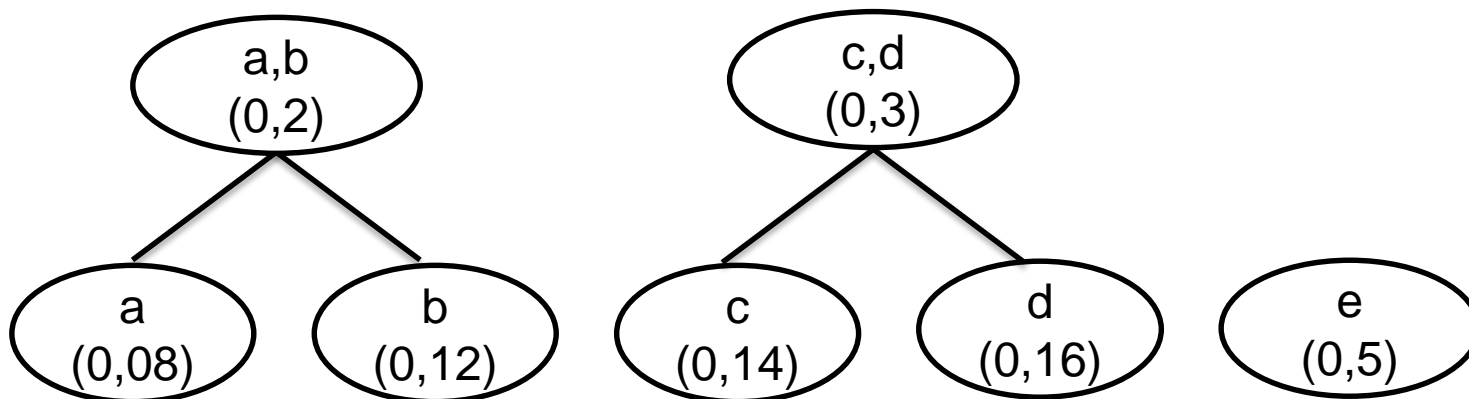
- Daten müssen gewichtet sein (Wahrscheinlichkeit)
 - Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
 - Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
 - Links: 0 Rechts: 1
-
- Beispiel:



Huffman-Codierung

- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
- Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
- Links: 0 Rechts: 1

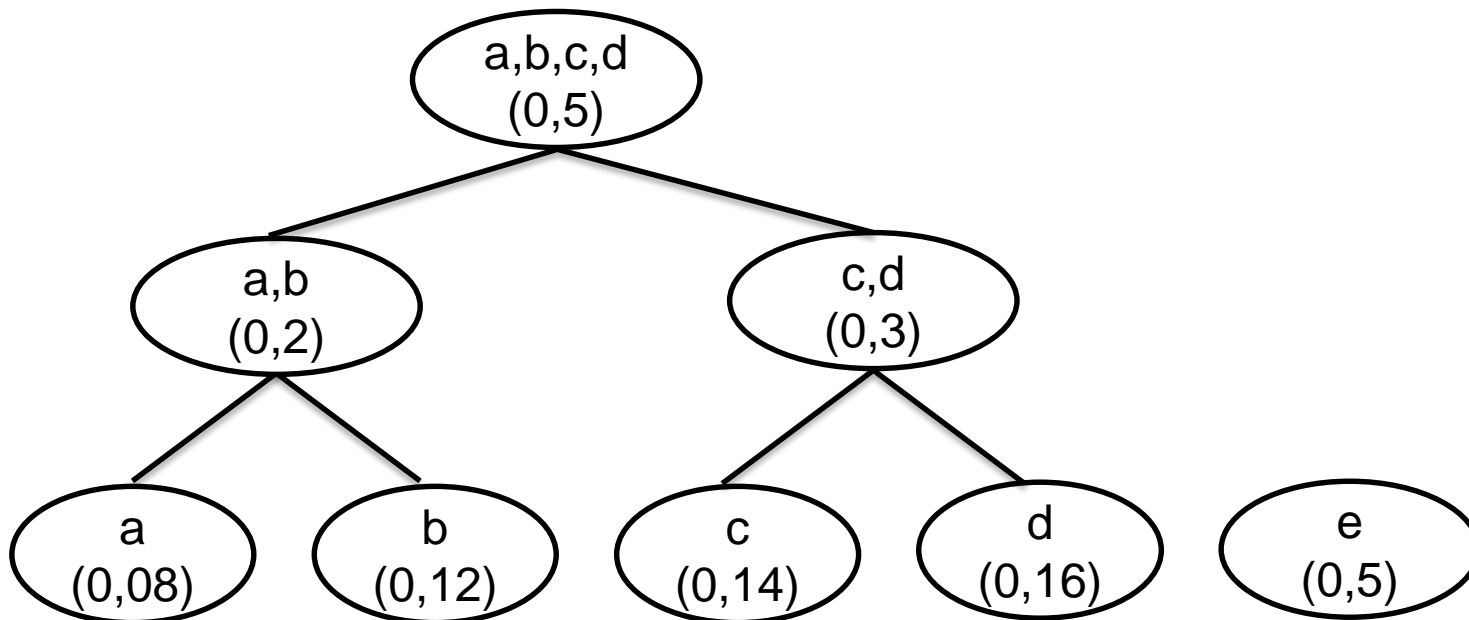
- Beispiel:



Huffman-Codierung

- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
- Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
- Links: 0 Rechts: 1

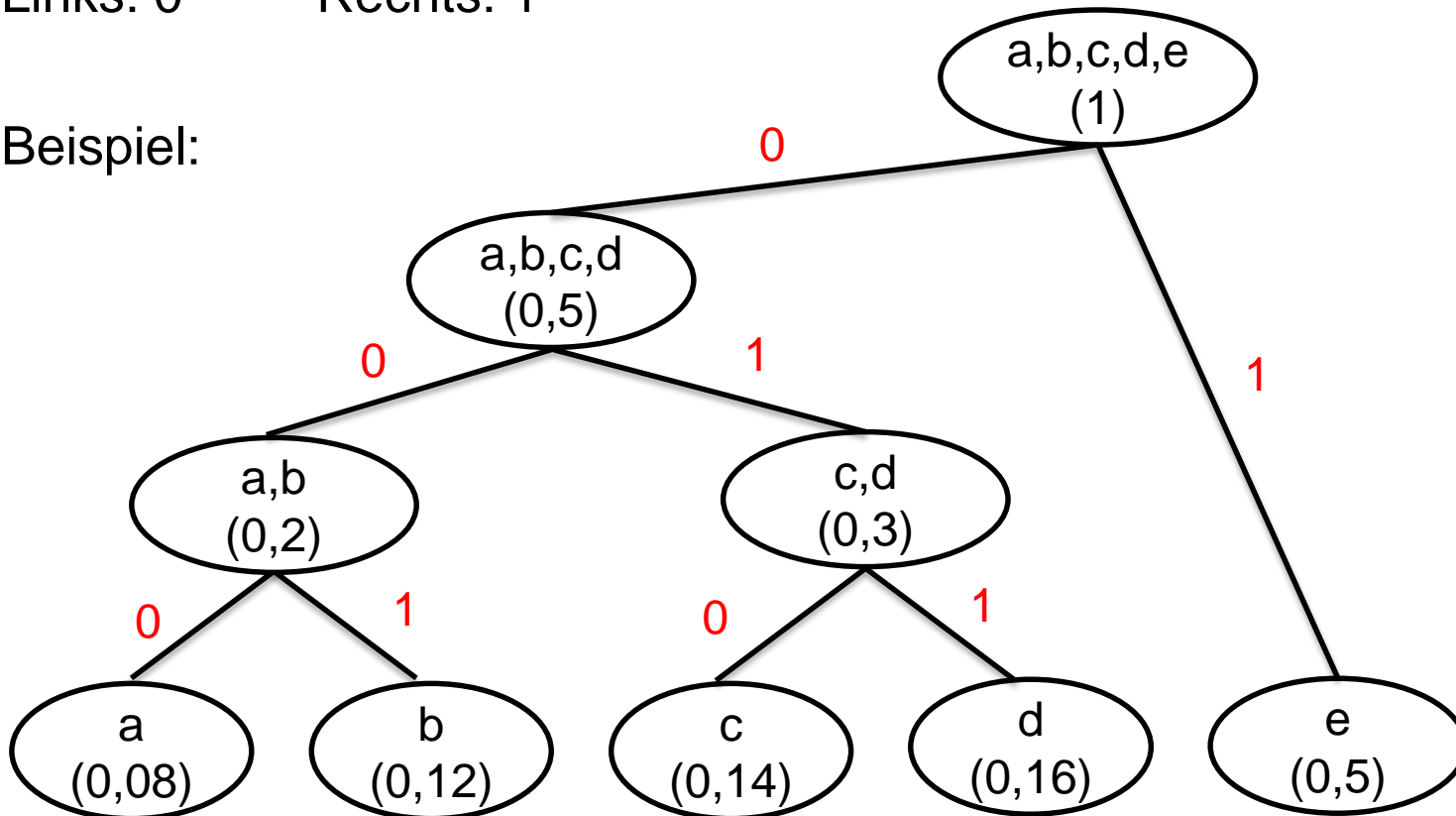
- Beispiel:



Huffman-Codierung

- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
- Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
- Links: 0 Rechts: 1

■ Beispiel:



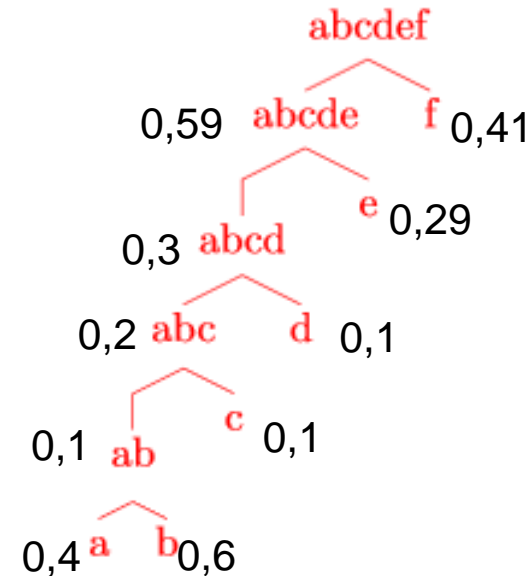
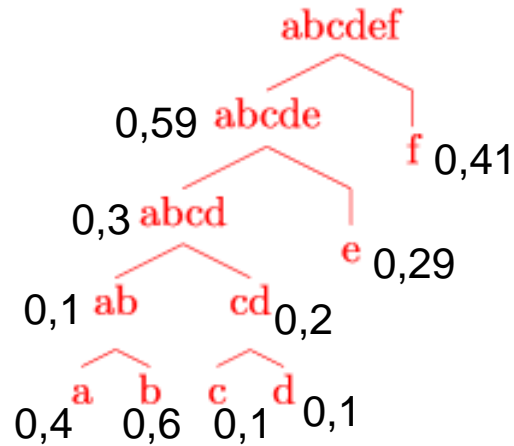
Bewertung des Codes

- Wie gut ist mein Code jetzt im Vergleich zu früher?
- Kann ich verschiedene optimale Codes vergleichen?

■ Mittlere Codewortlänge: $\bar{m} = \sum_{i=1}^n p(x_i) * m(x_i)$

■ Informationsgehalt H einer Quelle: $H = \sum_{i=1}^N p(i) * \log_2 \frac{1}{p(i)}$

Aufgabe 1.1



Aktivität	Wahrscheinlichkeit	Codierung
Liegen	41 %	f: 1/1
Stehen	10 %	d: 0011/001
Sitzen	29 %	e: 01/01
Gehen	10 %	c: 0010/0001
Laufen	6 %	b: 0001/00001
Radfahren	4 %	a: 0000/00000

Aufgabe 1.2

$$\bar{m} = 0,41 * 1 + 0,1 * 4 + 0,29 * 2 + 0,1 * 4 + 0,06 * 4 + 0,04 * 4 = \mathbf{2,19}$$

Aufgabe 1.3

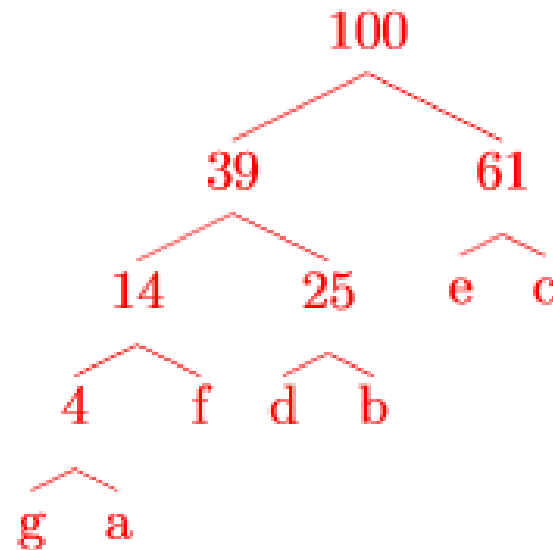
- Entropie der Quelle bzw. durchschnittlicher Informationsgehalt des Zeichenvorrats

$$H = \sum(p(x_i) * \lg\left(\frac{1}{p(x_i)}\right)) \approx \mathbf{2,139}$$

Aufgabe 1.4

$$\text{Kosten} = \frac{m * \overbrace{60 * 60 * 24}^{\text{Sek.} * \text{min.} * \text{h}}}{\underbrace{1024 * 8 * 10}_{\text{Kilo} * \text{byte} * 10}} * 0,1 \text{ €} = 0,23 \frac{\text{€}}{\text{Tag}}$$

Aufgabe 2.1



x_i	Beschreibung	$p(x_i)$	gefundener Code
a	Reaktor aus	3%	0001
b	zu niedrig	13%	011
c	optimal für Stufe 1	33%	11
d	Übergang	12%	010
e	optimal für Stufe 2	28%	10
f	zu hoch	10%	001
g	GEFAHR	1%	0000

Aufgabe 2.2

■ $\bar{m} = 0,04 * 4 + 0,35 * 3 + 0,61 * 2 = 2,43$

Aufgabe 2.3

- Entropie der Quelle bzw. durchschnittlicher Informationsgehalt des Zeichenvorrats

■ $H = \sum(p(x_i) * \lg\left(\frac{1}{p(x_i)}\right)) \approx 2,34$

Aufgabe 2.4

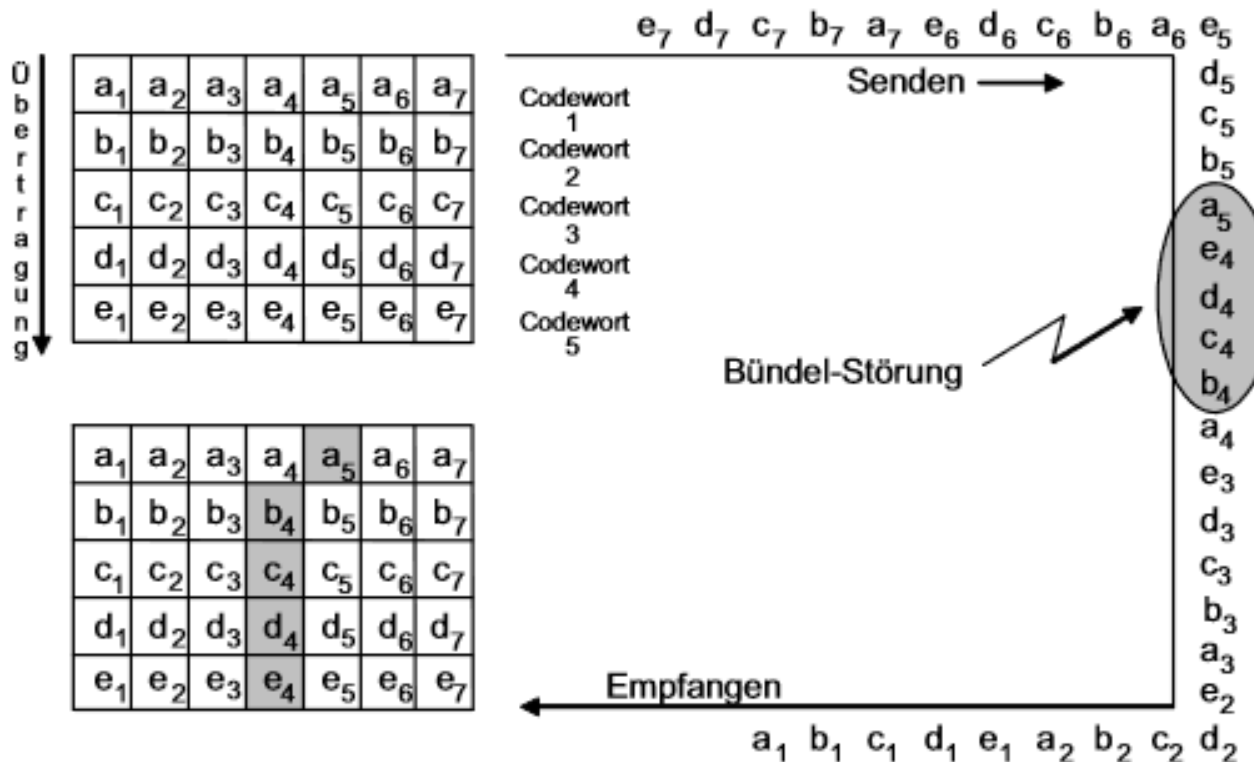
- Huffman hat geringste mittlere Codewortlänge
-> weniger Daten verschicken für gleiche Information

Aufgabe 2.5

- Huffman gleich gute oder bessere mittlere Codewortlänge als Shannon-Fano
- Shannon-Fano kann mehrere Lösungen haben
Huffman ist eindeutig

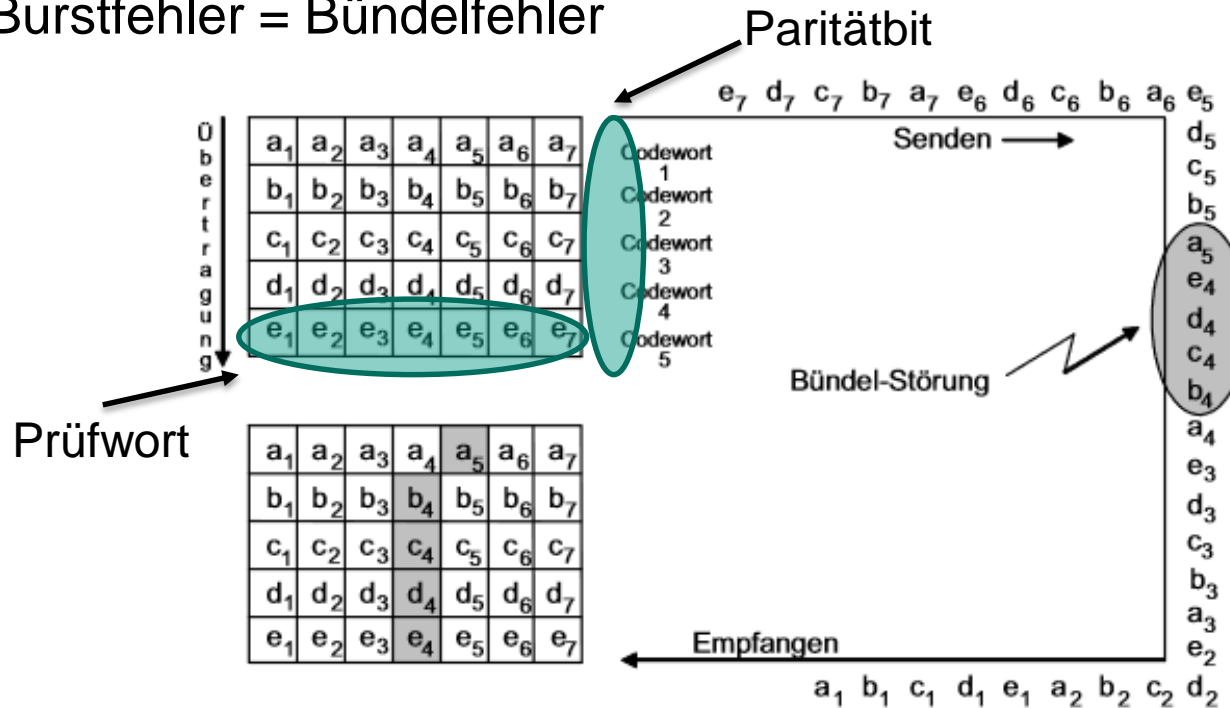
Blocksicherung und Scrambling

- Burstfehler = Bündelfehler



Blocksicherung und Scrambling

■ Burstfehler = Bündelfehler



■ $Overhead = \frac{\text{Bits für Sicherung}}{\text{Nutzdatenbits}}$

Aufgabe 3.1 & 3.2

- Burstfehler Länge 4 → 4 Zeilen
Blocksicherung → 1 Zeile als Prüfwort
→ 3 Zeilen je 8 Bits → 3 * 8bit = 24 Bit Nutzdaten

- Nutzdaten: 24 Bit
Prüfwort: 8 Bit (1 Paritätsbit pro Spalte)
Paritätsbit: 4 Bit (1 Paritätsbit pro Zeile)

$$Overhead = \frac{12 \text{ Bit}}{24 \text{ Bit}} = 0,5 \hat{=} 50\%$$

Aufgabe 3.3 & 3.4 & 3.5

- Länge Datenstrom: 63 Bit
Länge einer Zeile: 8 Bit Nutzdaten + 1 Bit Parität = 9 Bit
Anzahl der Zeilen: 63 Bit : 9 Bit = 7 Zeilen

- 7 Zeilen ergeben eine max. Länge eines Burstfehlers von 7

- Korrigiert: 01011110

max. Burstfehler (7 Zeilen)

1	1	0	1	0	1	0	1	1	1
2	1	0	0	0	1	1	1	0	0
3	0	1	0	0	1	1	1	0	1
4	1	0	1	1	0	1	0	1	1
5	1	1	1	0	0	1	1	0	1
6	0	1	0	0	1	0	0	0	0
7	0	1	1	0	0	0	0	0	0