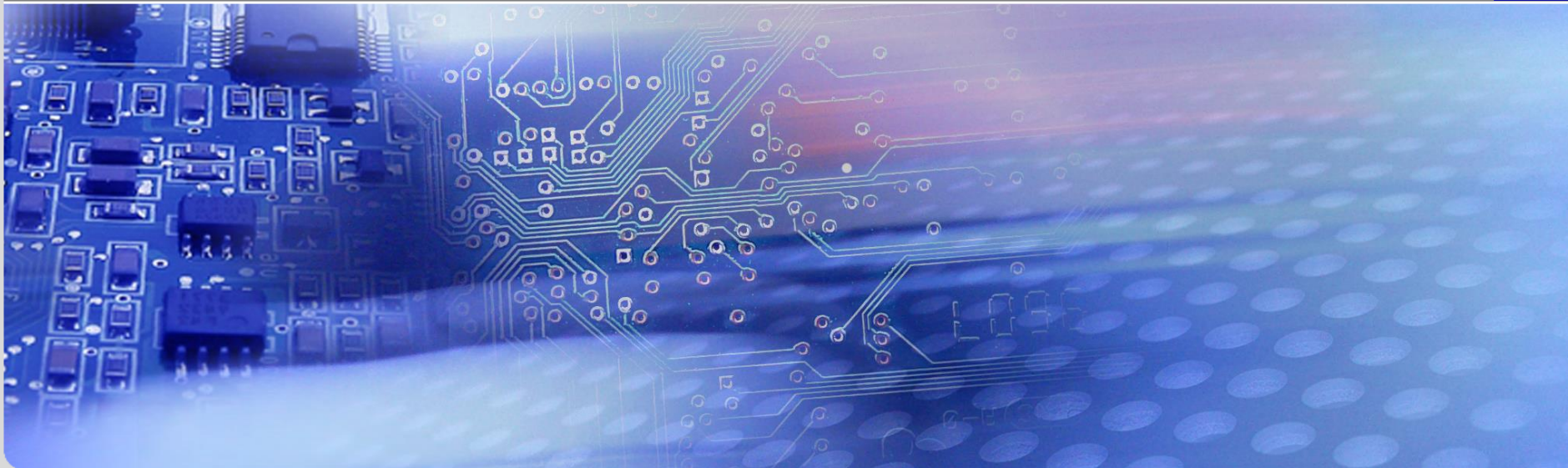


Digitaltechnik Tutorium 2

Institut für Technik der Informationsverarbeitung (ITIV)

itv



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

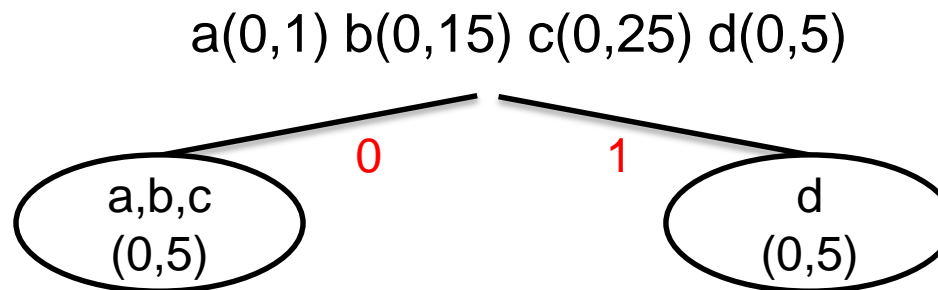
Shannon-Fano-Codierung

- 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(0,1) b(0,15) c(0,25) d(0,5)

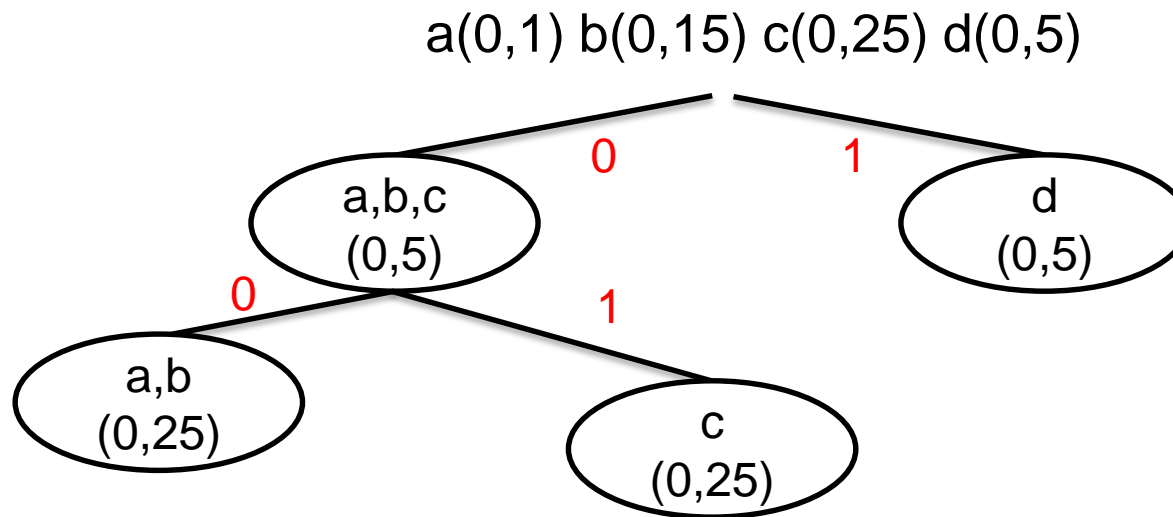
Shannon-Fano-Codierung

- 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:



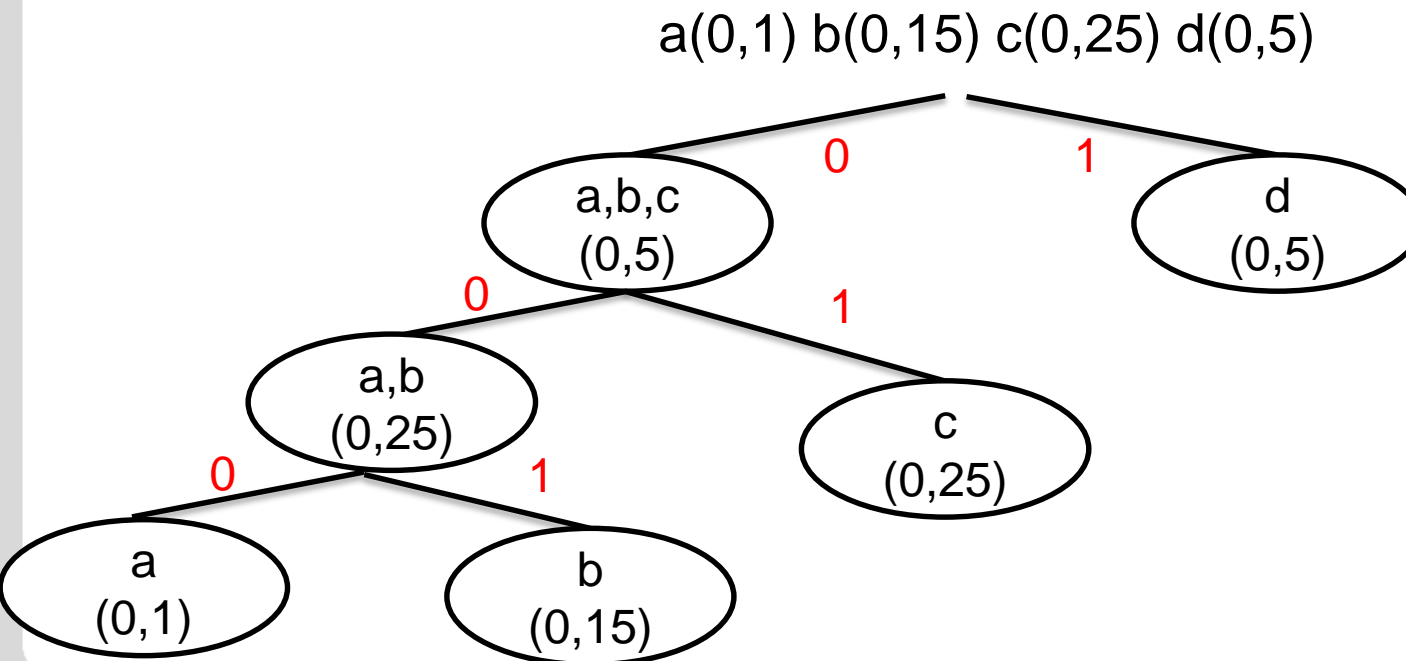
Shannon-Fano-Codierung

- 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:



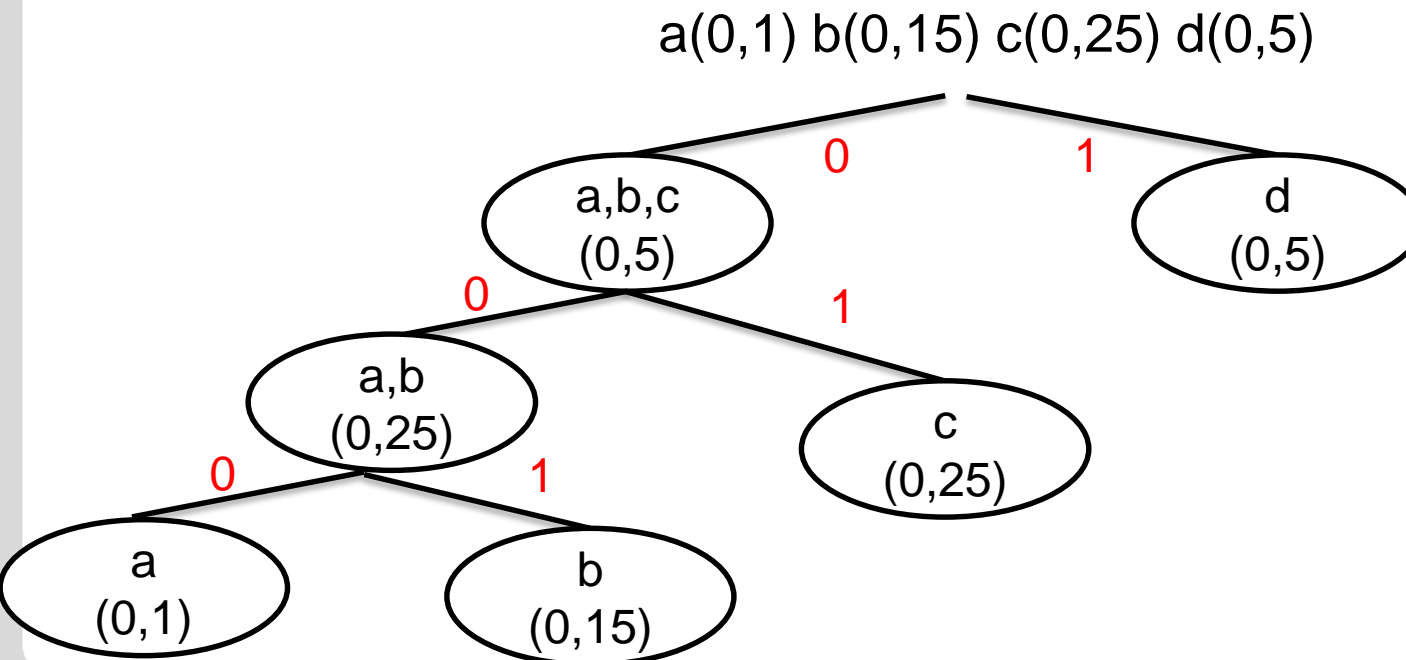
Shannon-Fano-Codierung

- 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:



Shannon-Fano-Codierung

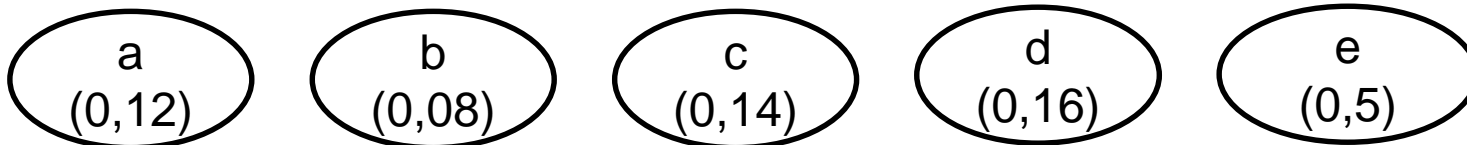
- Daten müssen gewichtet sein (Wahrscheinlichkeit)
- Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
Reihenfolge bleibt stets bestehen
- möglichst nach gleichen Wahrscheinlichkeiten aufteilen
- Links: 0 rechts: 1
- Beispiel:



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

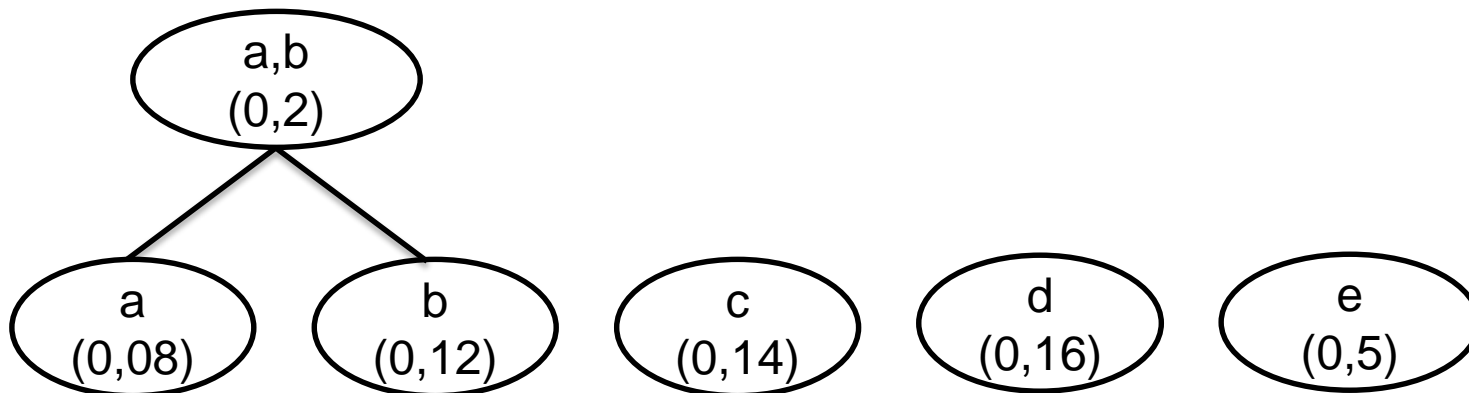
Huffman-Codierung

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



Huffman-Codierung

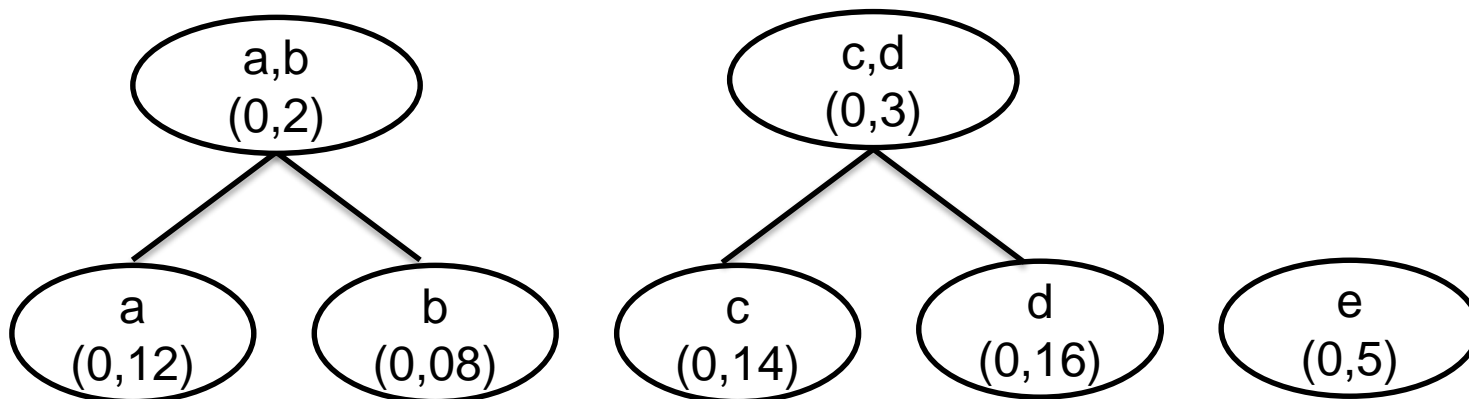
- Daten müssen gewichtet sein (Wahrscheinlichkeit)
 - Ereignisse nach Wahrscheinlichkeit sortieren (niedrig -> hoch)
 - Immer die 2 niedrigsten Wahrscheinlichkeiten zusammenfassen
 - Links: 0 Rechts: 1
- Beispiel:



Huffman-Codierung

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

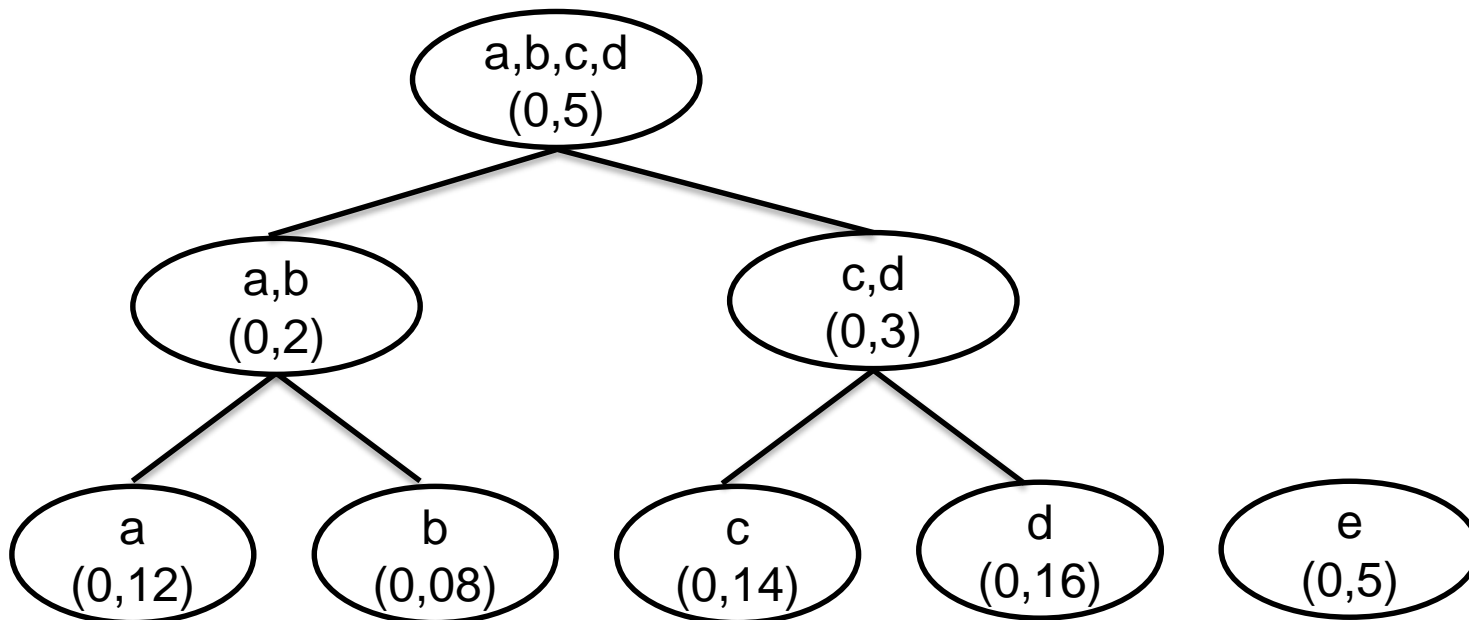
- Beispiel:



Huffman-Codierung

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

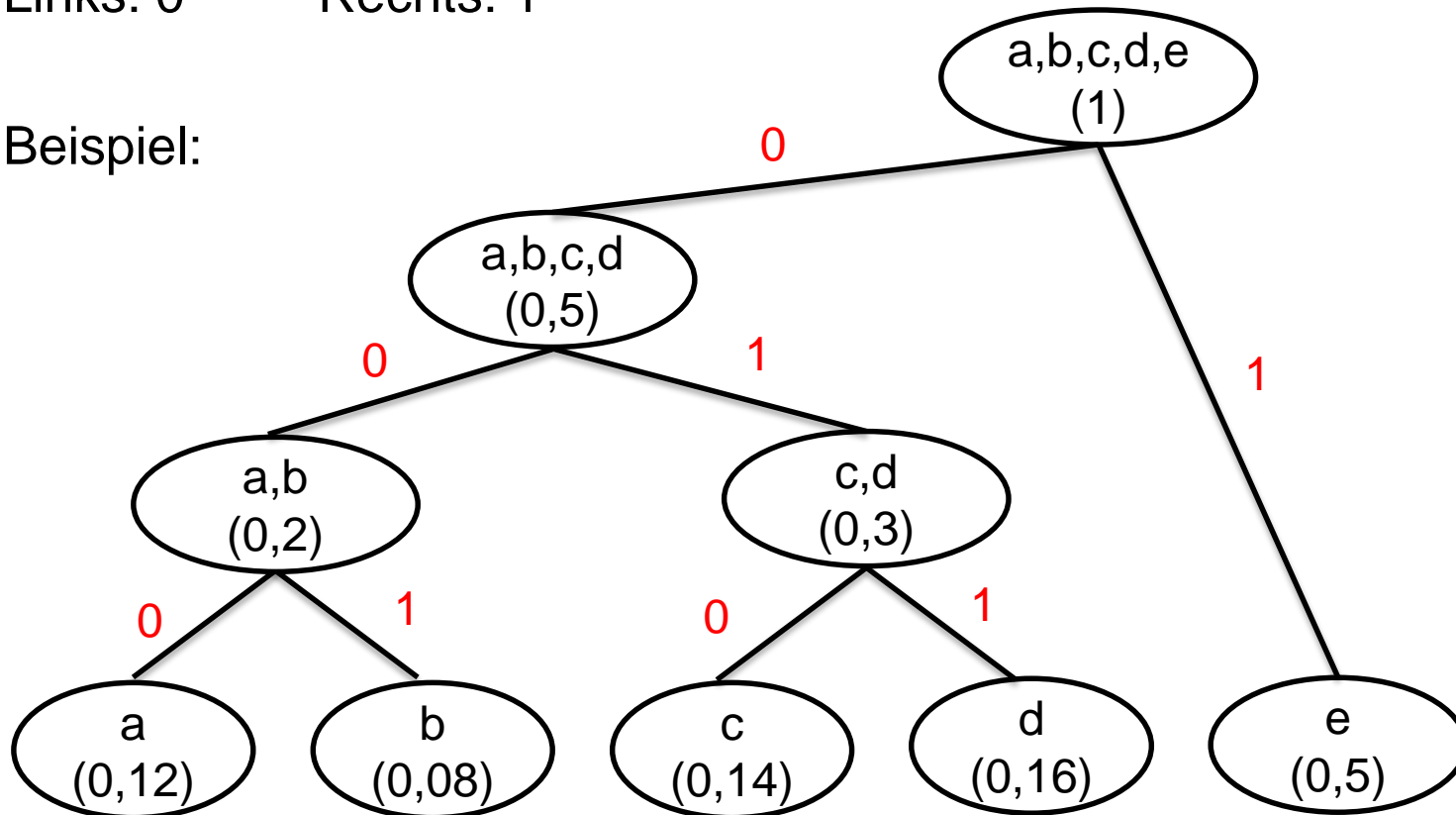
- Beispiel:



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

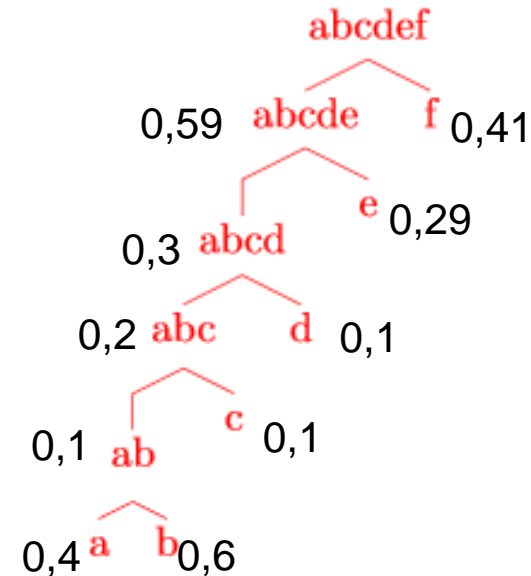
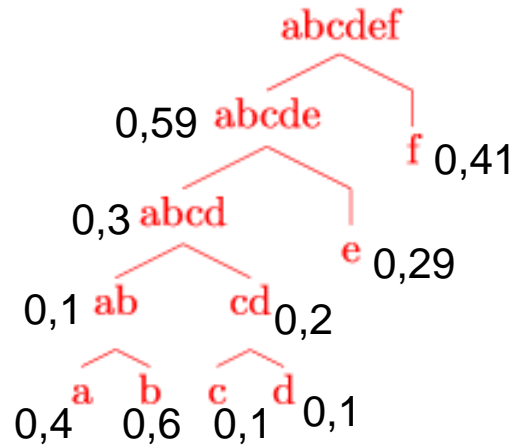
- Wie gut ist mein Code jetzt im Vergleich zu früher?
- Kann ich verschiedene optimale Codes vergleichen?

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

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

Der minimal erreichbare Idealwert ist dabei die Entropie der Sendequelle

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

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

Aufgabe 1.2

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

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 1.2

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

Aufgabe 1.3

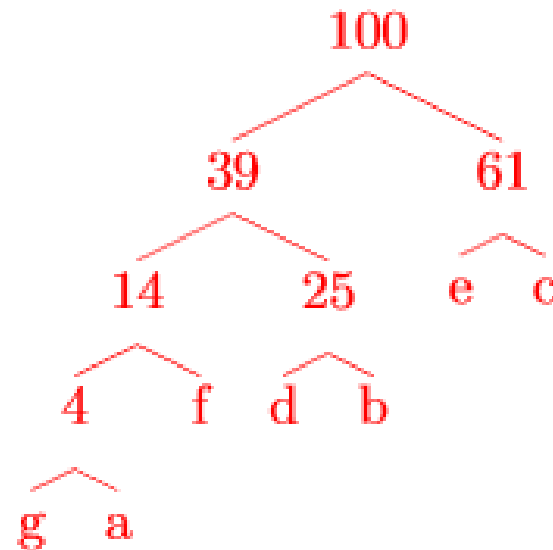
- Entropie der Quelle bzw. durchschnittlicher Informationsgehalt des Zeichenvorrats

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

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 2.2

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

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

- Codierung ist präfixfrei
 - kein Codewort, welches Präfix eines anderen Codewortes ist

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

- Blocksicherung: 2-dimensionales Hinzufügen eines Paritätsbits
- Burstfehler: Blockweise Störung des Signals => „Kippen“ mehrerer hintereinander gesendeter Bits
- Scrambling: Binärstellen eines Binärworts werden nicht nacheinander übertragen => länger anhaltender Störeinfluss (Bündelstörung)
bestimmter Länge verfälscht nur Binärstellen von unterschiedlichen Codewörtern

Blocksicherung und Scrambling

- **Aufgabe 3.1:** Berechnen Sie wie viele Nutzdaten in einem Block übertragen werden müssen, wenn durch das Scrambling Burstfehler der Länge 4 erkennbar sein soll.

Blocksicherung und Scrambling

- **Aufgabe 3.1:** Berechnen Sie wie viele Nutzdaten in einem Block übertragen werden müssen, wenn durch das Scrambling Burstfehler der Länge 4 erkennbar sein soll.

- **Lösung:**

Burstfehler der Länge 4 \Rightarrow 4 Zeilen durch Scrambling vorgegeben

Blocksicherung \Rightarrow letzte Zeile ist ein Prüf- und kein Datenwort

Blocksicherung und Scrambling

- **Aufgabe 3.1:** Berechnen Sie wie viele Nutzdaten in einem Block übertragen werden müssen, wenn durch das Scrambling Burstfehler der Länge 4 erkennbar sein soll.

- **Lösung:**

Burstfehler der Länge 4 \Rightarrow 4 Zeilen durch Scrambling vorgegeben

Blocksicherung \Rightarrow letzte Zeile ist ein Prüf- und kein Datenwort

\Rightarrow 3 Zeilen mit Nutzdaten zu je 8 Bit \Rightarrow 24 Bit Nutzdaten

Blocksicherung und Scrambling

- **Aufgabe 3.2:** Berechnen Sie den Overhead der durch die Blocksicherung entsteht. Verwenden Sie die in Aufgabe 3.1 berechnete Blockgröße

Blocksicherung und Scrambling

- **Aufgabe 3.2:** Berechnen Sie den Overhead der durch die Blocksicherung entsteht. Verwenden Sie die in Aufgabe 3.1 berechnete Blockgröße
- **Overhead** = Bits für Sicherung / Nutzdatenbits

Blocksicherung und Scrambling

- **Aufgabe 3.2:** Berechnen Sie den Overhead der durch die Blocksicherung entsteht. Verwenden Sie die in Aufgabe 3.1 berechnete Blockgröße.

- **Overhead** = Bits für Sicherung / Nutzdatenbits

- **Lösung:**

3 Zeilen Nutzdaten = 24 Bit Nutzdaten

Blocksicherung

=> ein Prüfwort mit 8 Bit (gesamte 4. Zeile)

=> ein Paritätsbit pro Zeile => Prüfspalte: 4 Bit

=> 24 Bit Nutzdaten und 12 Paritätsbits

=> Overhead = $12/24 = 50\%$

Blocksicherung und Scrambling

- **Aufgabe 3.3:** Rekonstruieren Sie die gesendeten Datenwörter unter der Annahme, dass 1 Datenwort 8 Bit lang ist.

- **Ansatz:**

Mit Blocksicherung und Scrambling gegen Fehler gesichert

⇒ Paritätsbits beachten

⇒ Wie lang ist eine Zeile?

Blocksicherung und Scrambling

- **Aufgabe 3.3:** Rekonstruieren Sie die gesendeten Datenwörter unter der Annahme, dass 1 Datenwort 8 Bit lang ist.

- **Lösung:**

Länge des Datenstroms: 63 Bit

Länge einer Zeile: 8 Bit Daten + 1 Paritätsbit = 9 Bit

Anzahl Zeilen: $63 / 9 = 7 \Rightarrow 6 \text{ Bit Daten} + 1 \text{ Paritätsbit pro Spalte}$

Blocksicherung und Scrambling

- **Aufgabe 3.3:** Rekonstruieren Sie die gesendeten Datenwörter unter der Annahme, dass 1 Datenwort 8 Bit lang ist.

- **Lösung:**

Zeile									
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

Paritätsbits sind grau hinterlegt

Blocksicherung und Scrambling

- **Aufgabe 3.4:** Welche Länge an Burstfehler lassen sich mit der verwendeten Blockgröße in Verbindung mit dem Scrambling maximal erkennen?

Blocksicherung und Scrambling

- **Aufgabe 3.4:** Welche Länge an Burstfehler lassen sich mit der verwendeten Blockgröße in Verbindung mit dem Scrambling maximal erkennen?

- **Lösung:**

7 Zeilen => maximale Länge eines Burstfehlers = 7 Bit

Blocksicherung und Scrambling

- **Aufgabe 3.5:** In dem übertragenen Datenstrom befindet sich ein Fehler. Beheben Sie diesen indem Sie die fehlerhafte Zeile und die fehlerhafte Spalte angeben. Markieren Sie das fehlerhafte Datenwort und geben Sie das korrigierte Datenwort an. Für die Blocksicherung wurde eine gerade Parität verwendet.

Zeile									
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

Blocksicherung und Scrambling

- **Aufgabe 3.5:** In dem übertragenen Datenstrom befindet sich ein Fehler. Beheben Sie diesen indem Sie die fehlerhafte Zeile und die fehlerhafte Spalte angeben. Markieren Sie das fehlerhafte Datenwort und geben Sie das korrigierte Datenwort an. Für die Blocksicherung wurde eine gerade Parität verwendet.

Zeile									
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

Blocksicherung und Scrambling

- **Aufgabe 3.5:** In dem übertragenen Datenstrom befindet sich ein Fehler. Beheben Sie diesen indem Sie die fehlerhafte Zeile und die fehlerhafte Spalte angeben. Markieren Sie das fehlerhafte Datenwort und geben Sie das korrigierte Datenwort an. Für die Blocksicherung wurde eine gerade Parität verwendet.

Zeile									
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

Blocksicherung und Scrambling

- **Aufgabe 3.5:** In dem übertragenen Datenstrom befindet sich ein Fehler. Beheben Sie diesen indem Sie die fehlerhafte Zeile und die fehlerhafte Spalte angeben. Markieren Sie das fehlerhafte Datenwort und geben Sie das korrigierte Datenwort an. Für die Blocksicherung wurde eine gerade Parität verwendet.

Zeile									
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

Korrigiertes Datenwort:
01011110