

Codierung u. Kompression

Codierung: bestimmtes **Format / Protokoll**, was zur **Übertragung und Speicherung von Daten** verwendet wird

Kompression: **Verringerung** der Dateigröße

verlustfrei

alle Daten können wiederhergestellt werden

verlustbehaftet

manche Daten gehen verloren

⊕ kein Datenverlust

⊕ höhere Kompressionsrate

z.B. **Huffman**

z.B. **JPEG**

Kompressionsrate:

$$\frac{\text{unkomprimiert}}{\text{komprimiert}} \%$$

Kompressionsfaktor:

$$\frac{\text{komprimiert}}{\text{unkomprimiert}}$$

hat nichts damit zu tun

kein Codewort ist Anfang eines anderen Codeworts?

Dann ist es ein **Präfixcode**

immer **eindeutig decodierbar**

codierte Daten bestehen aus **Codewörtern**

haben gleiche Länge?

Dann ist es ein **Blockcode**

Block aus verfügbaren Zeichen

↓
Wortlänge **Zeichenformat**

Alle Codewörter können im

als **Code** { **Codebaum** oder **Codetabelle**

dargestellt werden

Laufängencodierung (Run-length-Encoding, RLE)

- Bei mehreren gleichen hintereinander vorkommenden Zeichen / Zahlen / Symbolen
- Übertragung des Zeichens einmal + dessen Anzahl
- z.B. "AAAA" → "A4"

+
• einfach
• schnell
• verlustfrei

-
• vergleichsweise ineffizient
• nur einsetzbar, wenn gleiche Zeichen oft hintereinander vorkommen

Huffman-Codierung

- 💡 kürzerer Code für häufig vorkommende Zeichen
- ! Vorkommenswahrs. müssen bekannt sein, Optimierung auf einen bestimmten Fall
- + verlustfrei, kürzester präfixfreier Code

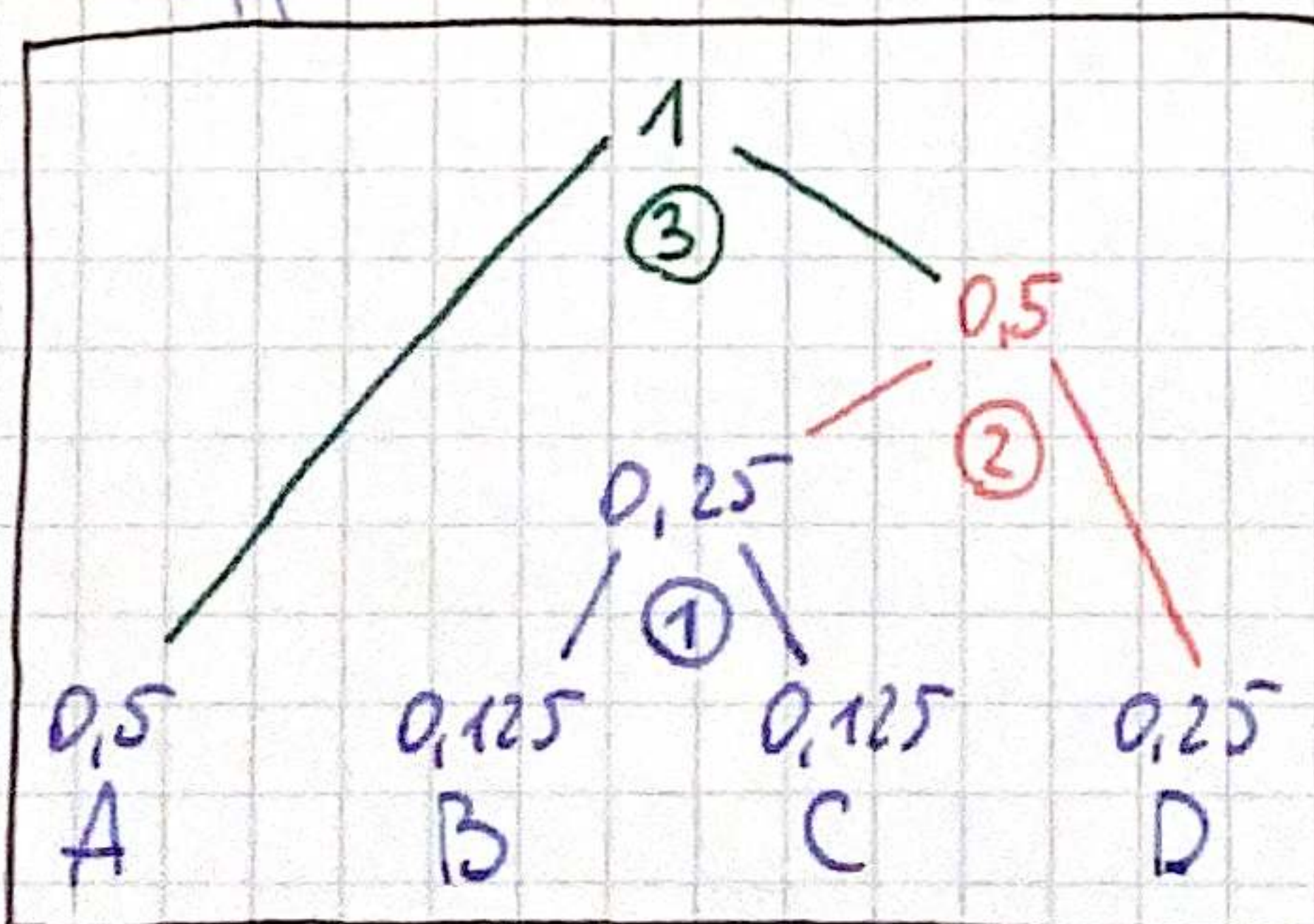
Vorgehen:

① "AABBCADAD"

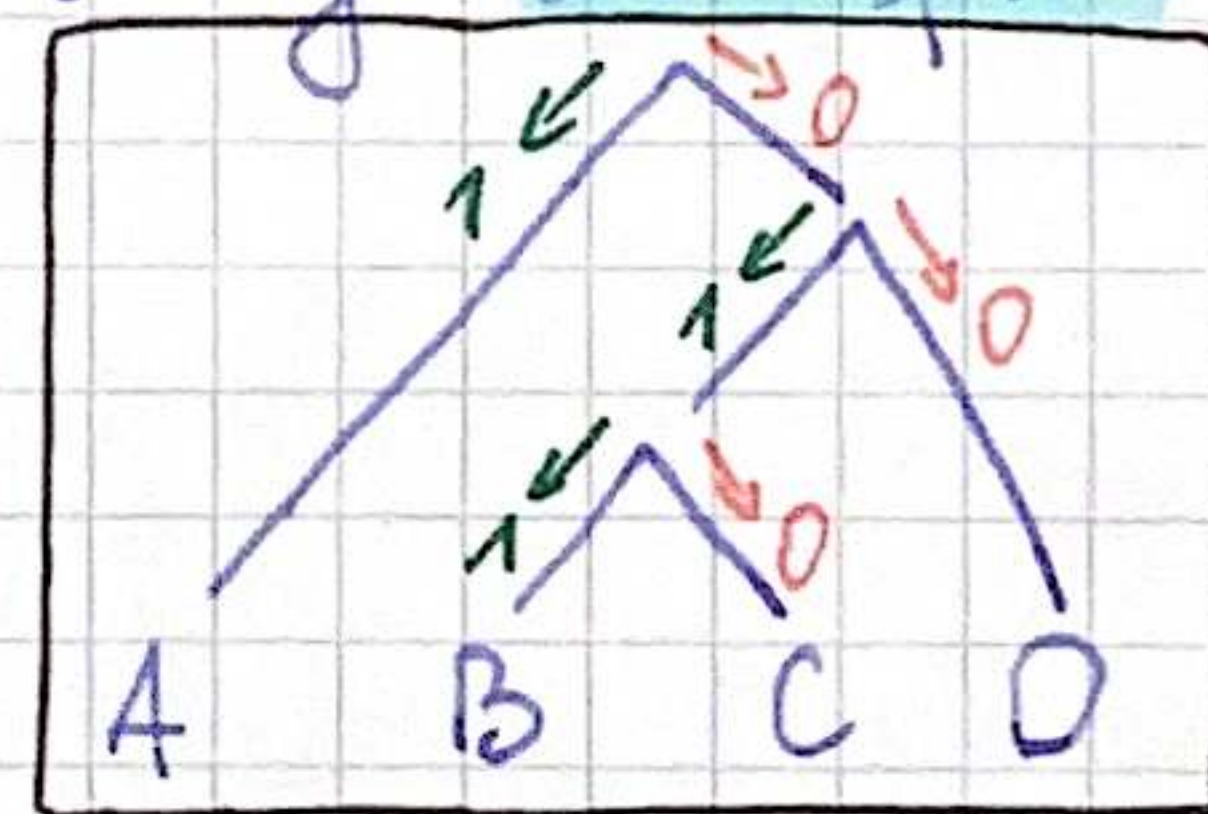
↓
② Wahrs. ermitteln

A: 0,5
B: 0,125
C: 0,125
D: 0,25

↓
③ Huffman-Baum erstellen



④ Zweige beschriften



1 links
0 rechts } konsistent!

⑤ Codes ablesen und in Tabelle eintragen

A: 1
B: 011
C: 010
D: 00

(Pfade ablaufen)

die beiden kleinsten Wahrs. jeweils zusammenführen, bis der Baum vollständig ist

Fehlererkennung: Repetitions codes

- bei der Übertragung können Fehler passieren
- Idee: wir übertragen jedes Bit zwei- oder dreimal
- Problem: ineffizient

Fehlererkennung: Paritätsbit

- Parität: # der Einsen-Bits
- Idee: an jedes Codewort/Bloch einen Paritätsbit anhängen

Codewort:	Parität:	Paritätsbit:
1000100	gerade	0
0110111	ungerade	1

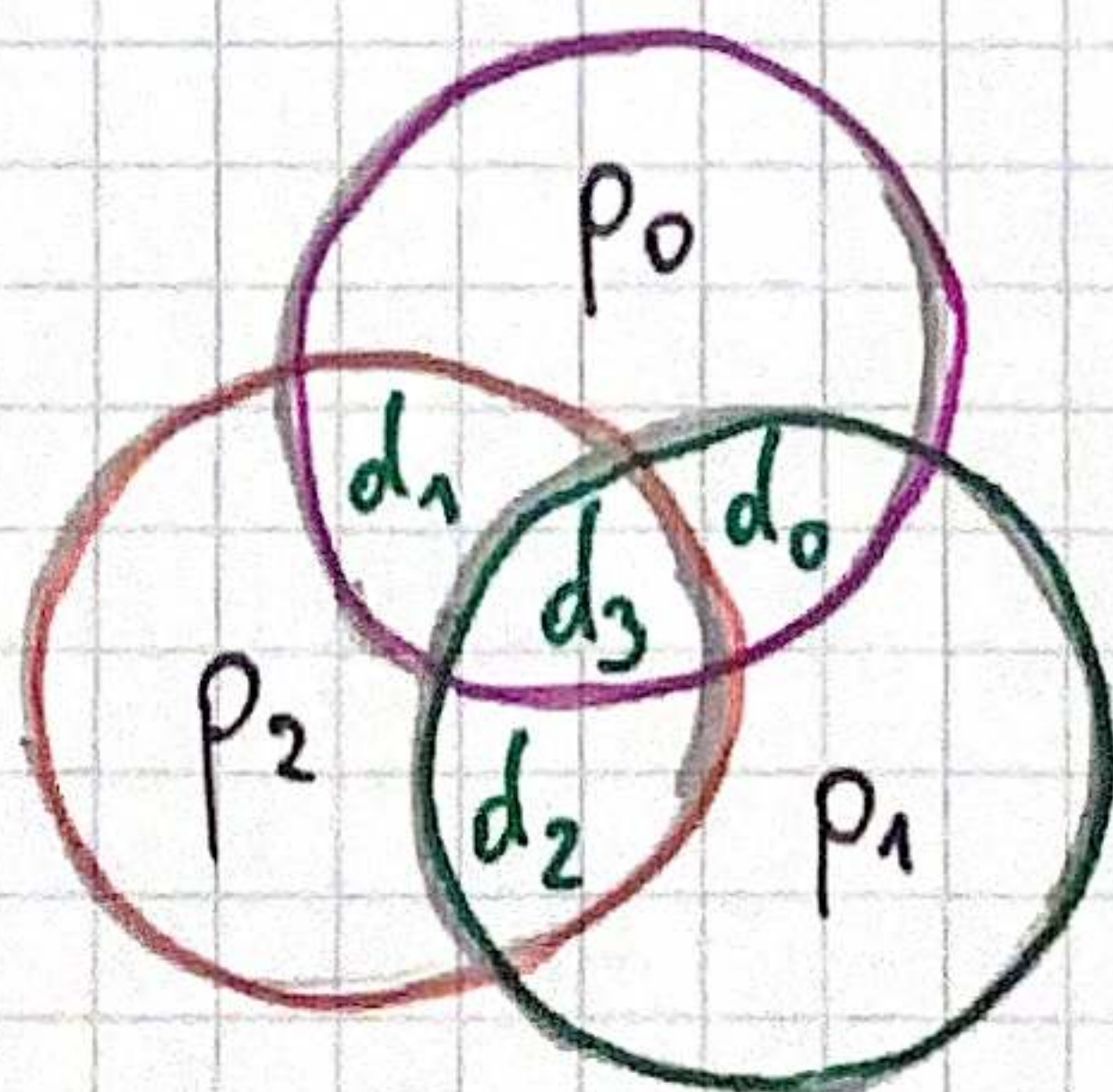
Fehlerkorrektur: (7,4)-Hamming-Code

- 4 Datenbits + 3 fehlerkorrigierende Bits = 1 Fehler kann korrigiert werden

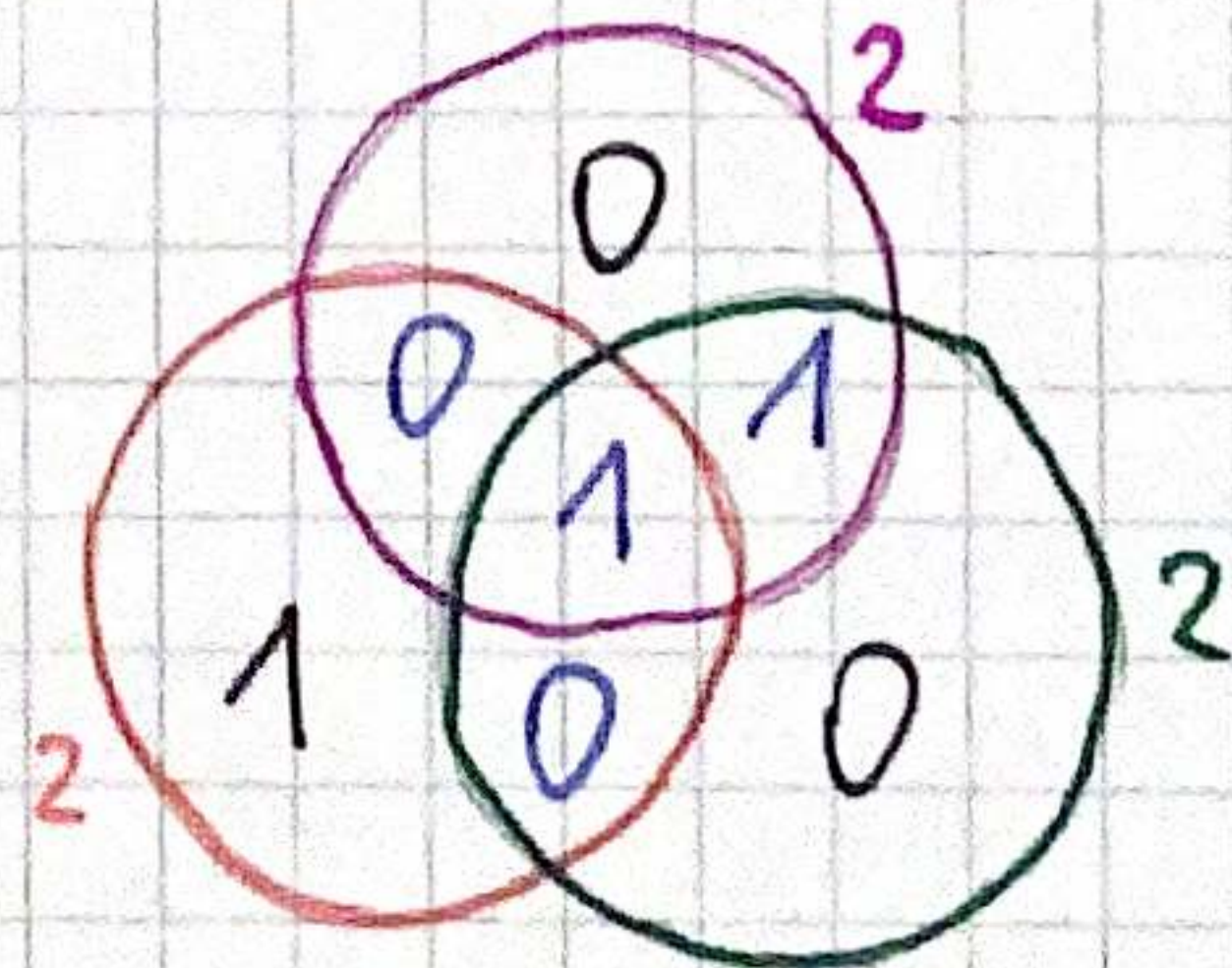
① 1001

② Reihenfolge: $p_0 p_1 d_0 p_2 d_1 d_2 d_3 \rightarrow p_0 p_1 1 p_2 001$

③ p_0, p_1, p_2 so bestimmen, dass



→ Summe der Bits in jedem Kreis soll gerade sein



④ Ergebnis: 0011001