Musterlösung zur Aufgabe A2.1

a) Richtig sind die Lösungsvorschläge 1 und 2. Bei der Leitungscodierung fügt man Redundanz hinzu, um das Sendesignal an die Spektraleigenschaften des Kanals anzupassen. Auch bei der Kanalcodierung fügt man gezielt Redundanz hinzu, in diesem Fall, um diese beim Empfänger zur Fehlererkennung und/oder Fehlerkorrektur nutzen zu können. Ziel von Quellencodierung ist dagegen eine größtmögliche Redundanzverminderung, um die Information der Nachrichtenquelle möglichst effizient speichern oder übertragen zu können.

b) Richtig ist hier die Antwort 3. Bei Leitungs– und Kanalcodierung wären verlustbehaftete Verfahren kontraproduktiv. Dagegen ist die Quellencodierung bei analogem Eingangssignal (Audio, Video, usw.) per se verlustbehaftet.

c) Zu den Leitungscodierverfahren zählt man

- die **4B3T–Codes** (es gibt mehrere Varianten, die alle blockweise arbeiten),
- den **AMI–Code** (symbolweise: Bei jedem Codierschritt wird ein Binärzeichen eingelesen und ein Ternärzeichen ausgegeben).

Demzufolge gilt $N_{\rm LC} = 2$.

d) Im Buch „Einführung in die Kanalcodierung” werden behandelt:

- die **Hamming–Codes,**
- die **Reed–Solomon–Codes,**
- die **Faltungscodes,**
- die **Turbocodes.**

Das richtige Ergebnis lautet dementsprechend $N_{\rm KC} = 4$.

e) Bei einem verlustlosen Quellencodierverfahren ist es dem Empfänger möglich, die Nachricht der Quelle vollständig zu rekonstruieren, wenn kein Übertragungsfehler aufgetreten ist.

Zu den verlustlosen Quellencodierverfahren gehören

- der **Huffman–Code,**
- die verschiedenen Varianten des **Lempel–Ziv–Algorithmus,**
- die so genannten **Run–Length–Codes,**
- das bekannte Komprimierungsprogramm **Winzip.**

$\Rightarrow N_{\rm QC\,(verlustlos)} = 4$. Alle diese Verfahren kann man nur bei digitalem Eingang anwenden.

f) Richtig sind die Aussagen 1 und 3. Nur **GIF und JPG** wendet man auf Bilder an, während **MP3** seit Jahren das am weitesten verbreitete Audio–Komprimierungsprogramm darstellt. Der **AMR–Codec** und der **EFR–Codec** finden Anwendung bei **GSM** und **UMTS.**

Musterlösung zur Aufgabe A2.2

a) Für die angegebenen Binärcodes gilt:

B1: $8 \cdot 2^{-3} = 1$ ⇒ Bedingung erfüllt,

B2: $1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 2 \cdot 2^{-4} = 1$ ⇒ Bedingung erfüllt,

B3: $1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 2 \cdot 2^{-4} = 1$ ⇒ Bedingung erfüllt,

B4: $1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 2 \cdot 2^{-3} + 1 \cdot 2^{-4} = 17/16$ ⇒ Bedingung nicht erfüllt.

b) Der Code B4, der die Kraftsche Ungleichung nicht erfüllt, ist mit Sicherheit auch nicht präfixfrei. Aber bei Erfüllung der Kraftschen Ungleichung ist noch nicht sicher, dass dieser Code auch präfixfrei ist. Beim Code B3 ist „10" der Beginn des Codewortes „1011". Dagegen sind die Codes B1 und B2 präfixfrei.

c) Richtig sind die Antworten 1 und 3. Der Code T2 ist dagegen nicht präfixfrei, da „1" der Beginn des Codewortes „10" ist. Die Kraftsche Ungleichung wird von allen drei Codes erfüllt.

d) N_i gibt an, wieviele Codeworte mit i Symbolen es im Code gibt. Für den Code T1 gilt:

$$N_1 = 1, \quad N_2 = 2, \quad N_3 = 6.$$

e) Nach der Kraftschen Ungleichung muss gelten

$$N_1 \cdot 3^{-1} + N_2 \cdot 3^{-2} + N_3 \cdot 3^{-3} \le 1 .$$

Bei gegebenem $N_1 = 1$ und $N_2 = 2$ wird dies erfüllt, solange gilt:

$$N_3 \cdot 3^{-3} \le 1 - 1/3 - 2/9 = 4/9 \quad \Rightarrow \quad N_3 \le 12 \quad \Rightarrow \quad \Delta N_3 = 6.$$

Die zusätzlichen Codeworte sind **210**, **211**, **212**, **220**, **221** und **222**.

f) Für den Code T3 gilt:

$$S(\text{T3}) = 2 \cdot 3^{-1} + 2 \cdot 3^{-2} + 1 \cdot 3^{-3} = 25/27 .$$

Wegen $S(\text{T3}) \le 1$ erfüllt der Ternärcode T3 die Kraftsche Ungleichung und er ist zudem auch präfixfrei. Betrachten wir nun die vorgeschlagenen neuen Codes.

- Code T4 ($N_1 = 2$, $N_2 = 2$, $N_3 = 5$):

$$S(\text{T4}) = S(\text{T3}) + 4 \cdot 3^{-3} = 29/27 > 1 \quad \Rightarrow \quad \text{T4 ist ungeeignet},$$

- Code T5 ($N_1 = 2$, $N_2 = 2$, $N_3 = 1$, $N_4 = 4$):

$$S(\text{T5}) = S(\text{T3}) + 4 \cdot 3^{-4} = 79/81 < 1 \quad \Rightarrow \quad \text{T5 ist geeignet},$$

- Code T6 ($N_1 = 2$, $N_2 = 2$, $N_3 = 2$, $N_4 = 3$):

$$S(\text{T6}) = S(\text{T3}) + 1 \cdot 3^{-3} + 3 \cdot 3^{-4} = \frac{75 + 3 + 3}{81} = 1 \quad \Rightarrow \quad \text{T6 ist geeignet}.$$

Richtig sind also die zwei letzten Lösungsvorschläge. Beispielsweise lauten die insgesamt 9 Codeworte des präfixfreien Codes T6: **0**, **1**, **20**, **21**, **220**, **221**, **2220**, **2221** und **2222**.

Musterlösung zur Zusatzaufgabe Z2.2

a) Die mittlere Codewortlänge ergibt sich allgemein zu

$$L_{\rm M} = p_{\rm A} \cdot L_{\rm A} + p_{\rm B} \cdot L_{\rm B} + p_{\rm C} \cdot L_{\rm C} + p_{\rm D} \cdot L_{\rm D}\,.$$

Sind die vier Quellensymbole gleichwahrscheinlich (alle Wahrscheinlichkeiten genau 1/4), so kann dafür auch geschrieben werden:

$$L_{\rm M} = 1/4 \cdot \left[L_{\rm A} + L_{\rm B} + L_{\rm C} + L_{\rm D}\right].$$

- Code C1: $L_{\rm M}$ = 2.00 bit/Quellensymbol,
- Code C2: $L_{\rm M}$ = 2.25 bit/Quellensymbol,
- Code C3: $L_{\rm M}$ = 2.25 bit/Quellensymbol.

b) Mit der Codetabelle $C1$ ergibt sich unabhängig von den Symbolwahrscheinlichkeiten stets die mittlere Codewortlänge $L_{\rm M}$ = 2 bit/Quellensymbol. Für die beiden anderen Codes erhält man

- Code C2: $L_{\rm M} = 1/2 \cdot 1 + 1/4 \cdot 2 + 1/8 \cdot 3 + 1/8 \cdot 3$ = 1.75 bit/Quellensymbol,
- Code C3: $L_{\rm M} = 1/2 \cdot 3 + 1/4 \cdot 2 + 1/8 \cdot 1 + 1/8 \cdot 3$ = 2.50 bit/Quellensymbol.

Man erkennt aus dem Beispiel das Prinzip: Wahrscheinliche Symbole werden durch wenige Binärsymbole dargestellt und unwahrscheinliche durch mehr. Bei gleichwahrscheinlichen Symbolen wählt man am besten auch die Codewortlängen gleich.

c) Richtig ist Lösungsvorschlag 1. Ein Code mit einheitlicher Länge aller Codeworte ist zwar präfixfrei, aber auch andere Codes können präfixfrei sein, zum Beispiel die Codes C2 und C3.

d) Bereits aus „00” am Anfang erkennt man, dass der Code C2 hier nicht in Frage kommt, da sonst die Quellensymbolfolge mit „AA” beginnen müsste. Tatsächlich wurde der Code C1 verwendet.

e) Richtig ist der Lösungsvorschlag 2. Der erste Lösungsvorschlag gibt die Quellensymbolfolge für den Code C2 an, wenn die Codesymbolfolge **001101111001100100111000** lauten würde.

Musterlösung zur Aufgabe A2.3

a) Zutreffend für den LZ78–Algorithmus ist Aussage 1. Die Vorbelegung gemäß Aussage 2 gilt dagegen für den **LZW–Algorithmus,** der 1983 gemeinsam mit Terry Welch veröffentlicht wurde.

b) **ε** bezeichnet das leere Zeichen. Da **ε** + **B** = **B** ergibt, sind die Aussagen 1 und 2 richtig. Dagegen würde die Aussage 3 wieder für die LZW–Komprimierung zutreffen.

c) Beide Aussagen treffen zu.

d) Im Wörterbuch wird unter dem Index $I = 1$ das Zeichen **B** gefunden und das nächste Zeichen **A** der Eingangsfolge wird angehängt: (1, **A**). Richtig sind somit die Aussagen 2 und 4. Die Aussage 3 kann schon deshalb nicht stimmen, da Z nur ein Zeichen sein kann und keine Zeichenfolge.

e) Der gesamte Codiervorgang ist in einer Tabelle zusammengefasst:

Index I		Inhalt	Eintrag	LZ78–Coderausgabe	
dezimal	binär		im Schritt	formal	binär
0	000	ε (leer)	$i = 0$	—	—
1	001	B	$i = 1$	(0, B)	00001
2	010	A	$i = 2$	(0, A)	00000
3	011	R	$i = 3$	(0, R)	00010
4	100	BA	$i = 4$	(1, A)	00100
5	101	RA	$i = 5$	(3, A)	01100
6	110	–	$i = 6$	(0, –)	00011
7	111	BAR	$i = 7$	(4, R)	10010

Wörterbuch

Man erkennt:

- Zu jedem Zeitpunkt i wird die bearbeitete Zeichenfolge in das Wörterbuch eingetragen.
- Zum Zeitpunkt $i = 7$ ist der Codiervorgang abgeschlossen.

f) Stellt man alle Indizes mit 3 Bit dar und die vier Zeichen (Character) mit je 2 Bit, so kommt man zu folgenden Ergebnissen:

- ohne Codierung: $N = 11 \cdot 2 =$ 22 Bit,
- mit LZ78–Codierung: $N = 7 \cdot (3 + 2) =$ 35 Bit.

Daran erkennt man:

- Eine jede LZ–Komprimierung macht erst bei einer größeren Datei Sinn, auch dann, wenn man glaubt, dass ein Text wie **BARBARA–BAR** dem LZ78–Algorithmus entgegenkommt.
- Mit variabler Bitanzahl für den Index entsprechend der **Theorieseite 5** und **Aufgabe A2.4** ergibt sich für dieses LZ78–Beispiel:

$$N = 1 \cdot 3 + 2 \cdot 4 + 4 \cdot 5 = 31 \text{ Bit}.$$

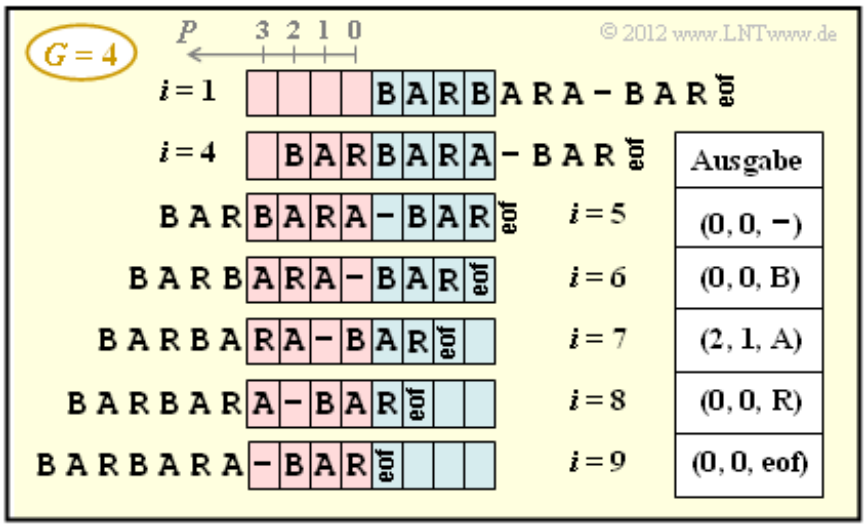
Musterlösung zur Zusatzaufgabe Z2.3

a) Richtig ist der Lösungsvorschlag 3. Im Vorschaupuffer steht zum betrachteten Zeitpunkt $i = 4$ die Zeichenfolge **BARA**, und im Suchpuffer in den letzten drei Stellen **BAR**:

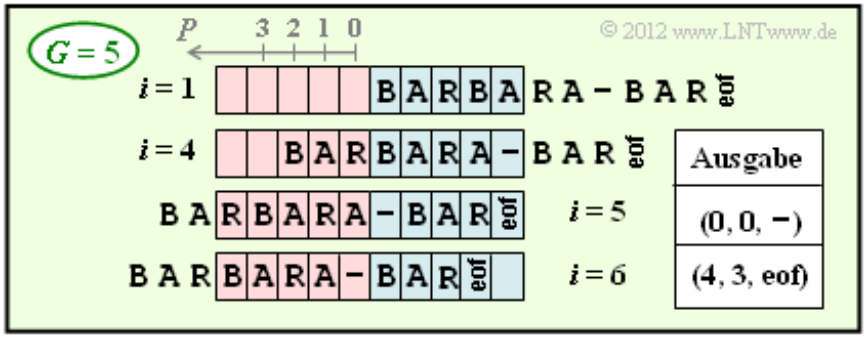
$$P = 2, \quad L = 3, \quad Z = \mathbf{A}.$$

b) Richtig sind die beiden ersten Lösungsvorschläge. Der Bindestrich findet sich zum Zeitpunkt $i = 5$ nicht im Suchpuffer, so dass (0, 0, –) ausgegeben wird.

c) Die folgende Grafik zeigt das *Sliding–Window* und die Coderausgabe zu den Zeitpunkten $i \geq 5$. Nach $i = 9$ Codierschritten ist der Codiervorgang unter Berücksichtigung von **eof** beendet.



d) Bei größerer Puffergröße ($G = 5$ anstelle von $G = 4$) ist die Codierung schon nach dem Codierschritt $i = 6$ abgeschlossen. Ein Vergleich der beiden Grafiken zeigt, dass sich für $G = 5$ gegenüber $G = 4$ nichts ändert bis einschließlich $i = 5$. Aufgrund des größeren Puffers lässt sich aber nun **BAR** gemeinsam mit **end–of–file** in einem einzigen Schritt codieren, während mit $G = 4$ hierfür vier Schritte notwendig waren.



e) Ein Nachteil von LZ77 ist das lokale Wörterbuch. Eigentlich schon bekannte Phrasen können nicht für die Datenkomprimierung verwendet werden, wenn sie mehr als G Zeichen vorher im Text aufgetreten sind. Dagegen sind bei LZ78 alle Phrasen im globalen Wörterbuch abgelegt. Richtig ist Aussage 1.

Die Aussage 2 trifft dagegen nicht zu.

- Richtig ist zwar, dass bei LZ78 nur Pärchen (I, Z) übertragen werden müssen, während bei LZ77 jeder Codierschritt durch ein Triple (P, L, Z) gekennzeichnet ist. Das bedeutet aber noch nicht, dass pro Codierschritt auch weniger Bit übertragen werden müssen.
- Betrachten wir beispielhaft die Puffergröße $G = 8$. Bei LZ77 muss man dann P mit 3 Bit und L mit 4 Bit dargestellen. Berücksichtigen Sie, dass die gefundene Übereinstimmung zwischen

Vorschaupuffer und Suchpuffer auch im Vorschaupuffer enden darf.

- Das neue Zeichen Z benötigt bei LZ78 genau die gleiche Bitanzahl wie bei LZ77 (nämlich 2 Bit), wenn man wie hier vom Symbolumfang $M = 4$ ausgeht). Die Aussage 2 wäre nur dann richtig, wenn N_I kleiner wäre als $N_P + N_L$, beispielsweise $N_I = 6$. Das würde aber bedeuten, dass man die Wörterbuchgröße auf $I = 2^6 = 64$ begrenzen müsste. Dies reicht für große Dateien nicht aus.
- Diese Überschlagsrechnung basiert allerdings auf einer einheitlichen Bitanzahl für den Index I. Mit **variabler Bitanzahl** für den Index kann man etliche Bit einsparen, indem man I nur mit so vielen Bit überträgt, wie es für den Codierschritt i erforderlich ist. Prinzipiell ändert das aber nichts an der Beschränkung der Wörterbuchgröße, was bei großen Dateien stets zu Problemen führen wird.

Musterlösung zur Aufgabe A2.4

a) Bei der LZW–Codierung des ersten Zeichens ($i = 1$) sind nur die vier Indizes $I = 0$, ... , $I = 3$ möglich, für die bereits zum Schritt $i = 0$ (Vorbelegung) ein Wörterbucheintrag vorgenommen wurde. Deshalb genügt es hier, den Index mit zwei Bit zu übertragen.

Bereits zum Schritt $i = 2$ werden dagegen drei Bit benötigt. Hätte die Eingangsfolge mit **AAA** begonnen, so hätte die LZW–Codierung $I_2 = I(i = 2)$ den Dezimalwert 4 ergeben. Dieser ist nicht mehr mit zwei Bit darstellbar und muss mit drei Bit codiert werden, ebenso wie I_3, I_4 und I_5.

Der vorgegebene Eingabestring

$$000000010101000001\text{...}$$

ist deshalb vom Decoder wie folgt aufzuteilen:

$$00|000|001|010|100|0001|\text{...}$$

Die Decoderergebnisse der ersten vier Schritte lauten somit:

$\underline{i = 1}$: $I =$ **00** (binär) $\underline{= 0}$ (dezimal) $\Rightarrow$ **A**,
$\underline{i = 2}$: $I =$ **000** (binär) $\underline{= 0}$ (dezimal) $\Rightarrow$ **A**,
$\underline{i = 3}$: $I =$ **001** (binär) $\underline{= 1}$ (dezimal) $\Rightarrow$ **B**,
$\underline{i = 4}$: $I =$ **010** (binär) $\underline{= 2}$ (dezimal) $\Rightarrow$ **C**.

b) Berücksichtigt man, dass

- für $i = 1$ zwei Bit verwendet werden,
- für $2 \le i \le 5$ drei Bit,
- für $6 \le i \le 19$ vier Bit,
- für $14 \le i \le 29$ fünf Bit,

so kommt man vom „kontinuierlichen Decoder–Eingangsstring"

$$000000010101000001100001110001000100111011101101101000110100 1$$

zum „eingeteilten Decoder–Eingabestring" (erste Zeile I_1, ... , I_8, in der zweiten Zeile I_9, ... , I_{16}):

$$00|000|001|010|100|0001|1000|0111|$$
$$0001|0001|0011|1011|1011|01101|00011|01001\,.$$

Damit lauten die gesuchten Ergebnisse für die Decodierschritte $i = 5$, ... , $i = 8$:

$\underline{i = 5}$: $I =$ **100** (binär) $\underline{= 4}$ (dezimal) $\Rightarrow$ **AA**,
$\underline{i = 6}$: $I =$ **0001** (binär) $\underline{= 1}$ (dezimal) $\Rightarrow$ **B**,
$\underline{i = 7}$: $I =$ **1000** (binär) $\underline{= 8}$ (dezimal) $\Rightarrow$ **AAB**,
$\underline{i = 8}$: $I =$ **0111** (binär) $\underline{= 7}$ (dezimal) $\Rightarrow$ **CA**.

c) Richtig ist <u>Aussage 2</u>. Man erhält folgende Decodierergebnisse (in verkürzter Form):

$I_9 = 1 \Rightarrow$ **B**, $I_{10} = 1 \Rightarrow$ **B**, $I_{11} = 3 \Rightarrow$ **D**, $I_{12} = 11 \Rightarrow$ **CAB**,
$I_{13} = 11 \Rightarrow$ **CAB**, $I_{14} = 13 \Rightarrow$ **BD**, $I_{15} = 3 \Rightarrow$ **D**, $I_{16} = 9 \Rightarrow$ **BA**.

d) Richtig sind die <u>Aussagen 1 und 4</u>, weil:

- Der neu ankommende Index ist 18 (dezimal) und im Wörterbuch wird unter dem Index $I = 18$ der Eintrag **DB** gefunden.
- Zum Decodierschritt $i = 17$ wird in das Wörterbuch die Zeile $I = 19$ eingetragen. Der Eintrag **BAD** setzt sich zusammen aus dem Decodierergebnis aus Schritt $i = 16$ (**BA**) und dem ersten Zeichen (**D**) des neuen Ergebnisses **DB**.

e) Richtig ist hier nur die Aussage 1:

- Für die erste Phrase genügen zwei Bit.
- Für die Phrasen I_2, ... , I_5 benötigt man drei Bit.
- Für die Phrasen I_6, ... , I_{13} benötigt man vier Bit.
- Für die Phrasen I_{14}, ... , I_{29} benötigt man fünf Bit.
- Für die Phrasen I_{30}, ... , I_{58} benötigt man sechs Bit.

Damit erhält man für die gesamte Bitanzahl:

$$N_{\rm Bit} = 1 \cdot 2 + 4 \cdot 3 + 8 \cdot 4 + 16 \cdot 5 + 29 \cdot 6 = 300\,.$$

Mit einheitlicher Bitlänge (6 Bit für jeden Index) wären $58 \cdot 6 = 348$ Bit (also mehr) erforderlich $\Rightarrow$ Aussage 2 ist prinzipiell falsch. Nun zur Aussage 3:

- Bei der uncodierten Übertragung von $N = 150$ Zeichen aus der Symbolmenge {**A**, **B**, **C**, **D**} würde man genau 300 Bit benötigen. Mit LZW benötigt man sicher mehr Bit, wenn die Quelle redundanzfrei ist (Zeichen gleichwahrscheinlich und statistisch unabhängig).
- Bei redundanter Quelle mit (beispielsweise) $H = 1.6$ Bit/Quellensymbol kann man die Bitanzahl auf $N_{\rm Bit} = N \cdot H$ begrenzen, vorausgesetzt, es handelt sich um eine sehr große Datei ($N \to \infty$).
- Bei einer eher kleinen Datei – wie hier lediglich mit $N = 150$ Bit – ist keine Aussage möglich, ob die Bitanzahl $N_{\rm Bit}$ kleiner, gleich oder größer als $150 \cdot 1.6 = 240$ sein wird. Auch $N_{\rm Bit} > 300$ ist durchaus möglich. Dann sollte man auf die „LZ–Komprimierung" besser verzichten.

Musterlösung zur Zusatzaufgabe Z2.4

a) Wir bezeichnen mit $W(I)$ ein Feld (Array), welches das Wörterbuch beschreibt und dessen Elemente Character oder Zeichenfolgen beinhalten. Die Codierung von **ABABABBAA** läuft wie folgt ab:

- $i = 1$: **A** → $\underline{I = 0}$, $W(I = 2)$ = **AB**,
- $i = 2$: **B** → $\underline{I = 1}$, $W(I = 3)$ = **BA**,
- $i = 3$: **AB** → $\underline{I = 2}$, $W(I = 4)$ = **ABA**,
- $i = 4$: **AB** → $\underline{I = 2}$, $W(I = 5)$ = **ABB**,
- $i = 5$: **BA** → $\underline{I = 3}$, $W(I = 6)$ = **BAA**.

Es ist anzumerken, dass das letzte Zeichen (**A**) des Eingabestrings **ABABABBAA** zum Zeitpunkt $i = 5$ zwar bereits beim Wörterbucheintrag berücksichtigt ist, aber noch nicht codiert wurde.

b) Für die Schritte 1 bis 3 ändert sich nichts gegenüber der Teilaufgabe (a). Danach gilt:

- $i = 4$: **ABA** → $\underline{I = 4}$, Wörterbuch ($I = 5$) = **ABAB**,
- $i = 5$: **BA** → $\underline{I = 3}$, Codierung abgeschlossen, kein neuer Wörterbucheintrag möglich.

c) Der Vergleich mit den obigen Ergebnissen zeigt, dass das Wörterbuch des Coders genau nach $\underline{i = 4}$ Codierschritten die gezeigten Einträge aufweist. Beim Decoder ergibt sich demgegenüber eine Zeitverzögerung um einen Schritt: $\underline{i = 5}$.

d) Die Sonderfallregelung der Decodierung ist (im vorliegenden Beispiel) dann notwendig, wenn im Codierschritt i der Index $I = i$ ausgegeben wird. Bei der Decodierung findet er dann die erforderliche Zuordnung Index → Zeichenfolge nicht, da das generierte Wörterbuch zum Zeitpunkt i nur Einträge mit Indizes $I < i$ enthält.

Für die Folge **ABABABBAA** gilt entsprechend Teilaufgabe a) stets $I < i$. Dagegen ergäbe sich bei der Folge **ABABABABA** folgende Indizes:

$$i = 1\text{: } I = 0\,,\ \ i = 2\text{: } I = 1\,,\ \ i = 3\text{: } I = 2\,,\ \ \underline{i = 4\text{: } I = 4}\,,\ \ i = 5\text{: } I = 3\,.$$

Richtig ist dementsprechend der Lösungsvorschlag 2.

Hier noch zusammenfassend die gesamte Decodierung von **ABABABABA**. Die Vorbelegung des Wörterbuchs beinhaltet $I = 0$: **A** und $I = 1$: **B**. Dann gilt mit dem Wörterbuch–Array $W(I)$:

- $i = 1$: Decodierung $I = 0$ → **A**,
- $i = 2$: Decodierung $I = 1$ → **B**, $W(I = 2)$ = **AB**,
- $i = 3$: Decodierung $I = 2$ → **AB**, $W(I = 3)$ = **BA**,
- $i = 4$: Ein Eintrag mit dem Index $I = 4$ ist nicht vorhanden ⇒ Sonderfallregelung. Man nimmt das letzte Decodierergebnis (hier **AB**) und fügt das erste Zeichen dieser Sequenz hinten an ⇒ **ABA**. Danach wird **ABA** im Wörterbuch unter dem Index $I = 4$ abgelegt.
- $i = 5$: Decodierung $I = 3$ → **BA**. Ende der Decodereingangsfolge.

Musterlösung zur Aufgabe A2.5

a) Die Näherung $r'(N)$ stimmt definitionsgemäß für die Eingangsfolgenlänge $N = 10000$ mit der per Simulation ermittelten Restredundanz $r(N) = 0.265$ exakt überein. Damit ist

$$A = 4 \cdot r(N = 10000) = 4 \cdot 0.265 \underline{= 1.06}.$$

b) Aus der Beziehung $A/\lg(N) \le 0.05 \quad \Rightarrow \quad A/\lg(N_b) = 0.05$ folgt:

$$\lg N_b = \frac{A}{0.05} = 21.2 \quad \Rightarrow \quad N_b = 10^{21.2} \underline{= 1.58 \cdot 10^{21}}.$$

c) Allgemein gilt:

$$r(N) = 1 - \frac{H}{K(N)}.$$

BQ1 hat die Entropie $H = 0.5$ bit/Symbol. Daraus folgt wegen $r(N) \approx r'(N)$ für $K(N_c) = 0.6$:

$$r(N_c) = 1 - \frac{0.5}{0.6} = 0.167 \Rightarrow \lg N_c = \frac{A}{0.167} = 6.36 \Rightarrow N_c = 10^{6.36} \underline{= 2.29 \cdot 10^6}.$$

d) Für $N = 10000$ gilt $r'(N) = r(N) = 0.190$:

$$\frac{A}{\lg 10000} = 0.19 \quad \Rightarrow \quad A = 0.19 \cdot 4 = 0.76.$$

Die Ergebnisse sind in nebenstehender Tabelle zusammengefasst. Man erkennt die sehr gute Übereinstimmung zwischen $r(N)$ und $r'(N)$. Die gesuchten Zahlenwerte sind in der Tabelle rot markiert.

Für den Komprimierungsfaktor gilt (der Apostroph weist darauf hin, dass von der Näherung $r'(N)$ ausgegangen wurde):

$$K'(N) = \frac{1}{1 - r'(N)}.$$

Damit gilt für die Länge des LZW–Ausgabestrings:

$$L'(N) = K'(N) \cdot N.$$

BQ2: redundanzfrei;
$H = 1$ bit/Quellensymbol

N	$r(N)$	$r'(N)$	$K'(N)$
1000	0.257	0.253	1.339
2000	0.231	0.230	1.299
5000	0.210	0.205	1.258
10000	0.190	0.190	1.235
20000	0.175	0.177	1.215
50000	0.161	0.162	1.193
10^6		0.127	1.145
10^9		0.084	1.092
10^{12}		0.063	1.067

e) Nach ähnlicher Vorgehensweise wie in der Teilaufgabe d) erhält man für die Binärquelle BQ3 den Anpassungsparameter $A = 1.36$ und daraus die Ergebnisse gemäß der blau hinterlegten Tabelle.

Hinweis: Die letzte Spalte dieser Tabelle ist nur bei Kenntnis der Teilaufgabe (f) verständlich. Dort wird gezeigt, dass die Quelle BQ3 die Entropie $H = 0.25$ bit/Quellensymbol besitzt.

In diesem Fall gilt für den Komprimierungsfaktor:

$$K'(N) = \frac{H}{1 - r'(N)} = \frac{0.25}{1 - r'(N)}.$$

Für $N = 10^{12}$ weicht also der Komprimierungsfaktor (0.282) noch deutlich von der Entropie (0.25) ab, die für $N \to \infty$ erreicht werden kann (Quellencodierungstheorem).

BQ3: Binärquelle mit H=0.25 bit/Quellensymbol

N	$r(N)$	$r'(N)$	$K'(N)$
1000	0.506	0.453	0.457
2000	0.436	0.412	0.425
5000	0.389	0.368	0.396
10000	0.340	0.340	0.379
20000	0.320	0.316	0.365
50000	0.285	0.289	0.351
10^6		0.227	0.323
10^9		0.151	0.294
10^{12}		0.113	0.282

f) Die einzelnen Näherungen $r'(N)$ unterscheiden sich nur durch den Parameter A. Dabei haben wir festgestellt:

- Quelle BQ2 ($H = 1.00$): $A = 0.76$,
- Quelle BQ1 ($H = 0.50$): $A = 1.06$,
- Quelle BQ3 (H unbekannt): $A = 1.36$.

Je kleiner die Entropie H ist, um so größer ist offensichtlich der Anpassungsfaktor A. Da genau eine Lösung möglich ist, muss H = 0.25 bit/Quellensymbol richtig sein $\Rightarrow$ Antwort 4.

Tatsächlich wurden bei der Simulation für die Quelle Q3 die Wahrscheinlichkeiten $p_A = 0.96$, $p_B = 0.04 \Rightarrow H \approx 0.25$ verwendet.

Musterlösung zur Zusatzaufgabe Z2.5

a) Der Komprimierungsfaktor ist definiert als der Quotient der Längen von LZW–Ausgangsfolge (L) und Eingangsfolge ($N = 10000$):

$$\text{BQ1}: \quad K = \frac{6800}{10000} \underline{= 0.680},$$

$$\text{BQ2}: \quad K = \frac{12330}{10000} \underline{= 1.233}.$$

Die Lempel–Ziv–Codierung macht natürlich nur bei der redundanten Binärquelle BQ1 Sinn. Hier kann die Datenmenge um 32% gesenkt werden. Bei der redundanzfreien Binärquelle BQ2 führt dagegen die LZ–Codierung zu einer um 23.3% größeren Datenmenge.

b) Aus der angegebenen Gleichung erhält man mit $N = 10000$:

$$\text{BQ1}: \quad H = 0.5\,, \;\; r(N = 10000) = 1 - \frac{0.5 \cdot N}{L} = 1 - \frac{5000}{6800} \underline{\approx 26.5\,\%},$$

$$\text{BQ2}: \quad H = 1.0\,, \;\; r(N = 10000) = 1 - \frac{N}{L} = 1 - \frac{10000}{12330} \underline{\approx 19\,\%}.$$

Die Restredundanz gibt die (relative) Redundanz der LZ–Ausgangsfolge an. Obwohl die Quelle BQ1 für die LZ–Codierung besser geeignet ist als die redundanzfreie Quelle BQ2, ergibt sich bei BQ1 wegen der Redundanz in der Eingangsfolge eine größere Restredundanz.

Eine kleinere Restredundanz $r(N)$ bei gegebenem N sagt also nichts darüber aus, ob der Einsatz von Lempel–Ziv für die vorliegende Quelle sinnvoll ist. Hierzu muss der Komprimierungsfaktor K betrachtet werden. Allgemein gilt folgender Zusammenhang zwischen $r(N)$ und $K(N)$:

$$r(N) = 1 - \frac{H}{K(N)}, \;\; K(N) = H \cdot \big(1 - r(N)\big)\,.$$

c) Aus der Tabelle auf der Angabenseite kann man ablesen (bzw. daraus ableiten)

- für die redundante Binärquelle BQ1:

$$L(N = 50000) = 32100\,, \;\; K(N = 50000) = 0.642\,, \;\; r(N = 50000) \underline{= 22.2\,\%},$$

- für die redundanzfreie Binärquelle BQ2:

$$L(N = 50000) = 59595\,, \;\; K(N = 50000) = 1.192\,, \;\; r(N = 50000) \underline{= 16.1\,\%}.$$

Richtig sind somit die Aussagen 1 und 2. Für beide Quellen ist $K(N = 50000) < K(N = 10000)$ und $r(N = 50000) < r(N = 10000)$. In beiden Fällen ergeben sich also bei größerem N „günstigere" Werte, auch dann, wenn eigentlich wie bei der redundanzfreien Binärquelle BQ2 die Anwendung von Lempel–Ziv zu einer Verschlechterung führt.

Musterlösung zur Aufgabe A2.6

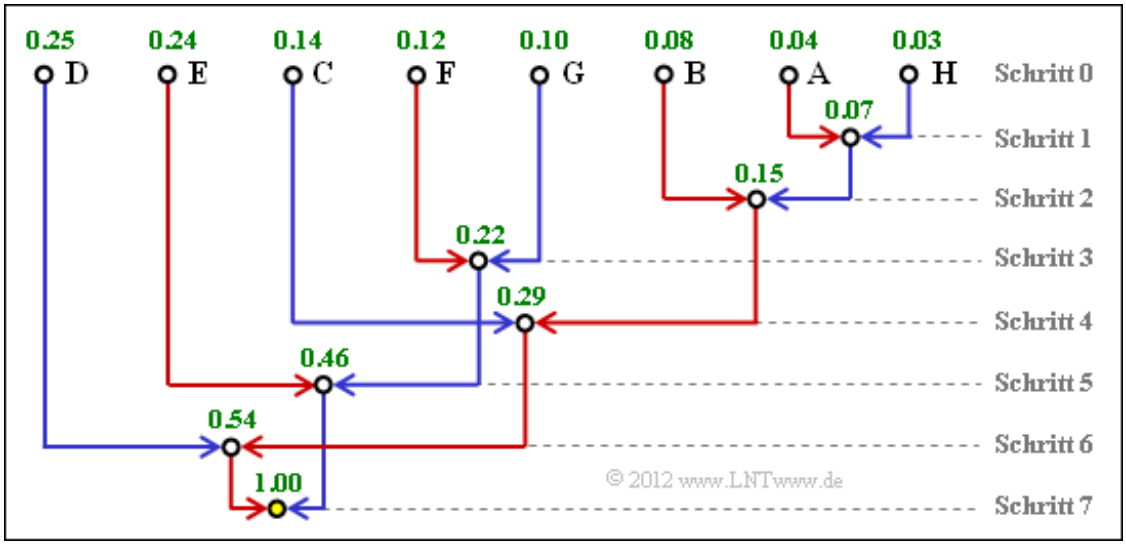
a) Vor dem Huffman–Algorithmus müssen die Symbole nach ihren Auftrittswahrscheinlichkeiten sortiert werden. Da die zwei unwahrscheinlichsten Symbole schon im Schritt 1 zusammengefasst werden, nehmen die Auftrittswahrscheinlichkeiten von oben nach unten ab (in der unteren Grafik zu dieser Musterlösung von links nach rechts). Durch Vergleich mit dem Angabenblatt erhält man:

Symbol **A**: Eingang 7, Symbol **B**: Eingang 6, Symbol **C**: Eingang 3, Symbol **D**: Eingang 1.

b Der Knoten R ist die Baumwurzel (*Root*). Unabhängig von den Auftrittswahrscheinlichkeiten ist dieser stets mit $R = 1$ belegt. Für die weiteren Werte gilt:

- Schritt 1: $U = p_{\rm A} + p_{\rm H} = 0.04 + 0.03 = 0.07$,
- Schritt 2: $V = U + p_{\rm B} = 0.07 + 0.08 = 0.15$,
- Schritt 3: $W = p_{\rm F} + p_{\rm G} = 0.12 + 0.10 = 0.22$,
- Schritt 4: $X = V + p_{\rm C} = 0.15 + 0.14 = 0.29$,
- Schritt 5: $Y = W + p_{\rm E} = 0.22 + 0.24 = 0.46$,
- Schritt 6: $Z = X + p_{\rm D} = 0.29 + 0.25 = 0.54$.

Damit lässt sich das Baumdiagramm auch wie folgt angeben.



c) Den Huffman–Code für das Symbol **A** erhält man, wenn man den Weg von der *Root* (gelber Punkt) zum Symbol **A** zurückverfolgt und jeder roten Verbindungslinie eine „**1**" zuordnet, jeder blauen eine „**0**".

- Symbol **A**: rot–rot–rot–blau–rot → **11101**,
- Symbol **B**: rot–rot–rot–rot → **1111**,
- Symbol **C**: rot–rot–blau → **110**,
- Symbol **D**: rot–blau– → **10**,
- Symbol **E**: blau–rot → **01**,
- Symbol **F**: blau–blau–rot → **001**,
- Symbol **G**: blau–blau–blau → **000**,
- Symbol **H**: rot–rot–rot–blau–blau → **11100**.

d) Die Codierung unter Punkt c) hat ergeben, dass

- die Symbole **D** und **E** mit 2 Bit,

- die Symbole **C**, **F** und **G** mit 3 Bit,
- das Symbol **B** mit 4 Bit, und
- die Symbole **A** und **H** mit 5 Bit

dargestellt werden. Damit erhält man:

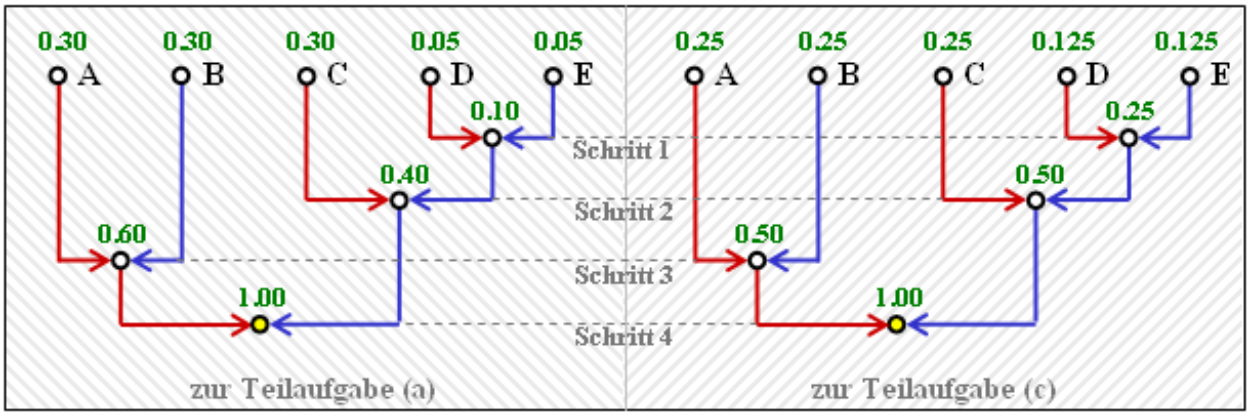
$$L_{\rm M} = (p_{\rm D} + p_{\rm E}) \cdot 2 + (p_{\rm C} + p_{\rm F} + p_{\rm G}) \cdot 3 + p_{\rm B} \cdot 4 + (p_{\rm A} + p_{\rm H}) \cdot 5 =$$
$$= 0.49 \cdot 2 + 0.36 \cdot 3 + 0.08 \cdot 4 + 0.07 \cdot 5 \underline{= 2.73\,\text{bit/Quellensymbol}}.$$

e) Die mittlere Codewortlänge $L_{\rm M}$ kann nicht kleiner sein als die Quellenentropie H. Damit scheiden die Antworten 2 und 3 aus. Richtig ist allein <u>Antwort 1</u>.

Man erkennt, dass die vorliegende Huffman–Codierung die durch das Quellencodierungstheorem vorgegebene Grenze $L_{\rm M,\,min} = H = 2.71$ bit/Quellensymbol nahezu erreicht.

Musterlösung zur Zusatzaufgabe Z2.6

a) Die Grafik zeigt die Konstruktion des Huffman–Codes mittels Baumdiagramm. Mit der Zuordnung rot → **1** und blau → **0** kommt man zu folgendem Code **A** → **11**, **B** → **10**, **C** → **01**, **D** → **001**, **E** → **000**. Richtig ist der Lösungsvorschlag 1.



Die linke Grafik gilt für die Wahrscheinlichkeiten gemäß Teilaufgabe (a). Das rechte Diagramm gehört zur Teilaufgabe (c) mit etwas anderen Wahrscheinlichkeiten. Es liefert den genau gleichen Code.

b) Nach dem **Quellencodierungstheorem** gilt stets $L_{\rm M} \ge H$. Voraussetzung für $L_{\rm M} = H$ ist allerdings, dass alle Symbolwahrscheinlichkeiten in der Form 2^{-k} (k = 1, 2, 3, ...) dargestellt werden können. Richtig ist demnach Lösungsvorschlag 3, wie auch die folgende Rechnung (mit „$\log_2$" ⇒ „ld") zeigt:

$$L_{\rm M} = (0.3 + 0.3 + 0.3) \cdot 2 + (0.05 + 0.05) \cdot 3 = 2.1\,{\rm bit/Quellensymbol}\,,$$
$$H = 3 \cdot 0.3 \cdot {\rm ld}\,(1/0.3) + 2 \cdot 0.05 \cdot {\rm ld}\,(1/0.05) \approx 2.0\,{\rm bit/Quellensymbol}\,.$$

c) **A**, **B**, **C** werden beim Code 1 durch 2 Bit dargestellt, **E**, **F** durch 3 Bit. Damit erhält man für

- die mittlere Codewortlänge

$$L_{\rm M} = p_{\rm A} \cdot 2 + p_{\rm B} \cdot 2 + p_{\rm C} \cdot 2 + p_{\rm D} \cdot 3 + p_{\rm E} \cdot 3\,,$$

- für die Quellenentropie:

$$H = p_{\rm A} \cdot {\rm ld}\,\frac{1}{p_{\rm A}} + p_{\rm B} \cdot {\rm ld}\,\frac{1}{p_{\rm B}} + p_{\rm C} \cdot {\rm ld}\,\frac{1}{p_{\rm C}} + p_{\rm D} \cdot {\rm ld}\,\frac{1}{p_{\rm D}} + p_{\rm E} \cdot {\rm ld}\,\frac{1}{p_{\rm E}}\,.$$

Durch Vergleich aller Terme kommt man zum Ergebnis:

$$p_{\rm A} = p_{\rm B} = p_{\rm C} = 0.25\,,\quad p_{\rm D} = p_{\rm E} = 0.125$$
$$\Rightarrow \quad L_{\rm M} = H = 2.25\,{\rm bit/Quellensymbol}\,.$$

Man erkennt: Mit diesen „günstigeren" Wahrscheinlichkeiten ergibt sich sogar eine größere mittlere Codewortlänge. Die Gleichheit ($L_{\rm M} = H$) ist allein auf die nun größere Quellenentropie zurückzuführen.

d) Beispielsweise liefert eine (von vielen) Simulationen mit den Wahrscheinlichkeiten gemäß der Teilaufgabe (c) die Folge **EBDCCBDABEBABCCCCCBCAABECAACCBAABBBCDCAB** (mit N = 40 Zeichen). Damit ergibt sich:

$$L'_{\rm M} = (34 \cdot 2 + 6 \cdot 3)/50 = 2.15\,{\rm bit/Quellensymbol}\,,$$

also ein kleinerer Wert als für die unendlich lange Folge (L_M = 2.25 bit/Quellensymbol). Bei anderem Startwert des Zufallsgenerators ist aber auch $L'_M \geq L_M$ möglich. Alle Aussagen sind zutreffend.

e) Richtig ist nur der Lösungsvorschlag 1.

- Code 1 ist ein Huffman–Code, wie schon in den vorherigen Teilaufgaben gezeigt wurde. Dies gilt zwar nicht für alle Symbolwahrscheinlichkeiten, aber zumindest für die Parametersätze gemäß den Teilaufgaben (a) und (c).
- Code 2 ist kein Huffman–Code, da ein solcher stets präfixfrei sein müsste. Die Präfixfreiheit ist hier aber nicht gegeben, da **0** der Beginn des Codewortes **01** ist.
- Code 3 ist ebenfalls kein Huffman–Code, da er eine um p_C (Wahrscheinlichkeit von **C**) größere mittlere Codewortlänge aufweist als erforderlich (Code 1). Er ist somit nicht optimal: Es gibt keine Symbolwahrscheinlichkeiten p_A, ... , p_E, die es rechtfertigen würden, das Symbol **C** mit **010** anstelle von **01** zu codieren.

Musterlösung zur Aufgabe A2.7

a) Bei redundanzfreier Binärquelle ($p_X = p_Y = 1/2$) erhält man $p_A = p_B = p_C = p_D = 1/4$ und mit der angegebenen Gleichung:

$$L_M = \big[p_A \cdot L_A + p_B \cdot L_B + p_C \cdot L_C + p_D \cdot L_D\big]/2 = \\ = \big[L_A + L_B + L_C + L_D\big]/8\,.$$

Berücksichtigt man die angegebenen Zuordnungen, so erhält man für

- Code 1: L_M = 1.000 bit/Quellensymbol,
- Code 2: L_M = 1.125 bit/Quellensymbol,
- Code 3: L_M = 1.250 bit/Quellensymbol.

Im Verlauf der Aufgabe wird sich zeigen, dass die beiden ersten Codes durchaus als Ergebnis des Huffman–Algorithmus möglich sind (natürlich nur bei geeigneten Symbolwahrscheinlichkeiten). Code 3 ist zwar ebenfalls **präfixfrei,** aber hinsichtlich der mittleren Codewortlänge nie optimal.

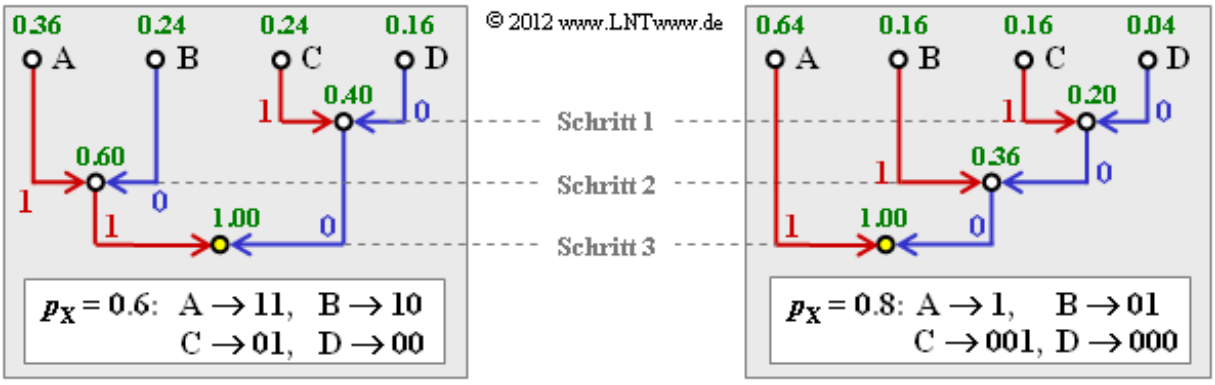
b) Die Wahrscheinlichkeiten der möglichen Zweiertupel lauten:

$$p_A = 0.6^2 = 0.36\,,\;\; p_B = 0.6 \cdot 0.4 = 0.24 = p_C\,,\;\; p_D = 0.4^2 = 0.16\,.$$

Damit ergibt sich das linke Baumdiagramm (siehe Grafik) und der folgende Huffman–Code:

A → 11, **B → 10**, **C → 01**, **D → 00**.

Es handelt sich um den Code 1 ⇒ Lösungsvorschlag 1.



c) Jedes Zweiertupel wird durch zwei Bit dargestellt. Damit ist L_M = 1 bit/Quellensymbol.

d) Hier lauten die Wahrscheinlichkeiten der einzelnen Zweiertupel:

$$p_A = 0.8^2 = 0.64\,,\;\; p_B = 0.8 \cdot 0.2 = 0.16 = p_C\,,\;\; p_D = 0.2^2 = 0.04\,.$$

Entsprechend dem rechten Baumdiagramm ergibt sich nun Code 2 ⇒ Lösungsvorschlag 2:

A → 1, **B → 01**, **C → 011**, **D → 010**.

e) Hier gilt für die mittlere Zweiertupellänge bzw. die mittlere Codewortlänge:

$$L'_M = 0.64 \cdot 1 + 0.16 \cdot 2 + (0.16 + 0.04) \cdot 3 = 1.56\,\text{bit/Zweiertupel}$$

$$\Rightarrow \quad L_M = \frac{L'_M}{2} = 0.78\,\text{bit/Quellensymbol}\,.$$

f) Beispielsweise ist für $p_X = 0.8$ entsprechend der Teilaufgabe (d) der Code 2 optimal und die mittlere Codewortlänge beträgt $L_M = 0.78$ bit/Quellensymbol. Für $p_X = 0.6$ ist dagegen Code 1 optimal und die mittlere Codewortlänge ist $L_M = 1$ bit/Quellensymbol (dieses Ergebnis ist unabhängig von p_X).

Der gesuchte Maximalwert $p_{X,\,max}$ wird zwischen 0.6 und 0.8 liegen. Die Bestimmungsgleichung ist dabei, dass für den Grenzfall $p_X = p_{X,\,max}$ beide Codes genau die gleiche mittlere Codewortlänge $L_M = 1$ bit/Quellensymbol besitzen, bzw. $L'_M = 2$ bit/Zweiertupel.

Mit der Abkürzung $p = p_{X,\,max}$ lautet die Gleichung:

$$L'_M\ (\text{Code 2}) = p^2 \cdot 1 + p \cdot (1-p) \cdot 2 + p \cdot (1-p) \cdot 3 + (1-p)^2 \cdot 3 \stackrel{!}{=} 2\,.$$

Dies führt zum zahlenmäßigen Ergebnis:

$$p^2 + p - 1 \stackrel{!}{=} 0 \quad \Rightarrow \quad p_{X,\,max} = p = \frac{\sqrt{5}-1}{2} \underline{\approx 0.618}\,.$$

Da sich die grundsätzliche Huffman–Struktur durch Vertauschen von **X** und **Y** nicht ändert, gilt für die untere Grenze:

$$p_{X,\,min} = 1 - p_{X,\,max} \underline{\approx 0.382}\,.$$

Die Darstellung der Zweiertupel durch unterschiedlich lange Bitfolgen (Code 2) macht also nur dann Sinn, wenn sich die Symbolwahrscheinlichkeiten von **X** und **Y** signifikant unterscheiden. Liegen diese dagegen zwischen 0.382 und 0.618, so ist Code 1 anzuwenden.

Die Aufteilung einer Strecke der Länge 1 in zwei Abschnitte der Länge 0.618... und 0.382... bezeichnet man als **Goldenen Schnitt,** auf den man in den verschiedensten Fachgebieten immer wieder stößt.

Musterlösung zur Zusatzaufgabe Z2.7

a) Die mittlere Codewortlänge ergibt sich mit $p_X = 0.7$, $L_X = 1$, $p_Y = 0.2$, $L_Y = 2$, $p_Z = 0.1$, $L_Z = 2$ zu

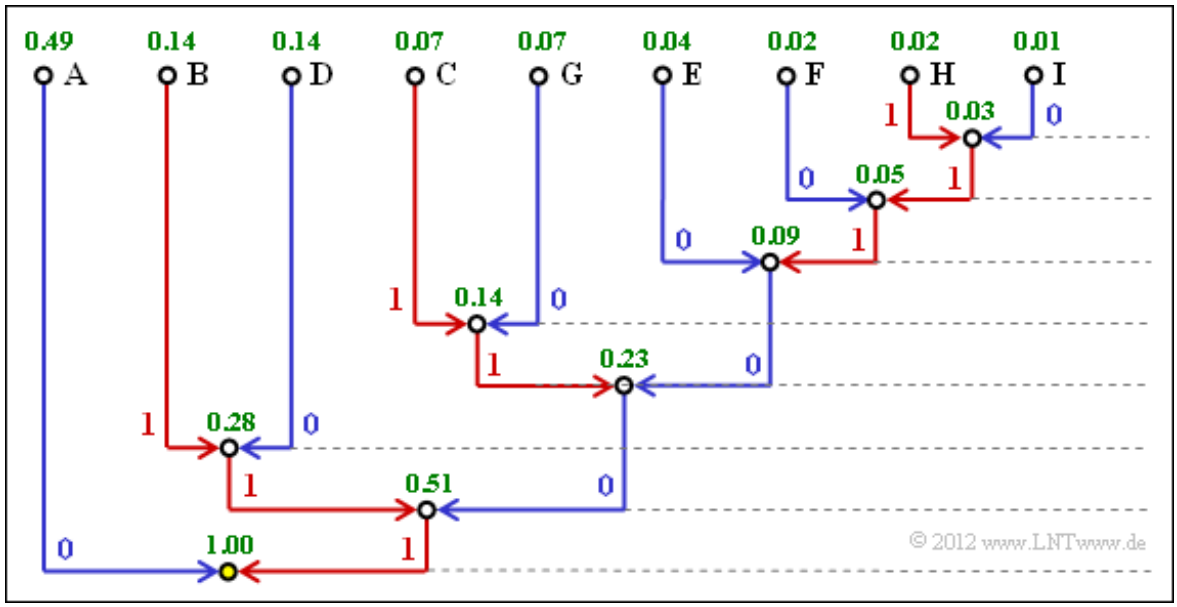
$$L_M = p_X \cdot 1 + (p_Y + p_Z) \cdot 2 \underline{= 1.3 \text{ bit/Quellensymbol}}.$$

Dieser Wert liegt noch deutlich über der Quellenentropie $H = 1.157$ bit/Quellensymbol.

b) Es gibt $M' = M^2 = 3^2 = 9$ Zweiertupel mit folgenden Wahrscheinlichkeiten:

$$p_A = \text{Pr}(\mathbf{XX}) \underline{= 0.49}, \quad p_B = \text{Pr}(\mathbf{XY}) \underline{= 0.14}, \quad p_C = \text{Pr}(\mathbf{XZ}) \underline{= 0.07},$$
$$p_D = \text{Pr}(\mathbf{YX}) = 0.14, \quad p_E = \text{Pr}(\mathbf{YY}) = 0.04, \quad p_F = \text{Pr}(\mathbf{YZ}) = 0.02,$$
$$p_G = \text{Pr}(\mathbf{YX}) = 0.07, \quad p_H = \text{Pr}(\mathbf{YY}) = 0.02, \quad p_I = \text{Pr}(\mathbf{YZ}) = 0.01.$$

c) Die Grafik zeigt den Huffman–Baum für die Anwendung mit $k = 2$.



Damit erhält man

- für die einzelnen Zweiertupels folgende Binärcodierungen:

  XX = A → 0, **XY = B → 111**, **XZ = C → 1011**, **YX = D → 110**, **YY = E → 1000**, **YZ = F → 10010**, **ZX = G → 1010**, **ZY = H → 100111**, **ZZ = I → 100110** .

- für die mittlere Codewortlänge:

$$L'_M = 0.49 \cdot 1 + (0.14 + 0.14) \cdot 3 + (0.07 + 0.04 + 0.07) \cdot 4 +$$
$$+ 0.02 \cdot 5 + (0.02 + 0.01) \cdot 6 = 2.33 \text{ bit/Zweiertupel}$$
$$\Rightarrow \quad L_M = L'_M/2 \underline{= 1.165 \text{ bit/Quellensymbol}}.$$

d) Richtig ist <u>Aussage 1</u>, auch wenn L_M mit wachsendem k nur sehr langsam abfällt.

- Die letzte Aussage ist falsch, da L_M auch für $k \to \infty$ nicht kleiner sein kann als $H = 1.157$ bit/Quellensymbol.
- Aber auch die zweite Aussage ist falsch: Da mit $k = 2$ weiterhin $L_M > H$ gilt, führt $k = 3$ zu einer Verbesserung.

Musterlösung zur Aufgabe A2.8

a) Bei der Quellensymbolfolge 2 erkennt man sehr viel weniger Symbolwechsel als in der roten Folge. Die blaue Symbolfolge 2 wurde mit dem Parameter

$$q = \Pr(\mathbf{X}\,|\,\mathbf{X}) = \Pr(\mathbf{Y}\,|\,\mathbf{Y}) = 0.8$$

erzeugt und die rote Symbolfolge 1 mit $q = 0.2$ ⇒ Richtig ist der Lösungsvorschlag 2.

b) Da hier die Quellensymbole **X** und **Y** gleichwahrscheinlich angenommen wurden, macht die direkte Anwendung von Huffman keinen Sinn. Dagegen kann man die inneren statistischen Bindungen der Markovquelle zur Datenkomprimierung nutzen, wenn man k–Tupel bildet ($k \ge 2$). Richtig sind demnach die Antworten 2 und 3. Je größer k ist, desto mehr nähert sich die Codewortlänge $L_{\rm M}$ der Entropie H.

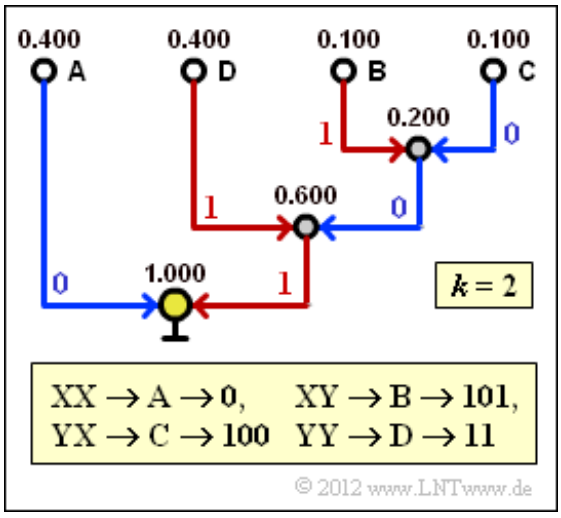
c) Die Symbolwahrscheinlichkeiten $p_{\rm X}$ und $p_{\rm Y}$ sind jeweils 0.5. Damit erhält man für die Zweiertupel:

$$\begin{aligned}
p_{\rm A} &= \Pr(\mathbf{XX}) = p_{\rm X} \cdot \Pr(\mathbf{X}\,|\,\mathbf{X}) = 0.5 \cdot q = 0.5 \cdot 0.8 \underline{= 0.4},\\
p_{\rm B} &= \Pr(\mathbf{XY}) = p_{\rm X} \cdot \Pr(\mathbf{Y}\,|\,\mathbf{X}) = 0.5 \cdot (1-q) = 0.5 \cdot 0.2 \underline{= 0.1},\\
p_{\rm C} &= \Pr(\mathbf{YX}) = p_{\rm Y} \cdot \Pr(\mathbf{X}\,|\,\mathbf{Y}) = 0.5 \cdot (1-q) = 0.5 \cdot 0.2 \underline{= 0.1},\\
p_{\rm D} &= \Pr(\mathbf{YY}) = p_{\rm Y} \cdot \Pr(\mathbf{Y}\,|\,\mathbf{Y}) = 0.5 \cdot q = 0.5 \cdot 0.8 \underline{= 0.4}.
\end{aligned}$$

d) Nebenstehender Bildschirmabzug des Programms **Shannon–Fano– und Huffman–Codierung** zeigt die Konstruktion des Huffman–Codes für $k = 2$ mit den soeben berechneten Wahrscheinlichkeiten. Damit gilt für die mittlere Codewortlänge:

$$\begin{aligned}
L'_{\rm M} &= 0.4 \cdot 1 + 0.4 \cdot 2 + (0.1 + 0.1) \cdot 3 =\\
&= \ 1.8\ \text{bit/Zweiertupel}
\end{aligned}$$

$$\Rightarrow \quad L_{\rm M} = \frac{L'_{\rm M}}{2} \underline{= 0.9\,\text{bit/Quellensymbol}}.$$



e) Nach dem Quellencodierungstheorem gilt $L_{\rm M} \ge H$. Wendet man aber Huffman–Codierung an und lässt dabei Bindungen zwischen nicht benachbarten Symbolen außer Betracht ($k = 2$), so gilt als unterste Grenze der Codewortlänge nicht $H = 0.722$, sondern $H_2 = 0.861$ (auf den Zusatz bit/Quellensymbol wird für den Rest der Aufgabe verzichtet) ⇒ Lösungsvorschlag 2.

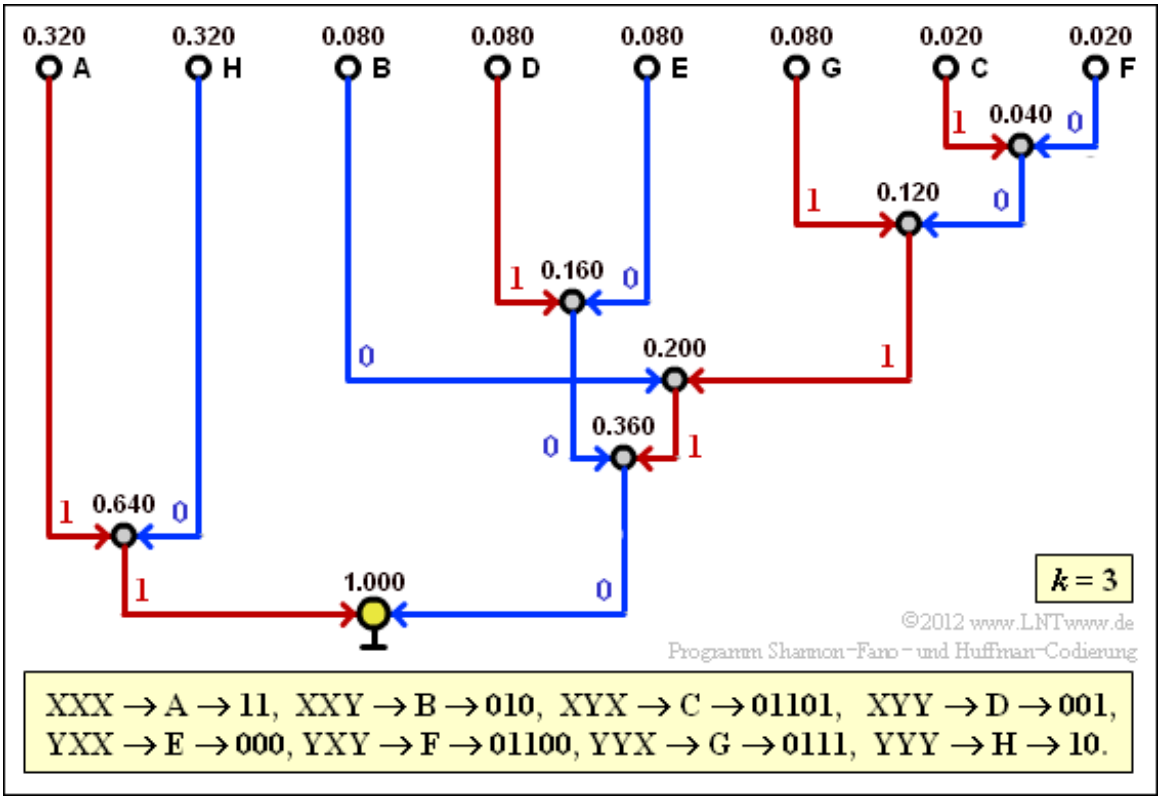
Das Ergebnis der Teilaufgabe d) war $L_{\rm M} = 0.9$. Würde eine unsymmetrische Markovkette vorliegen und zwar derart, dass sich für die Wahrscheinlichkeiten $p_{\rm A}$, ... , $p_{\rm D}$ die Werte 50%, 25% und zweimal 12.5% ergeben würden, so käme man auf die mittlere Codewortlänge $L_{\rm M} = 0.875$. Wie die genauen Parameter dieser unsymmetrischen Markovquelle aussehen, weiß ich (G. Söder) nicht. Auch nicht, wie sich der Wert 0.875 auf 0.861 senken ließe. Der Huffman–Algorithmus ist hierfür jedenfalls ungeeignet.

f) Mit $q = 0.8$ und $1 - q = 0.2$ erhält man:

$$p_{\rm A} = {\rm Pr}(\mathbf{XXX}) = 0.5 \cdot q^2 \underline{= 0.32} = p_{\rm H} = {\rm Pr}(\mathbf{YYY}),$$
$$p_{\rm B} = {\rm Pr}(\mathbf{XXY}) = 0.5 \cdot q \cdot (1-q) \underline{= 0.08} = p_{\rm G} = {\rm Pr}(\mathbf{YYX}),$$
$$p_{\rm C} = {\rm Pr}(\mathbf{XYX}) = 0.5 \cdot (1-q)^2 \underline{= 0.02} = p_{\rm F} = {\rm Pr}(\mathbf{YXY}),$$
$$p_{\rm D} = {\rm Pr}(\mathbf{XYY}) = 0.5 \cdot (1-q) \cdot q \underline{= 0.08} = p_{\rm E} = {\rm Pr}(\mathbf{YXX}).$$

g) Der Bildschirmabzug des Flash–Moduls verdeutlicht die Konstellation des Huffman–Codes für $k = 3$. Damit erhält man für die mittlere Codewortlänge:

$$L'_{\rm M} = 0.64 \cdot 2 + 0.24 \cdot 3 + 0.04 \cdot 5 = 2.52 \ {\rm bit/Dreiertupel}$$
$$\Rightarrow \quad L_{\rm M} = L'_{\rm M}/3 \underline{= 0.84 \, {\rm bit/Quellensymbol}}.$$



Man erkennt die Verbesserung gegenüber (d). Die für $k = 2$ gültige informationstheoretische Schranke $H_2 = 0.861$ wird nun unterschritten ($L_{\rm M}$). Die neue Schranke für $k = 3$ ist $H_3 = 0.815$. Um die Quellenentropie $H = 0.722$ zu erreichen (besser gesagt: diesem Endwert bis auf ein ε näher zu kommen), müsste man allerdings unendlich lange Tupel bilden ($k \to \infty$).

Musterlösung zur Aufgabe A2.9

a) Richtig ist der Vorschlag 3. Nachfolgend sehen Sie die durch Hochkommata eingeteilte Codesymbolfolge **1′01′001′000′1′1′000′01′001′1** ⇒ Quellensymbolfolge **ABCDAADBCA**.

b) Mit einem Bitfehler an Position 1 erhält man das folgende Decodierergebnis:

001′001′000′1′1′000′01′001′1 ⇒ CCDAADBCA ⇒ Lösungsvorschlag 1.

Das heißt: **AB** wird durch **C** ersetzt, der weitere Text **CDAADBCA** ist unverändert, allerdings um eine Position verschoben. Vergleicht man jedoch die ersten neun Symbole des Originals mit der Decodierung *Stelle für Stelle*, wie es ein Automat machen würde, so erkennt man acht unterschiedliche Symbole.

c) Richtig sind die Antworten 1 und 3:

- Durch einen zusätzlichen Bitfehler an Position 2 (**0 → 1**) wird **AB** zu **BA** verfälscht, aber alle weiteren Symbole wieder richtig erkannt.
- Ein zusätzlicher Bitfehler an Position 15 (**0 → 1**) führt zu **001′001′000′1′1′000′1′ 1′ 001′ 1** und damit zur Sinkensymbolfolge **CCDAADAACA**. Das neunte und das zehnte Symbol (beide grün markiert) und eventuell weitere Symbole werden richtig erkannt.
- Durch den ersten Bitfehler an Position 1 wird **AB** in **C** verfälscht, also ein Zeichen „verschluckt”. Ein weiterer Bitfehler an Position 10 macht aus **AA** ein **B**. Insgesamt verschluckt so der Decoder zwei Zeichen, und alle nachfolgend decodierten Zeichen stehen nicht an der richtigen Position.

d) Aus **001** wird **000**. Das bewirkt, dass insgesamt nur ein Fehler **C → D** entsteht:

101′000′000′1′1′000′01′001′1 ⇒ ABDDAADBCA ⇒ Lösungsvorschlag 2.

Musterlösung zur Aufgabe A2.10

a) Für den angegebenen Huffman–Code erhält man:

$$L_{\rm M} = 0.4 \cdot 1 + (0.17 + 0.14 + 0.10) \cdot 3 + 0.09 \cdot 4 + 0.05 \cdot 5 + (0.03 + 0.02) \cdot 6 =$$
$$= \underline{2.54\ \text{bit/Quellensymbol}}.$$

b) Vor Anwendung des Shannon–Fano–Algorithmus müssen die Zeichen zuerst noch nach ihren Auftrittswahrscheinlichkeiten sortiert werden. Damit ist Antwort 1 falsch.

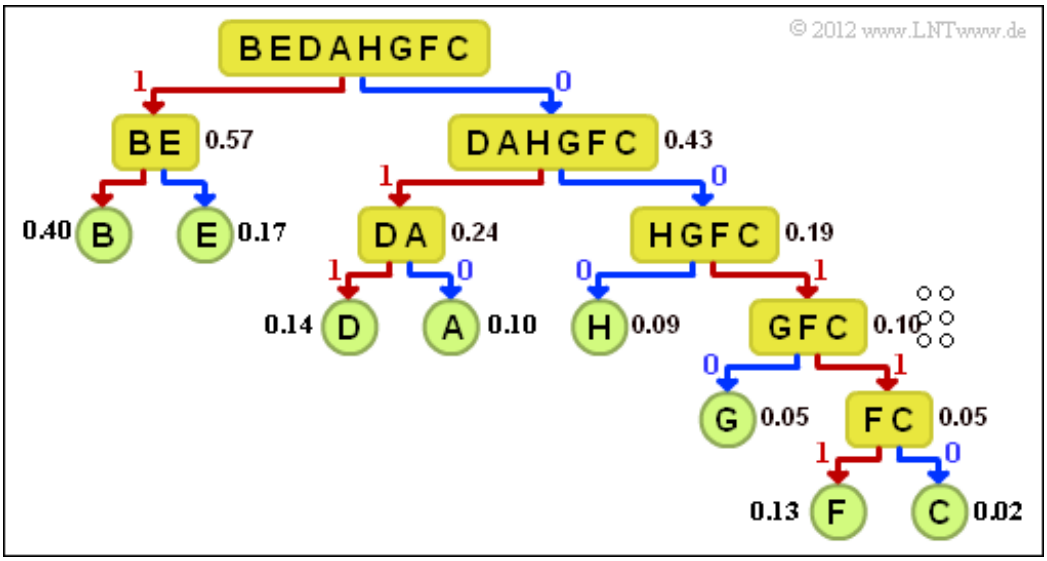
Richtig ist Antwort 2. Alle sortierten Zeichen müssen so in zwei Gruppen eingeteilt werden, dass die Gruppenwahrscheinlichkeiten möglichst gleich sind. Für den ersten Schritt bedeutet dies:

$$\Pr(\mathbf{BE}) = 0.57, \quad \Pr(\mathbf{DAHGFC}) = 0.43.$$

Bei der Aufteilung gemäß Lösungsvorschlag 3 würde die Gleichverteilung noch weniger erreicht:

$$\Pr(\mathbf{B}) = 0.40, \quad \Pr(\mathbf{EDAHGFC}) = 0.60.$$

c) Die Grafik zeigt das Baumdiagramm der Shannon–Fano–Codierung.



Daraus ergibt sich folgende Zuordnung (eine rote Verbindung weist auf **1** hin, eine blaue auf **0**):

A → 010, **B → 11**, **C → 00110**, **D → 011**, **E → 10**, **F → 00111**, **G → 0010**, **H → 000**.

Richtig sind alle vorgegebenen Lösungsvorschläge.

d) Mit dem Ergebnis der Teilaufgabe (c) erhält man:

$$L_{\rm M} = (0.40 + 0.17) \cdot 2 + (0.14 + 0.10 + 0.09) \cdot 3 + 0.05 \cdot 4 + (0.03 + 0.02) \cdot 5 =$$
$$= \underline{2.58\ \text{bit/Quellensymbol}}.$$

e) Richtig sind die Aussagen 2 und 3. Im vorliegenden Beispiel ergibt sich bei Shannon–Fano ein ungünstigerer Wert als bei Huffman. In den meisten Fällen – so auch im Beispiel auf der **Angabenseite** – ergibt sich für Huffman und Shannon–Fano ein gleichwertiger Code und damit auch die gleiche mittlere Codewortlänge. Einen effektiveren Code als Huffman liefert Shannon–Fano dagegen nie.

Musterlösung zur Aufgabe A2.11

a) Aus der Grafik auf der Angabenseite kann man die Wahrscheinlichkeiten ablesen:

$$p_{\rm X} = 0.7\,,\ \ p_{\rm Y} = 0.1\,,\ \ p_{\rm Z} = 0.2\,.$$

b) Auch das zweite Symbol ist **X**. Bei gleichem Vorgehen wie in der Aufgabenbeschreibung erhält man

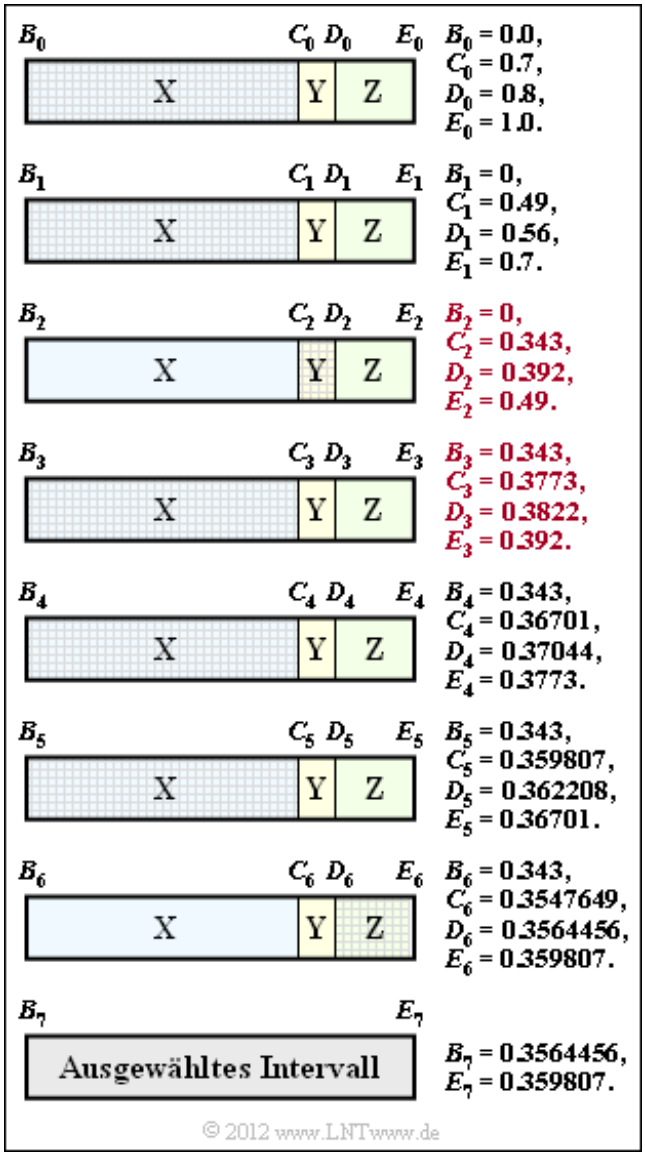
$$B_2 = \underline{0}\,,\ \ C_2 = 0.49 \cdot 0.7 = \underline{0.343}\,,$$
$$D_2 = \underline{0.392}\,,\ \ E_2 = C_1 = \underline{0.49}\,.$$

c) Aufgrund von **Y** als drittes Symbol gelten nun die Begrenzungen $B_3 = C_2$ und $E_3 = D_2$:

$$B_3 = \underline{0.343}\,,\ \ C_3 = \underline{0.3773}\,,$$
$$D_3 = \underline{0.3822}\,,\ \ E_3 = \underline{0.392}\,.$$

d) Aus $B_4 = 0.343 = B_3$ (abzulesen in der Grafik auf dem Angabenblatt) folgt zwingend, dass das vierte zu codierende Quellensymbol ein **X** war ⇒ <u>Lösungsvorschlag 1.</u>

e) Die Grafik zeigt die Intervallschachtelung mit allen bisherigen Ergebnissen.



- Man erkennt aus der Schraffierung, dass der zweite Lösungsvorschlag die richtige Symbolfolge angibt: **XXYXXXZ**.
- Die Intervallbreite Δ kann wirklich gemäß dem Lösungsvorschlag 3 ermittelt werden:

$$\Delta = 0.359807 - 0.3564456 = 0.003614\,,$$
$$\Delta = p_{\rm X}^5 \cdot p_{\rm Y} \cdot p_{\rm Z} = 0.7^5 \cdot 0.1 \cdot 0.2 = 0.003614\,.$$

Richtig sind somit die <u>Lösungsvorschläge 2 und 3.</u>

f) Der Vorschlag 1: $(0.101100)_{\rm binär}$ ist auszuschließen, da der zugehörige Dezimalwert $r_1 > 0.5$ ist. Auch der letzte Lösungsvorschlag ist falsch, da $(0.001011)_{\rm binär} < (0.01)_{\rm binär} = 0.25_{\rm dezimal}$ gilt.

Richtig ist der <u>Lösungsvorschlag 2</u>: $(0.010111)_{\rm binär}$, wegen:

$$r_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1 \cdot 2^{-6} = 0.359375\,.$$

Musterlösung zur Aufgabe A2.12

a) Das ausgewählte Intervall beginnt bei $B_3 = 0.343$ und endet bei $E_3 = 0.392$. Die Intervallbreite ist somit $\Delta_3 = 0.049$ und damit gilt mit dem *Logarithmus dualis* „$\log_2$” → „ld”:

$$N_{\rm Bit} = {\rm ld}\ \left\lceil \frac{1}{0.049} \right\rceil + 1 \underline{= 6}.$$

b) Das ausgewählte Intervall ergibt sich zu $I = [0.343,\ 0.392)$. Die Mitte liegt bei $M_3 = 0.3675$. Zur Bestimmung des arithmetischen Codes versuchen wir, die Intervallmitte durch eine Binärdarstellung möglichst gut zu erreichen. Da uns gerade kein entsprechendes Tool zur Lösung dieser Aufgabe zur Verfügung steht, gehen wir von folgenden Nebenrechnungen aus:

$H_4 = 2^{-2} + 2^{-4} = 0.3125$ ⇒ gehört nicht zum Intervall I,

$H_5 = H_4 + 2^{-5} = 0.34375 \in I$ ⇒ Binärdarstellung: 0.01011 ⇒ Code: **01011**.

$H_6 = H_5 + 2^{-6} = 0.359375 \in I$ ⇒ Binärdarstellung: 0.010111 ⇒ Code: **010111**.

$H_7 = H_6 + 2^{-7} = 0.3671875 \in I$ ⇒ Binärdarstellung: 0.0101111 ⇒ Code: **0101111**.

$H_{12} = H_7 + 2^{-12} = 0.3674316406 \in I$ ⇒ binär: 0.010111100001 ⇒ Code: **010111100001**.

Der entsprechende 6 Bit–Code lautet somit AC = **010111** ⇒ Richtig ist der <u>Vorschlag 2</u>.

c) Hier ergibt sich mit dem Beginn $B_7 = 0.3564456$ und dem Ende $E_7 = 0.359807$ die Intervallbreite $\Delta_7 = 0.0033614$ und damit

$$N_{\rm Bit} = \left\lceil {\rm ld}\ \frac{1}{0.0033614} \right\rceil + 1 = \lceil {\rm ld}\ 297.5 \rceil + 1 \underline{= 11}.$$

d) Die Binärdarstellung des Codes **01011100001** ergibt

$$2^{-2} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-11} = 0.3598632813 > E_7.$$

Richtig ist also <u>NEIN</u>. Wegen

$$2^{-2} + 2^{-4} + 2^{-5} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-11} = 0.3579101563$$
$$\Rightarrow \quad B_7 \le 0.3579101563 < E_7$$

ist der gültige arithmetische Code gleich **01011011101**.

e) <u>Alle Aussagen</u> sind richtig. Sie können sich davon beispielsweise in **[BCK02]** überzeugen.

Musterlösung zur Aufgabe A2.13

a) Die Binärfolge besteht aus insgesamt $N = 1250$ Binärsymbolen (ablesbar aus der letzten Spalte in der Tabelle). Damit benötigt man ohne Codierung ebenso viele Bit: $N_{\rm Bit} = \underline{1250}$.

b) Die gesamte Symbolfolge der Länge $N = 1250$ beinhaltet $N_{\rm B} = 25$ Symbole **B** und $N_{\rm A} = 1225$ Symbole **A**. Damit gilt für die relative Häufigkeit von **B**:

$$h_{\rm B} = \frac{N_{\rm B}}{N} = \frac{25}{1250} = \underline{0.02} = 2\%\,.$$

c) Wir betrachten nun *Run–Length Coding* (RLC), wobei jeder Abstand zwischen zwei **B**–Symbolen mit 8 Bit dargestellt wird ($D = 8$). Damit ergibt sich mit $N_{\rm B} = 25$:

$$N_{\rm Bit} = N_{\rm B} \cdot 8 = \underline{200}\,.$$

d) *Run–Length Coding* mit 7 Bit pro Codewort erlaubt für L_i nur Werte zwischen 0 und 127. Der Eintrag „226" in Zeile 19 ist aber größer ⇒ NEIN.

e) Auch bei *Run–Length Limited Coding* (RLLC) sind für die „echten" Abstände L_i mit $D = 7$ nur Werte zwischen 1 und $2^7 – 1 = 127$ zulässig. Der Eintrag „226" in Zeile 19 wird bei RLLC ersetzt durch

- Zeile 19a: **S** = **0000000** ⇒ Sonderzeichen, steht für „+ 127",
- Zeile 19b: **1100011** ⇒ Dezimal 99.

Damit erhält man insgesamt 26 Worte zu je 7 Bit:

$$N_{\rm Bit} = 26 \cdot 7 = \underline{182}\,.$$

f) Nun müssen bei RLLC gegenüber RLC (siehe Tabelle) folgende Änderungen vorgenommen werden:

- Zeile 1: $122 = 1 \cdot 63 + 59$ (ein Wort mehr),
- Zeile 6: $70 = 1 \cdot 63 + 7$ (ein Wort mehr),
- Zeile 7: $80 = 1 \cdot 63 + 17$ (ein Wort mehr),
- Zeile 12: $79 = 1 \cdot 63 + 18$ (ein Wort mehr),
- Zeile 13: $93 = 1 \cdot 63 + 30$ (ein Wort mehr),
- Zeile 19: $226 = 3 \cdot 63 + 37$ (drei Worte mehr),
- Zeile 25: $97 = 1 \cdot 63 + 34$ (ein Wort mehr).

Damit erhält man insgesamt 34 Worte zu je 6 Bit:

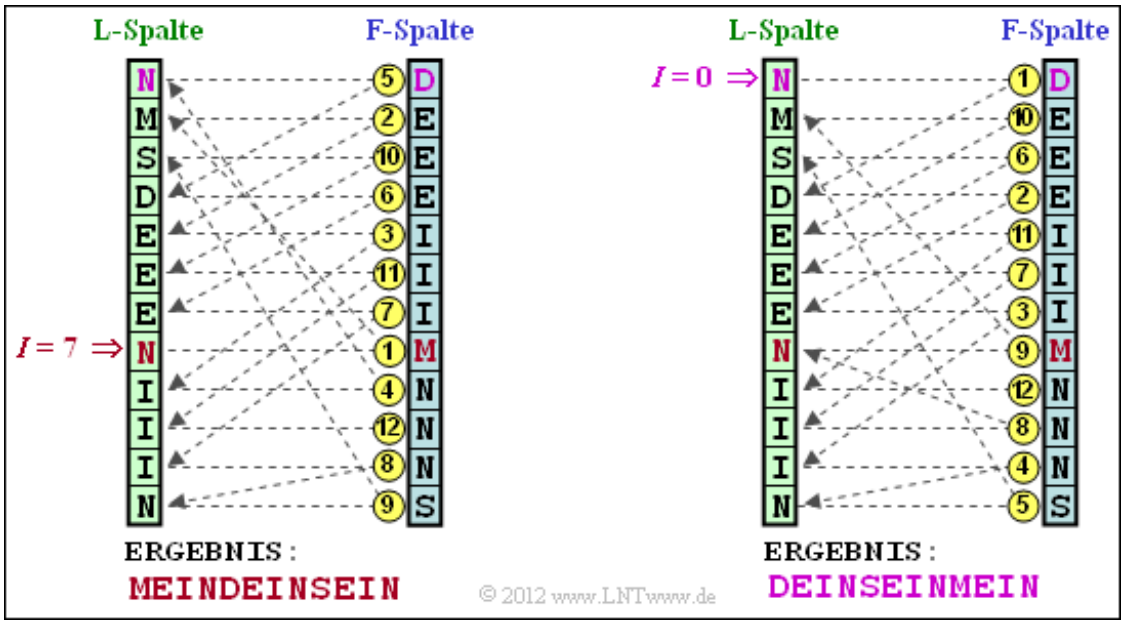
$$N_{\rm Bit} = 34 \cdot 6 = \underline{204}\,,$$

also ein schlechteres Ergebnis als mit 7 Bit gemäß Teilaufgabe (e).

Musterlösung zur Aufgabe A2.14

a) Man bezeichnet die erste Spalte der BWT–Matrix auch als F–Spalte und die letzte Spalte als L–Spalte (von „*First*" bzw. „*Last*"). Weitergegeben zur nächsten Codierstufe wird nur die L–Spalte. Die F–Spalte, die zur Rücktransformation ebenfalls benötigt wird, ergibt sich aus der L–Spalte durch lexikografisches Sortieren ⇒ Richtig ist der Lösungsvorschlag 3.

b) Richtig ist der Lösungsvorschlag 1: **MEINDEINSEIN**, wie aus der linken Darstellung hervorgeht. Beachten Sie, dass die obersten Zeile jeweils für die Zeilennummer „0" steht:



Zur Erklärung:

- Man beginnt die Decodierung mit der Zeile $I = 7$ der F–Spalte. Der Inhalt ist „M".
- Man sucht das entsprechende „M" in der L–Spalte und findet es in der Zeilennummer „1".
- Von Zeile 1 der L–Spalte geht man horizontal zur F–Spalte und findet das Symbol „E".
- In gleicher Weise findet man das dritte Ausgabesymbol „I" in der Zeile 4 der F–Spalte.
- Der Decodieralgorithmus endet mit dem Ausgabesymbol „N" in der drittletzten Zeile.

c) Richtig ist der Lösungsvorschlag 2: **DEINSEINMEIN**, wie aus der rechten Grafik hervorgeht.

d) Richtig ist der Lösungsvorschlag 3. Bei der BWT sind hier vier Zeichen gleich ihren Vorgängern, im Original kein einziges. In der F–Spalte wären zwar aufgrund der lexikografischen Sortierung noch mehr Zeichen gleich wie die jeweiligen Vorgänger (insgesamt 6), aber diese Sortierung lässt sich nicht verlustlos rückgängig machen. Auch der Lösungsvorschlag 1 ist falsch: Original und BWT beinhalten genau die gleichen Zeichen (dreimal **E**,dreimal **I**, dreimal **N** sowie je einmal **D**, **M** und **S**).

Musterlösung zur Zusatzaufgabe Z2.14

a) Die Grafik auf der Angabenseite zeigt, dass die Lösungsvorschläge 1 und 2 richtig sind und der Vorschlag 3 falsch ist. **E** und **I** treten zwar gruppiert auf, aber nicht die **N**–Zeichen.

b) Richtig sind die Lösungsvorschläge 2 und 3. Die Eingangsfolge wird Zeichen für Zeichen abgearbeitet. Auch die Ausgangsfolge hat somit die Länge $N = 12$.

Tatsächlich wird die Eingangsmenge {**D**, **E**, **I**, **N**, **M**, **S**} in die Ausgangsmenge {**0**, **1**, **2**, **3**, **4**, **5**} gewandelt. Allerdings nicht durch einfaches *Mapping*, sondern durch einen Algorithmus, der nachfolgend skizziert wird.

c) Die folgende Tabelle zeigt den MTF–Algorithmus. Der Schritt $i = 0$ (rote Hinterlegung) gibt die Vorbelegung an. Die Eingabe der MTF ist gelb hinterlegt, die Ausgabe grün.

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	$i=9$	$i=10$	$i=11$	$i=12$
$I=0$	D	N	M	S	D	E	E	E	N	I	I	I	N
$I=1$	E	D	N	M	S	D	D	D	E	N	N	N	I
$I=2$	I	E	D	N	M	S	S	S	D	E	E	E	E
$I=3$	M	I	E	D	N	M	M	M	S	D	D	D	D
$I=4$	N	M	I	E	E	N	N	N	M	S	S	S	S
$I=5$	S	S	S	I	I	I	I	I	I	M	M	M	M
Ausgabe		4	4	5	3	4	0	0	4	5	0	0	1

- Im Schritt $i = 1$ wird das Eingangszeichen **N** entsprechend der Spalte $i = 0$ durch den Index $I = \mathbf{4}$ dargestellt. Anschließend wird **N** nach vorne sortiert, während die Reihenfolge der anderen Zeichen gleich bleibt.
- Das Eingangszeichen **M** im zweiten Schritt erhält entsprechend der Spalte $i = 1$ ebenfalls den Index $I = \mathbf{4}$. In gleicher Weise macht man weiter bis zum 12. Zeichen **N**, dem der Index $I = \mathbf{1}$ zugeordnet wird.

Richtig ist Lösungsvorschlag 2. Man erkennt aus obiger Tabelle weiter, dass zu den Zeitpunkten $i = 6$, $i = 7$, $i = 10$ und $i = 11$ der Ausgabeindex jeweils $I = \mathbf{0}$ ist.

d) Richtig sind die Aussagen 1 und 2. Die Vorverarbeitungsschritte BWT und MTF haben lediglich die Aufgabe, möglichst viele Nullen zu generieren.

e) Alle Aussagen sind richtig. Nähere Angaben zum Huffman–Algorithmus finden Sie im **Kapitel 2.3.**