A2.1: Codierung mit und ohne Verlust

Man unterscheidet drei Arten von Codierverfahren, nämlich:

- **Leitungscodierung,**
- **Kanalcodierung,**
- **Quellencodierung.**

Alle diese grundlegenden Codierverfahren haben gemeinsam, dass das Quellensignal $q(t)$ durch eine Codesymbolfolge $\langle c_\nu \rangle$ dargestellt wird. Bei einer digitalen Quelle (mit oder ohne Gedächtnis) kann das Quellensignal $q(t)$ auch durch die Quellensymbolfolge $\langle q_\nu \rangle$ beschrieben werden.

Beim Empfänger wird aus der regenerierten Symbolfolge $\langle r_\nu \rangle$ die Sinkensymbolfolge $\langle v_\nu \rangle$ bzw. das Sinkensignal $v(t)$ gewonnen. Man spricht von *Decodierung*, manchmal auch von *Signalrekonstruktion*.

Alle rechts aufgeführten Begriffe gehören zu einer der drei oben aufgeführten Disziplinen, zwischen denen zwar eine gewisse Verwandtschaft besteht, die sich aber in Zielrichtung und mathematischer Handhabung durchaus unterscheiden.

Ein weiteres Unterscheidungsmerkmal bei codierter Übertragung ist:

- Man spricht dann von einem *verlustlosen Codierverfahren*, wenn nach der Decodierung $\langle v_\nu \rangle = \langle q_\nu \rangle$ bzw. $v(t) = q(t)$ gilt. Andernfalls ist das Codierverfahren *verlustbehaftet*.
- Voraussetzung für diese Klassifizierung ist eine fehlerfreie Übertragung: $\langle r_\nu \rangle = \langle c_\nu \rangle$.

Hinweis: Die Aufgabe gehört zum **Kapitel 2.1.** Die folgenden Fragen (c) bis (f) beziehen sich auf die Schlagworte in der obigen Grafik.

Fragebogen zu "A2.1: Codierung mit und ohne Verlust"

a) Bei welchen Codierverfahren wird Redundanz hinzugefügt?

- ☐ Verfahren zur Leitungscodierung,
- ☐ Verfahren zur Kanalcodierung,
- ☐ Verfahren zur Quellencodierung.

b) Welche Codierverfahren können durchaus verlustbehaftet sein?

- ☐ Verfahren zur Leitungscodierung,
- ☐ Verfahren zur Kanalcodierung,
- ☐ Verfahren zur Quellencodierung.

c) Wieviele der genannten Verfahren zählt man zur Leitungscodierung?

$$N_{\rm LC} =$$

d) Wieviele der genannten Verfahren zählt man zur Kanalcodierung?

$$N_{\rm KC} =$$

e) Wieviele Begriffe sind der verlustlosen Quellencodierung zuzuordnen?

$$N_{\rm QC\ (verlustlos)} =$$

f) Welche Aussagen gelten für die verlustbehafteten Quellencodierverfahren?

- ☐ AMR und EFR verwendet man im zellularen Mobilfunk.
- ☐ GIF, JPG und MP3 sind Komprimierungsverfahren für Bilder.
- ☐ MP3 wird zur Komprimierung von Audiodateien verwendet.

A2.2: Kraftsche Ungleichung

B1	B2	B3	B4
0 0 0	0	0	0
0 0 1	1 0	1 0	1 0
0 1 0	1 1 0	1 1 0	1 1 0
0 1 1	1 1 1 0	1 1 1 0	1 1 1
1 0 0	1 1 1 1	1 0 1 1	1 1 1 1
1 0 1			
1 1 0			
1 1 1			
T1	**T2**	**T3**	
0	0	0	
1 0	1	1	
1 1	1 0	2 0	
1 2 0	2 1	2 1	
1 2 1	2 2 0	2 2 0	
1 2 2			
2 0 0			
2 0 1			
2 0 2			

In der rechten Abbildung sind einige beispielhafte Binär– und Ternärcodes angegeben.

Beim Binärcode B1 werden alle möglichen Quellensymbole q_μ (mit Laufindex $\mu = 1, \ ... \ , 8$) durch jeweils eine Codesymbolfolge $\langle c_\mu \rangle$ einheitlicher Länge $L_\mu = 3$ dargestellt. Dieser Code ist aus diesem Grund zur Datenkomprimierung ungeeignet.

Die Möglichkeit zur Datenkomprimierung ergibt sich erst dann, wenn

- die M Quellensymbole nicht gleichwahrscheinlich sind,
- und die Länge L_μ der Codeworte unterschiedlich sind.

Diese Eigenschaft weist zum Beispiel der Binärcode B2 auf: Je ein Codewort hat hier die Länge 1, 2 und 3 ($N_1 = N_2 = N_3 = 1$) und zwei Codeworte haben die Länge $L_\mu = 4$ ($N_4 = 2$).

Voraussetzung für die Decodierbarkeit eines solchen Codes ist, dass der Code **präfixfrei** ist. Das heißt, dass kein Codewort der Präfix (der Beginn) eines längeren Codewortes sein darf.

Eine notwendige Bedingung dafür, dass ein Code zur Datenkomprimierung präfixfrei sein kann, wurde 1949 von Leon Kraft angegeben, die **Kraftsche Ungleichung:**

$$\sum_{\mu=1}^{M} D^{-L_\mu} \le 1 \, .$$

Hierbei bezeichnen

- M die Anzahl der möglichen Quellensymbole q_μ,
- L_μ die Länge des zum Quellensymbol q_μ gehörigen Codewortes c_μ,
- $D = 2$ einen Binärcode (**0** oder **1**) und $D = 3$ einen Ternärcode (**0**, **1**, **2**).

Ein Code kann nur dann präfixfrei sein, wenn die Kraftsche Ungleichung erfüllt ist. Die Umkehrung gilt nicht: Wird die Kraftsche Ungleichung erfüllt, so bedeutet das noch lange nicht, dass dieser Code tatsächlich präfixfrei ist.

Hinweis: Die Aufgabe bezieht sich auf das **Kapitel 2.1.**

Fragebogen zu "A2.2: Kraftsche Ungleichung"

a) Welche der Binärcodes erfüllen die Kraftsche Ungleichung?

- ☐ B1,
- ☐ B2,
- ☐ B3,
- ☐ B4.

b) Welche der vorgegebenen Binärcodes sind präfixfrei?

- ☐ B1,
- ☐ B2,
- ☐ B3,
- ☐ B4.

c) Welche der vorgegebenen Ternärcodes sind präfixfrei?

- ☐ T1,
- ☐ T2,
- ☐ T3.

d) Wie lauten die Kenngrößen des Codes T1?

Ternärcode T1: N_1 =

N_2 =

N_3 =

e) Wieviel 3–wertige Codeworte ($L_\mu = 3$) könnte man hinzufügen, ohne dass sich an der Präfixfreiheit etwas ändert?

Ternärcode T1: ΔN_3 =

f) Der Ternärcode T3 soll auf insgesamt $N = 9$ Codeworte erweitert werden. Wie erreicht man das ohne Verletzung der Präfixfreiheit?

- ☐ Ergänzung um vier 3–wertige Codeworte.
- ☐ Ergänzung um vier 4–wertige Codeworte.
- ☐ Ergänzung um ein 3–wertiges und drei 4–wertige Codeworte.

Z2.2: Mittlere Codewortlänge

Code C1	Code C2	Code C3
A → 0 0	A → 0	A → 0 0 1
B → 0 1	B → 1 0	B → 0 1
C → 1 0	C → 1 1 0	C → 1
D → 1 1	D → 1 1 1	D → 0 0 0

Ziel von Datenkomprimierung ist es, die Nachricht einer Quelle mit möglichst wenigen Binärzeichen darzustellen.

Wir betrachten hier eine wertdiskrete Nachrichtenquelle mit dem Symbolvorrat {**A**, **B**, **C**, **D**} ⇒ Symbolumfang $M = 4$ und den Auftrittswahrscheinlichkeiten

- $p_A = p_B = p_C = p_D = 1/4$ (Teilaufgabe a),
- $p_A = 1/2$, $p_B = 1/4$, $p_C = p_D = 1/8$ (ab Teilaufgabe b).

Vorausgesetzt wird, dass es zwischen den Quellensymbolen keine statistischen Bindungen gibt.

Ein Maß für die Güte eines Komprimierungsverfahrens ist die **mittlere Codewortlänge** L_M mit der Zusatzeinheit „bit/Quellensymbol”. Vorgegeben sind drei Zuordnungen. Anzumerken ist:

- Jeder dieser Binärcodes C1, C2 und C3 ist für eine spezielle Quellenstatistik ausgelegt.
- Alle Codes sind **präfixfrei** und somit ohne weitere Angabe sofort decodierbar.

Hinweis: Die Aufgabe bezieht sich auf das **Kapitel 2.1.**

Fragebogen zu "Z2.2: Mittlere Codewortlänge"

a) Bestimmen Sie die mittlere Codewortlänge für $p_A = p_B = p_C = p_D = 1/4$.

C1: L_M = bit/Quellensymbol

C2: L_M = bit/Quellensymbol

C3: L_M = bit/Quellensymbol

b) Welche Werte ergeben sich für $p_A = 1/2$, $p_B = 1/4$, $p_C = p_D = 1/8$.

C1: L_M = bit/Quellensymbol

C2: L_M = bit/Quellensymbol

C3: L_M = bit/Quellensymbol

c) Woran erkennt man präfixfreie Codes?

- ☐ Kein Codewort ist der Beginn eines anderen Codewortes.
- ☐ Alle Codeworte haben gleiche Länge.

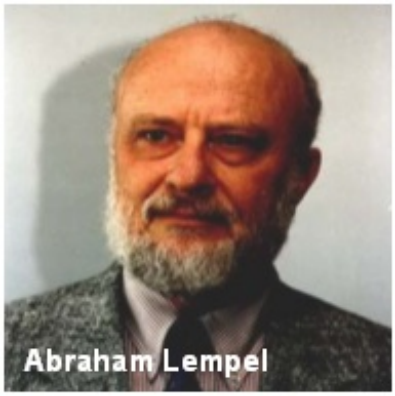
d) Für die spezielle Quellenfolge **ADBDCBCBADCA** ergibt sich die Codefolge **001101111001100100111000**. Welcher Code wurde verwendet?

- ☐ der Code C1,
- ☐ der Code C2.

e) Nach Codierung mit C3 erhält man **001101111001100100111000**. Wie lautet die zugehörige Quellensymbolfolge?

- ☐ **AACDBACABADAAA ....**
- ☐ **ACBCCCACAACCD ....**

A2.3: Zur LZ78–Komprimierung



Im Gegensatz zur **Entropiecodierung** nach Huffman oder nach Shannon, bei der man die Quellenstatistik (möglichst genau) kennen muss, sind solche Einschränkungen bei den von **Abraham Lempel** und **Jacob Ziv** entwickelten Komprimierungsverfahren nicht gegeben. Man spricht von universeller Quellencodierung.

Wir betrachten in dieser Aufgabe die 1978 erstmals veröffentlichte Variante **LZ78.** Codiert werden soll der String **BARBARA–BAR**.

- LZ78 arbeitet mit einem globalen Wörterbuch, das zu Beginn nur mit einem leeren Zeichen (**ε**) unter dem Index $I = 0$ gefüllt ist. Dadurch unterscheidet sich LZ78 von seinem Vorgänger **LZ77** (lokales Wörterbuch) und auch von seinem Nachfolger **LZW** (Wörterbuch ist mit den möglichen Zeichen vorbelegt).
- Wird ein Zeichen oder ein Wortfragment (mehrere Zeichen) des Eingabestrings im Wörterbuch gefunden, so wird der Index I_0 dieses Eintrags zusammen mit dem nächsten Eingangszeichen Z ausgegeben. In jedem Schritt i lautet also die Ausgabe: (I_0, Z).
- Anschließend wird der neue String unter dem nächsten freien Index $I_{\rm neu}$ ins Wörterbuch eingetragen. Betrachtet man das Wörterbuch als ein Feld $W[I]$ mit $I \ge 0$, bei dem ein jedes Element eine Zeichenkette beliebiger Länge enthält, so gilt mit der Character–Variablen Z:

$$W(I_{\rm neu}) = W(I_0) + Z\,.$$

Zur Verdeutlichung ein einfaches Beispiel:

- Zu einem gegebenen Zeitpunkt ist das Wörterbuch bis zum Index $I_{\rm akt} = 20$ gefüllt.
- Zur Codierung steht **Handy** an. Im Wörterbuch findet man unter dem Index $I = 11$ den Eintrag **Ha** und unter dem Index $I = 16$ den Eintrag **Han**.
- Somit lautet die aktuelle Coderausgabe $(I_0, Z) = (16, \mathbf{d})$ und ins Wörterbuch wird als neue Phrase eingetragen: $W(21) =$ **Hand**.
- Nun liegt der String **y** zur Codierung an. Findet man hierfür keinen passenden Eintrag, so wird $(0, \mathbf{y})$ ausgegeben und $W(22) = \mathbf{\varepsilon} + \mathbf{y} = \mathbf{y}$ neu ins Wörterbuch eingetragen.

Für die Teilaufgabe f) können Sie von folgenden Voraussetzungen ausgehen:

- Die Dezimalzahl I (Index) wird durch 3 Bit binär dargestellt.
- Das Zeichen $Z \in \{\mathbf{A}, \mathbf{B}, \mathbf{R}, –\}$ wird mit jeweils 2 Bit binärcodiert.

Hinweis: Die Aufgabe bezieht sich auf das **Kapitel 2.2.** Ähnliche Aufgaben zu anderen LZ–Methoden finden Sie unter folgenden Links:

- **Aufgabe Z2.3:** LZ77–Codierung von **BARBARA–BAR,**
- **Aufgabe A2.4:** LZW–(De–)Codierung einer binären Eingangsfolge.

Fragebogen zu "A2.3: Zur LZ78–Komprimierung"

a) Welche Aussagen gelten für die Vorbelegung des LZ78–Wörterbuches?

- ☐ Vorbelegt ist nur der Index $I = 0$ mit dem Leerzeichen (ε).
- ☐ Vorbelegt sind die Indizes $I = 0$ bis $I = 3$ mit den vier Zeichen.

b) Was geschieht im Codierschritt $i = 1$?

- ☐ Die Coderausgabe lautet (0, **B**).
- ☐ Der Wörterbucheintrag lautet: $W(I = 1)$ = **B**.
- ☐ Der Wörterbucheintrag lautet: $W(I = 1)$ = **BA**.

c) Welche Aussagen gelten für die Codierschritte $i = 2$ und $i = 3$?

- ☐ Für $i = 2$ gilt: Ausgabe (0, **A**), Eintrag $W(I = 2)$ = **A**.
- ☐ Für $i = 3$ gilt: Ausgabe (0, **R**), Eintrag $W(I = 3)$ = **R**.

d) Welche Aussagen gelten für den Codierschritt $i = 4$?

- ☐ Bei Schritt $i = 4$ wird **BAR** gemeinsam codiert.
- ☐ Bei Schritt $i = 4$ wird **BA** gemeinsam codiert.
- ☐ Die Coderausgabe lautet (2, **AR**).
- ☐ Die Coderausgabe lautet (1, **A**).

e) Vervollständigen Sie die LZ78–Codierung. Nach welchem Codierschritt i ist **BARBARA–BAR** vollständig codiert?

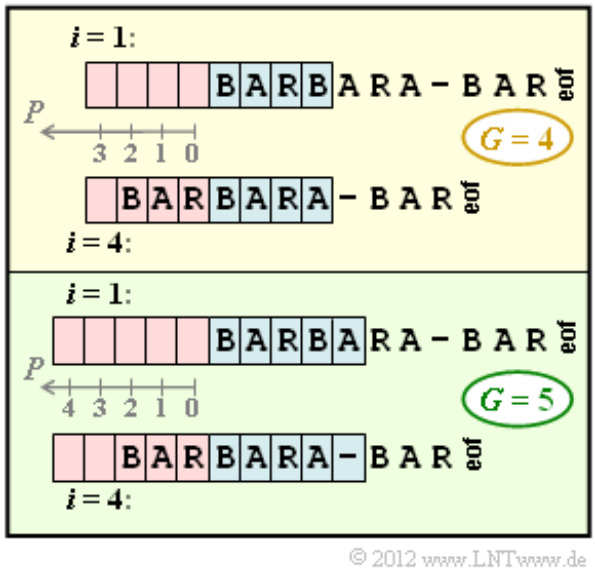
$\boldsymbol{i}$ =

f) Wieviele Binärzeichen benötigt man, um **BARBARA–BAR** zu codieren? Beachten Sie die Hinweise auf der Angabenseite.

ohne Codierung: N = Bit

mit LZ78–Codierung: N = Bit

Z2.3: Zur LZ77–Codierung



In der **Aufgabe A2.3** sollten Sie **BARBARA–BAR** (String der Länge 11, vier verschiedene Zeichen) mit dem LZ78–Algorithmus komprimieren. In dieser Aufgabe verwenden wir den gleichen Text zur Demonstration der **LZ77–Komprimierung.**

Anzumerken ist:

- Während beim Nachfolger LZ78 sukzessive ein globales Wörterbuch aufgebaut wird, verwendet LZ77 ein lokales Wörterbuch.
- Das LZ77–Verfahren arbeitet mit einem *Sliding Window*, das schrittweise über den Eingabetext verschoben wird.
- Dieses „gleitende Fenster" ist unterteilt in den Vorschaupuffer (in der Grafik blau hinterlegt) und den Suchpuffer (rote Hinterlegung). Beide Puffer haben eine Größe von G Speicherplätzen.
- Jeder Codierschritt i wird durch ein Zahlentriple (P, L, Z) charakterisiert. Hierbei sind P und L Integergrößen und Z ein Character. Übertragen werden die Binärdarstellungen von P, L und Z.
- Nach der Übertragung wird das *Sliding Window* um eine oder mehrere Positionen nach rechts verschoben und es beginnt der nächste Codierschritt $i + 1$.

Die obere Grafik zeigt die Anfangsbelegung mit Puffergröße $G = 4$ zu den Zeitpunkten $i = 1$ sowie $i = 4$. Zum Zeitpunkt $i = 1$ ist der Suchpuffer leer, so dass die Coderausgabe (0, 0, **B**) lautet. Nach der Verschiebung um eine Position beinhaltet der Suchpuffer ein **B**, aber keinen String, der mit **A** anfängt. Das zweite Zahlentriple ist somit (0, 0, **A**). Die Ausgabe für $i = 3$ lautet (0, 0, **R**), da im Suchpuffer auch jetzt keine Zeichenfolge zu finden ist, die mit **R** beginnt.

Die Momentaufnahme zum Zeitpunkt $i = 4$ ist ebenfalls in der Grafik angegeben. Gesucht ist nun die Zeichenfolge im Suchpuffer, die mit dem Vorschautext **BARA** am besten übereinstimmt. Übertragen wird wieder ein Zahlentriple (P, L, Z), aber nun mit folgender Bedeutung:

- P gibt die Position im (roten) Suchpuffer an, bei der die gefundene Übereinstimmung beginnt. Die jeweiligen P–Werte für die einzelnen Speicherplätze können der Grafik entnommen werden.
- L bezeichnet die Anzahl der Zeichen im Suchpuffer, die beginnend bei P mit dem aktuellen String im Vorschaupuffer übereinstimmen.
- Z bezeichnet schließlich das erste Zeichen im Vorschaupuffer, das sich vom gefundenen Übereinstimmungs–String im Suchpuffer unterscheidet.

Je größer der LZ77–Parameter G ist, um so leichter findet man eine möglichst lange Übereinstimmung. In der Teilaufgabe (d) werden Sie feststellen, dass die LZ77–Codierung mit $G = 5$ ein besseres Ergebnis liefert als diejenige mit $G = 4$. Aufgrund der nachfolgenden Binärdarstellung von P wird man allerdings G stets als Zweierpotenz wählen, so dass G mit ld P Bit darstellbar ist ($G = 8 \Rightarrow$ dreistellige Binärzahl P). Das heißt, ein *Sliding Window* mit $G = 5$ hat eher einen geringen Praxisbezug.

Hinweis: Die Aufgabe gehört zum Themengebiet von **Kapitel 2.2.**

Fragebogen zu "Z2.3: Zur LZ77–Codierung"

a) Wie lautet die LZ77–Ausgabe mit $G = 4$ bei Schritt $i = 4$?

- ☐ (0, 0, **B**),
- ☐ (2, 1, **A**),
- ☐ (2, 3, **A**).

b) Welche Aussage gilt für die gleiche Puffergröße $G = 4$ bei Schritt $i = 5$?

- ☐ Im Suchpuffer steht **BARA**.
- ☐ Im Vorschaupuffer steht **–BAR**.
- ☐ Die Ausgabe lautet (0, 0, **A**).

c) Nach welchem Schritt i ist die Codierung beendet?

$G = 4$: Nach i = Codierschritten

d) Nun gelte $G = 5$. Nach welchem Schritt i ist dann die Codierung beendet?

$G = 5$: Nach i = Codierschritten

e) Welche Vorteile hat LZ78 gegenüber LZ77 bei sehr großen Dateien?

- ☐ Man findet häufiger bereits abgelegte Phrasen im Wörterbuch.
- ☐ Pro Codierschritt müssen weniger Bit übertragen werden.

A2.4: LZW–Algorithmus

Index	Binär	Inhalt
0	00	A
1	01	B
2	10	C
3	11	D
4	100	AA
5	101	AB
6	110	BC
7	111	CA
8	1000	AAB
9	1001	BA
10	1010	AABC
11	1011	CAB
12	1100	BB
13	1101	BD
14	1110	DC
15	1111	CABC
16	10000	CABB
17	10001	BDD
18	10010	DB

Der Komprimierungsalgorithmus **LZW** – benannt nach den Erfindern **Abraham Lempel, Jacob Ziv** und **Terry Welch** – arbeitet ebenso wie **LZ78** mit einem globalen Wörterbuch. Dieses ist hier zu Beginn mit allen möglichen Zeichen – im Beispiel **A**, **B**, **C** und **D** – vorbelegt und wird während der Codierung sukzessive erweitert.

Bei der Decodierung entsteht genau das gleiche Wörterbuch, nur erfolgt der gleiche Eintrag mit dem Index I einen Schritt später als während der Codierung. Zur Bezeichnung der Decodierschritte verwenden wir hier die Laufvariable i.

Hier noch einige Hinweise zur LZW–Codierung und –Decodierung:

- Bei der Codierung wird zu jedem Zeitpunkt i im Wörterbuch nach einer möglichst langen Zeichenkette gesucht, die mit dem aktuell anliegenden Eingabe–String übereinstimmt.
- Der gefundene Wörterbuchindex I_i wird stets in Binärform übertragen. Gleichzeitig wird ins Wörterbuch unter dem nächsten freien Index $W(I_i) + Z$ eingetragen.
- Hierbei bezeichner $W(I_i)$ ein Zeichen oder eine Zeichenfolge, und Z ist das erste Zeichen des anstehenden Eingabe–Strings (also ebenfalls ein Character), das in $W(I_i)$ nicht mehr berücksichtigt ist.
- Bei $M = 4$ möglichen Zeichen wird der erste Index I_1 mit 2 Bit übertragen, die Indizes I_2, ..., I_5 mit 3 Bit, die nächsten 8 Indizes mit 4 Bit, danach 16 Indizes mit 5 Bit usw. Die Begründung für diese bitsparende Maßnahme finden Sie in der **Musterlösung** zur Aufgabe.

Nach der Codierung in der hier beschriebenen Art und Weise über 16 Codierschritte ergibt sich die folgende Binärfolge der Länge $N_{\rm Bit} = 61$:

$$00000001010100000110000111000100010011101110110110100011010 01.$$

Aufgabe des Decoders (genauer gesagt: Ihre Aufgabe) ist es nun,

- aus dieser Binärsequenz die Indizes I_1, ... , I_{16} zu rekonstruieren, wobei die unterschiedliche Bitanzahl zu berücksichtigen ist (Beschreibung der 16 Decodierergebnisse durch Dezimalzahlen),
- anschließend aus dem Wörterbuch entsprechend den Indizes die zugehörigen Zeichen bzw. Zeichenfolgen auszulesen und schließlich – mit einem Schritt Verzögerung – den neuen Wörterbucheintrag zu generieren.

Hinweis: Die Aufgabe bezieht sich auf das **Kapitel 2.2.**

Fragebogen zu "A2.4: LZW–Algorithmus"

a) Welche Indizes stehen zu den ersten vier Schritten zur Decodierung an? Geben Sie alle I als Dezimalzahlen ein.

$$i = 1: \quad I_1 =$$

$$i = 2: \quad I_2 =$$

$$i = 3: \quad I_3 =$$

$$i = 4: \quad I_4 =$$

b) Setzen Sie die Unterteilung des Decoder–Eingabestrings bis zum Ende fort. Welche Indizes ergeben sich bei den aufgeführten Decodierschritten?

$$i = 5: \quad I_5 =$$

$$i = 6: \quad I_6 =$$

$$i = 7: \quad I_7 =$$

$$i = 8: \quad I_8 =$$

c) Wie lautet der Ausgabe–String nach $i = 16$ Decodierschritten?

- ☐ **AABCAABAABCAABCABBDBBDCDBA**,
- ☐ **AABCAABAABCABBDCABCABBDDBA**,
- ☐ **AABCAABAABCABDBBABCCDBAABDCDBA**.

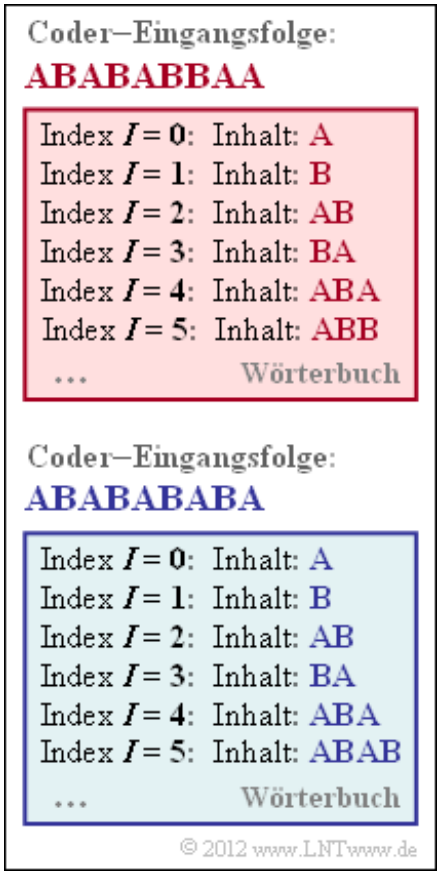
d) Der nächste Index lautet: I_{17} = 10010 (binär) = 18 (dezimal). Welche der folgenden Aussagen treffen zu?

- ☐ Die Decoderausgabe zum Schritt $i = 17$ lautet **DB**.
- ☐ Die Decoderausgabe zum Schritt $i = 17$ lautet **BAD**.
- ☐ Neuer Wörterbucheintrag **DB** in die Zeile $I = 19$.
- ☐ Neuer Wörterbucheintrag **BAD** in die Zeile $I = 19$.

e) Der Decoder–Eingabestring besteht aus $N_{\rm Bit} = 300$ Binärzeichen (Bit). Welche der folgenden Aussagen sind dann sicher zutreffend?

- ☐ Übertragen wurden 58 LZW–Phrasen mit variabler Bitlänge.
- ☐ Mit einheitlicher Indexbitlänge wären weniger Bit erforderlich.
- ☐ Übertragen wurde eine Datei mit $N = 150$ Quaternärzeichen.

Z2.4: LZW–Codierung / Decodierung



Die obere Grafik zeigt eine Momentaufnahme des Wörterbuchs, das während der **LZW–Codierung** der Eingangssymbolfolge **ABABABBAA** entsteht. Das untere Wörterbuch entsteht bei der LZW–Codierung der Sequenz **ABABABABA**. In beiden Fällen wird vorausgesetzt, dass keine andere Zeichen als **A** und **B** vorkommen.

Gleiche Wörterbücher entstehen bei der **LZW–Decodierung,** doch erfolgen dann die Wörterbucheinträge erst einen Schritt später. In der Teilaufgabe (c) wird gefragt, für welchen Codierschritt bzw. für welchen Decodierschritt die dargestellten Momentaufnahmen gültig sind.

Bei der LZW–Codierung wird zu jedem Codierschritt i ein Index I ausgewählt und (binär) übertragen. Das Zeichenpaar **AB** wird bei den beiden Wörterbüchern durch den Index $I = 2$ dargestellt. Wir betrachten hier den Index I als Dezimalzahl und lassen bei dieser Aufgabe die Binärdarstellung außer Betracht.

Bei der LZW–Decodierung wird in gleicher Weise mit Hilfe des Wörterbuchs aus jedem Index I ein Zeichen bzw. eine Zeichenfolge generiert, zum Beispiel führt $I = 1$ zum Zeichen **B** und $I = 2$ zum Zeichenpaar **AB**.

Wird tatsächlich ein Wörterbucheintrag mit dem gewünschten Index I gefunden, so läuft die Decodierung problemlos ab. Dies ist aber nicht immer so:

- Wird bei der Codierung beim Schritt i ein neuer Index I eingetragen und ist dieses I gleichzeitig das Codierergebnis des Schrittes, so ist dieser Index beim Decodierschritt i im Wörterbuch noch nicht belegt. Der Grund dafür ist, dass beim Decoder die Einträge um einen Schritt später erfolgen.
- Bei binärer Eingangsfolge (alle Zeichen seien **A** oder **B**) ist bei der LZW–Decodierung genau immer dann eine Sonderregelung anzuwenden, wenn im Codierschritt i der Eintrag mit dem Index $I = i$ vorgenommen wurde.

Diese Sonderregelung soll an einem Beispiel veranschaulicht werden:

- Zum Schritt i gibt es keinen zum Index I passenden Eintrag im Decoder–Wörterbuch.
- Wir nehmen an, dass das Decodierergebnis beim vorherigen Schritt $(i – 1)$ **ABBABA** war.
- Dann ergänzt man diese Zeichenfolge um das erste Zeichen der Folge. Hier: **ABBABAA**.
- Anschließend trägt man die Sequenz **ABBABAA** in das Wörterbuch unter dem Index I ein.

Hinweis: Die Aufgabe gehört zum Themengebiet von **Kapitel 2.2.** Beachten Sie bei der Lösung dieser Aufgabe, dass beim LZW–Algorithmus nicht von einem leeren Wörterbuch ausgegangen wird. Vielmehr beinhalten die Indizes $I = 0$ bis $I = M–1$ alle M zulässigen Zeichen.

Fragebogen zu "Z2.4: LZW–Codierung / Decodierung"

a) Codieren Sie die Eingangsfolge **ABABABBAA**. Welche Indizes ergeben sich zu den Schritten $i = 1, \ldots, 5$?

ABABABBAA, $i = 1$: I =

$i = 2$: I =

$i = 3$: I =

$i = 4$: I =

$i = 5$: I =

b) Codieren Sie nun die Eingangsfolge **ABABABABA**. Geben Sie die Indizes zu den Schritten $i = 4$ und $i = 5$ an.

ABABABABA, $i = 4$: I =

$i = 5$: I =

c) Für welchen Schritt (i) gilt die Momentaufnahme des auf der Angabenseite dargestellten Wörterbuchs bezüglich

Codierung: i =

Decodierung: i =

d) Wann muss man auf die Decodier–Sonderfallregelung zurückgreifen?

- ☐ Bei der Decodierung von **ABABABBAA** im Schritt $i = 4$.
- ☐ Bei der Decodierung von **ABABABABA** im Schritt $i = 4$.
- ☐ Bei der Decodierung von **ABABABABA** im Schritt $i = 5$.

A2.5: Relative Restredundanz

Wir gehen hier von einer binären Eingangsfolge der Länge N aus und betrachten drei verschiedene binäre Nachrichtenquellen BQ1, BQ2 und BQ3:

- **BQ1:** mit den Symbolwahrscheinlichkeiten $p_A = 0.89$, $p_B = 0.11$, also unterschiedlich
  ⇒ Entropie $H = 0.5$ bit/Quellensymbol
  ⇒ Quelle ist redundand.
- **BQ2:** $p_A = p_B = 0.5$ (gleichwahrscheinlich)
  ⇒ Entropie $H = 1$ bit/Quellensymbol
  ⇒ Quelle ist redundanzfrei.
- **BQ3:** Hier gibt es keine konkreten Angaben zur Statistik. In der Teilaufgabe (f) sollen Sie die Entropie H dieser Quelle abschätzen.

N	$r(N)$	$r'(N)$	$r(N)$	$r'(N)$	$r(N)$	$r'(N)$
1000	0.352	0.353	0.257	???	0.506	???
2000	0.328	0.321	0.231	???	0.436	???
5000	0.289	0.287	0.210	???	0.389	???
10000	0.265	0.265	0.190	0.190	0.341	0.341
20000	0.242	0.246	0.175	???	0.320	???
50000	0.221	0.226	0.161	???	0.292	???
10^6		0.177		???		???
10^9		0.118		???		???
10^{12}		0.088		???		???
	BQ1		BQ2		BQ3	

Für diese drei Quellen wurden per Simulation die jeweilige *Restredundanz* $r(N)$ ermittelt, die nach der **Lempel–Ziv–Welch–Codierung** in der Binärfolge verbleibt. Die Ergebnisse sind in der jeweils ersten Spalte obiger Tabelle für die Quellen

- BQ1 (gelbe Hinterlegung),
- BQ2 (grüne Hinterlegung),
- BQ3 (blaue Hinterlegung)

eingetragen, wobei wir uns bei der Simulation auf Folgenlängen $N \le 50000$ beschränkt haben.

Die *relative Redundanz der Ausgangsfolge* – vereinfachend *Restredundanz* genannt – kann aus

- der Länge N der Eingangsfolge,
- der Länge $L(N)$ der Ausgangsfolge und
- der Quellenentropie H

in folgender Weise berechnet werden:

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

Hierbei ist berücksichtigt, dass bei perfekter Quellencodierung die Länge der Ausgangsfolge bis auf den Wert $L_{min} = N \cdot H$ herabgesenkt werden könnte. Bei nichtperfekter Quellencodierung gibt $L(N) - N \cdot H$ die verbleibende Redundanz (mit der Pseudo–Einheit „bit") an. Nach Division durch $L(N)$ erhält man die relative Redundanz $r(N)$ mit dem Wertebereich zwischen 0 und 1; $r(N)$ sollte möglichst klein sein.

Eine zweite Kenngröße zur Effizienzmessung der LZW–Codierung ist der *Komprimierungsfaktor*

$$K(N) = L(N)/N,$$

der ebenfalls klein sein sollte. *Hinweis*: $K(N)$ und $r(N)$ hängen deterministisch zusammen.

Im **Theorieteil** wurde gezeigt, dass die Restredundanz $r(N)$ durch die Funktion

$$r'(N) = A/\lg\,(N) \quad \text{mit} \quad A = 4 \cdot r(N = 10000)\,.$$

oft gut angenähert wird. Die Näherung $r'(N)$ ist für BQ1 in der zweiten Spalte obiger Tabelle angegeben. In den Teilaufgaben (d) und (e) sollen Sie die Approximation für die Quellen BQ2 und BQ3 vornehmen.

Hinweis: Die Aufgabe bezieht sich auf **Seite 7** und **Seite 8** von **Kapitel 2.2.**

Fragebogen zu "A2.5: Relative Restredundanz"

a) Mit welchem Parameter A wurde die Näherung $r'(N)$ der Restredundanz für die Binärquelle BQ1 erstellt?

BQ1: A =

b) Wie groß muss $N = N_b$ mindestens sein, damit die Restredundanz die Bedingung $r(N) \le 5\%$ erfüllt? *Hinweis:* Ersetzen Sie $r(N)$ durch $r'(N)$.

BQ1: N_b =

c) Wie groß muss $N = N_c$ mindestens sein, damit der Komprimierungsfaktor $\Rightarrow$ $K(N) = L(N)/N$ nicht größer ist als 0.6?

BQ1: N_c =

d) Bestimmen Sie nun die Redundanznäherung $r'(N)$ für die redundanzfreie Binärquelle BQ2, insbesondere:

BQ2: $r'(N = 50000)$ =

$r'(N = 10^6)$ =

$r'(N = 10^{12})$ =

e) Welche Werte liefert die Redundanznäherung $r'(N)$ für die nicht näher spezifizierte Binärquelle BQ3? Insbesondere:

BQ3: $r'(N = 50000)$ =

$r'(N = 10^6)$ =

$r'(N = 10^{12})$ =

f) Welche Quellenentropie H könnte BQ3 nach diesem Ergebnis besitzen? *Hinweis:* Es ist genau eine Antwort richtig.

- $H = 1.00$ bit/Quellensymbol,
- $H = 0.75$ bit/Quellensymbol,
- $H = 0.50$ bit/Quellensymbol,
- $H = 0.25$ bit/Quellensymbol.

Z2.5: LZW–Komprimierung

N	$L(N)$	$L(N)$
1000	772	1345
2000	1488	2600
5000	3516	6329
10000	6800	12330
20000	13263	24242
50000	32100	59595
	BQ1	BQ2

Wir betrachten wie in **Aufgabe A2.5** die Datenkomprimierung mit dem 1983 veröffentlichten **Lempel–Ziv–Welch–Algorithmus.** Dabei gilt:

- Die Eingangsfolge habe die Länge N.
- Die Länge der LZW–Coderausgabe ist L.

Die Grafik zeigt für zwei verschiedene binäre Nachrichtenquellen BQ1 und BQ2 den Zusammenhang zwischen den Folgenlängen N und L, dargestellt durch den Funktionsverlauf $L(N)$. BQ1 und BQ2 besitzen die gleichen statistischen Eigenschaften wie in Aufgabe A2.5:

- **BQ1** ist aufgrund von ungleichen Symbolwahrscheinlichkeiten (p_A = 0.89, p_B = 0.11) redundant. Es bestehen keine Bindungen zwischen den einzelnen Symbolen. Die Entropie ist H = 0.5 bit/Quellensymbol.
- **BQ2** ist redundanzfrei und weist die Entropie H = 1 bit/Quellensymbol auf.

Weiter benötigen Sie für die Lösung dieser Aufgabe noch zwei Definitionen:

- Der *Komprimierungsfaktor* ist definitionsgemäß $K(N) = L(N)/N$.
- Die relevante Redundanz der LZW–Coderfolge (im Folgenden *Restredundanz* genannt) ist

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

Hinweis: Die Aufgabe bezieht sich auf das **Kapitel 2.2.**

Fragebogen zu "Z2.5: LZW–Komprimierung"

a) Welche Komprimierungfaktoren K ergeben sich jeweils mit $N = 10000$?

$N = 10000$, BQ1: K =

BQ2: K =

b) Wie groß ist die zugehörige Restredundanz (in Prozent)?

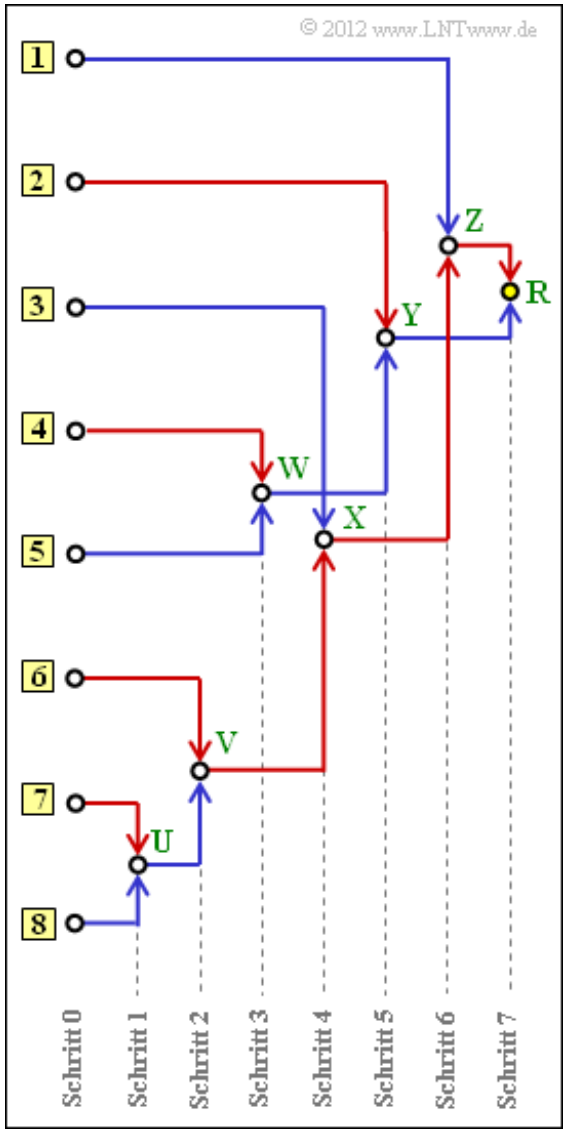
$N = 10000$, BQ1: r = %

BQ2: r = %

c) Welche Aussagen liefert der Vergleich von $N = 10000$ und $N = 50000$?

- ☐ Bei beiden Quellen ist $K(N = 50000)$ kleiner als $K(N = 10000)$.
- ☐ Bei beiden Quellen ist $r(N = 50000)$ kleiner als $r(N = 10000)$.
- ☐ Nur bei BQ1 ergeben sich mit $N = 50000$ günstigere Werte.

A2.6: Huffman–Codierung



Wir betrachten hier eine Quellensymbolfolge $\langle q_\nu \rangle$ mit dem Symbolumfang $M = 8$:

$$q_\nu = \{q_\mu\} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{H}\}.$$

Sind die Symbole gleichwahrscheinlich, also wenn gilt

$$p_{\rm A} = p_{\rm B} = \ldots = p_{\rm H} = 1/M,$$

so macht Quellencodierung keinen Sinn. Bereits mit dem Dualcode **A** → **000**, **B** → **001**, ... , **H** → **111**, erreicht die mittlere Codewortlänge $L_{\rm M}$ ihre untere Schranke H gemäß dem **Quellencodierungstheorem:**

$$L_{\rm M,\,min} = H = 3\ \text{bit/Quellensymbol}.$$

H bezeichnet hierbei die *Quellenentropie*.

Die Symbolwahrscheinlichkeiten seien aber in dieser Aufgabe wie folgt gegeben:

$$p_{\rm A} = 0.04,\ p_{\rm B} = 0.08,\ p_{\rm C} = 0.14,\ p_{\rm D} = 0.25,$$
$$p_{\rm E} = 0.24,\ p_{\rm F} = 0.12,\ p_{\rm G} = 0.10,\ p_{\rm H} = 0.03.$$

Es liegt hier also eine redundante Nachrichtenquelle vor, die man durch **Huffman–Codierung** komprimieren kann. Der Algorithmus wurde 1952 – also kurz nach Shannons bahnbrechenden Arbeiten zur Informationstheorie – von **David Albert Huffman** veröffentlicht und erlaubt die Konstruktion von optimalen präfixfreien Codes.

Der Algorithmus soll hier ohne Herleitung und Beweis angegeben werden, wobei wir uns auf Binärcodes beschränken (die Codesymbolfolge besteht nur aus Nullen und Einsen):

1. Man ordne die Symbole nach fallenden Auftrittswahrscheinlichkeiten.
2. Man fasse die zwei unwahrscheinlichsten Symbole zu einem neuen Symbol zusammen.
3. Man wiederhole Schritt 1 und 2, bis nur zwei (zusammengefasste) Symbole übrig bleiben.
4. Die wahrscheinlichere Symbolmenge wird mit **1** binär codiert, die andere Menge mit **0**.
5. Man ergänze schrittweise (von unten nach oben) die aufgespaltenen Teilcodes mit **1** bzw. **0**.

Oft wird dieser Algorithmus durch ein **Baumdiagramm** veranschaulicht. Die obige Grafik zeigt dieses für den vorliegenden Fall. Sie haben folgende Aufgaben:

- (a): Zuordnung der Symbole **A**, ... , **H** zu den mit [1], ... , [8] bezeichneten Eingängen.
- (b): Bestimmung der Summenwahrscheinlichkeiten U, ... , Z sowie R (*Root*).
- (c) Zuordnung der Symbole **A**, ... , **H** zu den entsprechenden Huffman–Binärfolgen; eine rote Verbindung im Baumdiagramm entspricht einer **1** und eine blaue Verbindung einer **0**.

Sie werden feststellen, dass die mittlere Codewortlänge

$$L_{\rm M} = \sum_{\mu=1}^{M} p_\mu \cdot L_\mu$$

bei Huffman–Codierung nur unwesentlich größer ist als die Quellenentropie H. In dieser Gleichung gelten für den vorliegenden Fall folgende Werte:

- $M = 8$ sowie $p_1 = p_{\rm A}$, ... , $p_8 = p_{\rm H}$,
- L_1, ... , L_8: Jeweilige Bitanzahl der Codesymbole für **A**, ... , **H**.

Hinweis: Die Aufgabe bezieht sich auf das Themengebiet von **Kapitel 2.3.**

Fragebogen zu "A2.6: Huffman–Codierung"

a) Welche Eingänge im Baumdiagramm stehen für

Symbol A: Eingang =

Symbol B: Eingang =

Symbol C: Eingang =

Symbol D: Eingang =

b) Welche Zahlenwerte sollten bei den Knoten im Baumdiagramm stehen?

Knoten U =

Knoten V =

Knoten W =

Knoten Z =

Root R =

c) Welche Binärcodes (darzustellen mit Nullen und Einsen) ergeben sich für

Symbol A: Binärcode =

Symbol B: Binärcode =

Symbol C: Binärcode =

Symbol D: Binärcode =

d) Wie groß ist die mittlere Codewortlänge?

L_M = bit/Quellensymbol

e) Wie groß ist die Quellenentropie H? *Hinweis:* Es gibt genau eine Lösung.

- H = 2.71 bit/Quellensymbol.
- H = 2.75 bit/Quellensymbol.
- H = 3.00 bit/Quellensymbol.

Z2.6: Nochmals zum Huffman–Code

Symbol	Code 1	Code 2	Code 3
A	11	1	11
B	10	10	10
C	01	01	010
D	001	001	001
E	000	000	000

Der Algorithmus von David A. Huffman realisiert eine Entropiecodierung mit folgenden Eigenschaften:

- Der entstehende Binärcode ist **präfixfrei** und somit in einfacher Weise (und sofort) decodierbar.
- Der Code führt bei einer gedächtnislosen Quelle zur kleinstmöglichen mittleren Codewortlänge L_M.
- L_M ist aber nie kleiner als die Quellenentropie H. Diese beiden Größen sind allein aus den M Symbolwahrscheinlichkeiten berechenbar.

Vorausgesetzt wird für diese Aufgabe eine gedächtnislose Quelle mit dem Symbolumfang $M = 5$ und dem Alphabet {**A**, **B**, **C**, **D**, **E**}. In obiger Grafik sind drei Codes vorgegeben. Sie sollen entscheiden, welche dieser Codes durch Anwendung des Huffman–Algorithmus entstanden sind (oder sein könnten).

Hinweis: Die Aufgabe gehört zum **Kapitel 2.3.** Weitere Informationen zum Huffman–Algorithmus finden Sie auch im Angabenblatt zur **Aufgabe A2.6.** Zur Kontrolle Ihrer Ergebnisse verweisen wir auf das Interaktionsmodul **Shannon–Fano– und Huffman–Codierung.**

Fragebogen zu "Z2.6: Nochmals zum Huffman–Code"

a) Welche Codes liefert Huffman für $p_A = p_B = p_C = 0.3$, $p_D = p_E = 0.05$?

- ☐ Code 1,
- ☐ Code 2,
- ☐ Code 3.

b) Wie stehen mittlere Codewortlänge L_M und Entropie H in Relation?

- ○ $L_M < H$,
- ○ $L_M = H$,
- ○ $L_M > H$.

c) Mit welchen Symbolwahrscheinlichkeiten würde hier $L_M = H$ gelten?

$$\boldsymbol{p_A} =$$
$$\boldsymbol{p_B} =$$
$$\boldsymbol{p_C} =$$
$$\boldsymbol{p_D} =$$
$$\boldsymbol{p_E} =$$

d) Die Angaben zu (c) gelten weiter. Die mittlere Codewortlänge wird aber nun für eine Folge der Länge $N = 40$ ermittelt $\Rightarrow$ L_M'. Was ist möglich?

- ☐ $L_M' < L_M$,
- ☐ $L_M' = L_M$,
- ☐ $L_M' > L_M$.

e) Welcher Code könnte überhaupt ein Huffman–Code sein?

- ☐ Code 1,
- ☐ Code 2,
- ☐ Code 3.

A2.7: Zweiertupel – Huffman

Symbol	Code 1	Code 2	Code 3
A = XX	11	1	0
B = XY	10	01	10
C = YX	01	001	110
D = YY	00	000	1110

Die Anwendung des **Huffman–Algorithmus** in seiner ursprünglichen Form setzt einen Symbolumfang $M > 2$ voraus und ist deshalb zur Datenkomprimierung von Binärquellen unbrauchbar.

Fasst man aber mehrere aufeinanderfolgende Binärzeichen der Nachrichtenquelle zu einem neuen Symbol zusammen, so kann man auf die neue Symbolmenge die Huffman–Datenkomprimierung sinnvoll anwenden.

Wir gehen in dieser Aufgabe von der Symbolmenge $\{\mathbf{X}, \mathbf{Y}\}$ $\Rightarrow$ $M = 2$ aus und bilden gemäß der obigen Tabelle Zweiertupel mit dem Symbolvorrat $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ $\Rightarrow$ $M' = M^2 = 4$. Aus der binären Quellensymbolfolge **XYXXYXXXYY** wird somit die quaternäre Folge **BACAD**.

Desweiteren sind in obiger Tabelle drei Codes angegeben, von denen manche durch den Huffman–Algorithmus entstanden sind. Die binären Ausgangsfolgen ergeben sich dann für unser Beispiel wie folgt:

- Code 1: **1011011100**,
- Code 2: **0110011000**,
- Code 3: **10011001110**.

Nochmals zum Verständnis: Aus der ursprünglichen Symbolmenge $\{\mathbf{X}, \mathbf{Y}\}$ erhält man durch die Bildung von Zweiertupeln eine Quaternärmenge mit dem Symbolvorrat $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$. Die Folgenlänge N wird dadurch halbiert. Durch Huffman–Codierung ergibt sich wieder eine Binärfolge, deren Symbolmenge zur besseren Unterscheidung mit $\{\mathbf{0}, \mathbf{1}\}$ bezeichnet wird. Die Anwendung der Huffman–Codierung macht genau dann Sinn, wenn die Länge der Ausgangsfolge (im statistischen Mittel) kleiner ist als N.

Mit dieser Aufgabe soll geklärt werden, welche der vorgegebenen Binärcodes bei welchen Randbedingungen sinnvoll sind. Die binäre Nachrichtenquelle $\{\mathbf{X}, \mathbf{Y}\}$ sei gedächtnislos und wird allein durch die Symbolwahrscheinlichkeiten p_X beschrieben. Die zweite Wahrscheinlichkeit ist dann stets $p_Y = 1 - p_X$.

Hinweis: Die Aufgabe bezieht sich auf die **letzte Theorieseite** von **Kapitel 2.3.** Die Idee zu dieser Aufgabe entstand bei einem Vortrag von Prof. **Robert Fischer** von der Universität Ulm zum Thema „Der goldene Schnitt in der Nachrichtentechnik".

Für die mittlere Codewortlänge pro Zweiertupel gilt:

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

Bezogen auf ein Quellensymbol ergibt sich $L_{\rm M} = L'_{\rm M}/2$.

Fragebogen zu "A2.7: Zweiertupel – Huffman"

a) Geben Sie die Codewortlängen bei redundanzfreier Binärquelle an.

Code 1: L_M = bit/Quellensymbol

Code 2: L_M = bit/Quellensymbol

Code 3: L_M = bit/Quellensymbol

b) Ermitteln Sie den Huffman–Code hinsichtlich Zweiertupel für $p_X = 0.6$.

- ☐ Es ergibt sich Code 1.
- ☐ Es ergibt sich Code 2.
- ☐ Es ergibt sich Code 3.

c) Wie groß ist die zugehörige mittlere Codewortlänge?

$p_X = 0.6$: L_M = bit/Quellensymbol

d) Ermitteln Sie den Huffman–Code hinsichtlich Zweiertupel für $p_X = 0.8$.

- ☐ Es ergibt sich Code 1.
- ☐ Es ergibt sich Code 2.
- ☐ Es ergibt sich Code 3.

e) Wie groß ist die zugehörige mittlere Codewortlänge?

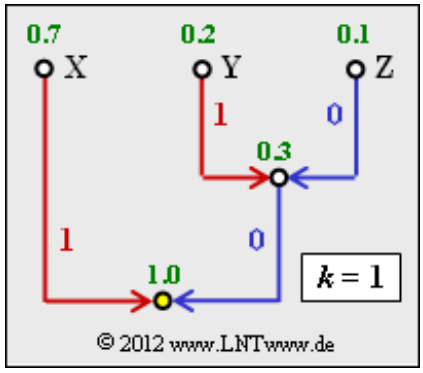
$p_X = 0.8$: L_M = bit/Quellensymbol

f) In welchem Bereich darf die Wahrscheinlichkeit p_X für das Symbol **X** liegen, damit sich Code 1 als Huffman–Code ergibt?

$p_{X,\,max}$ =

$p_{X,\,min}$ =

Z2.7: Ternärquelle –Zweiertupel



Wir betrachten den gleichen Sachverhalt wie in der **Aufgabe A2.7:** Der Huffman–Algorithmus führt zu einem besseren Ergebnis, das heißt zu einer kleineren mittleren Codewortlänge $L_{\rm M}$, wenn man ihn nicht auf einzelne Symbole anwendet, sondern vorher k–Tupel bildet. Dadurch erhöht man den Symbolumfang von M auf $M' = M^k$.

Für die hier betrachtete Nachrichtenquelle gilt:

- Symbolumfang: $M = 3$,
- Symbolvorrat: {**X**, **Y**, **Z**},
- Wahrscheinlichkeiten: $p_{\rm X} = 0.7$, $p_{\rm Y} = 0.2$, $p_{\rm Z} = 0.1$,
- Entropie: $H = 1.157$ bit/Ternärsymbol.

Die Grafik zeigt den Huffman–Baum, wenn man den Huffman–Algorithmus auf Einzelsymbole anwendet, also den Fall $k = 1$. In der Teilaufgabe (b) sollen Sie den entsprechenden Huffman–Code angeben, wenn vorher Zweiertupel gebildet werden ($k = 2$).

Hinweis: Die Aufgabe bezieht sich auf die **letzte Theorieseite** von **Kapitel 2.3.** Bezeichnen Sie die möglichen Zweiertupel mit

XX = A, **XY = B**, **XZ = C**, **YX = D**, **YY = E**, **YZ = F**, **ZX = G**, **ZY = H**, **ZZ = I** .

Fragebogen zu "Z2.7: Ternärquelle –Zweiertupel"

a) Wie groß ist die mittlere Codewortlänge, wenn der Huffman–Algorithmus direkt auf die ternären Quellensymbole **X**, **Y** und **Z** angewendet wird?

$$k = 1: \quad L_M = \quad \text{bit/Quellensymbol}$$

b) Wie groß sind die Tupel–Wahrscheinlichkeiten? Insbesondere:

$$p_A = \text{Pr(XX)} =$$

$$p_B = \text{Pr(XY)} =$$

$$p_C = \text{Pr (XZ)} =$$

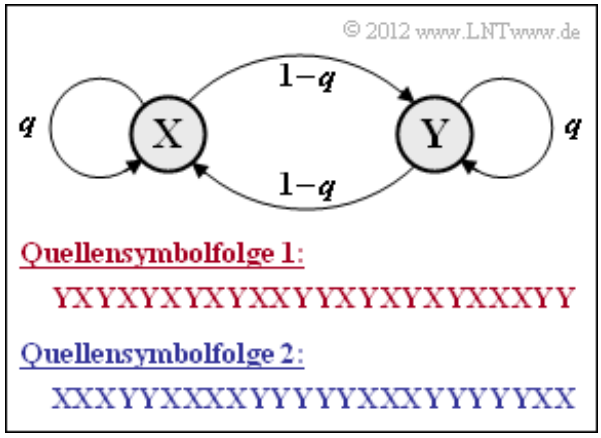
c) Wie groß ist die mittlere Codewortlänge, wenn man erst Zweiertupel bildet und darauf den Huffman–Algorithmus anwendet.

$$k = 2: \quad L_M = \quad \text{bit/Quellensymbol}$$

d) Welche der folgenden Aussagen sind zutreffend, wenn man mehr als zwei Ternärzeichen zusammenfasst ($k > 2$)?

- ☐ L_M fällt monoton mit steigendem k ab.
- ☐ L_M ändert sich nicht, wenn man k erhöht.
- ☐ Für $k = 3$ erhält man $L_M = 1.05$ bit/Quellensymbol.

A2.8: Markovquelle und Huffman



Wir betrachten hier die binäre symmetrische Markovquelle entsprechend nebenstehender Grafik, die durch den einzigen Parameter

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

vollständig beschrieben wird. Die angegebenen Quellensymbolfolgen gelten für $q = 0.2$ bzw. $q = 0.8$. In der Teilaufgabe (a) ist zu klären, welche Symbolfolge mit $q = 0.2$ und welche mit $q = 0.8$ generiert wurde.

Die Eigenschaften von Markovquellen werden im **Kapitel 1.2** ausführlich beschrieben. Aufgrund der hier vorausgesetzten Symmetrie bezüglich der binären Symbole **X** und **Y** ergeben sich einige gravierende Vereinfachungen, wie in **Aufgabe Z1.5** hergeleitet wird:

- Die Symbole **X** und **Y** sind gleichwahrscheinlich, das heißt. es ist $p_X = p_Y = 0.5$. Damit lautet die **erste Entropienäherung:**

  $$H_1 = 1 \text{ bit/Quellensymbol}.$$

- Die **Entropie** der Markovquelle ergibt sich zu

  $$H = q \cdot \text{ld}\,\frac{1}{q} + (1-q) \cdot \text{ld}\,\frac{1}{1-q} = 0.722 \text{ bit/Quellensymbol}.$$

  Der Zahlenwert gilt nur für $q = 0.2$ sowie für $q = 0.8$.

- Bei Markovquellen sind alle **Entropienäherungen höherer Ordnung** durch H_1 und H bestimmt. Die folgenden Zahlenwerte gelten wieder für $q = 0.2$ und $q = 0.8$ gleichermaßen:

  $$H_2 = \frac{1}{2} \cdot \big[H_1 + H\big] = 0.861 \text{ bit/Quellensymbol},$$
  $$H_3 = \frac{1}{3} \cdot \big[H_1 + 2H\big] = 0.815 \text{ bit/Quellensymbol}.$$

Wie auf der **letzten Theorieseite** dieses Kapitels 2.3 soll hier der Huffman–Algorithmus auf k–Tupel angewandt werden, wobei wir uns auf $k = 2$ und $k = 3$ beschränken.

Hinweis: Die Aufgabe gehört zum Themengebiet von **Kapitel 2.3.** Nützliche Informationen finden Sie auch in **Aufgabe A2.7** und **Aufgabe Z2.7.** Für die Huffman–Codierung können Sie das folgende Interaktionsmodul benutzen:

Shannon–Fano– und Huffman–Codierung

Fragebogen zu "A2.8: Markovquelle und Huffman"

a) Welche der vorne angegebenen Beispielfolgen gilt für $q = 0.8$?

- ○ Quellensymbolfolge 1,
- ○ Quellensymbolfolge 2.

b) Welche der folgenden Aussagen treffen zu?

- ☐ Auch die direkte Anwendung von Huffman ist hier sinnvoll.
- ☐ Huffman macht bei Bildung von Zweiertupeln ($k = 2$) Sinn.
- ☐ Huffman macht bei Bildung von Dreiertupeln ($k = 3$) Sinn.

c) Wie lauten die Wahrscheinlichkeiten der Zweiertupel ($k = 2$) für $q = 0.8$?

$$\boldsymbol{q = 0.8;\ k = 2:}\quad \boldsymbol{p_A = \Pr(XX)} =$$

$$\boldsymbol{p_B = \Pr(XY)} =$$

$$\boldsymbol{p_C = \Pr(YX)} =$$

$$\boldsymbol{p_D = \Pr(YY)} =$$

d) Ermitteln Sie mit dem angegebenen Flash–Modul den Huffman–Code für $k = 2$. Wie groß ist in diesem Fall die mittlere Codewortlänge?

$$\boldsymbol{L_M} = \quad \text{bit/Quellensymbol}$$

e) Welche Schranke ergibt sich für die mittlere Codewortlänge, wenn Zweiertupel gebildet werden ($k = 2$)? Interpretation.

- ☐ $L_M \geq H_1 = 1$ bit/Quellensymbol,
- ☐ $L_M \geq H_2 \approx 0.861$ bit/Quellensymbol,
- ☐ $L_M \geq H_3 \approx 0.815$ bit/Quellensymbol,
- ☐ $L_M \geq H \approx 0.722$ bit/Quellensymbol,
- ☐ $L_M \geq 0.5$ bit/Quellensymbol.

f) Berechnen Sie die Wahrscheinlichkeiten für $k = 3$.

$$\boldsymbol{q = 0.8;\ k = 3:}\quad \boldsymbol{p_A = \Pr(XXX)} =$$

$$\boldsymbol{p_B = \Pr(XXY)} =$$

$$\boldsymbol{p_C = \Pr(XYX)} =$$

$$\boldsymbol{p_D = \Pr(XYY)} =$$

$$p_E = \mathrm{Pr(YXX)} =$$

$$p_F = \mathrm{Pr(YXY)} =$$

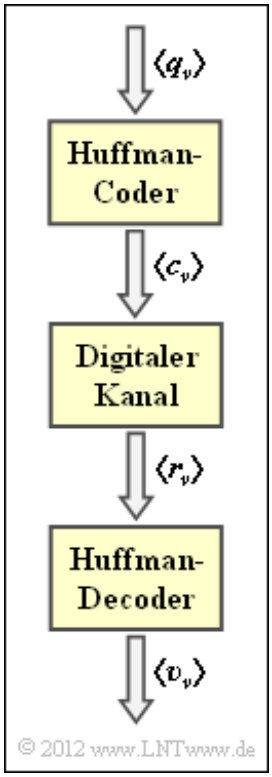
$$p_G = \mathrm{Pr(YYX)} =$$

$$p_H = \mathrm{Pr(YYY)} =$$

g) Ermitteln Sie mit dem genannten Flash–Modul den Huffman–Code für $k = 3$. Wie groß ist in diesem Fall die mittlere Codewortlänge?

$$L_M = \quad \text{bit/Quellensymbol}$$

A2.9: Huffman–Decodierung nach Fehlern



Wir betrachten die **Huffman–Codierung** gemäß folgender Zuordnung:

A → 1, **B → 01**, **C → 001**, **D → 000**.

Die Codierung nach Huffman ist stets *verlustlos*. Das bedeutet: Decodiert man die Codesymbolfolge $\langle c_ν \rangle$ nach dem Huffman–Codierer sofort wieder, so ist das Decodierergebnis $\langle v_ν \rangle$ gleich der Quellensymbolfolge $\langle q_ν \rangle$.

Stimmt dagegen die Empfangsfolge $\langle r_ν \rangle$ aufgrund von Fehlern bei der Übertragung (**0 → 1**, **1 → 0**) mit der erzeugten Codefolge $\langle c_ν \rangle$ nicht überein, so kann es zu einer Fehlerfortpflanzung kommen. Ein einziger Bitfehler kann dann dazu führen, dass (nahezu) alle nachfolgenden Zeichen falsch decodiert werden.

Hinweis: Die Aufgabe bezieht sich auf die **Seite 5** von **Kapitel 2.3.**

Fragebogen zu "A2.9: Huffman–Decodierung nach Fehlern"

a) Wir betrachten die Codesymbolfolge **10100100011000010011**. Wie lautet die dazugehörige Quellensymbolfolge?

- ☐ **CCDAADBCA**,
- ☐ **ABDDAADBCA**,
- ☐ **ABCDAADBCA**,
- ☐ Anders als die drei genannten.

b) Welche Folge ergibt sich nach der Decodierung, wenn das erste Bit verfälscht wird (**1** → **0**)? ⇒ Anliegende Folge **00100100011000010011**.

- ☐ **CCDAADBCA**,
- ☐ **ABDDAADBCA**,
- ☐ **ABCDAADBCA**,
- ☐ Anders als die drei genannten.

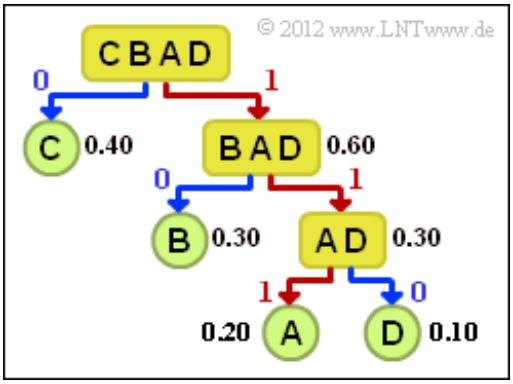
c) Ist es möglich, dass durch einen weiteren Bitfehler die späteren Symbole alle wieder richtig decodiert werden?

- ☐ Ja, durch einen zweiten Bitfehler an Position 2.
- ☐ Ja, durch einen zweiten Bitfehler an Position 10.
- ☐ Ja, durch einen zweiten Bitfehler an Position 15.
- ☐ Nein.

d) Welche Folge ergibt sich nach der Decodierung, wenn das sechste Bit verfälscht wird (**1** → **0**)?

- ☐ **CCDAADBCA**,
- ☐ **ABDDAADBCA**,
- ☐ **ABCDAADBCA**,
- ☐ Anders als die drei genannten.

A2.10: Shannon–Fano–Codierung



Ein weiterer Algorithmus zur Entropiecodierung wurde 1949 von **Claude Elwood Shannon** und **Robert Fano** angegeben, der im **Theorieteil** beschrieben ist.

Diese spezielle Art von Quellencodierung soll hier an einem einfachen Beispiel für den Symbolumfang $M = 4$ und folgende Symbolwahrscheinlichkeiten beschrieben werden:

$$p_{\rm A} = 0.2,\ \ p_{\rm B} = 0.3,\ \ p_{\rm C} = 0.4,\ \ p_{\rm D} = 0.1.$$

Die obige Grafik zeigt das dazugehörige Baumdiagramm. Man geht folgendermaßen vor:

1. Man ordnet die Symbole nach fallender Auftrittswahrscheinlichkeit, hier **C** – **B** – **A** – **D**.
2. Man teilt die Symbole in zwei etwa gleichwahrscheinliche Gruppen ein, hier **C** und **BAD**.
3. Der unwahrscheinlicheren Gruppe wird das Binärsymbol **0**, der anderen die **1** zugeordnet.
4. Sind in einer Gruppe mehr als ein Zeichen, so ist der Algorithmus rekursiv anzuwenden.

Für dieses Beispiel ergibt sich die folgende Codezuordnung (in obigem Baumdiagramm markiert eine rote Verbindung eine **1** und eine blaue eine **0**:

A → 111, **B → 10**, **C → 0**, **D → 110**.

Damit ergibt sich für die mittlere Codewortlänge:

$$L_{\rm M} = 0.4 \cdot 1 + 0.3 \cdot 2 + (0.2 + 0.1) \cdot 3 = 1.9\ \text{bit/Quellensymbol}.$$

Der **Huffman–Algorithmus** würde hier zwar einen geringfügig anderen Code erzeugen, aber auch bei diesem würde **C** mit einem Bit, **B** mit zwei Bit und **A** und **D** mit jeweils drei Bit codiert. Damit ergäbe sich ebenfalls $L_{\rm M} = 1.9$ bit/Quellensymbol.

In dieser Aufgabe sollen Sie den Shannon–Fano–Code für $M = 8$ und die Wahrscheinlichkeiten

$$p_{\rm A} = 0.10,\ \ p_{\rm B} = 0.40,\ \ p_{\rm C} = 0.02,\ \ p_{\rm D} = 0.14,$$
$$p_{\rm E} = 0.17,\ \ p_{\rm F} = 0.03,\ \ p_{\rm G} = 0.05,\ \ p_{\rm H} = 0.09$$

ermitteln. Sie werden erkennen, dass sich mit diesen Wahrscheinlichkeiten „Shannon–Fano" auch hinsichtlich Effizienz von „Huffman" unterscheiden wird. *Hinweis*: Beim Huffman–Code ergibt sich mit den vorliegenden Wahrscheinlichkeiten die folgende Zuordnung:

A → 100, **B → 0**, **C → 111100**, **D → 101**, **E → 110**, **F → 111101**, **G → 11111**, **H → 1110**.

Hinweis: Die Aufgabe bezieht sich auf die **Seite 1** von **Kapitel 2.4.** Zur Kontrolle können Sie das folgende Interaktionsmodul verwenden:

Shannon–Fano– und Huffman–Codierung

Fragebogen zu "A2.10: Shannon–Fano–Codierung"

a) Wie groß ist die mittlere Codewortlänge beim Huffman–Code?

Huffman: $L_{\rm M}$ = bit/Quellensymbol

b) Was geschieht im ersten Schritt der Shannon–Fano–Codierung? Alle anderen Symbole werden in der zweiten Gruppe zusammengefasst.

- ☐ Man fasst **A** und **B** zur ersten Gruppe zusammen.
- ☐ Man fasst **B** und **E** zur ersten Gruppe zusammen.
- ☐ Die erste Gruppe besteht nur aus dem Symbol **B**.

c) Welche Zuordnungen ergeben sich für den Shannon–Fano–Algorithmus?

- ☐ Das Zeichen **A** wird binär mit **010** codiert.
- ☐ Das Zeichen **B** wird binär mit **11** codiert.
- ☐ Das Zeichen **C** wird binär mit **00110** codiert.

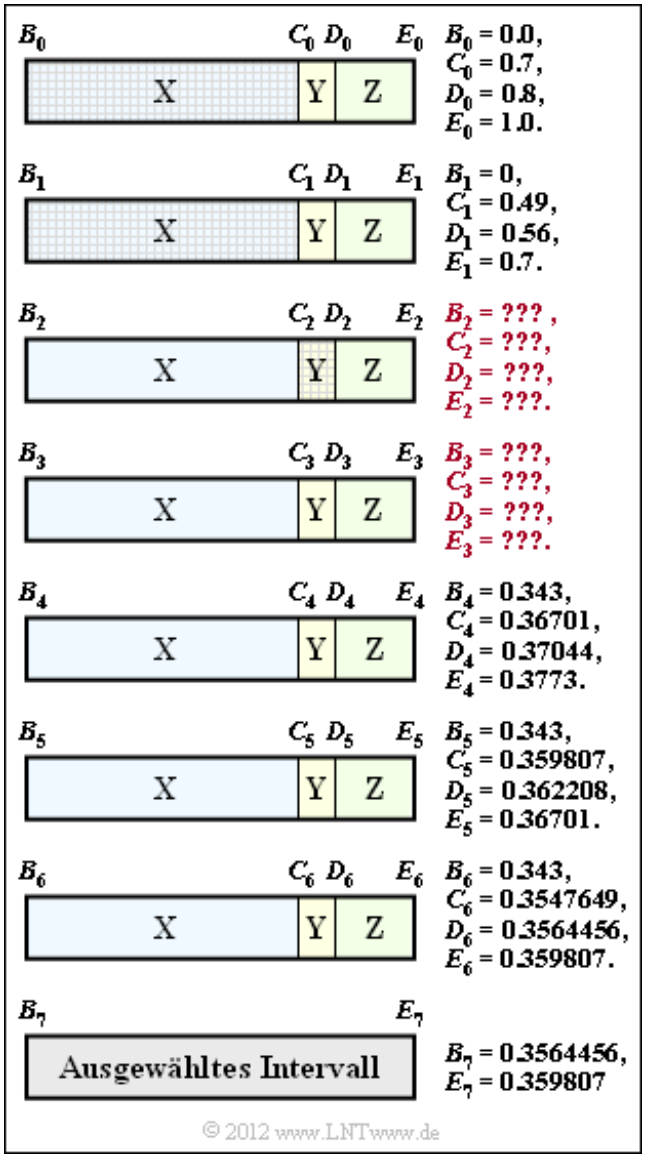
d) Wie groß ist die mittlere Codewortlänge beim Shannon–Fano–Code?

Shannon–Fano: $L_{\rm M}$ = bit/Quellensymbol

e) Welche Aussagen gelten für beliebige Wahrscheinlichkeiten?

- ☐ $L_{\rm M}$ könnte bei Shannon–Fano kleiner sein als bei Huffman.
- ☐ $L_{\rm M}$ könnte bei Shannon–Fano größer sein als bei Huffman.
- ☐ $L_{\rm M}$ könnte bei Shannon–Fano und Huffman gleich groß sein.

A2.11: Arithmetische Codierung



Die arithmetische Codierung ist eine spezielle Form der Entropiecodierung: Auch hier müssen die Symbolwahrscheinlichkeiten bekannt sein. In dieser Aufgabe gehen wir von $M = 3$ Symbolen aus, die wir mit **X**, **Y**, **Z** benennen.

Im Gegensatz zur **Huffman–Codierung** wird bei der Arithmetischen Codierung (AC) eine Symbolfolge der Länge N gemeinsam codiert. Das Codierergebnis ist ein reeller Zahlenwert r aus dem Intervall

$$I = [B, E) = [B, B + \Delta) .$$

Diese Notation bedeutet:

- Der Beginn B gehört zum Intervall I.
- Das Ende E ist nicht mehr in I enthalten.
- Die Intervallbreite ist $\Delta = E - B$.

Von den unendlich vielen möglichen Werten $r \in I$ (da r reellwertig ist, also kein Integer) wird derjenige Zahlenwert ausgewählt, der mit der geringsten Bitanzahl auskommt. Hierzu zwei Beispiele zur Verdeutlichung:

- Der Dezimalwert $r = 3/4$ lässt sich mit zwei Bit darstellen:

$$r = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 0.75 \quad \Rightarrow \quad \text{binär} : 0.11 \quad \Rightarrow \quad \text{Code} : \mathbf{11},$$

- Der Dezimalwert $r = 1/3$ benötigt dagegen unendlich viele Bit:

$$r = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + \ldots$$
$$\Rightarrow \quad \text{binär} : 0.011101 \quad \Rightarrow \quad \text{Code} : \mathbf{011101}.$$

In dieser Aufgabe beschränken wir uns auf die Bestimmung des aktuellen Intervalls I, gekennzeichnet durch den Beginn B sowie dem Ende E bzw. der Breite Δ. Diese Bestimmung geschieht entsprechend der Intervallschachtelung in obiger Grafik. An der Schraffierung ist zu erkennen, dass die Folge mit den Ternärsymbolen **XXY** beginnt.

Der Algorithmus funktioniert wie folgt:

- Vor Beginn (quasi beim Symbol 0) wird der gesamte Wahrscheinlichkeitsbereich nach den Wahrscheinlichkeiten p_X, p_Y und p_Z in drei Bereiche unterteilt. Die Grenzen liegen bei

$$B_0 = 0 , \quad C_0 = p_X , \quad D_0 = p_X + p_Y , \quad E_0 = p_X + p_Y + p_Z = 1 .$$

- Das erste Symbol ist **X** $\Rightarrow$ das ausgewählte Intervall wird durch B_0 und C_0 begrenzt. Dieses Intervall wird mit neuem Beginn $B_1 = B_0$ und neuem Ende $E_1 = C_0$ in gleicher Weise aufgeteilt wie

der Gesamtbereich im Schritt 0. Die Zwischenwerte sind C_1 und D_1.

- Die weitere Intervall–Aufteilung ist Ihre Aufgabe. Beispielsweise sollen in der Teilaufgabe (b) die Grenzen B_2, C_2, D_2, E_2 für das zweite Symbol **X** und in der Teilaufgabe (c) die Grenzen B_3, C_3, D_3, E_3 für das dritte Symbol **Y** ermittelt werden.

Hinweis: Die Aufgabe bezieht sich auf die **Seite 2a** und die **Seite 2b** in **Kapitel 2.4.** Die Binärdarstellung des ausgewählten Intervalls wird in **Aufgabe A2.12** behandelt.

Fragebogen zu "A2.11: Arithmetische Codierung"

a) Welche Wahrscheinlichkeiten sind der Grafik zugrundegelegt?

p_X =

p_Y =

p_Z =

b) Wie lauten die Bereichsgrenzen nach der Codierung des zweiten Symbols?

2. Symbol ist „X": B_2 =

C_2 =

D_2 =

E_2 =

c) Wie lauten die Bereichsgrenzen nach der Codierung des dritten Symbols?

3. Symbol ist „Y": B_3 =

C_3 =

D_3 =

E_3 =

d) Nach der Codierung des vierten Symbols ist $B_4 = 0.343$. Was folgt daraus?

- ☐ Das vierte Symbol war **X**.
- ☐ Das vierte Symbol war **Y**.
- ☐ Das vierte Symbol war **Z**.

e) Nach weiteren Symbolen wird das Ergebnisintervall durch $B_7 = 0.3564456$ und $E_7 = 0.359807$ begrenzt. Welche Aussagen treffen zu?

- ☐ Die zur Codierung anstehende Symbolfolge lautet **XXYXXZX**.
- ☐ Die zur Codierung anstehende Symbolfolge lautet **XXYXXXZ**.
- ☐ Die Breite des resultierenden Intervalls ist $\Delta = p_X^5 \cdot p_Y \cdot p_Z$.

f) Welche reellen Zahlen (in Binärform) fallen in das ausgewählte Intervall?

- ☐ $r_1 = (0.101100)_{\text{binär}}$
- ☐ $r_2 = (0.010111)_{\text{binär}}$

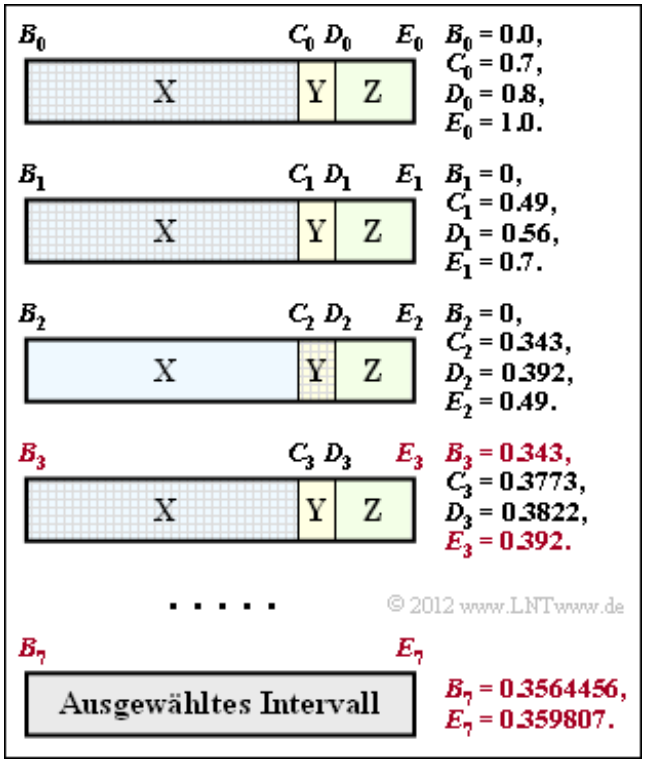
- [ ] $r_3 = (0.001011)_{\text{binär}}$

A2.12: Nochmals AC

Wir betrachten hier die arithmetische Codierung (AC). Alle notwendigen Informationen zu dieser Art von Entropiecodierung finden Sie in der **Aufgabe A2.11.**

Auch nebenstehende Grafik ist das Ergebnis der letzten Aufgabe. Die für die aktuelle Aufgabe wichtigen Zahlenwerte für die Codierschritte 3 und 7 sind farblich hervorgehoben:

- Das Intervall für $N = 3$ (Symbolfolge **XXY**) beginnt bei $B_3 = 0.343$ und reicht bis zum Endwert $E_3 = 0.392$.
- Das Intervall für $N = 7$ (Symbolfolge **XXYXXXZ**) beginnt bei $B_7 = 0.3564456$ und endet bei $E_7 = 0.359807$.



In dieser Aufgabe geht es nur um die Zuweisung von Binärfolgen zu den ausgewählten Intervallen. Man geht wie folgt vor:

- Das Intervall I wird bestimmt durch den Beginn B, das Ende E, die Intervallbreite $\Delta = E - B$ sowie die Intervallmitte $M = (B + E)/2$.
- Das Intervall I wird gekennzeichnet durch die Binärdarstellung (mit begrenzter Auflösung) eines beliebigen reellen Zahlenwertes $r \in I$. Beispielsweise wählt man $r \approx M$.
- Die erforderliche Bitanzahl ergibt sich aus der Intervallbreite nach folgender Gleichung (die nach unten offenen Klammern bedeuten „nach oben runden”):

$$N_{\rm Bit} = \lceil {\rm ld}\ 1/\Delta \rceil + 1\,.$$

- Beispielsweise steht für $N_{\rm Bit} = 5$ der Binärcode **01001** für die folgende reellwertige Zahl r:

$$r = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} = 0.28125\,.$$

Hinweis: Die Aufgabe gehört zum Themengebiet von **Kapitel 2.4.** Allerdings wird in *LNTwww* die arithmetische Codierung nur sehr knapp behandelt. Eine Übersicht finden Sie auch in **WIKIPEDIA** und eine ausführlichere Abhandlung in **[BCK02].**

Fragebogen zu "A2.12: Nochmals AC"

a) Wie viele Bit werden zur Darstellung der Folge **XXY** benutzt?

$$N = 3\text{:}\quad N_{\rm Bit} =$$

b) Welcher arithmetischer Code (AC) gilt für diesen Fall?

- ☐ AC = **01011**,
- ☐ AC = **010111**,
- ☐ AC = **110111**.

c) Wie viele Bit werden zur Darstellung von **XXYXXXZ** benutzt?

$$N = 7\text{:}\quad N_{\rm Bit} =$$

d) Ist **01011100001** ein gültiger Code für die Symbolfolge **XXYXXXZ**?

- ○ Ja.
- ○ Nein.

e) Welche Aussagen gelten für die arithmetische Codierung allgemein?

- ☐ Es handelt sich um eine gemeinsame Codierung ganzer Folgen.
- ☐ Eine 32 Bit–Rechnerarchitektur begrenzt die Folgenlänge N.
- ☐ Dieses Problem lässt sich durch Integer–Realisierung umgehen.
- ☐ Eine Integer–Realisierung erhöht die Codiergeschwindigkeit.

A2.13: Run–Length Coding und RLLC

Wir betrachten eine Binärquelle mit dem Symbolvorrat **A** und **B**, wobei **B** nur sehr selten auftritt.

- Ohne Quellencodierung würde man pro Quellensymbol genau ein Bit benötigen, und dementsprechend bei einer Quellensymbolfolge der Länge N für die Codefolge ebenfalls N Bit.
- Entropiecodierung macht hier ohne weitere Maßnahme (Zusammenfassen mehrerer Symbole zu einem Tupel) wegen der ungünstigen Symbolwahrscheinlichkeiten wenig Sinn.
- Abhilfe schafft **Run–Length Coding** (RLC), das unter dem genannten Link im Theorieteil beschrieben ist. Zum Beispiel:

  Quellensymbolfolge: **ABAABAAAABBAAB ...**
  Run–Lenght Coding: **2; 3; 5; 1; 3; ...**
- Natürlich muss man die Längen $L_1 = 2, L_2 = 3$, ... der einzelnen, jeweils durch **B** getrennten Substrings vor der Übertragung binär darstellen. Verwendet man für alle L_i jeweils $D = 3$ Bit, so erhält man die RLC–Binärfolge **010′011′101′001′011′...**
- Ein Problem der RLC ist der unbegrenzte Wertebereich der Größen L_i. Mit $D = 3$ kann kein Wert $L_i > 7$ dargestellt werden und mit $D = 2$ lautet die Beschränkung $1 \le L_i \le 3$.

Zeile	RLC (binär)	RLC (dez.)	RLC (kum.)
1	01111010	122	122
2	00101110	46	168
3	00011110	30	198
4	00000011	3	201
5	00001001	9	210
6	01000110	70	280
7	01010000	80	360
8	00111010	58	418
9	00001101	13	431
10	00111001	57	488
11	00001110	14	502
12	01001111	79	583
13	01011101	93	676
14	00110111	55	731
15	00010110	22	753
16	00001000	8	761
17	00001011	11	772
18	00001110	14	786
19	11100010	226	1012
20	00111100	60	1072
21	00000001	1	1073
22	00001011	11	1084
23	00111110	62	1146
24	00000111	7	1153
25	01100001	97	1250

- Das Problem umgeht man mit *Run–Length Limited Coding* (RLLC). Ist ein Wert $L_i \ge 2^D$, so ersetzt man L_i durch ein Sonderzeichen **S** und die Differenz $L_i - 2^D + 1$.
- Beim RLLC–Decoder wird dieses Sonderzeichen wieder expandiert.

Beispiel: Wir gehen wieder von obiger Binärfolge und dem Parameter $D = 2$ aus:

- Quellensymbolfolge: **ABAABAAAABBAAB ...**
- RLC–Dezimalfolge: **2; 3; 5; 1; 3; ...**
- RLLC–„Dezimalfolge": **2; 3; S; 2; 1; 3; ...**
- RLLC–Binärfolge: **01′11′ 00′10′01′11′...**

Man erkennt, dass hier das Sonderzeichen **S** als **00** binär–codiert ist (dies ist nur ein Beispiel – es muss nicht so sein). Da mit $D = 2$ für alle echten RLC–Werte $1 \le L_i \le 3$ gilt, erkennt der Decoder das Sonderzeichen und ersetzt es wieder durch $2^D - 1$ (im Beispiel drei) **A**–Symbole.

Die obere Grafik zeigt das das zu analysierende RLC–Ergebnis. In Spalte 2 und 3 sind die Substringlängen L_i binär bzw. dezimal angegeben und in Spalte 4 in kumulierter Form (Werte von Spalte 3 aufsummiert).

Hinweis: Die Aufgabe bezieht sich auf die **letzte Theorieseite** von **Kapitel 2.4.**

Fragebogen zu "A2.13: Run–Length Coding und RLLC"

a) Wieviele Bit würde man *ohne Quellencodierung* benötigen, also mit der Zuordnung **A** → **0** und **B** = **1**?

uncodiert: $N_{\rm Bit}$ =

b) Wie groß ist die relative Häufigkeit des Symbols **B**?

$h_{\rm B}$ =

c) Wie viele Bit benötigt man für *Run–Length Coding* (RLC) entsprechend der angegebenen Tabelle?

RLC, $D = 8$**:** $N_{\rm Bit}$ = Bit

d) Ist hier *Run–Length Coding* mit 7 Bit–Codeworten möglich?

- Ja.
- Nein.

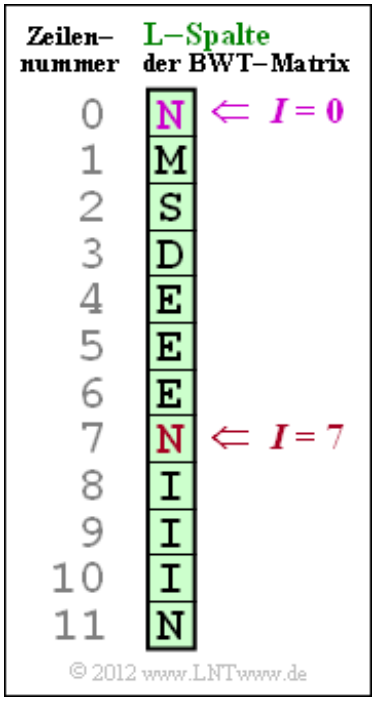
e) Wie viele Bit benötigt man mit *Run–Length Limited Coding* (RLLC) mit 7 Bit pro Codewort?

RLLC, $D = 7$**:** $N_{\rm Bit}$ =

f) Wie viele Bit benötigt man mit *Run–Length Limited Coding* (RLLC) mit 6 Bit pro Codewort?

RLLC, $D = 6$**:** $N_{\rm Bit}$ =

A2.14: Burrows–Wheeler–Rücktransformation



Die Burrows–Wheeler–Transformation – abgekürzt BWT – bewirkt eine blockweise Sortierung der Zeichen eines Textes mit dem Ziel, den Text für die effiziente Datenkomprimierung mit Hilfe einer **Lauflängencodierung** oder einer **Entropiecodierung** aufzubereiten.

- Zunächst wird aus einem Block der Länge N eine $N\times N$–Matrix erzeugt, wobei sich jede Zeile dieser ersten BWT–Matrix aus der Vorgängerzeile durch zyklische Linksverschiebung ergibt.
- Danach wird die Matrix lexikografisch (ohne Sonderzeichen: alphabetisch) sortiert. Das Ergebnis der BWT ist die letzte Zeile der neuen BWT–Matrix, die so genannte *L–Spalte*.
- Weiter wird in dieser Aufgabe auf die *F–Spalte* (erste Zeile der BWT–Matrix) Bezug genommen, die man für die inverse Burrows–Wheeler–Transformation benötigt ⇒ Rekonstruktion des Originaltextes aus der L–Spalte.
- Für die inverse BWT benötigt man ferner den so genannten *Primärindex I*. Dieser gibt diejenige Zeile der BWT–Matrix an, in welcher der Algorithmus gestartet werden muss.

Die Grafik zeigt das Ergebnis einer BWT, genauer gesagt die L–Spalte. Aus dieser soll der Originaltext entsprechend der Beschreibung auf der **entsprechenden Theorieseite** rekonstruiert werden, wobei in Teilaufgabe (b) von Primärindex $I = 7$ und in Teilaufgabe (c) von $I = 0$ auszugehen ist.

Hinweis: Die Aufgabe bezieht sich auf die **Seite 4a** und die **Seite 4b** von **Kapitel 2.4.** Weitere Informationen finden Sie in **[Abel03a]** und **[Abel03b].**

Fragebogen zu "A2.14: Burrows–Wheeler–Rücktransformation "

a) Wie lautet die zur vorgegebenen L–Spalte zugehörige F–Spalte?

- ☐ **SEINMEINDEIN**,
- ☐ **NIIINEEEDSMN**,
- ☐ **DEEEIIIMNNNS**.

b) Wie lautet das Ergebnis der Rekonstruktion mit dem Primärindex $I = 7$?

- ☐ **MEINDEINSEIN**,
- ☐ **DEINSEINMEIN**,
- ☐ **NIESNIEDNIEM**.

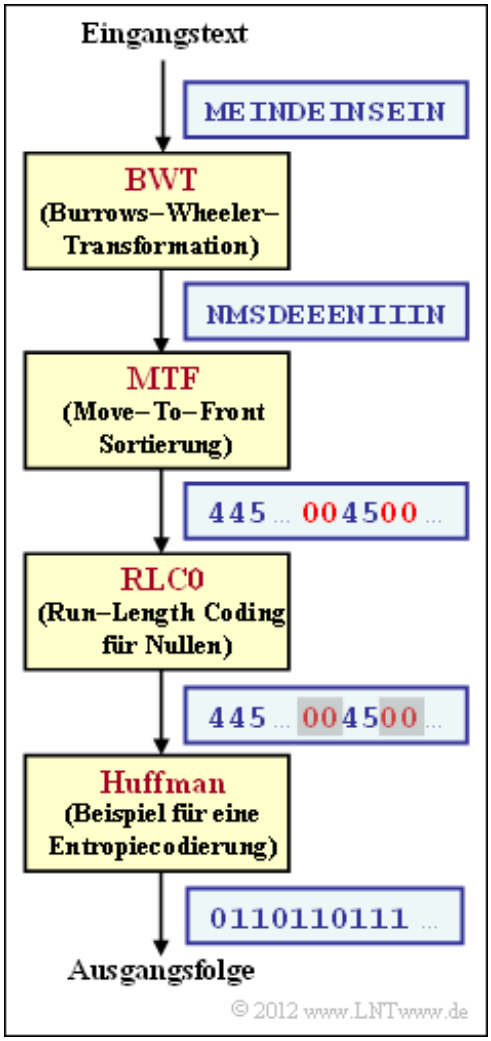
c) Was passiert, wenn bei der Rekonstruktion (BWT–Rücktransformation) vom falschen Primärindex ausgegangen wird, zum Beispiel von $I = 0$?

- ☐ Es ergibt sich der Originaltext, von hinten nach vorne gelesen.
- ☐ Das Ergebnis ist eine zyklische Vertauschung des Originals.

d) Warum ist die Burrows–Wheeler–Transformierte eines Textes hinsichtlich einer späteren Datenkomprimierung besser geeignet als das Original?

- ☐ Es ergeben sich günstigere Zeichenhäufigkeiten.
- ☐ Alle Zeichen sind lexikografisch sortiert.
- ☐ Gleiche Zeichen folgen in der BWT öfter aufeinander.

Z2.14: Kombination BWT & MTF



Wir beziehen uns auf die **letzte Theorieseite** von **Kapitel 2.4** und betrachten das rechts skizzierte Codiersystem, bestehend aus den Blöcken

- *Burrows–Wheeler–Transformation* (BWT) gemäß der Beschreibung in **Aufgabe A2.14;** Zeichenmengen am Ein– und Ausgang sind gleich: {**D**, **E**, **I**, **M**, **N**, **S**};
- *Move–to–Front* (MTF), ein Sortieralgorithmus, der eine gleich lange Zeichenfolge (im Beispiel $N = 12$), aber mit anderem Alphabet {**0**, **1**, **2**, **3**, **4**, **5**} ausgibt;
- *RLC0* – eine Lauflängencodierung speziell für die nach BWT und MTF (möglichst) häufige Null; alle anderen Indizes werden durch RLC0 nicht verändert;
- *Huffman* als Beispiel eines Entropiecodierers gemäß der Beschreibung in **Kapitel 2.3;** häufige Zeichen werden durch kurze Binärfolgen dargestellt, seltene durch lange.

Der MTF–Algorithmus lässt sich wie folgt beschreiben:

- Bei $M = 6$ Eingangssymbolen ist die Ausgangsfolge des MTF eine Aneinanderreihung von Indizes I aus der Menge $I = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}\}$.
- Vor Beginn des eigentlichen MTF–Algorithmus werden die möglichen Eingangssymbole lexikografisch sortiert und den folgenden Indizes zugeordnet:

  D → **0**, **E** → **1**, **I** → **2**, **M** → **3**, **N** → **4**, **S** → **5**.
- Der MTF–Eingabestring sei hier **N M S D E E E N I I I N**. Dies war das BWT–Ergebnis in der **Aufgabe A2.14.** Das erste **N** wird gemäß Voreinstellung mit $I = \mathbf{4}$ dargestellt.
- Anschließend wird das **N** in der Sortierung an den Anfang gestellt, so dass nach dem Codierschritt $i = 1$ folgende Zuordnung gilt:

  N → **0**, **D** → **1**, **E** → **2**, **I** → **3**, **M** → **4**, **S** → **5**.
- In gleicher Weise fährt man fort, bis der gesamte Eingangstext abgearbeitet ist. Steht ein Zeichen bereits an Position **0**, so ist keine Neusortierung erforderlich.

Hinweis: Die Aufgabe gehört zu **Kapitel 2.4.** Informationen zum Huffman–Code finden Sie in **Kapitel 2.3.** Für die Lösung dieser Aufgabe sind diese Informationen aber nicht erforderlich.

Fragebogen zu "Z2.14: Kombination BWT & MTF"

a) Welche Aussagen gelten für den Block „BWT" des Codiersystems?

- ☐ Die Eingangszeichenmenge ist {**D**, **E**, **I**, **M**, **N**, **S**}.
- ☐ Die Ausgangszeichenmenge ist {**D**, **E**, **I**, **M**, **N**, **S**}.
- ☐ In der Ausgangsfolge treten alle $M = 6$ Zeichen gruppiert auf.

b) Welche Aussagen gelten für den Block „MTF" des Codiersystems?

- ☐ Die Ausgangszeichenmenge ist {**D**, **E**, **I**, **M**, **N**, **S**}.
- ☐ Die Ausgangszeichenmenge ist {**0**, **1**, **2**, **3**, **4**, **5**}.
- ☐ Die MTF–Ausgangsfolge hat die Länge $N = 12$.

c) Wie lautet die MTF–Ausgangsfolge?

- ☐ **230000100405**,
- ☐ **445340045001**,
- ☐ **543120345123**.

d) Welche Aussagen gelten für den Block „RLC0" des Codiersystems?

- ☐ Der Eingangswert **0** erfährt eine Sonderbehandlung.
- ☐ Je häufiger eine **0** auftritt, um so effektiver ist dieser Block.
- ☐ Am besten wäre Pr(**0**) ≈ Pr(**1**) ≈ ... ≈ Pr(**5**).

e) Welche Aussagen gelten für den abschließenden Block „Huffman"?

- ☐ Die Ausgangsfolge ist binär.
- ☐ Er bewirkt eine möglichst kleine mittlere Codewortlänge.
- ☐ Die Dimensionierung richtet sich nach den anderen Blöcken.