H. Völz

VBASIC-Programm Entropie

Stand vom 22.11.06

Erklärung

Das Windows-Programm ist eine Arbeitsversion. Es muss nicht installiert werden. Die Entropie.exe kann direkt dort, wo sie abgelegt ist, aufgerufen werden. Eventuell sind die VBASIC 6 Standard-Run-Dateien (u.a. *.dll) erforderlich. Doch meist sind sie auf Windows-Rechnern automatisch vorhanden. Beim Aufruf des Programms werden keine zusätzlichen Daten (in Registry usw.) erzeugt. Das Programm ist nicht vollständig getestet. Bei „unüblichen“ Bedienungen kann es daher zur Blockierung oder zum Absturz kommen. Dann muss das Programm über den Task-Manager beendet werden. Für Interessenten liegt der Quell-Code bei.

Zweck des Programms

Das Programm benötigt einen Datensatz von Wahrscheinlichkeiten, deren Summe 1 beträgt. Von ihm können mehrere informationstheoretische Eigenschaften berechnet und angezeigt werden (Details dazu s.u.):

- Shannon, Alpha- (= Renyi-) und Bongard-Weiß-Entropie,
- Auffälligkeiten,
- Texttemperatur mit Abweichungen zur dazugehörenden Anpassung,
- Huffmann-, Shannon-Fano-, Langbaum- und zwei verschiedene Shannon-Codierungen.

Dieses Programm soll die theoretischen Aussagen[1] zu diesen Inhalten numerisch unterstützen. Hierzu existiert die Programmoberfläche von Bild 1. Beim Aufruf des Programms sind jedoch zunächst nur das Wahlmenü mit „Eingaben“ und „Zeichenzahl“ vorhanden. Die anderen in Bild 1 gezeigten Details der Oberfläche werden bei Nutzung des Programms je nach Notwendigkeit schrittweise ein- bzw. ausgeblendet. Es gibt ein Eingabefenster für die Daten, die auf unterschiedliche Weise übernommen bzw. erzeugt werden können (oberes Teilbild ①). Das Ausgabefenster steht erst nach Berechnungen zu Verfügung. Seine Eigenschaftsleiste und Länge hängt von den jeweils erzeugten Daten ab. Die einzelnen Rollbalken werden nur immer dann sichtbar, wenn sie auch für die Berechnungen genutzt werden können. Sie bleiben danach teilweise zur Orientierung über die jeweils aktuellen Werte stehen. Zur besseren Übersicht lassen sie sich jedoch immer mit „Ende“ ausblenden. Ähnliches gilt auch für das Eingabe- und Ausgabefenster. Hier erfolgt das Ausblenden jedoch mit „Beenden“ bzw. „Ausblenden“. Das Ausgabefenster kann immer wieder mit dem Wahlmenü „Ausgabe“ eingeblendet werden. Es besitzt wie auch „Zeichenzahl“ kein Untermenü. Die Bildausschnitte①, ② und ③ zeigen das vollständige Wahlmenü mit den jeweiligen Untermenüs.

Die Eingaben

Die Dateneingabe kann auf dreierlei Weise erfolgen. Bei Aufruf von „**Werte**“ werden 10 Zeilen der Art

A0 0
A1 0
A2 0
... bis
B0 0

erzeugt. Ihre Anzahl kann über den dabei erscheinenden Rollbalken „Zeichenzahl“ verändert werden. Danach muss aber – wie in gelb zu lesen ist – erneut „Werte“ aufgerufen werden. Die Benennung der Zeichen (A0 usw.) kann im Fenster editiert werden. Anstelle der „0“ müssen die entsprechenden Wahrscheinlichkeit mit „.“ Eingeben werden, also z.B. 0.534 oder auch nur .534. Ein Komma wird nicht akzeptiert! Bei dieser Eingabe ist zu beachten dass der Freiraum (Tabulator) nicht zerstört wird. Er kann leider nicht nachträglich eingegeben werden. Die letzte Wahrscheinlichkeit wird nicht eingegeben sondern durch „1“ ersetzt. Dies ist deshalb notwendig, weil die Summenwahrscheinlichkeit genau 1 sein muss. Durch die Eingabe von „1“ berechnet das Programm dann diesen Rest. Dies ist deshalb notwendig, weil sonst Rundungsfehler auftreten könnten, die sich später als Fehler bemerkbar machen. Nachdem diese Eingabe erledigt wurde, kann die Bestätigung durch Anklicken von „Übernehmen“ bestätigt werden. Wenn alles richtig ist erscheint darunter „Summe =1“. Andernfalls gibt es eine Fehleranzeige und die Daten können editiert werden. Sollen auch Daten ergänzt werden, sie ist zunächst der Regler für die Zeichenzahl zu verstellen und erneut „Werte“ aufzurufen. Die Zählung der Zeichenzahl beginnt hier mit

[1] Siehe z.B. H. Völz: Wissen - Erkennen - Information. Allgemeine Grundlagen für Naturwissenschaft, Technik und Medizin. Shaker Verlag, Aachen 2001. Auch auf dieser CD enthalten.

Null. Mit „Beenden“ wird die Standardberechnung ausgeführt und im Ausgabefenster erscheinen zeilenweise die Daten für die Zeichen, die Wahrscheinlichkeiten, $p_i \cdot \mathrm{ld}(p_i)$ und die Auffälligkeit. Darunter folgen dann die Shannon-Entropie, die maximale Entropie für Gleichverteilung und die Symbolanzahl.

Diese Eingabe mit Werte ist nur für eine kleine Anzahl von Daten sinnvoll. Für größere Mengen ist die Eingabe nach „**Nutzen**“ besser geeignet. Sie wird mit einem beliebigen Texteditor oder einem Textverarbeitungsprogramm vorbereitet. Dort Zeilenweise Benennung das Zeichens, Tabulator, Wahrscheinlichkeit (nicht in % sondern absolut) und „Enter“ (Return) einzugeben. Wichtig ist, dass auch die letzte Zeile mit „1“ mit Enter abgeschlossen wird. Diese Daten können dann auch als Textdatei gespeichert werden (Beispieldateien Alphabetx.txt sind im Programmpaket vorhanden). Die Daten werden dann markiert und mit Strg +C in die Zwischenablage kopiert. Nach Aufruf von Nutzen werden sie dann mit Strg + V in das leere Eingangsfenster kopiert. Die Eingabe wird dann wie oben geschildert mit Übernehmen fortgesetzt.

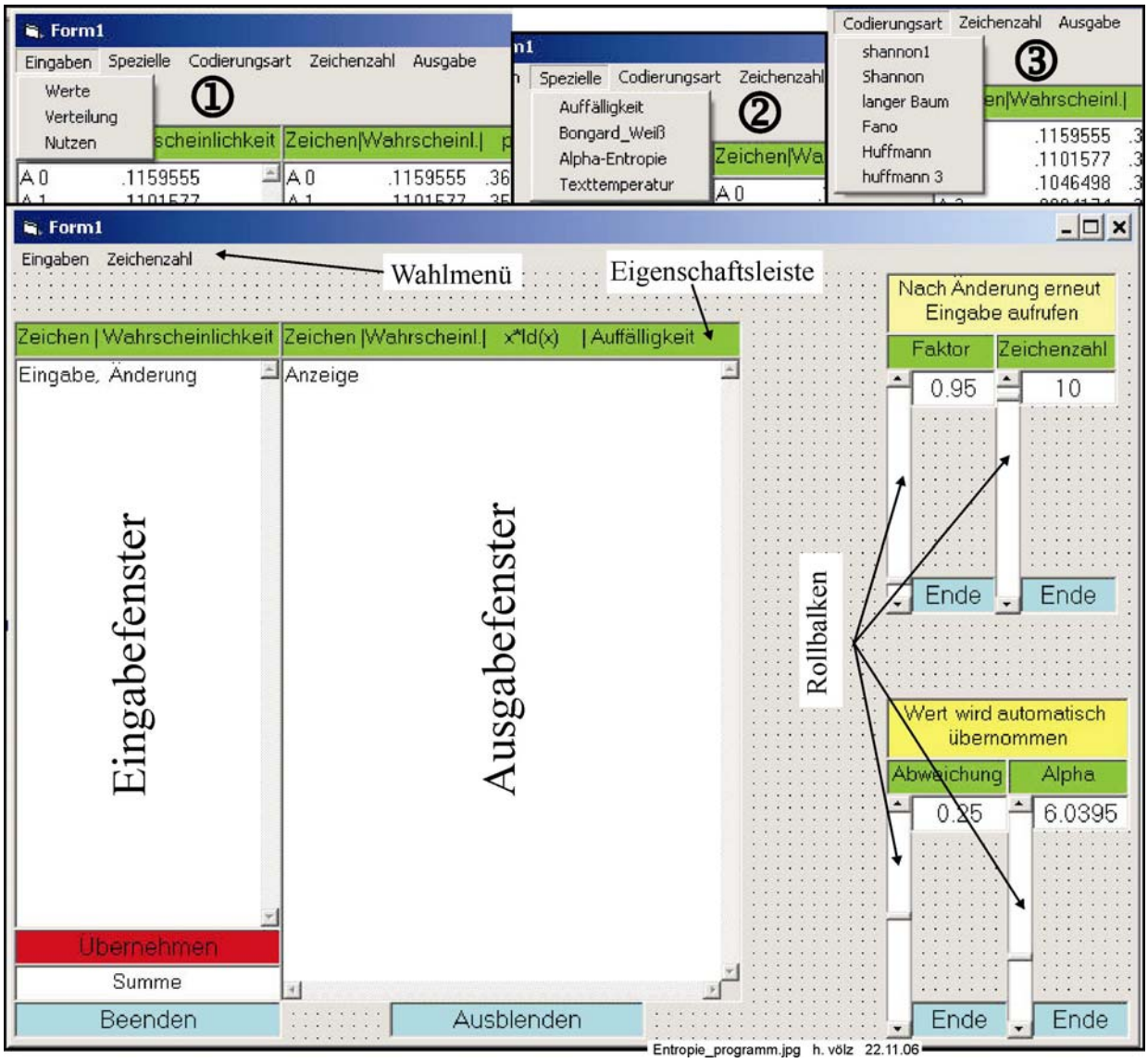

Bild 1 Überblick zur „vollständigen“ Oberfläche des Pogramms. Die Teilbilder ①, ② und ③ zeigen Ausschnitte, für die drei Untermenüs.

Die dritte Eingabemöglichkeit erfolgt mit „Verteilung“. Hierbei werden Wahrscheinlichkeiten mit dem „Faktor“ k (Regler links oben) zunächst nach dem Gesetz

$$p_0 = k;\ p_i = k \cdot p_{i-1}$$

erzeugt. Dann wird $x = \Sigma p_i$ gebildet. Anschließend werden alle p_i durch x dividiert und der letzte Wert auf 1 gesetzt. Diese Verteilung ist theoretisch interessant und hat einige spezielle Anwendungen

Spezielle Anzeigen

In diesem Menü sind 4 Varianten auswählbar. Die Variante „**Auffälligkeit**“ ist die immer zunächst automatisch aufgerufen Standardanzeige. Sie hängt mit dem Goldenen Schnitt zusammen. Für die einzelnen Zeichen entspricht dem Wert $p \cdot \mathrm{ld}(p)/max$, mit $max = 1/e \cdot \mathrm{ld}(1/e)$ und $e \approx 2{,}71...$ (Eulersche Zahl). Die Auffälligkeit[2] eines Zeichens besitzt bei $p = 1/e$ das Maximum von 1.

Bei der **Bongard-Weiß-Entropie** wird auch die subjektive empfundene Wahrscheinlichkeit *q* benutzt. Sie weicht vielfach deutlich von der objektiven Wahrscheinlichkeit *p* ab. Allgemein gilt dabei, dass bei kleinen *p* der Wert für *q* größer angenommen wird. Umgekehrt ist Werden sehr große *p* subjektiv zu klein empfunden. Die Zusammenhänge hierzu sind nur näherungsweise geklärt. Brauchbar vereinfacht ist im Programm der Zusammenhang nach Bild 2 modelliert. Der Faktor *A* ist mit dem Regler „Faktor“ zwischen 0 und 1 einstellbar. Bei A = 0 wird die Shannon-Entropie angenommen. Es ist zu beachten, dass vielfach gilt $\Sigma q \neq 1$. Meist gilt sogar $\Sigma q > 1$. Dies ergibt sich u.a. daraus, dass bei den meisten Wahrscheinlichkeitsverteilungen, die kleinen Werte überwiegen. Es kommt sogar relativ selten vor, dass ein Wert mit p > 0,5 vorhanden ist. Bei dem dann q kleiner ist. Infolge dessen fällt auch die Bongard-Weiß-Entropie meist deutlich größer als die Shannon-Entropie aus.

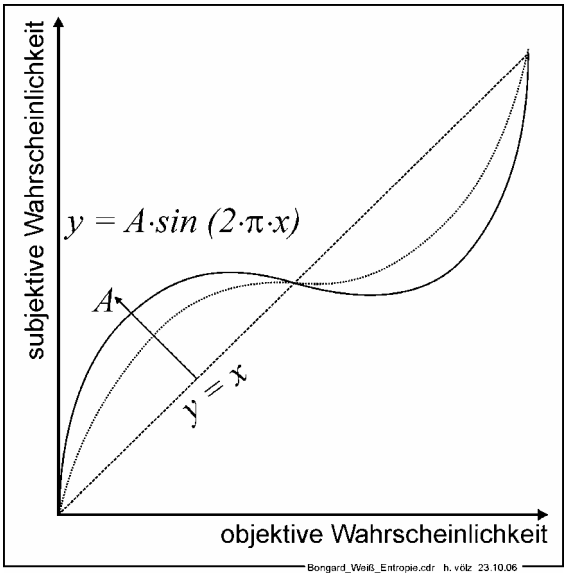

Bild 2. gewählter Zusammenhang zwischen objektiver und subjektiver Wahrscheinlichkeit.

Die **Alpha**- bzw. **Rényi-Entropie**[3] wurde von dem ungarischen Mathematiker ALFRÉD RÉNYI als Verallgemeinerung der Shannon-Entropie vorgeschlagen. Sie benutzt die Formel

$$H_\alpha = \frac{1}{1-\alpha} \cdot \mathrm{ld}\left(\sum_{1=1}^{n} p_i^\alpha \right).$$

Die Konstante α kann mit dem Regler „Alpha“ verstellt werden. Dabei werden immer automatisch die neu eingestellten Ergebnisse angezeigt. Für α =1 wird die Shannon-Entropie angenommen. Die Rényi-Entropie ist nur theoretisch interessant.

Die letzte Möglichkeit ist die **Texttemperatur**. Diese Begriff wurde u.a. von BENOÎT B. MANDELBROT (*1924) und GEORGE KINGSLEY ZIPF (1902 – 1950) eingeführt und ist als Zipf-Mandelbrot-Gesetz bekannt. Es geht auf die Erfahrung zurück, dass (fast) immer bei einer nach der Größe geordneten Wahrscheinlichkeitsverteilung und dem zugehörigen Rang R der Objekte bzw. Zeichen ein gesetzmäßiger Zusammenhang besteht (s. Fußnote [1]). Die einzelnen Formeln dafür weichen ein wenig voneinander ab. Fast immer gilt recht genau die einfache Regression

$$p = a \cdot R^{-1/T}.$$

Darin wird *T* als Texttemperatur bezeichnet. Mit diesem Programmteil werden T und a bestimmt, aber nur *T* und die Korrelation ρ^2 bezüglich der Wahrscheinlichkeitsverteilung summarisch angezeigt. Für die einzelnen Werte

[2] Siehe z.B. Literatur in Fußnote [1] oder: Völz, H.: Computer und Kunst. Reihe akzent 87. 2. Aufl. Urania - Verlag Leipzig Jena - Berlin 1990 (auch auf der Homepage www.rosw.cs.tu-berlin.de/voelz des Autors als Volltext mit Bildern vorhanden.

[3] Renyi, A.: Tagebuch über die Informationstheorie. Deutscher Verlag der Wissenschaften, Berlin 1982

wird die absolute Abweichung gegenüber der Regressionskurve angezeigt. Für die meisten „natürlich“ auftretenden Wahrscheinlichkeitsverteilungen beträgt ρ^2 nahezu 1. Die Texttemperatur liegt in der Nähe von 1.

Codierungen

Die Codierung Wahrscheinlichkeitsverteilungen erfolgt im Programm nach 6 Verfahren. Je nach der Codierung ergibt sich ein Codeaufwand gemäß

$$C_a = \sum L_i \cdot p_i \,.$$

Darin bedeuten Li die Länge des Codewortes und pi die dazugehörende Wahrscheinlichkeit. Zunächst werden hier die drei wichtigsten Algorithmen verbal so beschrieben:

Shannon-Codierung

1. Es wird ein vollständiger, binärer Codebaum der Tiefe *k* erzeugt, wobei *k* die kleinste ganze Zahl ist, welche die Gleichung $2^k \geq n$ erfüllt.
2. Da 2^k größer als *n* sein kann, werden soviel endständige Zweige entfernt, d. h. zwei Endknoten zu einem neuen zusammengefasst, bis genau *n* Endknoten existieren.
3. Auf die zusammengefassten Endknoten werden die Zeichen mit der größten Wahrscheinlichkeit gelegt, an die weiteren Knoten die anderen.

Shannon-Fano-Codierung

1. Jedem Zeichen wird ein leerer Code-Vektor zugeordnet.
2. Die Zeichen werden nach fallender Wahrscheinlichkeit sortiert.
3. Die Anordnung der Zeichen wird durch einen Schnitt in zwei Gruppen mit möglichst gleich großer Wahrscheinlichkeit geteilt.
4. Die „obere“ Teilgruppe wird mit „1“ codiert, die untere mit „0“. Diese Zeichen werden dem jeweiligen Code-Vektor der Zeichen angehängt.
5. Mit jeder Teilgruppe wird ab Schritt 3. solange fortgefahren, bis jeweils einzelne Symbole erreicht sind.

Huffmann-Codierung

1. Jedem Zeichen wird ein leerer Code-Vektor zugeordnet.
2. Die Zeichen werden nach fallender Wahrscheinlichkeit sortiert.
3. Die beiden Zeichen mit der kleinsten Wahrscheinlichkeit werden mit 0 bzw. 1 codiert und dem Code-Vektor vorne angefügt.
4. Die beiden codierten Zeichen werden als neues Hilfszeichen eingeführt, das ihre Summenwahrscheinlichkeit erhält. Die beiden ursprünglichen Zeichen erscheinen nicht mehr direkt im Zeichenvorrat.
5. Mit dem neuen Zeichenvorrat wird bei Schritt 2 solange fortgefahren, bis nur zwei Hilfssymbole existieren. Das Vorsetzen der Codes 0 bzw. 1 erfolgt aber immer für alle Zeichen, die in dem Hilfszeichen zusammengefasst sind.

Im Programm sind der noch weitere Varianten vorhanden. Bei der Shannon-Codierung (CLAUDE ELWOOD SHANNON; 1916 – 2001) wird nicht genau mitgeteilt, wie die Kürzung erfolgen soll. Manchmal gibt es nämlich mehrere Varianten. Hiervon sind die beiden extreme in Bild 3a und b als **Shannon1** und **Shannon** gezeigt. In der Literatur ist der Fall von Bild 3a üblich. Die jedoch mit einer Kürzung zu erreichende kleinstmöglichen Entropien liefert das Verfahren von Bild 3b.

Ein ebenfalls selten behandelter Algorithmus führt zu einem möglichst **tiefen (langen) Baum** (Bild 3c). Er ist besonders bei sehr „steilen“ Wahrscheinlichkeitsverteilungen nützlich. Dort kann er sogar die Werte des Huffmann-Algorithmus erreichen.

Die Teilung bei der **Shannon-Fano-Codierung** (ROBERT FANO; *1917) ist rechentechnisch sehr aufwendig. Sie muss nämlich kompliziert rekursiv vorgenommen werden. Im Programm wurde dabei das Schema von Bild 4 benutzt. Es wird ein Feld T(i) für die Zählung und Bereichsteilung der Zeichen benutzt. Dabei zählt das Zeichen mit *p*(T(i)) jeweils zum oberen Bereich. Die erste Teilung für T(1) erfolg bei dem Zeichen das möglichst nahe zur Summenwahrscheinlichkeit ½ liegt. Sie möge bei diesem Zeichen p_{summe} ergeben. Die Zeichen oberhalb von T(1) werden nun zusätzlich mit 1 die darunter mit 0 codiert. Die werte werden an die bereits vorhandene hinten angehängt. Dann erfolgt die nächste Teilung T(2) bei einem Zeichen, das eine möglichst nahe Summenwahrscheinlichkeit von für $p_{summe}/2$ ergibt. Auch hier erhalten die obern eine 1 und die unteren eine 0 als Codierung angehängt. Dies wird iterativ soweit fortgesetzt, bis nur noch zwei Zeichen im Intervall vorhanden sind. Anschließend muss dieses Prinzip dann zusätzlich auf die entstandenen – im Bild gezeichneten – Teilintervalle fortgesetzt werden.

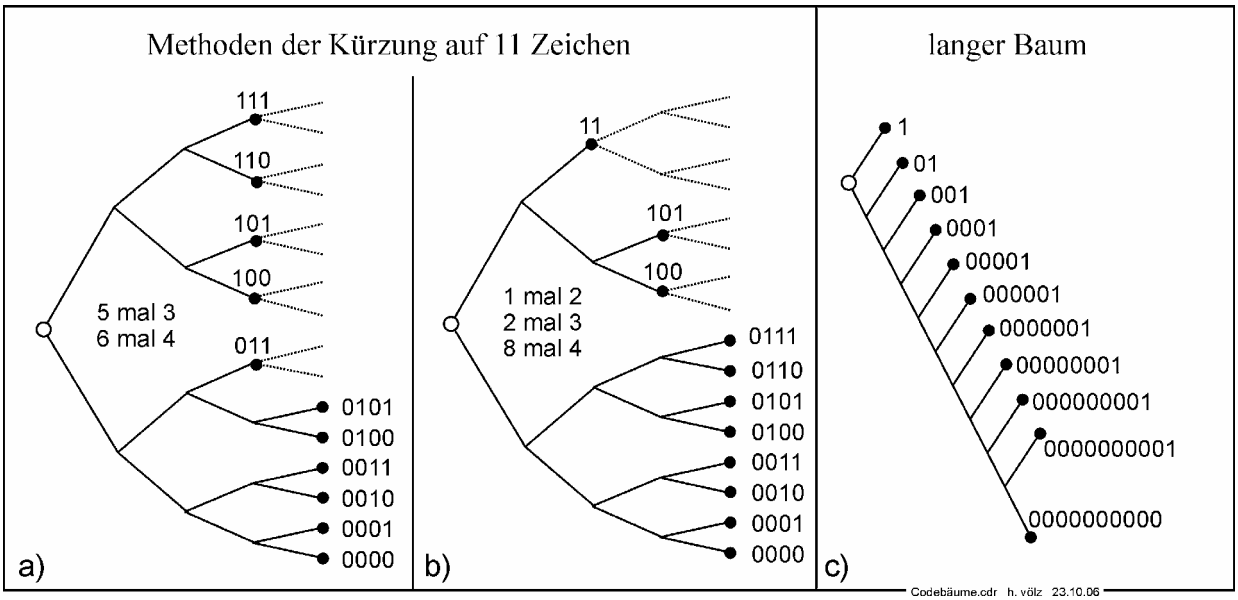

Bild 3. Die ersten drei Codierungsmöglichkeiten am Beispiel mit 11 Zeichen: a) Shannon1, b) Shannon und c) langer Baum.

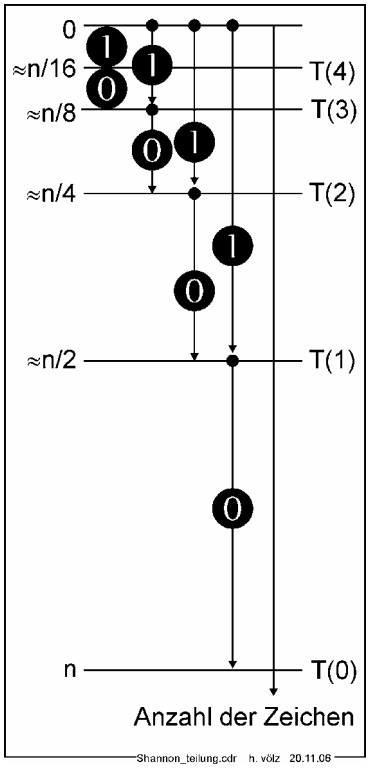

Bild 4 Schema für den beginn der Teilung bei der Shannon-Fano-Codierung.

Die **Huffmann-Codierung** (DAVID A. HUFFMAN; 1925 – 1999) sieht durch das immer zwischendurch erfolgende Zusammenfassen von zwei Zeichen zu einem neuen Zeichen und das anschließende Sortieren komplizierter aus. Sie ist jedoch wesentlich einfache zu programmieren. Es werden immer nur die beiden kleinsten Zeichen mit voranstellen von 0 bzw. 1 codiert. Doch es muss zusätzlich eine Kennung geben, welche jene Zeichen erfasst, die schon entsprechend Codiert wurden. Hierbei wurde ein kleiner Trick benutzt, der im Quellcode zu ersehen ist.

Bei der **Huffman3**-Codierung wird ein trinärer Baum benutzt. Diese Codierung wurde deshalb einbezogen, weil theoretisch nicht eindeutig entschieden ist, ob generell die binäre Huffmann-Codierung die beste aller Präfixcodierungen ist. Eventuell in Betracht kommen jedoch nur Verzweigungen mit Primzahlen, also 3, 5, 7, 11 usw. Hier wurde lediglich der Fall Basis 3 programmiert. Für den Codeaufwand ist dabei zu beachten, dass einer dreifachen Verzweigung auch eine dreifache Entscheidung entspricht. Daher gilt jetzt

$$C_a = ld(3) \cdot \sum L_i \cdot p_i \; .$$

Doch bei der trinären Codierung treten zuweilen auch binäre Endverzeigungen auf. Das ist immer dann der Fall wenn die Anzahl der noch zu codierenden Zeichen 2 ist. Deshalb ist der wirkliche Codeaufwand zuweilen niedriger als der obige Formelwert. In der Praxis ist diese Vereinfachen jedoch kaum nutzbar, diese Fälle müssten nämlich dem Empfänger zusätzlich mitgeteilt werden. Daher wurde im Programm die Berechnung des Codeaufwandes nach der obigen Formel vorgenommen.

Nebenbei sei bemerkt, dass die trinäre Codierung zuweilen auch praktische Bedeutung hat. Dies ist z.B. bei der folgenden Aufgabenstellung der Fall. Sie haben neun Kugeln, die alle bis auf eine leichtere das gleiche Gewicht besitzen. Wie stellen sie nun mit möglichst wenigen Schritten mittels einer Balkenwaage fest, welches die leichtere Kugel ist? Da bei jeder Wägung drei Möglichkeiten bestehen – Gleichgewichtes bzw. links oder rechts leichter gilt nun theoretisch

$$H_{trinär} = -\sum p_i \cdot \log_3(p_i) \, .$$

Es genügen zwei Wägungen. Die übliche Lösung besteht darin, beidseitig mit 3 Kugeln zu beginnen.