Zusammenfassung Modul 114

Codierungs- , Kompressions- und Verschlüsselungsverfahren einsetzen

Version: 01.00
Freigabe: 20.05.11

Janik von Rotz
Hoheneich 4, 6064 Kerns
Internet www.janikvonrotz.ch
Email contact@janikvonrotz.ch

Inhaltsverzeichnis

Änderungskontrolle

Version	Datum	Autor	Beschreibung der Änderung	Status
<<#>>	<<Datum>>	<<Name>>		

Referenzierte Dokumente

Nr.	Dok-ID	Titel des Dokumentes / Bemerkungen
<<#>>	<<#>>	<<Titel/Name des Dokumentes>>

Titel:	Zusammenfassung Modul 114	**Typ:**	Hanbuch	**Version:**	01.00
Thema:	Codierungs- , Kompressions- und Verschlüsselungsverfahren einsetzen	**Klasse:**	öffentlich	**Freigabe:**	20.05.11
Autor:	Janik von Rotz	**Status:**	Freigegeben	**PrtDat./gültig bis:**	20.05.11 / Mai 11
Ablage/Name:	c:\Dokumente und Einstellungen\ILZ32\Eigene Dateien\Dropbox\exchange\teil_abschluss_prüfungen\zusammenfassung\m114\modul114_zusammenfassung.docx			**Registratur:**	.

1. Codierungsverfahren

1.1 Zahlensysteme

1.1.1 Umrechnung von Dezimal in ein anderes System

Für die Basis wurde hier 2 genommen, heisst anstatt 2 kann man jede andere Basis eines Zahlensystems verwenden

- **Schritt 1:(Aufteilen)**

Die Dezimalzahl wird in den ganzzahligen Anteil und in die Nachkommastellen aufgeteilt.

→ 207,2210 = 20710 + 0.2210

- **Schritt 2:(Der ganzzahlige Anteil)**

Man teilt die Zahl so oft durch die Basis (2), bis das Ergebnis Null wird. Die Reste der Division (jeweils 0 oder 1) ergeben die neuen (binären) Ziffern mit steigender Wertigkeit.

Die Zahl 20710 ins binäre Zahlensystem umrechnen:

207 : 2 = 103 Rest 1 ↑
103 : 2 = 51 Rest 1 |
51 : 2 = 25 Rest 1 |
25 : 2 = 12 Rest 1 |
12 : 2 = 6 Rest 0 |
6 : 2 = 3 Rest 0 |
3 : 2 = 1 Rest 1 |
1 : 2 = 0 Rest 1 | => 20710 = 11001111

- **Schritt 3:(Die Nachkommastellen)**

Die Nachkommastellen werden mit der Basis B (2) multipliziert. Das Produkt besteht aus einem ganzzahligen Anteil und aus neuen Nachkommastellen. Der ganzzahlige Anteil ist schon Teil des Umwandlungsergebnisses. Die neu entstandenen Nachkommastellen werden wieder mit der Basis B multipliziert, usw. Das Verfahren ist fortzusetzen, bis die Nachkommastellen Null werden oder bis die gewünschte Genauigkeit erreicht ist.

Die Zahl 0.2210 ins binäre Zahlensystem umrechnen.

0,22 * 2 = 0.44 + ganzzahliger Teil = 0 |
0,44 * 2 = 0.88 + ganzzahliger Teil = 0 |
0,88 * 2 = 0.76 + ganzzahliger Teil = 1 |
0,76 * 2 = 0.52 + ganzzahliger Teil = 1 |
0,52 * 2 = 0.04 + ganzzahliger Teil = 1 |
0,04 * 2 = 0.08 + ganzzahliger Teil = 0 ↓ => 0.2210 = 0.001112

- **Schritt 4:(Zusammenfassen)**

Die Teilergebnisse der Schritte 2 & 3 zusammenfassen:

207,2410 = 110011112 + '.' + 001112 = 11001111. 001112

1.1.2 Umrechnung von jedem beliebigen System ins Dezimale

Hier einige Beispiele

Die Binärzahl 11001111,012 setzt sich aus folgendem Sachverhalt zusammen:

Z = 11001111,012 = 1•27 + 1•26 + 0•25 + 0•24 + 1•23 + 1•22 + 1•21 + 1•20 + 0•2-1 + 1•2-2
= 1•128 + 1•64 + 0•32 +0•16 + 1•8 + 1•4 + 1•2 + 1•1 + 0•0.5 + 1•0.25
= 207,2510

Oktalzahlen werden mit der Basis “8“ oder “O“ dargestellt $1278_8 = 127_O$

$125{,}48_O$ = 1•82 + 2•81 + 5•80 + 4•8-1
= 1•64 + 2•8 + 5•1 + 4•0,125
= 85,510

Die Formel dazu lautet also:

1.1.3 Umwandlung Dual <> Hexadezimal

Durch die Zusammenfassung von jeweils 4 nebeneinanderliegenden Ziffern kann eine Dualzahl in eine Hexadezimalzahl umgewandelt werden.

Für die Umwandlung einer Hexadezimalzahl in eine Dualzahl wird entsprechend jede Ziffer in eine Vierergruppe von Binärziffern umgerechnet (Nibbelbildung).

Ein Nibbel = 4 Bit

0011 0010 1011 0110 B
3 2 B 6H

1.1.4 Beispiele

Dual	Hexadezimal
1010’1011	
1’1111’1110	
1001’1001	
1000’0000’0000’0001	
	D16
	1234
	AB
	10011

1.1.5 Umwandlung Dualzahl ↔ Oktalzahl

Durch die Zusammenfassung von jeweils 3 nebeneinanderlegenden Ziffern kann eine Dualzahl in eine Oktalzahl umgewandelt werden.

Für die Umwandlung einer Oktalzahl in eine Dualzahl wird entsprechend jede Ziffer in eine Dreiergruppe von Binärziffern umgerechnet.

$$\underline{011}\ \underline{010}\ \underline{101}\ \underline{110}_{B}$$
$$3\quad 2\quad 5\quad 6_{O}$$

1.2 Rechnen mit Dualzahlen

1.2.1 Binäre Addition

Dualzahl werden in ähnlicher Weise addiert wie Zahlen im Dezimalzahlen:

Bei der Addition gelten folgende Rechenregeln:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 **Achtung, hier erhält das Übertragsbit, d.h. das Carry - Bit den Status 1!**

Da sich das Ergebnis 1 + 1 nicht mehr in einer einzigen Stelle darstellen lässt, muss eine zusätzliche Stelle eingeführt werden. Es entsteht ein Übertrag (Carry) auf die nächste, höherwertige Stelle.

1.2.1.1 Beispiel mit ganzen Zahlen

Dezimalsystem:

			5
	+	4	2
	+	5	4
Carry	**1**	**1**	
	1	0	1

Dualsystem:

				1	0	1_2
	+	1	0	1	0	0_2
	+	1	1	0	1	0_2
Carry	**1**	**1**	**1**	**1**	**1**	
	1	1	0	0	1	0 1_2

1.2.1.2 Beispiel mit gebrochenen Zahlen

Dezimalsystem

	1	,	5	
+	2	,	2	5
+	3	,	6	2 5
Carry 1				
	7	,	3	7 5

Dualsystem

			1	,	1_2		
+		1	0	,	0	1_2	
+		1	1	,	1	0	1_2
Carry	**1**	**1**	**1**				
	1	1	1	,	0	1	1_2

1.2.2 Zweier-Komplement

Das Zweier-Komplement einer negativen Zahl erhält man, indem man die Zahl positiv darstellt, danach sämtliche Bits invertiert (0→1 und 1 →0) und +1 rechnet.

1.2.2.1 Beispiel: (Wir möchten -6 im Zweierkomplement darstellen)

+6 = 0110

-6 = Alle Bits kehren ==> (1001) und +1 rechnen ==> 1010

Daraus ergibt sich folgende Formel:

1.2.2.2 Beispiel im Dezimalsystem (Basis B = 10):

Annahme: Es stehen drei Zeichen zur Verfügung (n=3),
das heisst, der Bereich geht von 000 bis 999.

Wir bilden nun das Komplement der Zahl -55:

K(-55) = 103 - 55= 945 (945 entspricht der Zahl -55 in Zweierkomplementdarstellung)

Nun wollen wir Testen, ob wir die Subtraktion wie die Addition rechnen können:
310 - 55 = 255 (normale Subtraktion)
310 + 945 = 1255 (!! nur 3 Stellen) → 1255 (Subtraktion im Komplement entspricht der Addition!)

Ist eine Zahl in Zweier-Komplement dargestellt, sagt die erste Ziffer aus, ob es sich um eine positive oder negative Zahl handelt (bei binären Zahlen: 0=pos, 1=neg, bei Dezimalzahlen 0..4= pos. 5..9=neg.).

1.2.2.3 Darstellung Dezimalzahl als Dualzahl im Zweierkomplement

Die Zahl -2510 soll als Dualzahl im Zweierkomplement (8 Stellen) dargestellt werden:

Das Zweier-Komplement einer negativen Dualzahl kann gebildet werden, indem man die Zahl Stellenweise invertiert und anschliessend 1 addiert.

+ 2510 = 000110012
Die Zweierkomplementsdarstellung ergibt sich nun durch die Kehrung aller Bits + 1:
-2510 = 111001102 + 000000012 = 111001112

Daraus ergibt sich folgende Formel:

2. Digitalisierung

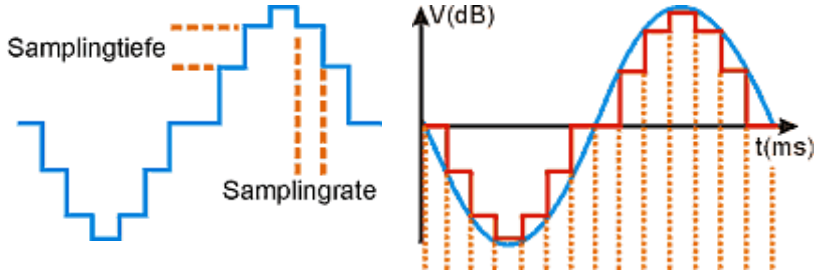
2.1 Quantisierung

Bei der analog/digital Wandlung wird die Lautstärke in kurzen Abständen gemessen und aufgezeichnet. Dadurch entsteht aus der ursprünglichen Schwingungskurve, eine Kurve aus kleinen Treppenstufen

Der digitale Ton wird mit einer durch die Samplingrate und Samplingtiefe vorgegebenen Auflösung punktuell abgetastet.

Die beiden Parameter bestimmen wie fein/grob die Rasterung des analogen Signals ausfällt.

Den Vorgang, etwas in ein vorgegebenes Raster zu übertragen nennt man Quantisierung.



2.2 Samplingrate

Die Anzahl der Messungen pro Sekunde wird als Samplingrate (Abtastfrequenz) bezeichnet.

2.3 Samplingtiefe

Die Samplingtiefe ist der zweite Parameter um eine Schwingung zu quantisieren.

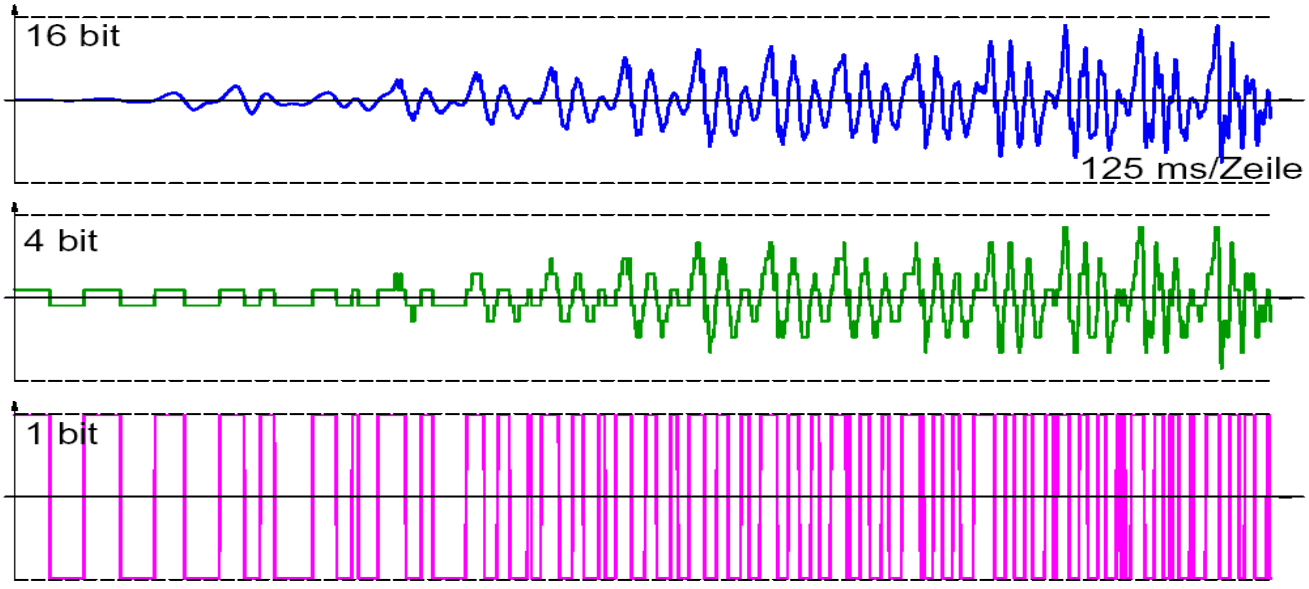
Den gemessenen Lautstärkepegeln wird jeweils ein Wert auf einer Skala zugeordnet und in Form eines Datenworts digital aufgezeichnet.

Dieses Datenwort besteht aus einer festgelegten Anzahl von Binärinformationen.

Bei 16 Bit Quantisierung lässt sich schon eine Skala von 65536 Werten darstellen.

Die Samplingtiefe gibt an, wie viel Bit für die Quantisierung der Pegelmessungen zur Verfügung stehen.

Je mehr Bit, desto mehr Lautstärkenabstufungen sind möglich und desto naturgetreuer wir die Aufzeichnung ausfallen.



2.4 Speicherplatz

Der Speicherbedarf lässt sich durch Multiplikation von Samplingrate und Samplingtiefe berechnen: z.B. 44100 (1/s) x 16 Bit x 60 s (pro Minute) x 2 (bei Stereo) = 84672000 Bit = (/8) 10584000 Byte = (/1028) 10296 kB

= (/1028) ca. 10 MB pro Minute; bei 96 kHz / 24 Bit sind es schon ca. 33 MB pro Minute.

2.5 Eigenschaften von Bildern

2.5.1 Zeichengrafik

- Relikt von früher, als auf dem Bildschirm und Drucker nur Textzeichen ausgegeben werden konnte.
- Bild wird mit Hilfe von Textzeichen vom ASCII-Code dargestellt.
- Findet man in Grossrechnern.
- Sehr schnell aus Drucker ausgegeben
- Diese ASCII-Arts werden in der Warez-Szene sehr aktiv angewendet und beworben. Es findet dort
- ein richtiger Wettbewerb unter den Erstellern statt.
- Der Host der Suva liefert auch solche „Bilder“.

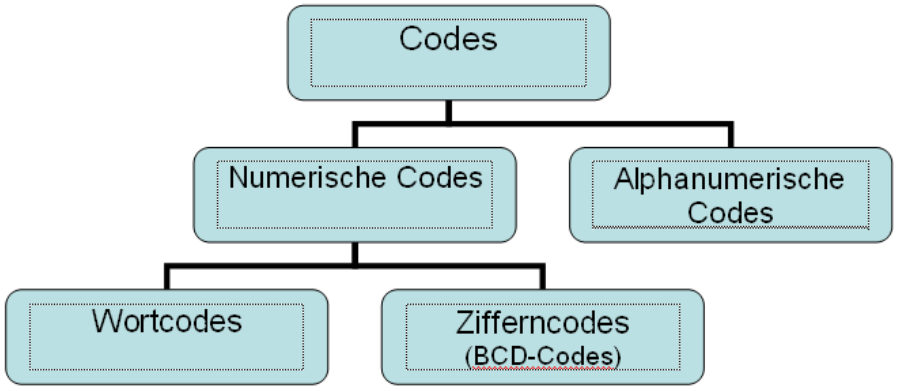
2.5.2 Pixelgrafik

- Bild besteht aus Bildpunkten (Pixel)
- Computer und Fernseher (768 x 576) können grundsätzlich nur Pixelgrafiken darstellen.
- Jeder Pixel besteht aus drei Elementen: Rot, Grün, Blau (RGB)
- Speicherbedarf: Anzahl der Bildpunkte horizontal und vertikal, Farbe der einzelnen Pixel
    - 640 x 480 mit Grautönen (256 Farben) = 640 * 480 * 1 (1Byte = 8 Bit; 28 = 256) = 307200 Byte
- Anwendung: Schnell auf dem Bildschirm dargestellt. Schnelle Bildfolge = Filmsequenz.
- OCR-Umwandlung möglich.
- Problem: Qualitätsverlust bei Vergrösserung

2.5.3 Vektorgrafik

- Keine Punkte sondern Bildelemente, welche genau beschrieben sind (Farbe, Dicke, Muster, Lage).
- Durch festgelegte Sprache beschrieben (z.B. SVG)
- Vergrössern ohne Qualitätsverlust
- Bildelemente einzeln manipulierbar
- Bei Ausgabe an Bildschirm oder Drucker wird die Vektorgrafik in eine Pixelgrafik umgewandelt
- Stiftplotter kann Vektorgrafiken als solche Ausgeben.

3. Codesysteme



3.1 Eigenschaften

Ein Code wird durch folgende Eigenschaften Beschrieben:

- Stellenzahl (Aus wie vielen Stellen besteht ein Codewort?)
- Bewertbarkeit (ist jeder Stelle eine bestimmte Wertigkeit zugewiesen? z.B.22 ,23)
- Hamminggewicht (Anzahl mit 1 belegten Stellen z.B. 0101 = Gewicht 2)
- Hammingdistanz (das Minimum aller Abstände zwischen Wörtern innerhalb des Codes)
- Stetig (ist die Distanz zwischen allen Codewörtern konstant)
- Redundanz (sind mehr Kombinationen vorhanden als die Codierung mathematisch (theoretisch) benötigt?

3.1.1 Anzahl nötiger Stellen

3.1.2 Redundanz

Anzahl Verwendete Stellen – Anzahl nötiger Stellen = Redundanz

3.2 Dualcode

Zugeordnete Ziffer	Wertigkeit						
	$2^{..}$	2^5	2^4	2^3	2^2	2^1	2^0
0	...	0	0	0	0	0	0
1	...	0	0	0	0	0	1
2	...	0	0	0	0	1	0
3	...	0	0	0	0	1	1
4	...	0	0	0	1	0	0
5	...	0	0	0	1	0	1
.....	....	...	...	...	...	...	...
63	...	1	1	1	1	1	1
.....	....	...	...	...	...	...	...

Stellenzahl	unen∞
Bewertbar	**Ja**
Gewicht	0..∞
Hammingdistanz	1..∞
Stetig	**Nein**
Redundanz	**0**

3.3 8-4-2-1-Code

Zugeordnete Ziffer	Wertigkeit			
	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	1	0	1
	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

Stellenzahl	**4**
Bewertbar	**Ja**
Gewicht	**0..4**
Hammingdistanz	**1..4**
Stetig	**Nein**
Redundanz	**0,678**

3.4 BCD-Zählcode

Zugeordnete Ziffer	Wertigkeit									
	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1

Stellenzahl	**10**
Bewertbar	**Nein**
Gewicht	**1..10**
Hammingdistanz	**1-9**
Stetig	**Ja**
Redundanz	**6,68**

3.5 1 aus 10 Code

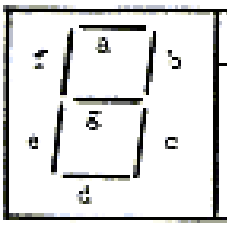
Zugeordnete Ziffer	Wertigkeit									
	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0

Stellenzahl	10
Bewertbar	Ja
Gewicht	1
Hammingdistanz	2
Stetig	Ja
Redundanz	6,68

3.6 7-Segment-Code für Ziffernanzeigen

Stellenzahl	7
Redundanz	**3.68**

Vervollständigen Sie die folgende Tabelle für den 7-Segment-Code:

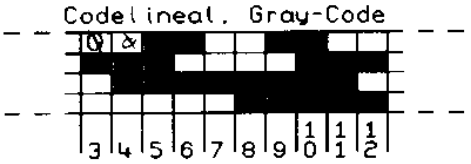


	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

3.7 Gray-Code

Zugeordnete Ziffer	Wertigkeit						
	-	-	-	-	-	-	-
0	...	0	0	0	0	0	0
1	...	0	0	0	0	0	**1**
2	...	0	0	0	0	**1**	1
3	...	0	0	0	0	1	**0**
4	...	0	0	0	**1**	1	0
5	...	0	0	0	1	1	**1**
6	....	0	0	0	1	**0**	1
7	...	0	0	0	1	0	**0**
8	....	0	0	**1**	1	0	0
9	....	0	0	1	1	0	**1**
10	....	0	0	1	1	**1**	1
11	....	0	0	1	1	1	**0**
…	....	...	...	...	...	...	...

Stellenzahl	∞
Bewertbar	**Nein**
Gewicht	∞
Hammingdistanz	**1..** ∞
Stetig	**Ja**
Redundanz	**0**



4. Kompressionsverfahren

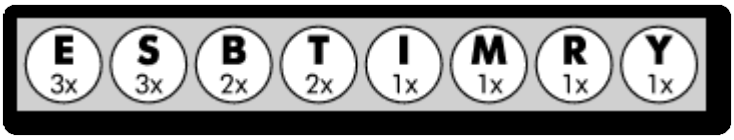
4.1 Huffman-Codebaum

Die Huffmann Codierung ist eine Codierung mit variabler Länge. Ihr Ziel ist es, die Länge der häufig auftretenden Zeichen zu minimieren. Die Theorie der Huffmann Codierung wurde bereits vor einigen Jahren vom gleichnamigen Mathematiker entwickelt. Sie wird in der Praxis sehr oft angewendet (MPEG, WinZip, etc.). Kein anderer Zeichencode führt zu einer kürzeren Kodierung als die Huffman-Kodierung.

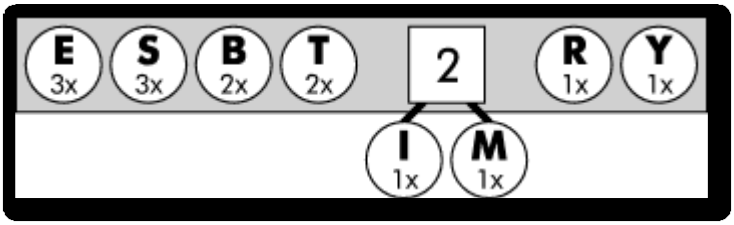
4.1.1 Ein Beispiel

Das zu kodierend Wort lautet: **„BETRIEBSSYSTEM“**

Um die Buchstaben ordentlich klassifizieren zu können, muss man sie zunächst einmal auszählen (hier der Häufigkeit nach sortiert):

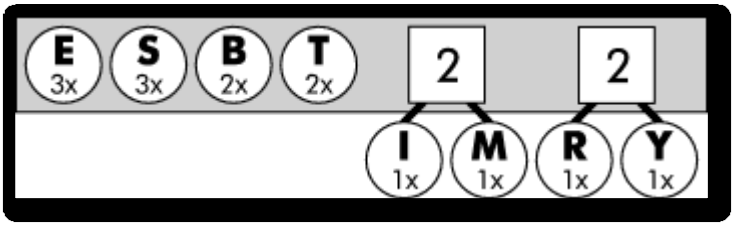


Im weiteren Verlauf baut man nun einen Binärbaum auf, indem man immer die beiden seltensten Symbole zu einem Knoten zusammenfasst. Der Knoten erhält auch eine Häufigkeitsangabe, welche die Summe aus den beiden Ursprungssymbolen ist.

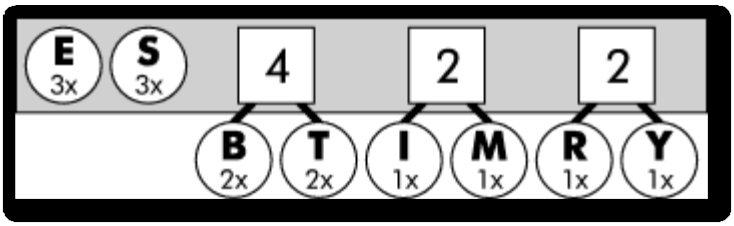


Hier wurden zwei (I; M) der vier Symbole mit nur einem Vorkommnis (I; M; R; Y) zu einem Knoten zusammengefasst. Dieser Knoten erhält also die Häufigkeit 1+1=2.

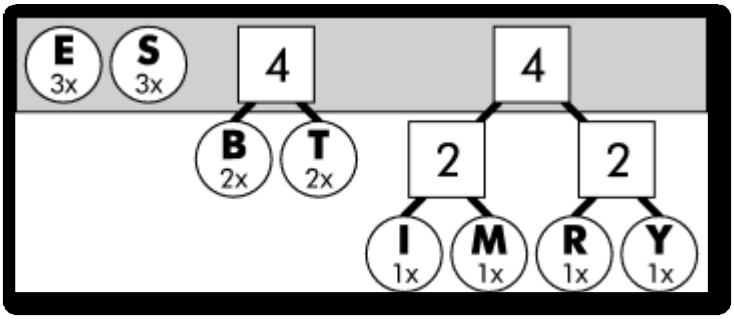
Das Spiel geht nun solange weiter, bis nur noch ein Knoten übrig ist:



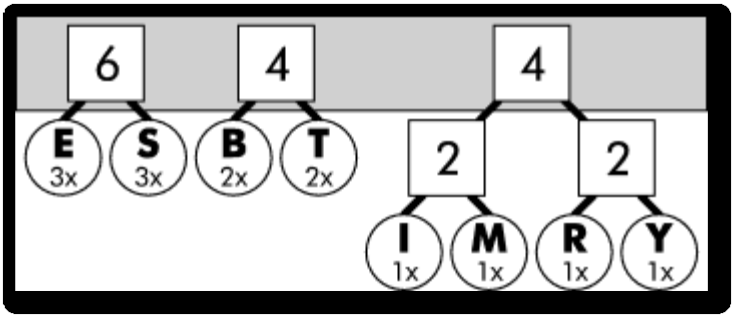
Hier wurden die beiden übrigen 1x-Symbole zusammengefasst.



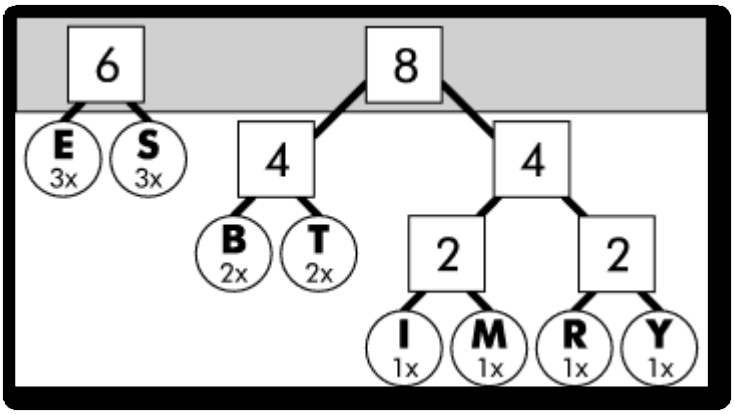
Wenn man die Liste an der Basis des Baumes von links nach rechts durchsucht, trifft man auf die Symbole B und T, die je zweimal vorkommen.



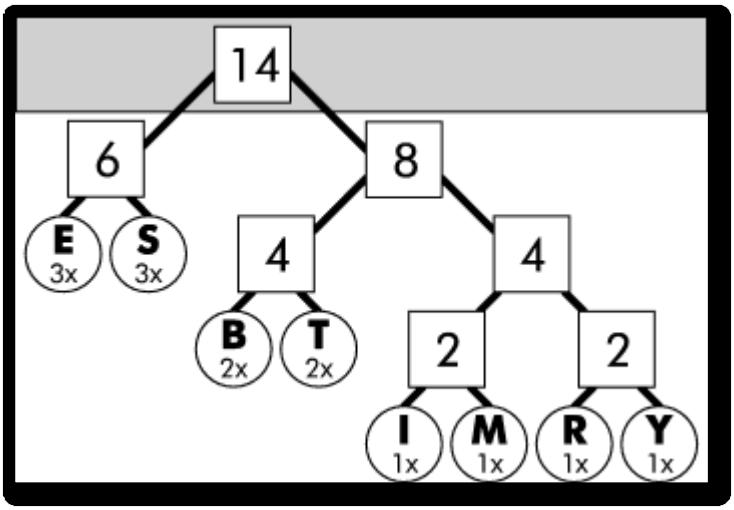
Die Situation hat sich verändert: Die beiden schon verknoteten Symbole (mit der Zwei als Häufigkeitswert) waren plötzlich die niedrigsten Zahlen in der Liste. Also werden sie an einen weiteren Knoten angehängt, der die Zahl 4 = 2+2 = (1+1)+(1+1) erhält.



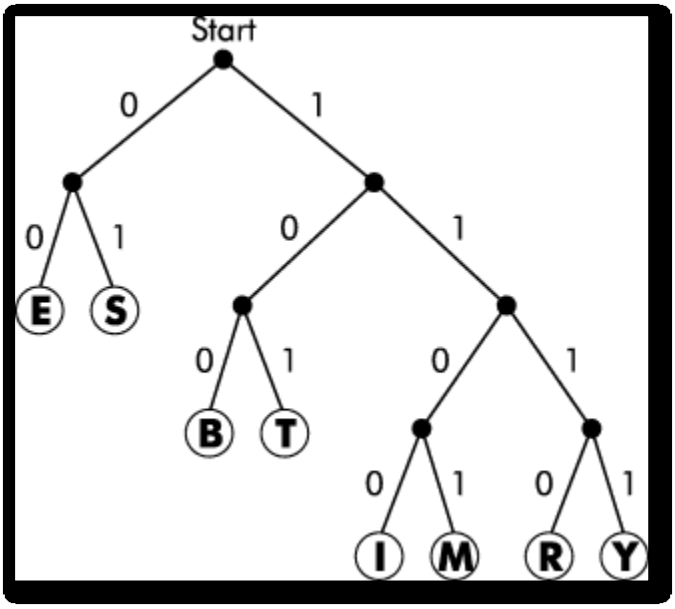
Die Symbole E und S sind wieder die niedrigsten Werte an der Basis gewesen, also wurden sie zusammengefasst.



Die beiden 4er-Knoten hatten den niedrigsten Wert, also wurden sie verknotet.



Jetzt ist der Baum fertig: Es gibt nur noch einen Knoten an der Basis des Baums, die so genannte Wurzel. Alle anderen Symbole hängen darunter. Zwischen den einzelnen Knoten existieren ja nun Verknüpfungen. Gibt man nun jeder Verknüpfung, die nach links weist, einen Wert 0 und jeder nach rechts weisenden Verknüpfung den Wert 1, erhält man einen Baum, der folgendermassen aussieht:



4.2 Verlustfreie Kompression

Bei der verlustfreien Kompression geht keine Information verloren.

Die Daten werden nur anders als vorher organisiert, indem bestimmte Redundanzen erkannt und zusammengefasst werden.

Zum Beispiel können sich wiederholende Bitfolgen einmal in einem Wörterbuch abgelegt und dann nur noch durch ihre Nummer repräsentiert werden.

Bekannte Verfahren sind die Lauflängenkodierung, LZW oder die Huffman-Codierung.

Es können beliebige allgemeine Komprimierungsverfahren verwendet werden, die sich auch auf andere Arten von Daten wie Text anwenden lassen.

4.3 Verlustbehaftete Kompression

Bei der verlustbehafteten Kompression wird versucht, den Informationsverlust unmerklich oder wenigstens ästhetisch erträglich zu halten.

Diese Methoden nutzen aus, dass kleine Farbänderungen für das Auge nicht sichtbar sind.

Ähnlich wie bei der verlustbehafteten Audiokomprimierung basiert die Bildkomprimierung auf einem Modell der menschlichen Wahrnehmung.

Der Komprimierungsalgorithmus soll bevorzugt die Bildinformationen entfernen, die über die Aufnahmefähigkeit der menschlichen Bildwahrnehmung hinausgehen.

Das Wahrnehmungsmodell ist jedoch, im Gegensatz zur Audiokompression, nicht explizit formuliert und in die Algorithmen eingearbeitet, sondern mehr intuitiv.

5. Verschlüsselungsverfahren

5.1 Public Key Verfahren

Asymmetrische Verschlüsselung

- Alice und Bob einigen sich auf ein Kryptosystem
- Alice bekommt Bob‘s Public Key
    - Von Bob direkt
    - Schlüsseldatenbank
- Alice verschlüsselt die Nachricht mit Bob‘s Public
- Key.
- Alice sendet die verschlüsselte Nachricht an Bob.
- Bob entschlüsselt sie mit mit seinem private Key.

5.2 Verschlüsselung durch Substitution

5.2.1 polyalphabetische Substitution

Dies ist ein Sonderfall der einfachen monoalphabetischen Substitution, wobei das zur Verschlüsselung verwendete Alphabet durch zyklisches Verschieben jedes einzelnen Buchstabens des Standardalphabets gewonnen wird.

Die Anzahl der Plätze, um die verschoben wird, ist der Schlüssel.

Schon Caesar benutzte dieses Verfahren, zumeist mit dem Schlüssel „C“, was einer Verschiebung um drei Buchstaben entspricht.

Beispiel für die Caesar-Verschlüsselung:

Klartext	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	Z
Geheimtext	D	E	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Bei diesem Beispiel wird das Wort „wikipedia“ als „ZLNLSHGLD“ verschlüsselt.

5.2.2 monoalphabetische Substitution

Im Gegensatz zur monoalphabetischen Substitution werden für die Zeichen des Klartextes mehrere Geheimtextalphabete verwendet.

6. Glossar

Begriff	Beschreibung
ECC	ECC = Error Correcting Code = Fehlerkorrekturverfahren Fehlerkorrekturverfahren dient dazu, Fehler bei der Speicherung und beim Übertragen von Daten zu erkennen und wenn möglich zu korrigieren. Für diese Überwachung werden zusätzliche Bits hinzugefügt.
Hammingdistanz	Ist der Bereich von den niedrigsten Anzahlen an 1 in einer Codewertigkeit bis zu höchsten Anzahl an 1 in einer Codewertigkeit.
Redundanz	Die Redundanz zeigt auf wieviele Stellen theoretisch für den Code von nöten wäre oder wieviel dieser höchsten braucht.
AIKEN-Code	Der AIKEN-Code ist eine Art Komplement des BCD-Code. Die Zahlen von 0 bis 4 im binären werden für die Zahlen 5 bis 9 spiegelbildlich abgebildet; 1=0001 => 8=1110
EAN, EAN128, EAN13	EAN = European Article Numbering = Europäische Artikel Nummerierung Das ist ein Europaweit einheitlicher Strichcode. Die richtige Bezeichnung lautet jedoch International Article Numbering.
ASCII	ASCII = American Standard Code for Information Interchange Dieser Code ist eine 7-Bit Zeichencodierung
ANSI, ISO, DIN	ANSI = American National Standards Institute ISO = International Organization for Standardization DIN = Deutsches Institut für Normung ANSI stellt die Normierung in Amerika für industrielle Verfahrensweise sicher und ist Mitglied von ISO. Das Gegenstück ist der DIN.
Unicode	Der Unicode vereint alle sinntragenden Schriftzeich und Textelement aller Schriftkulturen undZeichensysteme in einem digitalen Code.
Dualcode	Besser bekannt als Binärsystem ist ein Zahlencode der die Zahlen auf Basis 2 darstellt, optimal für digitale Geräte.
Graycode	Beim Gray-Code ändern sich die Codewörter zu Codewörter um immer genau eine Stelle. Hamming-Distanz = 1
FEC, resp. EDAC	FEC = forward error correction = Vorwärtsfehlerkorrektur Senkt die Fehlerrate beim Speichern und beim Übertragen von digitalen Daten. Der Sender kodiert die zu übertragenden Daten in redundanter Weise, so dass der Empfänger Übertragungsfehler ohne Rückfrage beim Sender erkennen und korrigieren kann.
Paritätsbit	Das Paritätsbit wird je nach dem ob die Parität ungerade oder gerade sein soll hinzugefügt oder nicht hinzugefügt und ermöglich so fehlerhaft übertragene Daten.
BCD-Code	BCD = Binary Coded Decimal Ist ein 8-4-2-1-Code und stellt die Dezimalzahlen von 0-9 Dual in einem 4 Bit Code (1 Nibble) dar.
1-aus-N-Code	1-aus-N-Code = One-Hot-Kodierung Dieser Code stellt die Dezimalzahlen von 0-9 in einem 10 Bit Code dar. Jeder Wert ist ein Bit mit dem Stellenwert der Zahl. Der Code weisst eine hohe Redundanz auf, dafür besitzt er eine niedrige Hamming-Distanz.