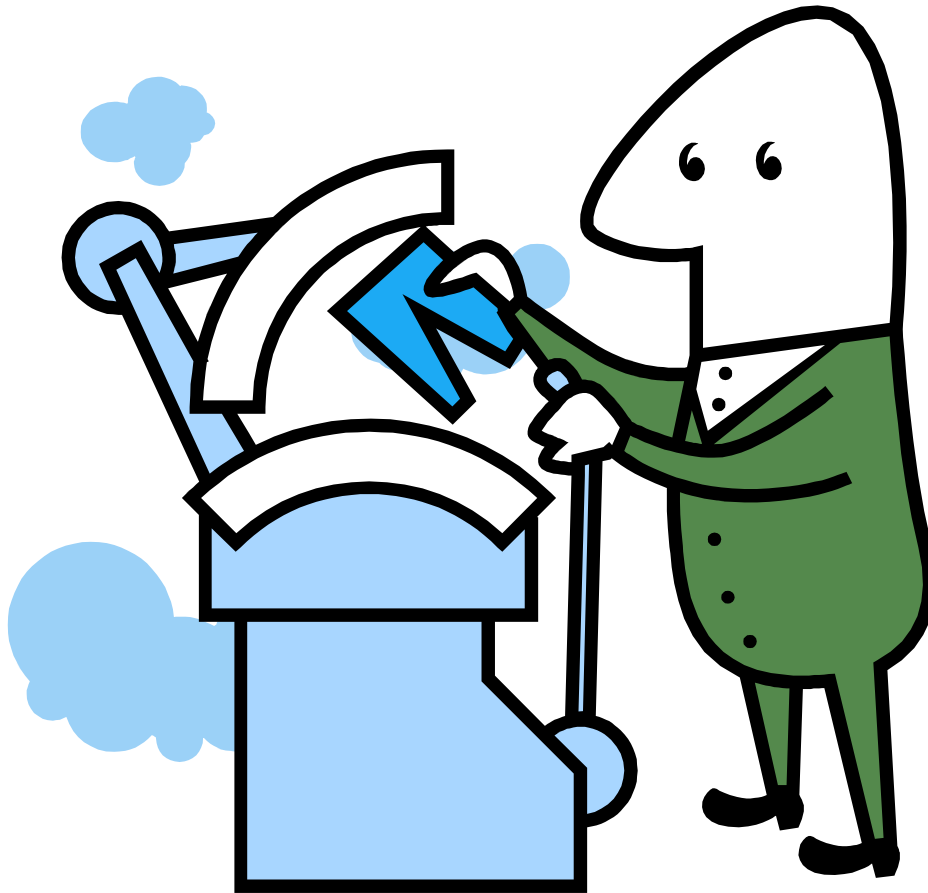


Datenkompression



Datenkompression I



- Wie der Hofmathematiker herausgefunden hatte, läßt sich eine Kodierung durch Wahl einer variablen Wortlänge reduzieren (optimaler Code).
- Dieser optimale Code gilt nur für eine ganz bestimmte Häufigkeitsverteilung der Codeworte (vgl. Morsecode).
- Es wurden verschiedenen Verfahren zur Datenreduktion und -kompression entwickelt.
- Eines der bekanntesten Verfahren ist die Huffman-Kodierung. Sie generiert anhand der Häufigkeiten einen optimalen Code.



- Grundsätzlich werden zwei Arten von Datenkompression unterschieden:
 - die verlustfreie und
 - die verlustbehaftete.
- Bei verlustfreier Datenkompression lassen sich die Daten nach der Kompression wieder Bit-genau dekomprimieren, so dass der Eingangs- und der Ausgangsdatenstrom identisch sind.
- Bei verlustbehafteter Komprimierung können die Daten bei der Dekompression nicht identisch, sondern nur annähernd oder nur in bestimmten Bereichen wieder hergestellt werden.

Huffman-Code I



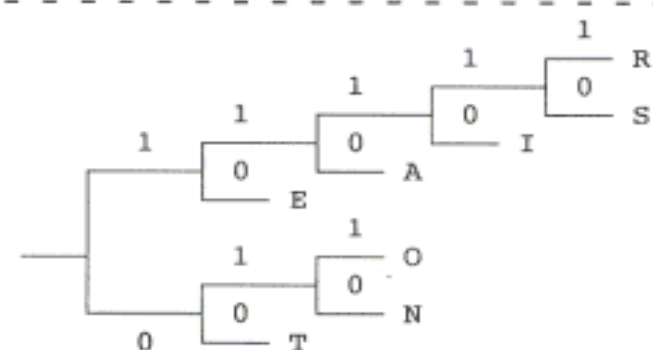
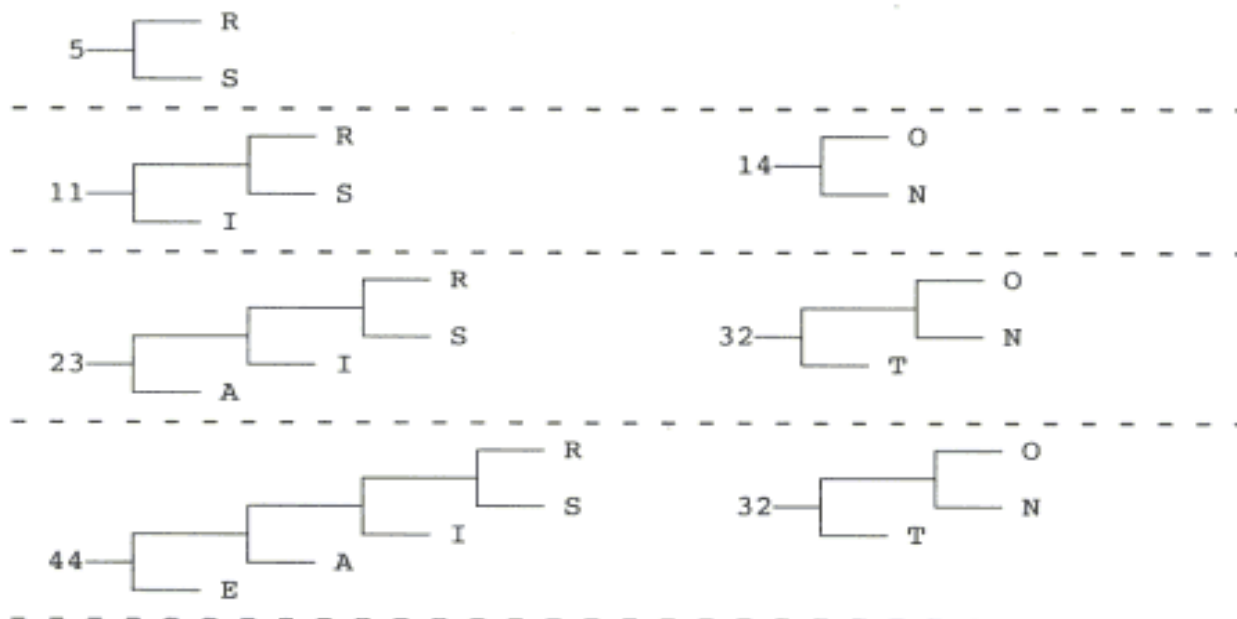
Die Huffman-Kodierung geht wie folgt vor:

- Suche aus der Menge aller Zeichen die beiden seltensten (geringste Häufigkeit)
- Bilde einen Teilbaum aus diesen Zeichen, weise den Ästen des Teilbaums die Werte 0 und 1 zu, der Wurzel die Summe der beiden Wahrscheinlichkeiten.
- Suche nun wiederum die beiden seltensten Teilbäume oder Zeichen und wiederhole die Gruppierung.
- Fahre fort, solange noch Teilbäume oder Zeichen existieren.

Huffman-Code II



E 21 T 18 A 12 O 8 N 6 I 6 R 3 S 2



z. B.: R = 11111
 I = 1110
 T = 00

Nachteile des Huffman-Code

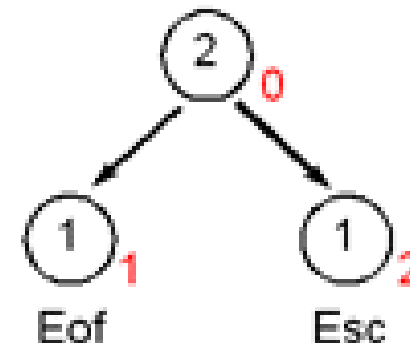


- Die Huffman-Kodierung hat einige schwerwiegende Nachteile:
 - Sie braucht zwei Durchläufe über die Eingangsdaten,
 - die Häufigkeitstabelle muß vom Encoder zum Decoder übertragen werden,
 - das Verfahren berücksichtigt nur die durchschnittlichen Wahrscheinlichkeiten des ganzen Eingabestroms, nicht lokale Besonderheiten.
- Diese Nachteile werden durch das Verfahren des sogenannten adaptiven Huffman-Codes ausgeglichen.

Adaptiver Huffman-Code I



- Kodierer und Dekodierer beginnen mit einem leeren Baum, der erst bei der Kodierung / Dekodierung aufgebaut und angepasst wird.
- Dem Dekodierer muss ein neu auftretendes Zeichen bekannt gemacht werden, damit dieser es in sein Modell einfügen kann.
- Daher wird vor einem neuen Zeichen ein NYT (not yet transmitted) gesendet und dann der unkomprimierte neue Character.
- Nahe der Wurzel des Baums stehen anfangs das NYT (aka Esc) sowie das 2. Sonderzeichen EOF (end-of-file):



Adaptiver Huffman-Code II



Die adaptive Huffman-Kodierung geht wie folgt vor:

- Die Knoten müssen gemäß der Häufigkeiten sortiert bleiben, sie werden zur Prüfung durchnummeriert (**rot**).
- Wenn das zu kodierende Zeichen noch nicht im Baum ist, sende NYT, gefolgt von dem neuen Zeichen.
- Der Knoten mit der höchsten Ordnungsnummer wird gesplittet, die „Kinder“ bekommen neue Nummern.
- Befindet sich das Zeichen bereits im Baum, sende den gegenwärtigen Code und passe die Häufigkeit des Zeichens an.
- Hänge ggfs. den angepassten Knoten im Baum um.

Lempel-Ziv-Algorithmus (LZ77) I



- Der LZ-Algorithmus benötigt zur Dekompression keine zusätzlichen Informationen über die Daten.
- Die Information über die Daten wird während des Dekodiervorganges gewonnen.
- Das ist der Hauptunterschied der LZ-Algorithmen gegenüber anderen verlustfreien Verfahren (z.B Huffman).
- Da nicht die gesamten Daten nach Mustern untersucht werden können, wird ein Datenfenster auf den Eingabestrom gelegt und dieses nach Übereinstimmungen durchsucht.

Lempel-Ziv-Algorithmus (LZ77) II



- Dieses Datenfenster wird im Datenstrom verschoben, so dass immer die zuletzt gelesenen Zeichen als Mustererkennung zur Verfügung stehen.
- Der Kompressor durchsucht das Datenfenster nach der längsten Zeichenkette, die mit der an der Kodierungsposition beginnenden Zeichenkette übereinstimmt.
- Kodiert wird dann ein Tupel aus (Position, Länge) und dem nächsten Zeichen nach der übereinstimmenden Zeichenkette im Vorausschaupuffer.
- Hat der Kompressor keine Übereinstimmung gefunden, gibt er (0,0) gefolgt von dem nächsten Zeichen aus.

Lempel-Ziv-Algorithmus (LZ77) III



Der LZ77 geht wie folgt vor:

- Setze die Kodierungsposition auf den Anfang des Eingabedatenstroms.
- Finde *rückwärts* die längste Übereinstimmung im Datenfenster mit der an der Kodierungsposition beginnenden Zeichenkette im Vorausschaupuffer.
- Gib das Tupel (Position, Länge) und das erste nicht passende Zeichen im Vorausschaupuffer aus.
- Ist das Ende des Eingabedatenstroms noch nicht erreicht, bewege die Kodierungsposition (Datenfenster) um $l+1$ Zeichen weiter (l bezeichnet die Länge der zuletzt gefundenen Zeichenkette), und fahre bei 2. fort.

Lempel-Ziv-Algorithmus (LZ77) IV



Kleinbuchstaben: Inhalt des Datenfensters

Großbuchstaben: Inhalt des Vorausschaupuffers

Ausgabe: (Position,Länge)Zeichen

Position	0	1	2	3	4	5	6	7	8
Eingabedatenstrom	A	A	B	C	A	A	A	B	C

Kodierungsvorgang

Kodierungspos.	Datenfenster:Vorschaupuffer	Pos.:Übereinstimmung	Ausgabe
0	:AABCAAABC	-	(0,0)A
1	a:ABCAAABC	1:A	(1,1)B
3	aab:CAAABC	-	(0,0)C
4	aabc:AAABC	4:AA	(4,2)A
7	aabcaaa:BC	5:BC	(5,1)C

Lempel-Ziv-Algorithmus (LZ77) V



Die Dekodierung geschieht folgendermaßen:

Eingabedatenstrom (0,0)A (1,1)B (0,0)C (4,2)A (5,1)C

Dekodierungsvorgang

Eingabe	Datenfenster:Ausgabe
(0,0)A	-:A
(1,1)B	a:AB
(0,0)C	aab:C
(4,2)A	aabc:AAA
(5,1)C	aabcaaa:BC

Beispiel LZ77



Nr.	Suchpuffer	Vorschau-puffer	Kodierung
1		EINE_MAUΣ_	(0, 0, "E")
2	E	INE_MAUΣ_B	(0, 0, "I")
3	EI	NE_MAUΣ_BA	(0, 0, "N")
4	EIN	E_MAUΣ_BAU	(3, 1, "_")
5	EINE_	MAUΣ_BAUT_	(0, 0, "M")
6	EINE_M	AUΣ_BAUT_E	(0, 0, "A")
7	EINE_MA	UΣ_BAUT_EI	(0, 0, "U")
8	EINE_MAU	S_BAUT_EIN	(0, 0, "S")
9	EINE_MAUΣ	_BAUT_EIN_	(5, 1, "B")
10	EINE_MAUΣ_B	AUT_EIN_HA	(5, 2, "T")
11	EINE_MAUΣ_BAUT	_EIN_HAUΣ.	(5, 1, "E")
12	EINE_MAUΣ_BAUT_E	IN_HAUΣ.	(15, 2, "_")
13	EINE_MAUΣ_BAUT_EIN_	HAUΣ.	(0, 0, "H")
14	EINE_MAUΣ_BAUT_EIN_H	AUΣ.	(14, 3, ".")
15	_MAUΣ_BAUT_EIN_HAUΣ.	>(14, 3, ".")	



- LZ78 und LZW arbeiten mit einer „Phrasentabelle“.
- Die Phrasentabelle wird während der Codierung erweitert.
- Wiederholungen der Phrasen werden nur noch als Zeiger auf die Position in der Phrasentabelle ausgegeben.
- Ggf. werden erweiterte Phrasen zusätzlich abgelegt.
- LZW ist im Gegensatz zu LZ78 so angelegt, dass keine Phrasen aus einzelnen Zeichen bestehen.

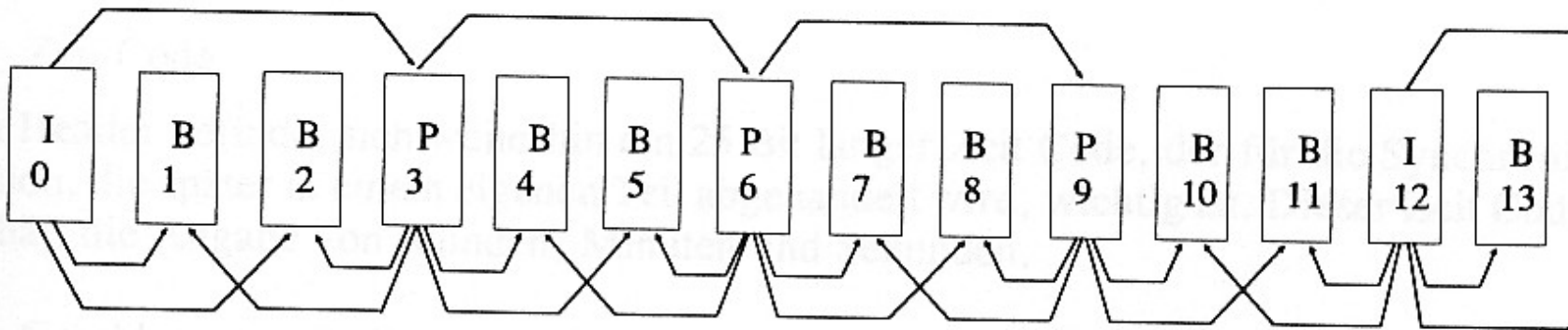


- Ein Bewegtbildfilm besteht aus einer Reihe von Einzelbildern (Pictures).
- Im MPEG Standard gibt es vier verschiedenen Arten von Bildern:
 - I Pictures (Intra Pictures), bei denen das gesamte Bild für sich ohne Informationen von anderen Bildern kodiert wird.
 - P Pictures (Predicted Pictures), die zur Kodierung die Information der vorhergehenden I oder P Pictures verwenden. Kodiert werden nur die Veränderungen hinsichtlich des letzten Bildes.
 - B Pictures (Bidirectionally Predicted Pictures) werden kodiert, indem Informationen vom letzten und/oder nächsten I oder P Picture zur Kodierung verwendet werden.

MPEG Kodierung II



- D Pictures, die nur bei der Dekodierung für die schnelle Suche vorwärts genutzt werden.
- Unten zu sehen ist eine typische Anordnung der Bildtypen in einer Group of Pictures (GOP) und ihre Abhängigkeiten.



MPEG Kodierung III



- In MPEG unterscheidet man die Wiedergabeordnung (wie im letzten Bild) und die Datenstromordnung, die die untereinander bestehenden Beziehungen berücksichtigt.
- Wenn der Dekodierer ein B Picture dekodieren will, muss er zunächst das dahinterliegende P oder I Picture erzeugt werden.
- Der Datenstrom in der Datenstromordnung sieht also folgendermaßen aus:

I	P	B	B	P	B	B	P	B	B	I	B	B
0	3	1	2	6	4	5	9	7	8	12	10	11

Komprimierung in MPEG



- Video ist ein Folge von Einzelbildern, ein Einzelbild ein zweidimensionales Feld von Pixeln.
- Videokomprimierung geschieht in zwei Ebenen:
 - die Komprimierung der Bildinformation der Einzelbilder,
 - eine Komprimierung basierend auf der Abfolge der Bilder.
- Bei der MPEG Videokomprimierung werden verschiedene Komprimierungstechniken kombiniert eingesetzt, die folgend vorgestellt werden

Zusammenfassen von Farbinformation



- Es handelt sich um eine sehr gebräuchliche Art der Video-komprimierung.
- Die Information über die Helligkeit eines Pixels wird für jedes Pixel abgespeichert.
- Die Farbinformation und die Farbsättigung werden nur einmal für jeweils vier Pixel gespeichert.
- Da Menschen Helligkeit stärker wahrnehmen als Farbe und Farbsättigung, beeinträchtigt die Zusammenfassung die Wahrnehmung kaum.
- Diese unwesentliche Verschlechterung der Qualität wird für eine Komprimierungsrate von 1:4 bei Farbton und Sättigung in Kauf genommen.

Interpolation / Variable Längenkodierung



- Bei der Bildinterpolation werden die Bildpunkte eines Bildes nicht übertragen, sondern aus denen des vorigen und nächsten Bildes bestimmt.
- Dazu werden die Werte dieser Pixel jeweils gemittelt.
- Diese Technik wird innerhalb der Motion Compensation angewendet.
- Die variable Längenkodierung ist die schon bekannte Kodierung nach Häufigkeit.

Laufängen-Kodierung (RLE)



- Das PCX-Bildformat wendet die Laufängen-Kodierung an.
- Dabei werden aufeinander folgende gleiche Farbwerte in der Weise kodiert, dass der Farbwert und die Anzahl der Wiederholungen gespeichert wird.
- Kodiert wird ein Bild immer zeilenweise.
- Bei grafischen, flächigen Bildern (besonders schwarz/weiß) ist diese Kodierung sehr effektiv (Bsp.: Fax).
- Die Folge 00000001110000011111111111011111 kann beschrieben werden:
7 x 0, 3 x 1, 5 x 0, 11 x 1, 1 x 0, 4 x 1 oder
7 x 0, 3, 5, 11, 1, 4

Quantisierung I



- Quantisierung bedeutet allgemein, dass für einen Wertebereich immer nur ein einzelner Wert steht.
- Es steht nicht von vornherein fest, wie hoch der Komprimierungsfaktor ist. Auch die Qualitätseinbuße ist verschieden.
- Beide Faktoren hängen von der Wahl der Quantisierungsschritte und den Eingabedaten ab.
- Beispiele für Quantisierung:
 - Jede reelle Zahl wird durch die ihr am nächsten liegende ganze Zahl abgebildet.
 - Auch Farbscanner bilden die an sich unendlich vielen Farben auf eine Palette ab (256, 262 T, ca. 18 Mio)

Quantisierung II



- Quantisierung wird bei allen Picture Typen in MPEG angewandt.
- Je größer die Quantisierungsschritte gewählt werden, desto kleiner wird die Datenmenge des kodierten Bildes, umso schlechter wird aber auch die Qualität.
- Eine variable Aufteilung der Quantisierungsschritte auf die Bilder ist die beste Möglichkeit:
 - Gibt es nur wenig Bewegung im Video werden die I Pictures mit vielen Bits (kleinen Quantisierungsschritten) kodiert, die P und B Pictures mit wenigen.
 - Bei viel Bewegung entfallen auf I Pictures weniger Bits, dafür auf P und B Pictures mehr.

Motion Compensation I



- Ein wesentlicher Teil der Motion Compensation ist die Technik des sogenannten vorhersagenden Kodierens.
- Dabei sagen sowohl Kodierer als auch Dekodierer, basierend auf den vorher kodierten Pixeln, für einen Block die Werte der Pixel voraus, die jetzt kodiert werden sollen.
- Der Kodierer vergleicht die vorausgesagten Werte mit den tatsächlichen Werten der Pixel und kodiert dann die Differenz.
- Der Dekodierer benutzt diese Differenz, um seine eigene Voraussage zu korrigieren.

Motion Compensation II



- Wie gut die Kompressionsrate dabei ist, liegt vor allem an der Anzahl der Schnitte und Zooms, die in einem Video vorkommen.
- Ist Bewegung in einem Film, bedingt das, dass ein Makroblock eines Bildes im nächsten eine andere Position haben wird.
- Die Verschiebung eines Makroblocks wird als Bewegungsvektor (zweidimensionaler Vektor) angegeben.
- Die Ermittlung des Bewegungsvektors und das vorher-sagende Kodieren machen beide zusammen die Technik der Motion Compensation aus.