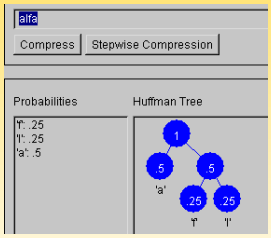
Algorithmierung/Programmierung

Datenkomprimierung



1.1 Datenkompression

aus Wikipedia, der freien Enzyklopädie

Datenkompression ist ein technisches Verfahren zur systematischen Verminderung der Menge an Daten, die notwendig ist, um einen gegebenen Inhalt (einen Text, eine Bilddatei, etc.) in einer computerlesbaren Form wiederzugeben. Dazu wird ein originaler Quellentext, ein Quellbild oder eine andere Computerdatei mit Hilfe eines Kompressionsverfahrens untersucht und in eine komprimierte, d.h. verkürzte Form umgeschrieben (***gepackt***). Mit Hilfe eines zu dem gewählten Verfahren passenden Leseverfahrens wird die verkürzte Fassung des Textes oder der Daten wieder in die ursprüngliche, längere zurückverwandelt (***entpackt).*** Dabei wird unterschieden, ob die ursprüngliche Information letztlich wortgetreu (***verlustfrei***) oder auch mit gewissen Änderungen (***verlustbehaftet***) rekonstruiert wird (***Datenreduktion***). Texte sollen meist wortgetreu, Bilder häufig nur detail- und farbgetreu verarbeitet werden.

Kompression:

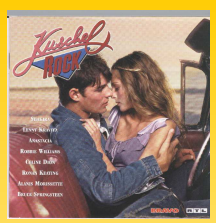
LANGE-ORIGINAL-QUELLDATEI (->Komprimierprogramm->) KOMPR.DATEI

Dekompression:

KOMPR.DATEI (->Entkomprimierprogramm->) LANGE-ORIGINAL-QUELLDATEI

1.2 Wann wird komprimiert ?

- Text



- Grafik

Größe: 11,9 MB (12.493.312 Bytes)

Dateiformat des Grafikprogramms (PhotoImpact)

- jpeg

Größe: 1,76 MB (1.852.842 Bytes)

Größe: 165 KB (169.035 Bytes)

Größe: 14,0 KB (14.409 Bytes)

•Audio

CD1 783 MB → mp3 71,4 MB

•Video

http://www.hc-leipzig.de

Video 08: Der letzte Wurf von HYPO und dann der grenzenlose Jubel der Spielerinnen! (14 sec / 1,1 MB)

•Video


Pinnacle Studio - Unbenannt *
Datei Bearbeiten Ansicht Film erstellen Toolbox Setup Hilfe
1 Aufnahme
2 Bearbeiten
3 Film erstellen
STUDIO
Band
AVI
MPEG
WebVideo
Upload
Status
Das Projekt erfordert einige Rendering-Operationen.
Bitte aktivieren Sie die Schaltfläche unten, um zu beginnen.
AVI...
Laufwerk (D:
Frei: 5765.5 MB
Belegt: 4465.9 MB
Geschätzte Größe
1240.4 MB
Video: DV Video Encoder, 720x576, 25
Audio: PCM, 16-Bit (Stereo) 48 kHz
Einstellungen
Nichts zu übertragen
Öffnet AVI...
0:00:00.00
Unbenannt *

•Video


Pinnacle Studio - Unbenannt *
Datei Bearbeiten Ansicht Film erstellen Toolbox Setup Hilfe
1 Aufnahme
2 Bearbeiten
3 Film erstellen
STUDIO
Band
AVI
MPEG
WebVideo
Upload
Status
Das Projekt erfordert einige Rendering-Operationen.
Bitte aktivieren Sie die Schaltfläche unten, um zu beginnen.
MPEG...
Laufwerk (I:)
Frei: 12385 MB
Belegt: 920 MB
Geschätzte Größe
103.3 MB
Video: VHS 352 x 288, 2400 KBit/Sek.
Audio: WAV, Stereo, 44,1 kHz
Einstellungen
0:00:00.00
Unbenannt *

1.3 Warum wird komprimiert ?

Gründe, um Daten zu komprimieren:

- Daten platzsparend archivieren
- Daten möglichst schnell übertragen, Übertragung ermöglichen (Telekommunikation)
- Unterhaltungselektronik, Multimedia enormer Speicherbedarf von unkomprimierten Grafiken, Audios und Videos
  - ⇒ Platz auf Massenspeichern sparen
  - ⇒ Übertragungszeiten verkürzen
  - ⇒ digitale Übermittlung von Datenströmen ökonomisch gestalten

Grundlage vieler dieser fortgeschrittenen Komprimierungstechniken ist die Textkomprimierung. Die wichtigsten Algorithmen dazu sollen in diesem Vortrag vorgestellt werden.

1.4 Das Prinzip

So funktioniert die Komprimierung:

Mathematische Formeln etwa ausgenommen, enthält jegliche Form von Information - Texte , Bilder, Musik Wiederholungen und Redundanzen(eine »unwichtige« oder »überflüssige« Information, die aus dem Kontext hervorgeht). Ein einfaches Beispiel: In der deutschen Sprache kommen häufig Wörter vor wie "und ", "die", "ein" oder Zeichenfolgen wie „"ung","sch","ver".

Am Computer bietet sich die Möglichkeit, zum Teil erhebliche Datenmengen einzusparen, indem häufig wiederkehrende Zeichenfolgen jeweils durch ein Zeichen kodiert werden.
Da Packprogramme zu diesem Zweck unterschiedliche Algorithmen verwenden, unterscheiden sie sich hinsichtlich Kompressionszeit und Packdichte.

Algorithmen zur Datenkompression kann man in zwei Klassen einteilen:

verlustfrei *(lossless)* ⇔ verlustbehaftet *(lossy)*

1.5 Verlustfreie Datenkompression

Verlustfreie Datenkompression erlaubt die exakte Wiederherstellung der ursprünglichen Eingabe. Hauptanwendungsgebiet ist für Texte, Programme und andere Daten, bei denen selbst ein falsches Bit das gesamte File unbrauchbar machen kann.Kompressionsfaktoren liegen für gewöhnlich zwischen 0.9 und 0.2.

- Anwendung Faxübertragung, Grundlage von UNIX compress, gzip, winzip, Einsatz in jpeg in Verbindung mit verlustbehafteten Verfahren

1.6 Verlustbehaftete Datenkompression

Verlustbehaftete Datenkompression kann die ursprüngliche Eingabe nur näherungsweise wiederherstellen. Durch den Kunstgriff unwichtige Details unter den Tisch fallen zu lassen, sind traumhafte Kompressionsraten erreichbar.Allerdings geht dies zu Lasten der Qualität.

- Anwendungsgebiete sind hauptsächlich Bilder, Musik und Filme.Es werden Daten weggelassen, die für den Menschen nicht wahrnehmbar sind.

2.1 Run Length Encoding - RLE

Hinter dem Verfahren der **Lauflängenkodierung** steckt folgende einfache Idee: Wenn das Symbol"s" **n**-mal nacheinander im Eingabestrom auftritt, so ist es zweckmäßiger, die **n**-malige Wiederholung des Zeichens durch den Ausdruck n"s" zu ersetzen. Anwendungen finden sich sowohl im Bereich der Text- als auch im Bereich der Bildkomprimierung.

RLE basiert auf der Idee, sich wiederholende Symbole durch ein einzelnes Symbol und die Anzahl der Wiederholungen darzustellen.

aufeinanderfolgende Vorkommen einer Dateneinheit ***d*** =***run*** (Lauf)

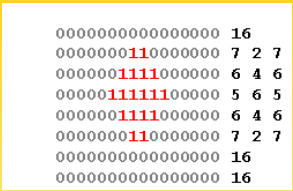
Anzahl der Wiederholungen ***n*** von ***d = run length*** (Lauflänge)

Beispiel:

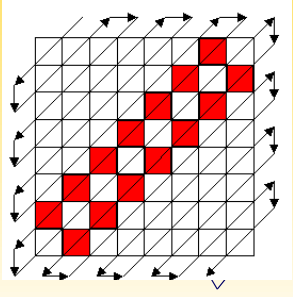
AAAABBBAABBBBBBCCCCCCCC

4A 3B AA 6B 8C

Dabei lohnt es sich nicht Läufe der Länge eins oder zwei zu kodieren, da für die Kodierung zwei Zeichen benötigt werden.



Bei Binärdateien ist sicher, dass ein "run" von Nullen stets von einem "run" Einsen gefolgt wird. Es genügt deshalb nur die Länge der "runs" zu speichern.



Bei der Anwendung der Lauflängenkodierung auf Bilddaten, ist oftmals die Ausleserichtung entscheidend für den erreichbaren Kompressionsgrad.

In vielen Fällen hat die Zickzack-Variante Vorteile gegenüber dem zeilen – oder spaltenweisen Auslesen der Daten.

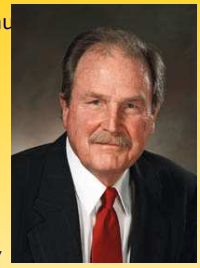
Die beiden bisher vorgestellten Verfahren eignen sich nicht zur Darstellung von Zeichenreihen , die aus einer Mischung von Buchstaben und Ziffern bestehen, da der Decoder nicht entscheiden könnte, ob es sich um eine Längenangabe oder ein Zeichen des Textes handelt.
Deshalb wurde eine Möglichkeit zur Codierung möglicher Folgen von Zeichen aus einem festen Alphabet entwickelt , bei dem nur die Zeichen dieses Alphabets verwendet werden.Nimmt man dabei ein reines Buchstaben – Alphabet, können auch nur Buchstaben zur Längenangabe genutzt werden.
Das i-te Zeichen des Alphabets wird dazu benutzt, um die Zahl i darzustellen.Um die Angaben der Buchstaben zu unterscheiden bedient man sich sogenannter Escape –Zeichen, d. h. ein bestimmtes Zeichen, das nur selten in der Datei vorkommt, wird vor jede Sequenz aus Zähler und sich wiederholendem Zeichen gestellt. Für das deutsche Alphabet wäre z.Bsp. das Q gut geeignet.
Da die Escape –Sequenz selbst drei Zeichen lang ist, lohnt sich die Verschlüsselung erst ab einer Folge von mehr als drei Zeichen. Sollte Q selbst verschlüsselt werden, implementiert man eine spezielle Sequenz, die zum Beispiel aus Q<Leerzeichen>bestehen kann.

Beispiel: **AAAABBBAABBBBBBCCCCCCCC**

QDA BBBAA QFB QHC

RLE ergibt nur bei speziellen Eingaben eine gute Kompression. Bei Font Bitmaps mit vielen Widerholungen eignet sich RLE sehr gut.
Problematisch ist die hohe Fehleranfälligkeit (Fehler in einem run).

2.3 Huffman Codierung



David A. Huffman

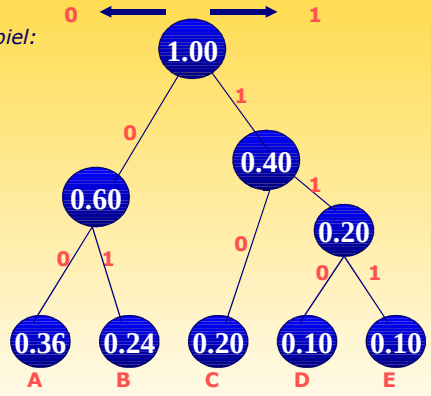
Bei diesem von David A.Huffman 1952 veröffentlichten Algorithmu handelt es sich um eines der bekanntesten und am weitesten verbreiteten Verfahren zur Datenkompression.
Der Algorithmus erzeugt einen optimalen Präfixcode variabler Länge, der durch einen binären Baum dargestellt wird.

Als Voraussetzung benötigt der Standard – Algorithmus die Häufigkeitsverteilung der verschiedenen Zeichen in der zu codierenden Datei.Dazu bedient man sich entweder einer vorgefertigten Statistik **(statisch)** zur gegebenen Sprache oder man implementiert einen Durchlauf über die zu codierende Datei, wobei die vorkommenden Zeichen zusammen mit der Häufigkeit ermittelt werden**(dynamisch).** Eine dritte Möglichkeit wäre **adaptierend:**
Es wird mit festen Vorgaben begonnen (z.B.: alle Zeichen treten gleich oft auf, oder das ***e*** ist das häufigste Zeichen, usw..), die während der Codierung an die realen Daten angepasst werden.

Nun kann der Baum, der den Code repräsentiert und dessen Blätter die Zeichen des Alphabets sind, erstellt werden.

1. Füge alle Symbole mit ihrer Wahrscheinlichkeit als Gewicht in einen Wald ein.
2. Kombiniere die beiden Bäume mit kleinstem Gewicht zu einem neuen Baum dessen Gewicht die Summe der beiden Bäume ist.
3. Verfahre rekursiv bis nur noch ein Baum übrig ist.
4. Schreibe 0 bzw. 1 an den linken bzw. rechten Ast jedes Knotens.

Beispiel:



A	00
B	01
C	10
D	110
E	111