Kodierungsalgorithmen

- Komprimierung
- Verschlüsselung

Komprimierung

Zielsetzung:

- **Reduktion der Speicherkapazität**
- **Schnellere Übertragung**

Prinzipien:

- **Wiederholungen in den Eingabedaten kompakter speichern**
- **Kodierung von Zeichen entsprechend der Häufigkeit des Auftretens**

Begriffe:

- **Encoder/Compressor: Programm zur Komprimierung**
- **Decoder/Decompressor: Programm zur Rekonstruktion der Originaldaten**
- **Rohdaten: Originaldaten**
- **Verlustfreie Komprimierung**
- **Verlustbehaftete Komprimierung**
- **Nichtadaptive Komprimierung: unabhängig von den Eingabedaten**
- **Adaptive Komprimierung**
- **Semiadaptive Komprimierung: Vorlauf zur Erzeugung von Statistiken über die Eingabedaten**
- **Symmetrische Komprimierung: Prinzipiell gleiche Algorithmen zur Kodierung und Dekodierung**
- **Asymmetrische Komprimierung: aufwendige Kodierung (selten), wenig aufwendige Dekodierung (oft)**
- **Compression ratio: Ausgabegröße/Eingabegröße**

- **Compression factor: Eingabegröße/Ausgabegröße**

RLE (Run Length Encoding):

N Wiederholungen einen Zeichens werden durch eine Sequenz

Escape Character – N – Zeichen

ersetzt. (Escape Character ist natürlich geeignet zu kodieren!)

Bei einer durchschnittlichen Lauflänge von L (und damit einer Laufanzahl M=Anzahl der Zeichen / L) bringt RLE eine Einsparung von M*L – M*3.

RLE wird häufig für Binärdateien verwendet, bei denen nur Folgen von 0 oder 1 als Läufe auftreten und es daher genügt, das erste Zeichen und die Lauflängen zu speichern. Allerdings ist zu beachten, dass dieses Verfahren ineffizient werden kann, wenn die Lauflängen ungeeignet gespeichert werden (z.B. Speicherung als Integer!)

RLE wird allerdings i.a. nicht allein angewandt, sondern in Verbindung mit anderen Verfahren!

Huffmann Codierung

Huffmann Codierung basiert auf dem Prinzip, einen statisch optimierten binären (Selektor-) Baum zu konstruieren, der sämtliche Zeichen enthält und für den die gewichtete Pfadlänge (Summe über Häufigkeit der Zeichen * Pfadlänge der Zeichen) möglichst gering ist.

Algorithmus:

1. **Generiere für jedes Zeichen einen Blattknoten, in dem das Zeichen und die relative Häufigkeit eingetragen werden.**
2. **Solange noch zwei unbearbeitete Knoten vorhanden sind, führe Schritte 3 bis 5 aus:**
3. **Bestimme zwei unbearbeitete Knoten mit minimaler relativer Häufigkeit und berechne die Summe.**
4. **Erzeuge einen neuen Knoten mit der Summe als relativer Häufigkeit und den beiden Knoten als Nachfolgern.**
5. **Markiere die Nachfolger als bearbeitet und den neuen Knoten als nicht bearbeitet.**
6. **Der Huffmann Code eines Zeichens ergibt sich durch Verfolgung des Pfads im konstruierten Baum, wobei eine Verzweigung in den linken Teilbaum als 0 und eine Verzweigung in den rechten Teilbaum als 1 dargestellt wird.**

Huffmann Codes sind i.a. nicht eindeutig!

Zur Dekodierung muß die Tabelle der Huffmann Codes mitübertragen werden, es sei denn, sie wird universell eingesetzt.

LZ77:

Das von Lempel und Ziv entwickelte Verfahren basiert auf der Idee, Wiederholungen in den Eingabedaten zu finden und in der Ausgabe in der Form

Position, Länge der Wiederholung

darzustellen. Um den Fall fehlender Wiederholungen korrekt zu behandeln, wird zusammen mit der jeweiligen Wiederholung das nächste Zeichen mit ausgegeben.

Algorithmus:

1. **Lies das erste Zeichen der Eingabe.**
2. **Solange das Ende der Eingabe nicht erreicht ist, führe die folgenden Schritte durch:**
3. **Beginnend mit dem aktuellen Eingabezeichen, suche den längsten Treffer in der bisherigen Eingabe.**
4. **Ausgegeben werden die Position und Länge des Treffers (jeweils 0 für nicht gefunden) und das erste Zeichen nach dem Treffer.**
5. **Lies das nächste Zeichen der Eingabe.**

Zur Reduktion der Speicher- und Verarbeitungsanforderungen wird in Schritt 3 im allgemeinen nicht die gesamte bisherige Eingabe betrachtet, sondern ein Fenster der Größe W über die bisherige Eingabe gelegt, so dass nur die letzten W Zeichen der bisherigen Eingabe betrachtet werden.

Der Kodierungsaufwand bei LZ77 ist sehr hoch, die Dekodierung ist jedoch sehr einfach und effizient!

LZ78 / LZW

Das von Lempel und Ziv entwickelte und von Welsh optimierte Verfahren basiert auf der gleichen Grundidee wie LZ77, allerdings werden die Wiederholungen in einem Dictionary abgelegt und bei der Kodierung anstelle von Verweisen in die bisherige Eingabe Verweise in das Dictionary gespeichert.

Algorithmus zur LZW Kodierung:

a. **Die Zeichen werden gemäß ihrem Zeichencode in dem Dictionary abgelegt (z.B. auf den Positionen 0 bis 255 für 8 bit – Zeichensätze).**

b. **Setze die Eingabe I auf das erste Eingabezeichen.**

c. **Solange das Ende der Eingabe nicht erreicht ist, führe die folgenden Schritte durch:**

d. **Lies das nächste Eingabezeichen x.**

e. **Solange Ix im Dictionary enthalten ist, führe Schritt 6 durch:**

f. **Setze I auf Ix, und lies das nächste Eingabezeichen x.**

g. **Gib die Position von I im Dictionary aus.**

h. **Füge Ix an der nächsten freien Position im Dictionary an.**

i. **Setze I auf x.**

Auf den ersten Blick erscheint LZW gegenüber LZ77 als nachteilig, da die zur Dekodierung das Dictionary notwendig erscheint. Dies ist jedoch nicht der Fall, das Dictionary kann bei der Dekodierung „on the fly“ aufgebaut werden.

Algorithmus zur LZW Dekodierung:

a. **Die Zeichen werden gemäß ihrem Zeichencode in dem Dictionary abgelegt (z.B. auf den Positionen 0 bis 255 für 8 bit – Zeichensätze).**

b. **Setze die Eingabe auf das erste Zeichen v.**

c. **Setze I auf den Dictionaryeintrag an Position v, und gib I aus.**

d. **Solange Zeichen vorhanden sind, führe die folgenden Schritte aus:**

e. **Lies ein Zeichen v.**

f. **Setze J auf den Dictionaryeintrag an Position v.**

g. **Ist J != null, so setze x auf das erste Zeichen von J, sonst auf das erste Zeichen von I.**

h. **Falls Ix nicht im Dictionary ist, füge Ix an der nächste freie Position im Dictionary an.**

i. **Falls J=null ist, setze J auf Ix.**

j. **Gib J aus.**

k. **Setze I auf J.**

Einsatz der Komprimierungsverfahren:

Compress	**LZW**
Gzip	**LZ77, Huffmann**
Gif	**LZW**
Jpeg	**DCT, Quantisierung, RLE, Huffmann**
Mpeg	**Frames, Verfahren für jpeg**

Verschlüsselung

Begriffe:

Kryptologie

- **Kryptographie: Entwicklung von Systemen für geheime Kommunikation**
- **Kryptoanalyse: Untersuchung von Methoden zur Entschlüsselung von solchen Systemen**

Kryptosystem:

- **Kombination aus Verschlüsselungssystem und Entschlüsselungssystem**

(Konkurrierende) Ziele:

Sicherheit des Verfahrens:

- **Unbefugter Empfänger soll die Nachricht nicht (mit vertretbarem Aufwand) entschlüsseln können**

Effizienz von Verschlüsselung und Entschlüsselung

Verfahren:

Cäsar Chiffre:

- **Jeder Buchstabe im Alphabet wird um eine feste Anzahl von Stellen verschoben.**

Tabellenkodierung:

- **Jedem Buchstaben im Alphabet wird eine Kodierung zugeordnet.**

Vigenere Chiffre (Vernam Chiffre):

- **Verwendung eines kurzen, sich wiederholenden Schlüssels, um für jedes Eingabezeichen die Verschiebung zu bestimmen**
- **Je länger der Schlüssel, desto sicherer das Verfahren**
- **Im Idealfall ist der Schlüssel so lang wie der Klartext**

Public Key Encryption

Prinzip:

- **Jeder Benutzer hat einen öffentlichen und einen privaten Schlüssel (public key, private key)**
- **Nachrichten werden verschüsselt mit dem öffentlichen Schlüssel des Empfängers.**
- **Nachrichten werden entschlüsselt mit dem privaten Schlüssel des Empfängers.**

Idee: Diffie, Hellmann 1976

Verfahren: RSA (Rivest, Shamir, Adleman) 1977

Grundidee von RSA:

Gegeben sind drei große Primzahlen x, y und s (der private Schlüssel). Der öffentliche Schlüssel p wird derart bestimmt, dass

p*s mod (x-1)*(y-1) =1

gilt.

Dann gilt

M^(p*s) mod N = M.

Mit

P(M)=M^p mod N (public key encryption)

und

S(C) = C^s mod N (private key decryption)

ergibt sich

S(P(M))=

S(M^p mod N) =

(M^p mod N)^s mod N =

M^(p*s) mod N = M.