Informatik-Propädeutikum

Dozentin: Dr. Claudia Ermel

Betreuer: Sepp Hartung, André Nichterlein, Clemens Hoffmann
Sekretariat: Christlinde Thielcke (TEL 509b)

TU Berlin
Institut für Softwaretechnik und Theoretische Informatik
Prof. Niedermeier
Fachgruppe Algorithmik und Komplexitätstheorie
http://www.akt.tu-berlin.de

Wintersemester 2013/2014

Gliederung

Information

Wikipedia: Information (lateinisch informare „bilden", "eine Form, Gestalt, Auskunft geben") ist der strukturelle Aspekt physikalischer Wechselwirkungen, der aktionsprägend auf komplexe Systeme wie Menschen oder auch Maschinen wirkt. Sie vermittelt einen Unterschied.

Information ist Gegenstand verschiedener Struktur- und Geisteswissenschaften, kann mathematischer, philosophischer oder empirischer (etwa soziologischer) Begriff sein.

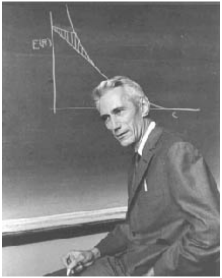
Beliebte Definition:
Informatik = Information + Automatik.
Bisher haben wir uns vorwiegend mit „Automatik", sprich Algorithmen beschäftigt. Nun zum Begriff der Information.

Wo **Daten** eher als das „Rohmaterial" der Informatik / Computer gesehen werden, da ist der Begriff der **Information** eher an den Anwendungskontext gebunden, sprich Information wird als „Rohmaterial" der jeweiligen Anwendung gesehen.

Information aus Informatiksicht

Was ist Information und wie kann man Informationsgehalt messen?

Shannon verknüpfte als erster den Informationsbegriff mit dem Begriff des *Zufalls*: Eine Binärzeichenkette aus lauter Nullen hat sehr geringen Informationsgehalt, eine völlig zufällige Binärzeichenkette aber sehr hohen Informationsgehalt (vergleiche Kolmogorov-Komplexität im vorigen Kapitel).
$\rightsquigarrow$ Begriff der **Entropie**, den Shannon 1948 einführte und damit zum Vater der Informationstheorie wurde.



Claude E. Shannon, 1916–2001, Mathematiker und Elektrotechniker

Nachfolgend gehen wir kurz ein auf

- Kodierungstheorie;
- Informationstheorie;
- Datenkomprimierung.

Kodierungstheorie

Wikipedia: Ein Code f über den Alphabeten A und B ist eine **injektive Abbildung** (= Kodierung) der Form $f\colon A \to B$. Er ordnet Wörtern aus Symbolen des Alphabets A Wörter aus dem Alphabet B zu. Als Code wird auch die Bildmenge von f, bezeichnet. Der Code heißt **entzifferbar**, wenn es eine eindeutige Umkehrabbildung f^{-1} gibt, die jedem Nachrichtenwort aus B wieder das ursprüngliche Wort aus A zuordnet.

Beispiele:

- Postleitzahlen.
- Binärkodierung. (Rückblick: Selbst Turing-Maschinen wurden binär kodiert, Gödelisierung).
- ASCII-Code.
- Morsecode.
- Strichcodes (QR-Codes) auf Produkten im Supermarkt.
- ...

Binärkodierung

Binärzahl mit k Stellen: $Z = z_k z_{k-1} \dots z_1$, $z_i \in \{0, 1\}$, z. B. 1011
Der Wert W von Z bzw. die Darstellung von Z im Dezimalsystem ist:

$$W = \sum_{i=1}^{k} z_i 2^{i-1}$$

Umrechnung von Dezimalzahlen in Binärzahlen durch wiederholtes Rechnen modulo 2:

$$\begin{array}{rcccll}
41 & :2 & = & 20 & \text{Rest} & \mathbf{1} \uparrow \\
20 & :2 & = & 10 & \text{Rest} & \mathbf{0} \\
10 & :2 & = & 5 & \text{Rest} & \mathbf{0} \\
5 & :2 & = & 2 & \text{Rest} & \mathbf{1} \\
2 & :2 & = & 1 & \text{Rest} & \mathbf{0} \\
1 & :2 & = & 0 & \text{Rest} & \mathbf{1}
\end{array}$$

$\rightsquigarrow [41]_{10} = [101001]_2$

Fehlerentdeckende Codes I

Problem: Bei der Übertragung von Informationen können Fehler auftreten (z.B. Bits werden „geflippt"). Wie erkennt man, ob die Übertragung korrekt war?
Lösung: Fehlerentdeckende Codes durch Ausnutzung von *Redundanz*.

Bei ISBN (International Standard Book Number) ist die letzte Ziffer der Zahl eine *Prüfziffer*.

Berechnung der Prüfziffer bei ISBN-10 (10 Stellen $b_1 b_2 \dots b_{10}$):

$$b_{10} = \left(\sum_{i=1}^{9} i \cdot b_i \right) \bmod 11$$

(Anstelle von $b_{10} = 10$ wird ggf. $b_{10} = X$ geschrieben.)
Beispiel: Die ISBN 3-411-05232-5 ist gültig:

$$(3 + 8 + 3 + 4 + 0 + 30 + 14 + 24 + 18) \bmod 11 = 104 \bmod 11 = 5$$

$\rightsquigarrow$ Wird eine ungültige ISBN empfangen, so kann dies **erkannt** werden! Aber der Fehler kann nur durch wiederholtes Übertragen behoben werden.

Fehlerentdeckende Codes II

Auch bei Kreditkartennummern gibt es eine Gültigkeitsprüfung:
Prüfungsvorschrift:

1. Ersetze Ziffern an ungerader Position durch folgende Ersetzung:
   $0 \mapsto 0,\ 1 \mapsto 2,\ 2 \mapsto 4,\ 3 \mapsto 6,\ 4 \mapsto 8,$
   $5 \mapsto 1,\ 6 \mapsto 3,\ 7 \mapsto 5,\ 8 \mapsto 7,\ 9 \mapsto 9$
2. Falls die Quersumme der neuen Zahl durch 10 teilbar ist, *könnte* die Nummer zulässig sein, ansonsten ist die Nummer ungültig.

Beispiel:
Kreditkartennummer: 4544 7000 1234 6789
$\overset{(1)}{\leadsto}$ 8584 5000 2264 3779
$\overset{(2)}{\leadsto}$ Quersumme ist 70, also durch 10 teilbar.
$\leadsto$ Nummer könnte zulässig sein.

Fehlerkorrigierende Codes I

Bisher: Fehler in der Übertragung können festgestellt werden, aber nicht ohne wiederholte Übertragung behoben werden.
Jetzt: Fehlerkorrigierende Codes.

Grundlegende Idee: Den „Abstand" der Codewörter untereinander erhöhen!

Beispiel: Kodierung zweier Wörter: Ja $\mapsto 111$, Nein $\mapsto$ 000.
$\rightsquigarrow$ zwei der drei Stellen der Codewörter sind redundant.

Die Übertragung liefert das Codewort „101".
$\rightsquigarrow$ *Wahrscheinlich* wurde das Wort „Ja" übertragen.

Fehlerkorrigierende Codes II

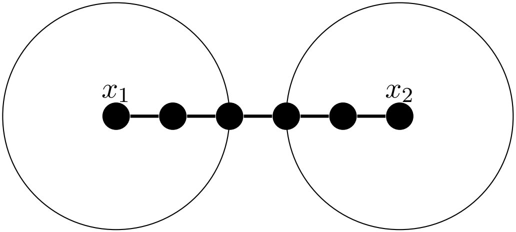
Die Anzahl der Bits, in denen sich zwei gleich lange 0-1-Wörter x und y unterscheiden, nennt man die **Hamming-Distanz** $d_H(x, y)$.

Die **Hamming-Distanz eines Codes** (also einer Menge von Codewörter) ist die kleinste vorkommende Hamming-Distanz $d_H(x, y)$ zwischen je zwei verschiedenen Codewörtern x und y.

Beispiele:
$d_H(111, 000) = 3$; $d_H(11000110, 01101100) = 4$
Die Hamming-Distanz der Binärkodierung von Dezimalzahlen ist eins.

Fehlerkorrigierende Codes III

Mitteilung: Bei einem Code mit Hamming-Distanz d kann man bis zu $\left\lfloor \frac{d-1}{2} \right\rfloor$ Bitfehler korrigieren.



x_1, x_2: gültige Codewörter
$\rightsquigarrow$ alle möglichen Wörter mit Hamming-Abstand $\left\lfloor \frac{d-1}{2} \right\rfloor$ zu x_1 können x_1 zugeordnet werden.
$\rightsquigarrow$ Anzahl gültiger Codewörter reduziert

Mitteilung: Für die Anzahl a der Codewörter in einem k Fehler korrigierenden Code der Länge n muss gelten:

$$a \leq \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{k}}$$

Informationstheorie

Ziel: Quantifizierung von Information.
Claude Shannon: Informationsbegriff wird Zufallsereignis zugeordnet

Münzwurf wird die Maßzahl **1 bit Informationsgehalt** zugeordnet
Werfen von zwei Münzen hat entsprechend 2 bit Informationsgehalt

Sinnvoll? Interpretation: Informationsgehalt eines Zufallsexperiments entspricht der durchschnittlichen Anzahl der Ja/Nein-Fragen, welche notwendig sind um den Experimentausgang zu bestimmen.

Beispiel: Jemand wählt zufällig eine Zahl zwischen 1 und einer Million.
Wieviele Fragen muss man stellen um Zahl sicher zu bestimmen?
Beste Strategie ist binäre Suche $\rightarrow \log_2(1.000.000) \approx 20$ Fragen

Entropie I

Extreme: Zufallsexperiment mit einem Ereignis der W'keit 1 und einem zweiten Ereignis mit W'keit 0. Informationsgehalt? Keiner!
Nun Informationsgehalt von Zufallsereignissen mit beliebiger Verteilung:

Definition: Eine Zufallsereignis mit Ereignissen $a_1, \ldots, a_k$ und entsprechenden W'keiten $p_1, \ldots, p_k$ (wobei $\sum_{i=1}^{k} p_i = 1$) hat den folgenden Informationsgehalt bzw. **Entropie**:

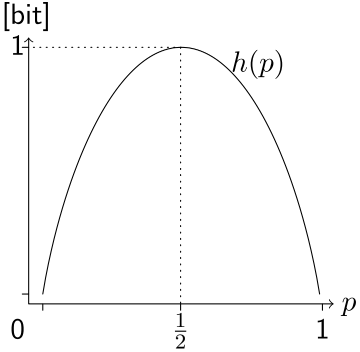
$$H(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \log_2(p_i) \text{ [bit]}$$

Für $p_i = 0$ setzen wir $p_i \log_2(p_i) = 0$.

Beispiel: Zufallsexperiment „Schwangerschaft":
$Pr[\text{Junge}] = 0,48$, $Pr[\text{Mädchen}] = 0,49$ und $Pr[\text{Zwillinge}] = 0,03$
Dann gilt $H(0,48; 0,49; 0,03) \approx 1,164$ bit.

Entropie II

Allgemeine Entropieverteilung für Zufallsexperiment mit zwei Ausgängen, d.h. $h(p) := H(p, 1-p)$.



Interpretation: Entropie als gewichteter Mittelwert.

- Informationsgehalt des Ereignisses a_i ist $-\log_2(p_i)$, d.h. je geringer W'keit für a_i desto größer ist Informationsgewinn bei Eintreten des Ereignisses.
- Informationsgehalt geht mit Gewicht p_i ein, entsprechend $\sum_{i=1}^{k} p_i = 1$.

Datenkomprimierung

Datenkomprimierung ist quasi Gegenspieler zur Fehlerentdeckung und -korrektur. Während man dort die Redundanz erhöht, will man sie bei der Komprimierung verringern.

Ausgangspunkt: Dateien

- enthalten Redundanzen,
- verbrauchen relativ viel Speicherplatz und
- dauern lange zu übertragen.

$\rightsquigarrow$ Datenkomprimierung kann **Speicherplatz** und **Übertragungszeit** sparen.

Beispiel: Ein unkomprimiertes FullHD-Video in 1080p50-Qualität benötigt ≈ 2.08 GBit/s. Zwei Stunden Film benötigen dann ≈ 15000 GB oder ca. 300 BlueRay-Disks.

Anwendungen von Datenkomprimierung

Generische Dateikomprimierung

- Dateien: gzip, bzip2.
- Archivierungsprogramm: pkzip, rar.
- Dateisystem: NTFS (New Technology File System).

Multimedia

- Bilder: GIF (Graphic Interchange Format), JPEG (Joint Photographic Experts Group)
- Audio: MP3 (MPEG-1 oder MPEG-2 Audio Layer III)
- Video: MPEG (Moving Picture Experts Group), DivX

Kommunikation

- ITU-T T4 Group 3 Fax
- V.42bis modem

Grundbegriffe der Komprimierung

$\boldsymbol{M}$**:** Eingangsdaten
$\boldsymbol{C(M)}$**:** Komprimierte Daten
$\boldsymbol{M'}$**:** Rekonstruierte Daten

$$M \longrightarrow \boxed{\text{Compressor}} \longrightarrow C(M) \longrightarrow \boxed{\text{Decompressor}} \longrightarrow M'$$

Wünschenswert: $|C(M)| < |M|$
$\rightsquigarrow$ **Komprimierungsrate** $= |C(M)|/|M|$

Die Komprimierung ist

- **verlustfrei** (engl. „lossless"), falls $M' = M$;
- **verlustbehaftet** (engl. „lossy"), sonst.

Verlustfreie Komprimierung

Typische Anwendungen:
Texte, Sourcecode, ausführbare Dateien.

Bekannte verlustfreie Komprimierungsprogramme:
gzip, bzip2, pkzip, rar.

Wichtige Komprimierungsverfahren:

- Lauflängenkodierung;
- Statistisch: Huffmankodierung und arithmetische Kodierung;
- Wörterbuchbasiert: Lempel-Ziv und ihre Varianten

Typische Komprimierungsraten: ca. 50%, für Text bis zu 25%.

Lauflängenkodierung („Run-Length Encoding“, RLE)

Idee: Ausnutzung von Wiederholungen in den Eingabedaten.

Beispiel „Bitmaps“: Eingabe besteht nur aus ‘0’ und ‘1’ $\rightsquigarrow$ Abwechselnd die Anzahlen der aufeinanderfolgenden ‘0’ oder ‘1’ speichern!
Diese bezeichnen wir als Lauflänge.

Frage: Wie kodiert man die Lauflänge?
Möglichkeiten:

- Kodierung mit fester Bitzahl
- Kodierung mittels variabler Bitzahl $\rightsquigarrow$ häufiger auftretende Lauflängen werden mit weniger Bits kodiert

Lauflängenkodierung Beispiel

M: Gescanntes ‘b’, s/w, gedreht

Unkomprimiert: 1 Bit/Pixel

28 Zeilen

```
0000000000000000000000000000000011111111111111111111110000000000
0000000000000000000000000000011111111111111111111111111110000000
0000000000000000000000000001111111111111111111111111111111100000
0000000000000000000000000111111111111111111111111111111111110000
0000000000000000000000011111111111111111111111111111111111111000
0000000000000000000001111111111111111111111111111111111111111110
0000000000000000000111111111111111111111111111111111111111111111
0000000000000000001111111111111111100000000000000001111111111111
0000000000000000001111111111111110000000000000000000000111111111
0000000000000000001111111111111000000000000000000000000001111111
0000000000000000011111111111100000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000011111111110000000000000000000000000000000011111
0000000000000000000111111111100000000000000000000000000000011111
0000000000000000000111111111100000000000000000000000000001111111
0000000000000000000001111111111000000000000000000000000111111100
0000000000000000000000011111111111111000000000000001111111110000
0000000000000000000000001111111111111111111111111111111111110000
0111111111111111111111111111111111111111111111111111111111111111
0111111111111111111111111111111111111111111111111111111111111111
0111111111111111111111111111111111111111111111111111111111111111
0111111111111111111111111111111111111111111111111111111111111111
0111111111111111111111111111111111111111111111111111111111111111
```

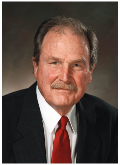
64 Spalten

RLE komprimiertes C(M)

```
32 22 10
29 28 7
27 32 5
25 35 4
23 38 3
21 42 1
19 45
18 17 16 13
18 15 22 9
18 13 26 7
17 12 30 5
17 10 32 5
17 10 32 5
17 10 32 5
17 10 32 5
17 10 32 5
17 10 32 5
17 10 32 5
19 10 30 5
19 10 28 7
21 10 24 7 2
23 14 14 9 4
24 36 4
1 63
1 63
1 63
1 63
1 63
```

Speicherverbrauch: $28 \cdot 64 + 6 = 1798$ Bits $\quad$ $95 \cdot 6 + 6 = 576$ Bits

Huffman-Kodierung



David Huffman
1925—1999

Grundidee

- Kodiere jedes Zeichen x mit einem Binärcode $C(x)$.
- Zeichen kommen im Text unterschiedlich häufig vor.
  $\rightsquigarrow$ Häufigere Zeichen werden mit weniger Bits kodiert.

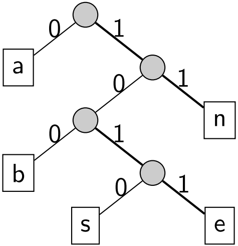
Huffman-Baum

- Kodierung wird durch einen Binärbaum beschrieben.
- Zeichen entsprechen Blättern im Baum.
- Pfad von Wurzel zum Blatt definiert Code (links = '0', rechts = '1')

Beispiel: Kodiere das Wort 'bananenananas'.

Zeichen	a	b	e	n	s	Länge
Naiv	000	001	010	011	100	39 Bits
Huffman	0	100	1011	11	1010	26 Bits

Frage: Wie erstellt man einen Huffman-Baum?

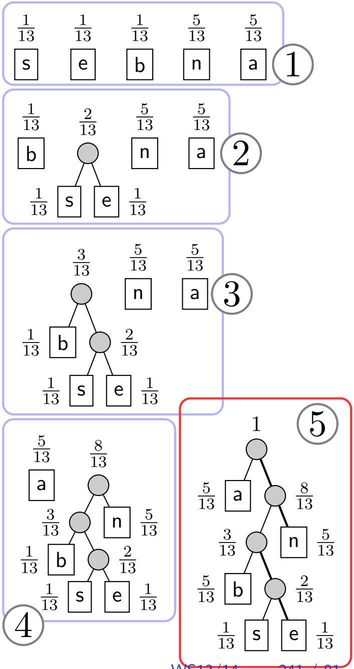


Erstellung des Huffman-Baums

Vorgehensweise: Bottom up

Erzeuge für jedes Zeichen x einen Teilbaum mit Knoten x und Gewicht $p_x :=$ relative Häufigkeit von x in M
while Es gibt mehr als einen Teilbaum **do**
  Suche zwei Teilbäume mit kleinsten Gewichten p_1 und p_2
  Vereinige die beiden Teilbäume zu einem mit Gewicht $p_1 + p_2$
end while

Bsp: 'bananenananas'



Mitteilung: Huffman-Kodierung ist **optimal**, d.h für eine gegebene Eingabe M mit relativen Häufigkeiten p_x gibt es keine Kodierung, die jedem Zeichen eine ganze Anzahl von Bits zuweist und M mit weniger Bits kodiert als die Huffman-Kodierung.

Lempel-Ziv

Grundidee

- Zeichenketten sind meist mehrmals in Eingabe vorhanden.
- Kodiere eine Zeichenkette durch ihren Index in einem Wörterbuch.
- Das Wörterbuch wird an die Eingabe angepasst.

Fragen Was steht im Wörterbuch? Was tut man, wenn eine Zeichenkette nicht im Wörterbuch steht?

Eine Lsg. LZW (Lempel-Ziv-Welch von 1984) verwendet Wörterbuch W mit maximal 2^{12} Einträgen, um eine Eingabe M zu komprimieren.

W enthält zunächst alle 256 möglichen 1-Byte-Zeichenketten.
while M ist nicht leer **do**
    Finde die längste Zeichenkette s in W, die mit dem Anfang von M übereinstimmt.
    Gib den Index von s in W aus und entferne s vom Anfang von M.
    Füge s gefolgt vom nächsten Zeichen in M zu W hinzu.
end while

Beispiel Lempel-Ziv-Welch

Eingabe:
'aabaaabaaabaaaab'; Länge in 8-Bit-ASCII: 128 Bit

Ablauf der Komprimierung
Initial.: $W[0] = \backslash 0, \ldots, W[97] = \mathsf{a}, W[98] = \mathsf{b}, \ldots, W[255] = \ddot{\mathsf{y}}$

Längste Zeichenkette s in W	Ausgabe	Eintrag in W
a	97	$W[256] :=$ a**a**
a	97	$W[257] :=$ a**b**
b	98	$W[258] :=$ b**a**
aa	256	$W[259] :=$ aa**a**
ab	257	$W[260] :=$ ab**a**
aaa	259	$W[261] :=$ aaa**b**
ba	258	$W[262] :=$ ba**a**
aaab	261	

Ausgabelänge Insgesamt $8 \cdot 12 = 96$ Bit $\rightsquigarrow$ Kompressionsrate $= 0.75$

Das Counting-Argument

Frage: Gibt es ein Verfahren, um ALLE Dateien verlustfrei zu komprimieren?

Antwort: Nein! There is no free lunch! Aber warum nicht?

Argumentation: Angenommen, man könnte alle Dateien der Länge n Bits um mindestens 1 Bit komprimieren. Es gibt 2^n Dateien der Länge n, und insgesamt $2^1 + 2^2 + \ldots + 2^{n-1} = 2^n - 2$ Dateien mit weniger Bits. Dann müssen mindestens zwei Eingabedateien auf die gleiche komprimierte Datei kodiert werden $\rightsquigarrow$ keine verlustfreie Dekomprimierung möglich.

Gegenstück in der Physik: Perpetuum Mobile

Verlustbehaftete Komprimierung

Grundidee
Tausche eine deutlich höhere Komprimierungsrate als bei verlustfreien Verfahren gegen (hoffentlich akzeptable) Abweichungen zwischen originalen und rekonstruierten Daten.

Typische Anwendungen
Multimedia-Dateien: Bilder, Musik, Videos.

Ausnutzung der Ungenauigkeit menschlicher Wahrnehmung:

- Psychoakustische Effekte,
- Helligkeitswerte feiner wahrgenommen als Farbwerte,
- Rekonstruktionsfehler bei Details in Bildern meist nicht sichtbar.

Wichtige Komprimierungsverfahren MP3, JPEG, MPEG

Typische Komprimierungsraten Audio 10%, Fotos 5%, Video 1%.

JPEG

Grundidee

- Entworfen für Komprimierung von Fotos
- Feine Details (hochfrequente Bildkomponenenten) werden weniger stark wahrgenommen als grobe Strukturen (niederfrequente Bildkomponenten)

Bild M → DCT → Quantisierung → RLE + Huffman → C(M)

Kernkomponenten

- DCT: Diskrete Kosinustransformation; transformiert das Bild blockweise in eine Frequenzdarstellung.
- Quantisierung: Reduziert numerische Genauigkeit der Daten $\rightsquigarrow$ weniger Entropie; nicht verlustfrei reversibel; hohe Frequenzen werden stärker reduziert als niedrige.
- RLE+Huffman-Codierer: Effiziente Komprimierung der quantisierten Daten.

JPEG Beispiel

Unkomprimiertes Bild: 512×512 Pixel, 256 KB

Abbildung: Schiffe am Strand

JPEG Beispiel

Komprimiertes Bild: 39 KB, Komprimierungsrate 15%

Abbildung: Schiffe am Strand

JPEG Beispiel

Komprimiertes Bild: 6 KB, Komprimierungsrate 2.3%

Abbildung: Schiffe am Strand (?)