Informatik der digitalen Medien

Ergänzungs-Studienangebot der Mediendidaktik für Lehramtstudenten
Dr. Harald Sack
Institut für Informatik
FSU Jena

Sommersemester 2007

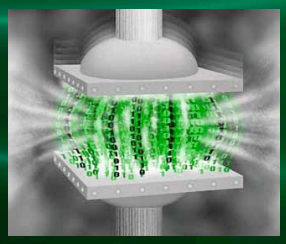
http://www.informatik.uni-jena.de/~sack/SS07/infod.htm

Informatik der digitalen Medien

2. Grundlagen der Digitalisierung – Datenrepräsentation im Computer (Teil 2)

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- Komprimierung und Redundanz
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - Redundanzfreie Codes
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



Komprimierung und Redundanz

- Information und Entropie
  - Was ist Information?
    - Maßgröße für die Ungewissheit des Eintretens von **Ereignissen** im Sinne der Wahrscheinlichkeitsrechnung
    - = beseitigte Ungewissheit (z.B. durch Auskunft, Aufklärung, Mitteilung, Benachrichtigung über Gegenstände)
    - Ereignisse = Zeichen (Nachrichtenelemente)
      - Werden durch Auswahlvorgang aus einem Zeichenvorrat von Nachrichtenquelle erzeugt

Komprimierung und Redundanz

- Information und Entropie
  - Was ist Information?
    - **Zeichenkette** = Folge von Elementen eines Alphabets

      Wirsing - Alphabet = {a,b,c,d,….,A,B,C,D,…}
      1001001 - Alphabet = {0,1}

    - **Nachricht** = übermittelte Zeichenkette, die meist nach bestimmten, vorgegebenen Regeln (**Syntax**) aufgebaut ist.

    - durch Verarbeitung erhält die Nachricht Bedeutung (**Semantik**)

Komprimierung und Redundanz

- Information und Entropie
  - Was ist Information?
    - **Information** = Nachricht, die einen **Sachverhalt** ausdrückt, einem **Zweck** dient oder eine **Aktion** auslöst
    - Wissensgewinn
    - ermöglicht Handlungen
    - stellt Bezüge her
    - erklärt Sachverhalte

Komprimierung und Redundanz

- Information und Entropie
  - Wie messe ich Information?
    - z.B. **kürzeste Beschreibung**, die eine Nachricht benötigt, welche dieselbe Bedeutung für den Empfänger besitzt, wie die ursprüngliche vorgegebene Information (**Beschreibungskomplexität**)
    - **Wie viele Bits benötige ich** mindestens, um eine Nachricht mit einem bestimmten Informationsgehalt zu kodieren?

Alphabet = {a,b,c,d}

Kodierung: Blockcode mit 2 Bit

a	00
b	01
c	10
d	11

Komprimierung und Redundanz

- Information und Entropie
  - Wie messe ich Information?
    - Alphabet = {a,b,c,d}
    - Nachricht: **abacbdbcbabbabbd**
    - Kodierte Nachricht:

00 01 00 10 01 11 01 10 01 00 01 01 00 01 01 11

    - Gesamtinformation: **32 Bit**
    - Mittlerer Informationsgehalt eines Zeichens: **2 Bit**
    - Informationsgehalt einer **kompletten Nachricht**?

a	00
b	01
c	10
d	11

Komprimierung und Redundanz

- Information und Entropie
  - Wie messe ich Information?
    - Betrachte **relative Zeichenhäufigkeit** der Nachricht

Nachricht:
abacbdbcbabbabbd

Kodierte Nachricht:
00 01 00 10 01 11 01 10 01 00 01 01 00 01 01 11

Zeichen	Code	Relative Häufigkeit
a	00	1/4
b	01	1/2
c	10	1/8
d	11	1/8

  - **Idee:**
    gebe häufigeren Zeichen kürzeren Code, damit die Nachricht kürzer wird.

Komprimierung und Redundanz

- Information und Entropie
  - Wie messe ich Information?
    - Informationsgehalt abhängig von Kodierung
    - Nach Claude E. Shannon: **Entropie H**

Claude E. Shannon
(1916-2001)

$$H(I) = \sum_i -p_i \log_2(p_i)$$

Nachricht I, besteht aus unterschiedlichen **Symbolen i**
jedes Symbol i kommt in Nachricht I mit bestimmter Häufigkeit (**Wahrscheinlichkeit**) p_i vor

Komprimierung und Redundanz

- Information und Entropie
  - Kleiner Exkurs: Logarithmus
    - Auflösung der Gleichung $y = a^x$ nach x

$$\log_a y = \log_a a^x = x$$

Bsp.: Logarithmus zur Basis 2

$$\log_2 2^x = x$$
$$\log_2 16 = \log_2 2^4 = 4$$
$$\log_2 256 = \log_2 2^8 = 8$$

$$\log_2 0{,}5 = \log_2 \frac{1}{2^1} = \log_2 2^{-1} = -1$$
$$\log_2 0{,}25 = \log_2 \frac{1}{2^2} = \log_2 2^{-2} = -2$$
$$\log_2 0{,}125 = \log_2 \frac{1}{2^3} = \log_2 2^{-3} = -3$$

Komprimierung und Redundanz

- **Information und Entropie**
  - Wie messe ich Information?
    - Entropie

$$H(I) = \sum_i -p_i \log_2(p_i)$$

$$H(I) = (-\frac{1}{4}\cdot\log_2\frac{1}{4}) + (-\frac{1}{2}\cdot\log_2\frac{1}{2}) + 2\cdot(-\frac{1}{8}\cdot\log_2\frac{1}{8}) =$$
$$(-\frac{1}{4}\cdot(-2)) + (-\frac{1}{2}\cdot(-1)) + 2\cdot(-\frac{1}{8}\cdot(-3)) =$$
$$\frac{1}{2} + \frac{1}{2} + 2\cdot\frac{3}{8} =$$
$$1\frac{6}{8} =$$
$$1{,}75$$

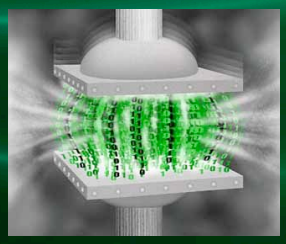
Zeichen c_i	Code	Relative Häufigkeit $p(c_i)$
a	00	1/4
b	01	1/2
c	10	1/8
d	11	1/8

  - Entropie der Nachricht I: **1,75 Bit**

⇒ **Es muss eine bessere Kodierung geben**

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- Komprimierung und Redundanz
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - Redundanzfreie Codes
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



Komprimierung und Redundanz

- Redundanz –
  Mehrwert oder Verschwendung?
  - [lat.]*redundare* = überlaufen
  - Anteile einer Nachricht, die **keine** zur Nachricht **beitragende Information** enthalten, also aus dieser entfernt werden können, ohne den eigentlichen Informationsgehalt zu verringern
  - Bsp.:
    - **Whnachtsman**

      = Weihnachtsmann

unsere Sprache enthält bereits Redundanz

Komprimierung und Redundanz

- Redundanz –
  Mehrwert oder Verschwendung?
  - Wozu ist Redundanz dann eigentlich gut?
    - **Bsp.:**
      - **„Frxxdrxch Sxhxllxr xnxversxtät Jxna“**
    - Information kann selbst bei
      - **unvollständiger Übermittlung** oder
      - **Übertragungsfehlern** rekonstruiert werden
    - Information ist **leichter zu lesen/interpretieren**

      ↑ **Fehlertoleranz und Vereinfachung**
      ↓ **größere Informationsmenge**

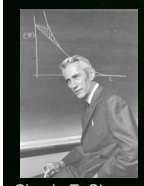
Komprimierung und Redundanz

- Redundanz –
  Mehrwert oder Verschwendung?
  - **Vorteile:**
    - Fehlertoleranz
    - Vereinfachung
  - **Nachteile:**
    - größere Informationsmenge

⇒ Einsatzzweck entscheidet über Redundanz

Komprimierung und Redundanz

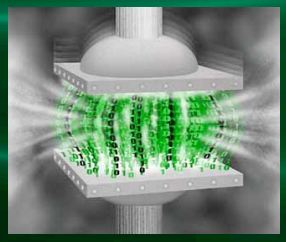
- Redundanz–
  Mehrwert oder Verschwendung?
  - Kann man Daten beliebig komprimieren?
  - **Claude E. Shannon** zeigte 1948 die Existenz einer maximalen Grenze, wie weit sich Information ohne Verlust komprimieren läst
  - Er definiert den **Informationsgehalt** einer Nachricht, die **Entropie H**
    - abhängig von **statistischer Natur** der Nachrichtenquelle
    - **keine weitere verlustfreie Komprimierung (kleiner als H) möglich!**



Claude E. Shannon
(1916-2001)

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - **Komprimierungsvarianten**
  - Redundanzfreie Codes
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung

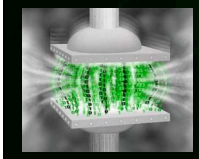


Komprimierung und Redundanz

- Komprimierungsvarianten
  - Definition Komprimierung:

    Verringerung der Redundanz einer Datenmenge

    - Varianten:
      - logische / physikalische Komprimierung
      - symmetrische / asymmetrische Komprimierung
      - adaptive / semiadaptive / nichtadaptive Komprimierung
      - verlustfreie / verlustbehaftete Komprimierung



Komprimierung und Redundanz

- Komprimierungsvarianten
  - **Logische Komprimierung**
    - fortlaufende Substitution von Symbolen durch andere Symbole
    - **Nutzung der inhärenten Information** der Daten
    - z.B.: **„USA“** statt **„United States of America“**
  - **Physikalische Komprimierung**
    - **ohne Nutzung inhärenter Information**
    - Austausch einer Kodierung durch eine kompaktere
    - kann leicht **automatisiert** werden

Komprimierung und Redundanz

- Komprimierungsvarianten
  - **Symmetrische Komprimierung**
    - Verfahren zur Kodierung und Dekodierung besitzen **dieselbe Berechnungskomplexität** (d.h. sind gleich schwierig)
  - **Asymmetrische Komprimierung**
    - Kodierungs- und Dekodierungsverfahren besitzen **unterschiedliche Berechnungskomplexität**
    - In der Regel ist **Kodierung** schwerer
      - ist dann sinnvoll, wenn nur selten auszuführen

Komprimierung und Redundanz

- Komprimierungsvarianten
  - **Adaptive / semiadaptive / nichtadaptive Komprimierung**
    - Betrifft **Wörterbuchbasierte Komprimierung**
      - spezifisches Medienformat

Komprimierung und Redundanz

- Komprimierungsvarianten
  - **Adaptive / semiadaptive / nichtadaptive Komprimierung**
    - Betrifft **Wörterbuchbasierte Komprimierung**
      - spezifisches Medienformat

z.B. Text

jedes Wort des Textes
findet sich im Wörterbuch

Wörterbuch

ordnet dem Wort einen
kürzeren Code zu

„Weihnachtsmann“ → 0123

Komprimierung und Redundanz

- Komprimierungsvarianten
  - **Nicht-adaptive Komprimierung**
    - Verwendet **statisches Wörterbuch** mit vorgegebenen Datenmustern (schnell, aufwändiges Wörterbuch)
  - **Adaptive Komprimierung**
    - Für den zu komprimierenden Text wird ein **eigenes Wörterbuch** erstellt (enthält nur Worte aus dem zu komprimierenden Text)
  - **Semi-adaptive Komprimierung**
    - **Mischform** aus adaptiver und nicht-adaptiver Komprimierung

Komprimierung und Redundanz

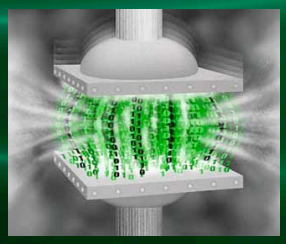
- Komprimierungsvarianten
  - **Verlustfreie Komprimierung**
    - Nach Kodierung und Dekodierung können die ursprünglichen Daten **unverändert** und **ohne Verlust** rekonstruiert werden
  - **Verlustbehaftete Komprimierung**
    - Beim Komprimieren **gehen** (unwichtige) Teile der ursprünglichen Information **verloren**, so dass diese nach dem Dekodieren nicht exakt mit den ursprünglichen Daten übereinstimmt

Komprimierung und Redundanz

- Komprimierungsvarianten
  - logische ←→ physikalische Komprimierung
  - symmetrische ←→ asymmetrische Komprimierung
  - adaptive ←→ semiadaptive ←→ nichtadaptive Komprimierung
  - verlustfreie ←→ verlustbehaftete Komprimierung

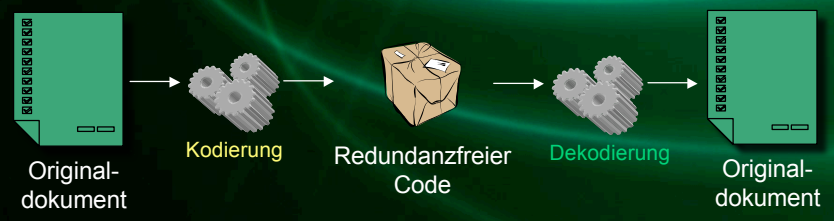
Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- Komprimierung und Redundanz
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - **Redundanzfreie Codes**
    - **Verlustfreie Komprimierung**
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



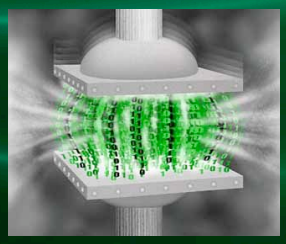
Komprimierung und Redundanz

- Redundanzfreie Codes
  - Verlustfreie Komprimierung
    - Ziel ist dabei einen möglichst **redundanzfreien Code** zu erzeugen, aus dem die ursprüngliche Information **eindeutig** und **ohne Informationsverlust** wieder rekonstruiert werden kann



Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- Komprimierung und Redundanz
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - **Redundanzfreie Codes**
    - Verlustfreie Komprimierung
    - **Lauflängenkodierung**
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



Komprimierung und Redundanz

- Redundanzfreie Codes
  - Lauflängenkodierung
    - z.B. Textdatei

AAAADEBBHHHHHCAAABCCCC

- **Folgen von sich wiederholenden Zeichen** lassen sich kompakter kodieren, indem man die Folge **nur einmal** angibt und **dazu die Anzahl** der jeweiligen Wiederholungen

⇒ **4ADE2B5HC3AB4C**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - Lauflängenkodierung
    - Immer rentabel?
    - Nur wenn **lange Folgen** desselben Zeichens auftreten
    - Im Computer sind alle Daten digital (0/1)

01110011000000100001111110100101

↓

03100116014061010010 1

Komprimierung und Redundanz

- Redundanzfreie Codes
  - Kodierung mit variabler Länge
    - Standard: Kodierung mit **fester Länge**
    - Bsp.:
      ordne jedem Buchstaben des Alphabets einen **5-Bit Code** zu
      (**i**-ter Buchstabe wird zu Binärzahl **i**)
    - Ergebnis:

ABRAKADABRA

00001	00010	10010	00001	01101	00001	00100	00001	00010	10010	00001
A	B	R	A	K	A	D	A	B	R	A
1	2	18	1	13	1	4	1	2	18	1

Gesamtlänge: 55 Bit

Komprimierung und Redundanz

- Redundanzfreie Codes
  - Kodierung mit variabler Länge

00001	00010	10010	00001	01101	00001	00100	00001	00010	10010	00001
A	B	R	A	K	A	D	A	B	R	A
1	2	18	1	13	1	4	1	2	18	1

  - Gesamtlänge: 55 Bit
  - Aber:
    - „**A**“ kommt 5-mal vor
    - „**K**“ kommt nur 1-mal vor
  - **Platzeinsparung**, wenn häufiger vorkommende Zeichen kürzer kodiert werden als seltenere

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Kodierung mit variabler Länge**
    - Ordne den Buchstaben einen **Code** zu, dessen Länge von der **relativen Häufigkeit** des Buchstabens abhängt
    - z.B.

Buchstabe	Häufigkeit	Code
A	5	0
B	2	1
R	2	01
K	1	10
D	1	11

0	1	01	0	10	0	11	0	1	01	0
A	B	R	A	K	A	D	A	B	R	A

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Kodierung mit variabler Länge**

0	1	01	0	10	0	11	0	1	01	0
A	B	R	A	K	A	D	A	B	R	A

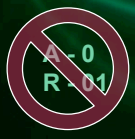
  - Gesamtlänge: 15 Bit (010101001101010)
  - Aber:
    - Problem der Rückübersetzung

01	0	01	001	10	01	10	0	01	1001	0
R	A	B	R	A	R	K	B	A	K	

    - **Rückübersetzung nicht eindeutig!**

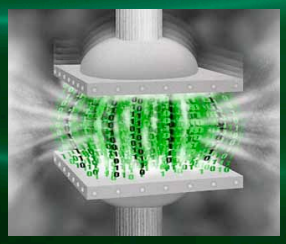
Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Kodierung mit variabler Länge**
    - Wie kann man Mehrdeutigkeiten vermeiden?
      - Einfügen eines **zusätzlichen Begrenzerzeichens** zwischen den einzelnen Codes
      - oder
      - Achte darauf, dass **kein Code zugleich der Beginn eines anderen Codes** ist



Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - **Redundanzfreie Codes**
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - **Huffman-Kodierung**
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung

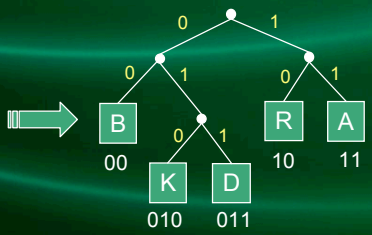


Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    - Achte darauf, dass **kein Code zugleich der Beginn eines anderen Codes** ist
    - Präfixfreier Code → **Fano-Bedingung**

Robert M. Fano
(*1917)

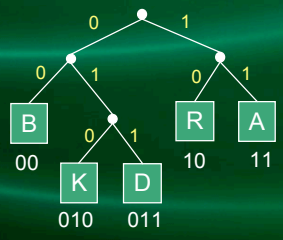
Buchstabe	Code
A	11
B	00
R	10
K	010
D	011



Baumkodierung

Komprimierung und Redundanz

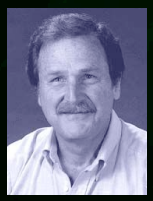
- Redundanzfreie Codes
  - **Huffman-Kodierung**
    - **Präfixfreie Kodierung:**
      - kein Code ist zugleich Anfang eines anderen Codes
      - lässt sich am leichtesten durch **Baumkodierung** erzeugen
    - an **inneren Knoten** wird nach links (**0**) oder nach rechts (**1**) verzweigt
    - Zeichen werden nur in den **Endknoten** (Blättern) kodiert



Baumkodierung

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    - Wie gewinnt man einen möglichst effizienten (**kurzen**) präfixfreien Code?
    - 1952 Huffman-Kodierung
      - Optimale Kodierung einer Textdatei lässt sich stets als Binärbaum darstellen (jeder innere Knoten besitzt 2 Nachfolger)
      - Tiefe eines Blattknotens gibt die Länge des zugeordneten Codes an


David Huffman
1925-1999

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    - Wie gewinnt man einen möglichst effizienten (**kurzen**) präfixfreien Code?

1. Ermittle die **relative Häufigkeit** der zu kodierenden Zeichen

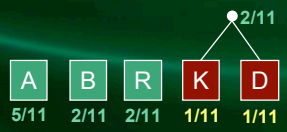
ABRAKADABRA 

Buchstabe	Anzahl	Häufigkeit
A	5	5/11
B	2	2/11
R	2	2/11
K	1	1/11
D	1	1/11

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**

2. Fasse die beiden Zeichen c_i und c_j mit der **geringsten Häufigkeit** $f(c_i)$ und $f(c_j)$ zusammen zu einem **neuen Knoten** mit der Häufigkeit $f(c_i)+f(c_j)$



Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    2. Fasse die beiden Zeichen c_i und c_j mit der **geringsten Häufigkeit** $f(c_i)$ und $f(c_j)$ zusammen zu einem **neuen Knoten** mit der Häufigkeit $f(c_i)+f(c_j)$
    3. Fahre fort, bis alle Blattknoten in einem gemeinsamen Baum verbunden sind

Komprimierung und Redundanz

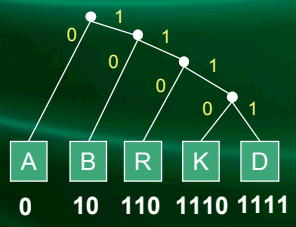
- Redundanzfreie Codes
  - **Huffman-Kodierung**
    2. Fasse die beiden Zeichen c_i und c_j mit der **geringsten Häufigkeit** $f(c_i)$ und $f(c_j)$ zusammen zu einem **neuen Knoten** mit der Häufigkeit $f(c_i)+f(c_j)$
    3. Fahre fort, bis alle Blattknoten in einem gemeinsamen Baum verbunden sind

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    2. Fasse die beiden Zeichen c_i und c_j mit der **geringsten Häufigkeit** $f(c_i)$ und $f(c_j)$ zusammen zu einem **neuen Knoten** mit der Häufigkeit $f(c_i)+f(c_j)$
    3. Fahre fort, bis alle Blattknoten in einem gemeinsamen Baum verbunden sind

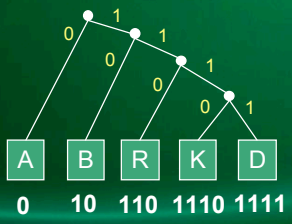
Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**
    2. Fasse die beiden Zeichen c_i und c_j mit der **geringsten Häufigkeit** $f(c_i)$ und $f(c_j)$ zusammen zu einem **neuen Knoten** mit der Häufigkeit $f(c_i)+f(c_j)$
    3. Fahre fort, bis alle Blattknoten in einem gemeinsamen Baum verbunden sind
    4. Interpretiere Baum als **Baumkodierung**



Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**

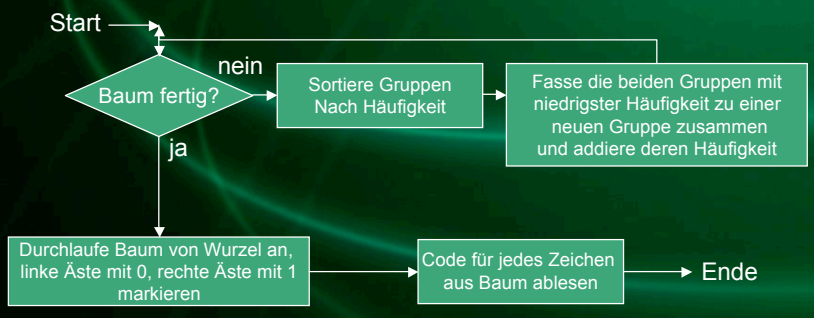


ABRAKADABRA ⇒ **0 10 110 0 1110 0 1111 0 10 110 0**
A B R A K A D A B R A

- Gesamtlänge: **23 Bit**
- **Achtung**: Es kann mehrere, unterschiedliche optimale Codes geben

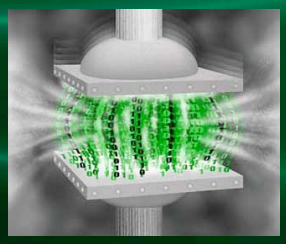
Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Huffman-Kodierung**



Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - **Redundanzfreie Codes**
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - **Wörterbuchbasierte Kodierung**
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**

z.B. Text

jedes Wort des Textes
findet sich im Wörterbuch

Wörterbuch

ordnet dem Wort einen
kürzeren Code zu

„Weihnachtsmann“ → 0123

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - Bekanntester Vertreter: **LZW-Algorithmus** (zip)
      - **L**empel, **Z**iv, **W**elch (1984)


      - Adaptives Verfahren
      - patentiert von Unisys/IBM
      - Prinzipieller Ablauf
        - **(1)** erzeuge aus zu komprimierenden Zeichenketten ein Wörterbuch
        - **(2)** Daten werden mit Wörterbuch kodiert (komprimiert)
        - **(3)** Wörterbuch muss mit übertragen (gespeichert) werden

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - **(1) Wörterbuch erzeugen**
      1. Lese Zeichen aus zu komprimierenden Daten und akkumuliere diese zu Zeichenkette **S**, solange **S** sich als Wörterbucheintrag findet
      2. Sobald Zeichen **x** gelesen wird und **Sx** ist nicht im Wörterbuch, nehme **Sx** in das Wörterbuch auf
      3. Wiederhole (1,2) mit **x** als nächstes Zeichen bis das Ende der zu komprimierenden Daten erreicht ist.

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Alphabet mit 6 Zeichen (**A,B,C,D,E,F**)
        - **Wörterbuch** mit 16 möglichen Einträgen (4-Bit Kodierung)

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110-1111	noch frei

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere: ABABACDCDAAAAAAE

ABABACDCDAAAAAAE

↓

0110

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110-1111	noch frei

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere:
          ABABACDCDAAAAAAE

ABABACDCDAAAAAAE

↓

0110

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	**AB**
0111-1111	**noch frei**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere:
          ABABACDCDAAAAAAE

A**BA**BACDCDAAAAAAE

↓

0111

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	AB
0111	**BA**
1000-1111	**noch frei**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere: ABABACDCDAAAAAAE

ABABACDCDAAAAAAE

↓

1000

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	AB
0111	BA
1000	**ABA**
1001-1111	**noch frei**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere: ABABACDCDAAAAAAE

ABAB**AC**DCDAAAAAAE

↓

1001

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	AB
0111	BA
1000	ABA
1001	**AC**
1010-1111	**noch frei**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere: ABABACDCDAAAAAAE

ABABA**CD**CDAAAAAAE

↓

1010

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	AB
0111	BA
1000	ABA
1001	AC
1010	**CD**
1011-1111	**noch frei**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Wörterbuchbasierte Komprimierung**
    - **LZW-Algorithmus** - vereinfachter Ablauf
      - Beispiel:
        - Komprimiere: ABABACDCDAAAAAAE

Code	Zeichen
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	AB
0111	BA

Code	Zeichen
1000	ABA
1001	AC
1010	CD
1011	DC
1100	CDA
1101	AA
1110	AAA
1111	AAAE

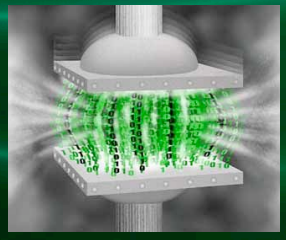
ABA	**1000**
BA	**0111**
CD	**1010**
CDA	**1100**
AA	**1101**
AAAE	**1111**

Komprimierung und Redundanz

- Redundanzfreie Codes
  - LZW-Algorithmus (vereinfacht)
    - **Wichtig**: Wörterbuch muss stets mit übertragen/gespeichert werden
    - durch geschickte Wahl des Verfahrens für Wörterbucheinträge und Verzicht auf maximal mögliche Komprimierung **kann auf Übertragung des Wörterbuchs verzichtet werden**
    - **Wörterbuch** wird dann **dynamisch** durch Dekomprimierungsalgorithmus wieder **aufgebaut**
    - Typische Wörterbuchgröße: 4.096 Byte (12-Bit Kodierungsschema)
    - wird für Grafikkodierung **GIF** und **TIFF** eingesetzt

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - **Redundanzfreie Codes**
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - **Arithmetische Kodierung**
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



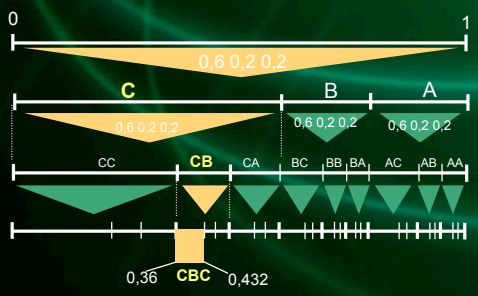
Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Arithmetische Kodierung**
    - **Zeichen** werden durch **Häufigkeitsintervalle** kodiert
    - **Zeichenfolgen** werden durch bedingte („geschachtelte“) Häufigkeitsintervalle kodiert
    - patentrechtlich geschütztes Verfahren (Q-Coder, IBM)
    - Nähert sich einer optimalen Kodierung bei sehr langen zu komprimierenden Nachrichten

Komprimierung und Redundanz

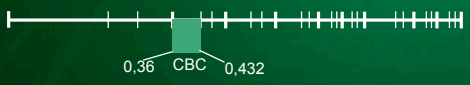
- Redundanzfreie Codes
  - **Arithmetische Kodierung**
    - Bsp.:
      - kodiere **CBC**

Zeichen	Relative Häufigkeit
A	0,2
B	0,2
C	0,6



Komprimierung und Redundanz

- Redundanzfreie Codes
  - **Arithmetische Kodierung**
    - Bsp.:
      - kodiere **CBC**



      - obere und untere Grenze als Binärzahl

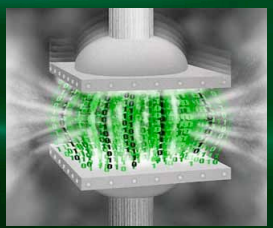
0,36	0,432
0,01011…	**0,011001…**

      - wähle **obere Grenze** und **schneide Binärzahl nach erster Stelle ungleich unterer Grenze ab**

CBC - **011**

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - Redundanzfreie Codes
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - **Geht es noch besser?**
    - **Verlustbehaftete Komprimierung**



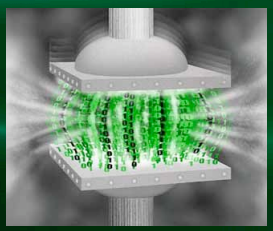
Komprimierung und Redundanz

- **Geht es noch besser?**
  - …nur, wenn wir **Information weglassen**
  - sinnvoll?
  - nur dann, wenn für uns die weggelassene Information „**nicht so wichtig**“ ist
  - z.B.
    - Audiodaten
      - Leise Geräusche werden von lauten übertönt
      - Subjektiver Eindruck bleibt, auch wenn leise Geräusche vor Komprimierung herausgefiltert werden
    - Bilddaten
      - Details im Hintergrund sind unwichtiger als der Bildvordergrund

demo

Grundlagen der Digitalisierung
Datenrepräsentation im Computer (2)

- **Komprimierung und Redundanz**
  - Information und Entropie
  - Redundanz – Mehrwert oder Verschwendung?
  - Komprimierungsvarianten
  - Redundanzfreie Codes
    - Verlustfreie Komprimierung
    - Lauflängenkodierung
    - Huffman-Kodierung
    - Wörterbuchbasierte Kodierung
    - Arithmetische Kodierung
  - Geht es noch besser?
    - Verlustbehaftete Komprimierung



Informatik der digitalen Medien

2. Grundlagen der Digitalisierung – Datenrepräsentation im Computer (2)
   - **Literatur**

- Ch. Meinel, H. Sack:
  WWW– Kommunikation, Internetworking, Web-Technologien,
  Springer, 2004.

- P.A. Henning:
  Taschenbuch Multimedia,
  3. Aufl., Fachbuchverlag Leipzig, 2003.