

Algorithmen und Datenstrukturen (für ET/IT)

Wintersemester 2012/13

Dr. Tobias Lasser

Computer Aided Medical Procedures
Technische Universität München



Weiterer Ablauf Vorlesung

- Klausurstoff: Kapitel 1 bis Kapitel 10 (einschließlich)
- nicht klausur-relevant:
 - Kapitel 11: Datenkompression (Huffmann-Codierung, JPEG)
 - Kapitel 12: Kryptographie (symm. Verfahren, RSA)
- Weiterer Ablauf:
 - heute: Kapitel 11 (Huffmann-Codierung)
 - morgen: **Wiederholung und Fragestunde**
 - 4. Februar: Probeklausur (über 2. Teil der Vorlesung)
 - 6. Februar: Vorlesung Kapitel 11 (JPEG)
 - 7. Februar: Vorlesung Kapitel 12 (Kryptographie)
 - 8. Februar: letzte Tutorfragestunde
 - 21. Februar: Klausur

Programm heute

7 Fortgeschrittene Datenstrukturen

8 Such-Algorithmen

9 Graph-Algorithmen

10 Numerische Algorithmen

11 Datenkompression

Einführung

Grundlagen Informationstheorie

Huffman Codes

Warum Datenkompression?

- Datenbanken:
 - Experimente vom LHC: ca. 15PB Daten pro Jahr
 - Steam-Plattform ca. 30PB Daten pro Monat (Stand: 2011)
 - Google bearbeitet ca. 24PB Daten pro Tag (Stand: 2008)

Warum Datenkompression?

- Datenbanken:
 - Experimente vom LHC: ca. 15PB Daten pro Jahr
 - Steam-Plattform ca. 30PB Daten pro Monat (Stand: 2011)
 - Google bearbeitet ca. 24PB Daten pro Tag (Stand: 2008)
- Filmmaterial:
 - 90 Minuten Film in PAL Auflösung, 25fps: 105GB
Kapazität DVD: 8.5GB (single side, dual layer)
 - 90 Minuten Film in Full HD (1080p), 24fps: 501GB
Kapazität Blu-Ray: 50GB (dual layer)

Warum Datenkompression?

- Datenbanken:
 - Experimente vom LHC: ca. 15PB Daten pro Jahr
 - Steam-Plattform ca. 30PB Daten pro Monat (Stand: 2011)
 - Google bearbeitet ca. 24PB Daten pro Tag (Stand: 2008)
- Filmmaterial:
 - 90 Minuten Film in PAL Auflösung, 25fps: 105GB
Kapazität DVD: 8.5GB (single side, dual layer)
 - 90 Minuten Film in Full HD (1080p), 24fps: 501GB
Kapazität Blu-Ray: 50GB (dual layer)
- Grafik:
 - Display mit Auflösung 2560x1600, 60Hz Refresh (z.B. Google Nexus 10): Datenrate $\sim 700\text{MB/s}$
 - 4K Display (Auflösung 4096x2160, 60Hz Refresh): $\sim 1.5\text{GB/s}$
 - Datenrate verdoppelt sich für 3D Displays

Warum Datenkompression?

- limitierte Speicherkapazitäten:
 - 3.5" Festplatte: bis zu 4TB (Stand: 2012)
 - 2.5" Solid State Disk (SSD): bis 500GB
(in 2013: bis 800GB)
 - Flash-Speicher in Handys/Tablets: typisch 16GB bis 64GB
(Stand: 2012)
 - DVD: 8.5GB, Blu-Ray: 50GB (jeweils dual layer)

Warum Datenkompression?

- limitierte **Speicherkapazitäten**:
 - 3.5" Festplatte: bis zu **4TB** (Stand: 2012)
 - 2.5" Solid State Disk (SSD): bis **500GB**
(in 2013: bis **800GB**)
 - Flash-Speicher in Handys/Tablets: typisch **16GB** bis **64GB**
(Stand: 2012)
 - DVD: **8.5GB**, Blu-Ray: **50GB** (jeweils dual layer)
- limitierte Kapazitäten von **Datenübertragung**:
 - Ethernet: typisch 1Gbit/s ($\sim 100\text{MB/s}$)
mit Glasfaser bis 100Gbit/s
 - WLAN: 802.11n bis zu 450MBit/s ($\sim 40\text{MB/s}$)
in Markteinführung: 802.11ac bis zu 1Gbit/s
 - Mobilfunk: LTE aktuell bis zu 100MBit/s ($\sim 12\text{MB/s}$)

Praxisbeispiele Datenkompression

- Speicherung von Bildern
 - GIF Format (LZW)
 - PNG Format (LZ77 mit Huffmann-Codierung)
 - JPEG Format (DCT mit Quantisierung und Huffmann-Codierung)
 - JPEG2000 Format (DWT mit Quantisierung und arithmetischer Codierung)
- Speicherung von Audio
 - MP3 (MDCT, Psycho-akustisches Modell und Huffmann-Codierung)
 - AAC / MPEG-4 (MP3 mit Verbesserungen)
- Speicherung von Video
 - MPEG-2 (DCT mit Quantisierung und Huffmann-Codierung)
 - H.264 / MPEG-4 (MPEG-2 mit Verbesserungen und arithmetischer Codierung)

Praxisbeispiele Datenkompression

- **Netzwerkübertragung**
 - PPP mit LZS (LZ77 mit Huffman-Codierung)
 - HTTP mit gzip (LZ77 mit Huffman-Codierung)
- **Datenspeicherung auf Festplatten**
 - ZIP Datei-Archive (LZ77 mit Huffman-Codierung)
 - bzip2 Datei-Kompression (BWT mit Huffman-Codierung)
 - 7zip LZMA (LZ77 mit Range-Codierung)
 - Office Dokument meist ge-ZIPt
 - Sandforce SSD Controller komprimieren Daten automatisch
 - NTFS Filesystem automatische Kompression (LZ77 Variante)

Datenkompression - wie?

- **Verlustfreie** Kompression
 - exakte Wiederherstellung des Originals
 - Codierung mit variabler Wortlänge
 - Kompressionsrate typisch 2:1 bis 4:1
 - Achtung: kann zu Datenexpansion führen!
 - Beispiele: Huffman- und arithmetische Codierung, LZ77, LZW
- **Verlustbehaftete** Kompression
 - Original nicht wiederherstellbar (Verlust)
 - Kompressionsrate fast beliebig wählbar (bei Qualitätsverlust!)
 - Ausnutzung menschlicher Wahrnehmungs-Beschränkungen
 - Beispiele: MP3, AAC, MPEG-2, H.264, JPEG, JPEG2000



Codierungen

Beispiel: "Computer"

- mit ASCII Code: jedes Zeichen 8 Bit
→ 8 mal 8 Bit = 64 Bit
- mit Unicode: jedes Zeichen 16 Bit
→ 8 mal 16 Bit = 128 Bit
- eigentlich: nur 8 verschiedene Zeichen, könnte in jeweils 3 Bit codiert werden
→ 8 mal 3 Bit = 24 Bit
 - Kompressionsrate: 62.5% im Vergleich zu ASCII
 - Kompressionsrate: 81.25% im Vergleich zu Unicode

Kompressionsrate

Kompressionsrate

Sei E Grösse der Eingabe, C Grösse der komprimierten Eingabe, dann ist die **Kompressionsrate** CR

$$CR = 1 - \frac{C}{E}$$

- Eingabe wird nicht komprimiert $\rightarrow CR = 0\%$
- Eingabe wird auf ein Drittel komprimiert $\rightarrow CR = 66.6\%$
- Eingabe wird auf 0 komprimiert $\rightarrow CR = 100\%$
- Eingabe wird verlängert $\rightarrow CR < 0$

Codierungen

Beispiel: "Abrakadabra"

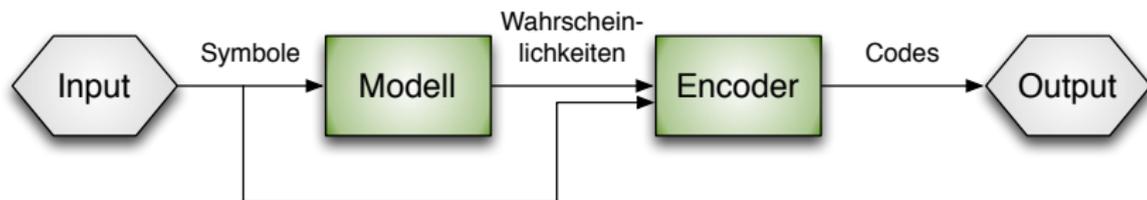
- Vorkommen:

Symbol	Häufigkeit	Huffman-Code
A	5	0
B	2	10
R	2	111
K	1	1100
D	1	1101

- Codierung:

- ASCII: $11 \cdot 8 = 88$ Bits
- Huffman: $5 \cdot 1 + 2 \cdot 2 + 2 \cdot 3 + 1 \cdot 4 + 1 \cdot 4 = 23$ Bits
- Kompressionsrate: $CR = 73.87\%$

Datenkompression



- **Modell:**
 - z.B. einfaches statistisches Modell ohne Gedächtnis
→ Häufigkeiten der Symbole
- **Encoder:**
 - z.B. Huffman-Codierer (s. später)

Programm heute

7 Fortgeschrittene Datenstrukturen

8 Such-Algorithmen

9 Graph-Algorithmen

10 Numerische Algorithmen

11 Datenkompression

Einführung

Grundlagen Informationstheorie

Huffman Codes

Eingabequelle

Eingabequelle ohne Gedächtnis

Wir betrachten eine **Eingabequelle Q ohne Gedächtnis**, die Wörter produziert (also ein Folge von Buchstaben oder Symbolen) mittels einem Alphabet

$$\{a_0, a_1, \dots, a_{N-1}\}$$

von N Symbolen. Wir bezeichnen

$$p_j = p(a_j) = \text{Wahrscheinlichkeit da\ss } a_j \text{ produziert wird}$$

für $j = 0, \dots, N - 1$.

- Beispiele für Alphabet:
 - $\{0, 1\}$ für eine binäre Quelle
 - $\{00000000, 00000001, \dots, 11111111\}$ für eine ASCII Quelle

Wahrscheinlichkeitsverteilung

Wahrscheinlichkeitsverteilung

Sei Q Eingabequelle mit Wahrscheinlichkeiten p_0, \dots, p_{N-1} . Dann heißt

$$\mathbf{p} = (p_0, \dots, p_{N-1})$$

Wahrscheinlichkeitsverteilung von Q .

- Sei $\mathbf{w} = a_{j_1} a_{j_2} \cdots a_{j_n}$ ein Wort der Länge n aus der Quelle Q . Die Annahme, daß Q ohne Gedächtnis ist äquivalent zu

$$p(\mathbf{w}) = p(a_{j_1})p(a_{j_2}) \cdots p(a_{j_n})$$

d.h. die Ereignisse sind unabhängig.

Informationsgehalt

Informationsgehalt

Sei \mathbf{w} Wort aus Quelle Q .

$$I(\mathbf{w}) = \log_2 \left(\frac{1}{p(\mathbf{w})} \right) = -\log_2 p(\mathbf{w})$$

heißt Informationsgehalt von \mathbf{w} .

- $I(\mathbf{w})$ ist invers proportional zu $p(\mathbf{w})$
“je seltener das Wort, desto interessanter ist es”
- es ist $I(a_{j_1} a_{j_2} \cdots a_{j_n}) = I(a_{j_1}) + I(a_{j_2}) + \dots + I(a_{j_n})$,
d.h. der Informationsgehalt eines Wortes ist die Summe der Informationsgehalte seiner Buchstaben

Informationsgehalt: Beispiele

- Informationsgehalt ist über Logarithmus zur Basis 2 definiert
→ Informationseinheit Bit!
- Beispiel Münzwurf:
 - Quelle Q produziert $a_0 = \text{Kopf}$, $a_1 = \text{Zahl}$
 - Wahrscheinlichkeitsverteilung $\mathbf{p} = (p_0, p_1) = (\frac{1}{2}, \frac{1}{2})$
 - dann ist $I(a_0) = I(a_1) = -\log_2 2^{-1} = 1$
 - Codierung z.B. Kopf 0, Zahl 1 (→ also 1 Bit)
- Beispiel Würfel:
 - Quelle Q produziert Zahlen zwischen 0 und 255
(sehr großer Würfel)
 - Wahrsch.-verteilung $\mathbf{p} = (p_0, \dots, p_{255}) = (\frac{1}{256}, \dots, \frac{1}{256})$
 - $I(a_0) = \dots = I(a_{255}) = -\log_2 2^{-8} = 8$
 - → Codierung mit 8 Bits

Entropie

Entropie

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$. Dann heißt der durchschnittliche Informationsgehalt pro Symbol (in Bits pro Symbol)

$$\begin{aligned} H(\mathbf{p}) &:= p_0 I(a_0) + \dots + p_{N-1} I(a_{N-1}) \\ &= -p_0 \log_2 p_0 - \dots - p_{N-1} \log_2 p_{N-1} \end{aligned}$$

Entropie der Quelle Q .

Entropie

Entropie

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$. Dann heißt der durchschnittliche Informationsgehalt pro Symbol (in Bits pro Symbol)

$$\begin{aligned} H(\mathbf{p}) &:= p_0 I(a_0) + \dots + p_{N-1} I(a_{N-1}) \\ &= -p_0 \log_2 p_0 - \dots - p_{N-1} \log_2 p_{N-1} \end{aligned}$$

Entropie der Quelle Q .

Sei das Alphabet $\{a_0, \dots, a_{N-1}\}$ statisch, wir variieren \mathbf{p} .

- $H(\mathbf{p}) = 0 \iff Q$ produziert nur ein Symbol (z.B. a_0)
 - d.h. $p(a_0) = 1$ und $I(a_0) = 0$
- $H(\mathbf{p})$ ist maximal $\iff p_0 = \dots = p_{N-1} = \frac{1}{N}$
 - dann ist $H(\mathbf{p}) = \log_2 N$

Entropie: Beispiel

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_7\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_7)$ mit

$$p_0 = \frac{1}{2}, \quad p_1 = \frac{1}{4}, \quad p_2 = p_3 = \frac{1}{16}, \quad p_4 = p_5 = p_6 = p_7 = \frac{1}{32}$$

- Informationsgehalt:

$$I(a_0) = 1, \quad I(a_1) = 2, \quad I(a_2) = I(a_3) = 4,$$

$$I(a_4) = I(a_5) = I(a_6) = I(a_7) = 5$$

- Entropie:

$$H(\mathbf{p}) = \frac{1}{2}1 + \frac{1}{4}2 + 2 \cdot \frac{1}{16}4 + 4 \cdot \frac{1}{32}5 = 2.125$$

Bits/
Symbol

Entropie-Codierung: Beispiel

Symbol	$p(a_j)$	$I(a_j)$	Codierung
a_0	$\frac{1}{2}$	1	0
a_1	$\frac{1}{4}$	2	10
a_2	$\frac{1}{16}$	4	1100
a_3	$\frac{1}{16}$	4	1101
a_4	$\frac{1}{32}$	5	11100
a_5	$\frac{1}{32}$	5	11101
a_6	$\frac{1}{32}$	5	11110
a_7	$\frac{1}{32}$	5	11111

- ohne Kompression: 8 verschiedene Symbole \rightarrow 3 Bit
 - 10000 Symbole benötigen 30000 Bits
- mit Entropie-Codierung: basierend auf Statistik der Quelle
 - 10000 Symbole benötigen 21250 Bits
- Kompressionsrate $CR = 29.16\%$

Präfixcode

Präfixcode

Ein Code heißt **Präfixcode**, falls kein Codewort Präfix eines anderen Codewortes ist.

- der Entropie-Code der vorigen Folie ist ein binärer Präfixcode
- betrachte den Code $A \mapsto 0$, $B \mapsto 01$, $C \mapsto 10$
 - decodiere 001010: nicht eindeutig!
 - kann sein: *AACC*, *ABAC* oder *ABBA*
 - Code von A ist Präfix von Code von B
- decodiere 0110100111011111110001000 mit Entropie-Code (vorige Folie)
 - decodiert: $a_0 a_3 a_0 a_0 a_5 a_7 a_2 a_0 a_1 a_0 a_0$
- in der Tat: jeder eindeutig zu decodierende Code ist ein Präfixcode (McMillan, 1956)

Kraft Ungleichung

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$.

- **Idee von Shannon (1948):** assoziiere zu jedem a_j einen binären Code mit Länge l_j , so daß

$$l_j = \lceil I(a_j) \rceil \quad \text{für alle } j = 0, \dots, N - 1$$

Kraft Ungleichung

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$.

- **Idee von Shannon (1948):** assoziiere zu jedem a_j einen binären Code mit Länge l_j , so daß

$$l_j = \lceil I(a_j) \rceil \quad \text{für alle } j = 0, \dots, N-1$$

- wann funktioniert das?

Kraft Ungleichung (1949)

Seien l_0, \dots, l_{N-1} vorgegebene Längen für binäre Codewörter. Dann gibt es einen **binären Präfixcode** mit diesen Längen genau dann, wenn

$$\sum_{j=0}^{N-1} 2^{-l_j} \leq 1.$$

Quellencodierungssatz von Shannon

Quellencodierungssatz von Shannon (1948)

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$. Sei C ein assoziierter binärer Präfixcode mit durchschnittlicher Länge der Codewörter $\bar{l} = \sum_{j=0}^{N-1} p_j l_j$ (in Bits pro Symbol). Dann gilt:

$$H(\mathbf{p}) \leq \bar{l}.$$

- in anderen Worten: es gibt keine verlustfreie Kompression unter der Entropie.
- für Code, der nach Shannon's Idee erstellt wurde, gilt:

$$\bar{l} < H(\mathbf{p}) + 1.$$

Programm heute

7 Fortgeschrittene Datenstrukturen

8 Such-Algorithmen

9 Graph-Algorithmen

10 Numerische Algorithmen

11 Datenkompression

Einführung

Grundlagen Informationstheorie

Huffman Codes

Algorithmus: Shannon-Fano Code

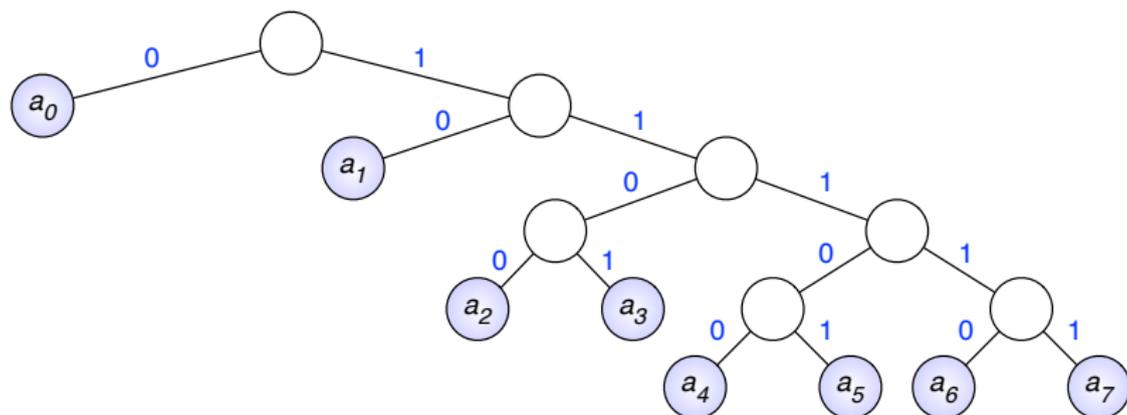
Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$.

- 1 Sortiere Symbole absteigend in eine Liste gemäss ihrer Häufigkeit
- 2 Unterteile Liste in zwei Teile, so daß die Summe der Häufigkeiten des ersten Teils möglichst nahe an der Summe der Häufigkeiten im zweiten Teil ist
- 3 Der erste Teil erhält 0 als Binärziffer, der zweite Teil 1
- 4 Wende Schritt 2 und 3 rekursiv an
- 5 Die sich ergebenden Folgen von Binärziffern sind der Shannon-Fano Code

Shannon-Fano Code: Beispiel

Symbol	Häufigkeit	Code
a_0	16	0
a_1	8	10
a_2	2	1100
a_3	2	1101
a_4	1	11100
a_5	1	11101
a_6	1	11110
a_7	1	11111

Präfixcode als Baum



- **Codieren:** finde Pfad von Wurzel zu Blatt a_j
- **Decodieren:** durchlaufe Baum von Wurzel gemäß Code

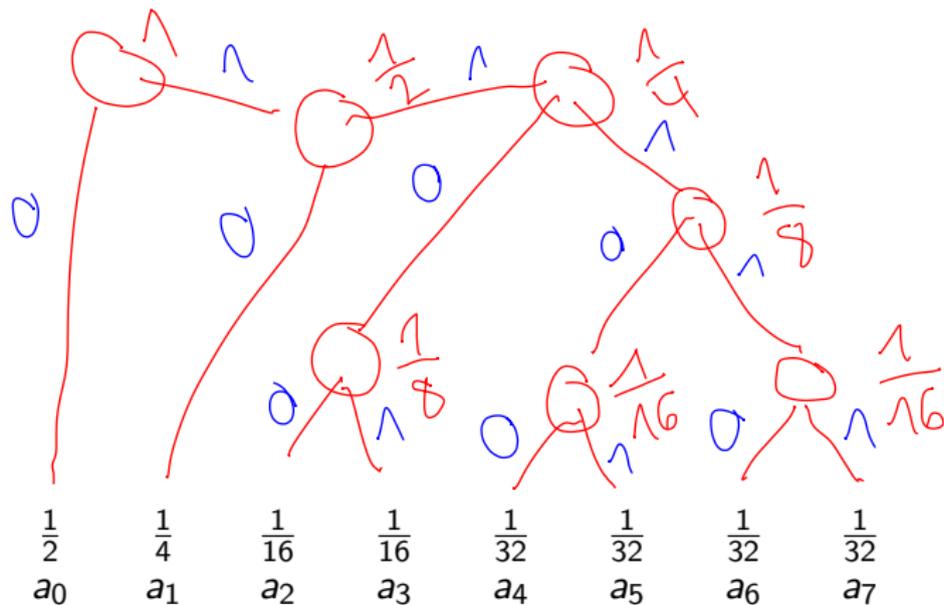
Algorithmus: Huffman-Baum

Sei Q Quelle mit Alphabet $\{a_0, \dots, a_{N-1}\}$ und Verteilung $\mathbf{p} = (p_0, \dots, p_{N-1})$.

- 1 Erzeuge eine Liste von Blättern mit je ein Blatt pro a_j mit Gewicht p_j für $j = 0, \dots, N - 1$
- 2 Für die zwei Knoten mit geringstem Gewicht erzeuge Vaterknoten mit Summe der Gewichte
- 3 Ersetze die zwei Knoten in Liste mit Vaterknoten
- 4 Wiederhole Schritt 2 und 3 bis nur 1 Knoten (die Wurzel) übrig ist

(weiteres Beispiel für Greedy-Algorithmus)

Huffman-Code: Beispiel



- decodiere: 01101001110111111110001000

$a_0 a_3 a_0 a_0 a_5 a_7 a_2 a_0 a_1 a_0 a_0$

Huffmann vs. Shannon-Fano

- **Shannon-Fano:** Baum wird von Wurzel abwärts erstellt
- **Huffman:** Baum wird von Blättern aufwärts erstellt
 - Huffman ist optimaler Code mit ganzzahliger Anzahl von Bits

Beispiel:

Symbol	a_0	a_1	a_2	a_3	a_4
Häufigkeit	15	7	6	6	5
p_j	0.385	0.179	0.154	0.154	0.128
Shannon-Fano	00	01	10	110	111
Huffman	0	100	101	110	111

Durchschnittliche Codelänge:

- **Shannon-Fano:**
 $\bar{l} = (2 \cdot (15 + 7 + 6) + 3 \cdot (6 + 5)) / 39 \approx 2.28 \text{ Bits/Symbol}$
- **Huffman:**
 $\bar{l} = (1 \cdot 15 + 3 \cdot (7 + 6 + 6 + 5)) / 39 \approx 2.23 \text{ Bits/Symbol}$

Huffman in der Praxis

- **Codieren:** Häufigkeiten / Wahrscheinlichkeiten müssen zuerst erzeugt werden
 - Quelle muß zweimal gelesen werden
- **Decodieren:** Baum muß zuerst übertragen werden
 - z.B. als Liste der Wahrscheinlichkeiten (kostet Platz!)
 - anderer Ansatz: starte mit Gleichverteilung, adaptiere Wahrscheinlichkeiten beim Lesen von Quelle

Ausblick: Codierung

- Codieren mit nicht-ganzzahliger Anzahl von Bits
 - z.B. mit geschachtelter Intervall-Unterteilung
 - → Arithmetisches Codieren
- bisherige Annahme: jedes Symbol ist unabhängig von den anderen Symbolen
 - in Text ist diese Annahme meist falsch!
 - weiteres Ausnutzen von Redundanzen durch Statistiken von Symbol-Paaren, -Tripeln etc.
 - sog. höhere statistische Modelle

Zusammenfassung

7 Fortgeschrittene Datenstrukturen

8 Such-Algorithmen

9 Graph-Algorithmen

10 Numerische Algorithmen

11 Datenkompression

Einführung

Grundlagen Informationstheorie

Huffman Codes