Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Proseminar Datenkompression im WS 2004/2005

Huffman-Kodierung

Engelmann, Viktor

Zimmermann, Martin

16. November 2004

Huffman-Kodierung

- Einführung und Motivation
- Grundlagen
  - Definitionen
  - Informationstheorie
  - Vorgänger der Huffman-Kodierung
- Huffman-Kodierung
  - Funktionsweise
  - Komplexität
  - Beweis der Optimalität
- Modifikationen
  - Nicht-binäre Bäume
  - Größere Wortlänge
  - Adaptive Huffman-Kodierung
- Zusammenfassung

Literatur

- Rudolf Mathar. Informationstheorie. Verlag der Augustinus Buchhandlung, 1993.
- Khalid Sayood. Introduction to data compression. Morgan Kaufmann, 2000.
- www.huffmancoding.com.

Einführung und Motivation

Motivation

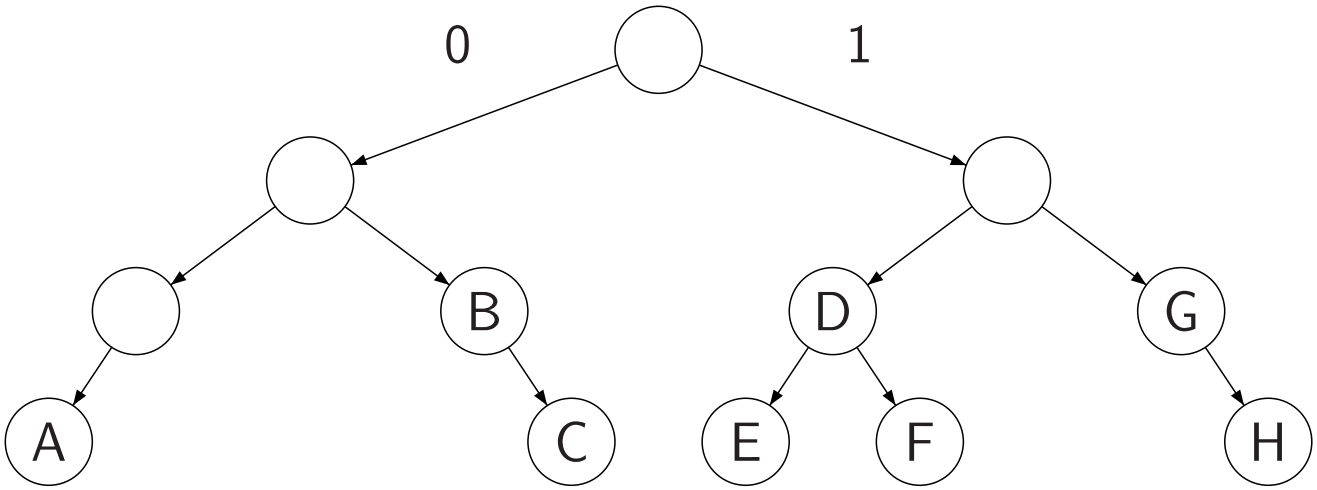
- Unkomprimierte Bilder benötigen mit einer Auflösung von 1024 x 768 Pixeln und 24 Bit Farbtiefe 2,25 Megabyte
- Eine CD mit unkomprimierten Audio-Daten benötigt 1100% des Speicherplatzes einer MP3 in CD-Qualität
- Ohne Kompression passen auf eine DVD mit einer Kapazität von 8,5 Gigabyte weniger als 5 Minuten Film ohne Ton

Einleitung

- Ein **Kompressions-Algorithmus** bildet eine Zeichenfolge A über einem Quellalphabet Σ auf eine kürzere Zeichenfolge B über einem Kodealphabet Y ab
- Kann man A aus B vollständig rekonstruieren, so heißt die Kompression **verlustlos**
- Kann man aus B eine Zeichenfolge $A' \simeq A$ konstruieren, so heißt die Kompression **verlustbehaftet**
- Bei der Kodierung von Zeichen können ihnen Worte variabler Länge über dem Kodealphabet zugewiesen werden

Darstellung einer Kodierung als Baum

- Bezeichne die Kanten mit Symbolen aus dem Kodealphabet
- Der Pfad zu einem Knoten entspricht der Kodierung des Symbols in dem Knoten



A$\widehat{=}$000, B$\widehat{=}$01, C$\widehat{=}$011, D$\widehat{=}$10, E$\widehat{=}$100, F$\widehat{=}$101, G$\widehat{=}$11, H$\widehat{=}$111

BH$\widehat{=}$01 111=01111=011 11$\widehat{=}$CG

Genau dann, wenn alle Symbole in Blättern liegen, ist die Kodierung präfixfrei

Präfixfreie Kodierungen sind eindeutig dekodierbar

- Die **Eigeninformation** eines Ereignisses, z.B. das Auftreten eines Symbols a_i in einem Wort A der Länge n, ist

$$i(a_i) := \log_d \left(\frac{1}{p_i} \right) = -\log_d (p_i)$$

  und entspricht der Länge einer optimalen Kodierung für a_i

- p_i ist hierbei die relative Häufigkeit von a_i in A, $p_i \cdot n$ ist die absolute Häufigkeit. Daher ist $n \cdot p_i \cdot i(a_i)$ die Länge einer optimalen Kodierung aller a_i

- Summiert man diesen Wert für das gesamte Quellalphabet und teilt durch n, erhält man

$$H(A) := -\sum_{i=1}^{k} p_i \cdot \log_d(p_i)$$

  Dieser Wert heißt die **Entropie** des Wortes A und entspricht der durchschnittlichen Kodewortlänge für jedes Symbol aus A in einer optimalen Kodierung und damit der erreichbaren Kompressionsrate

Satz von McMillan, Satz von Kraft

- Satz von McMillan (1959)

  Für alle eindeutig dekodierbaren Kodierungen über einem Kodierungsalphabet mit d Symbolen mit Kodewortlängen $(n_1, n_2, \ldots, n_k)$ gilt

$$\sum_{i=1}^{k} d^{-n_i} \leq 1$$

- Satz von Kraft (1949)

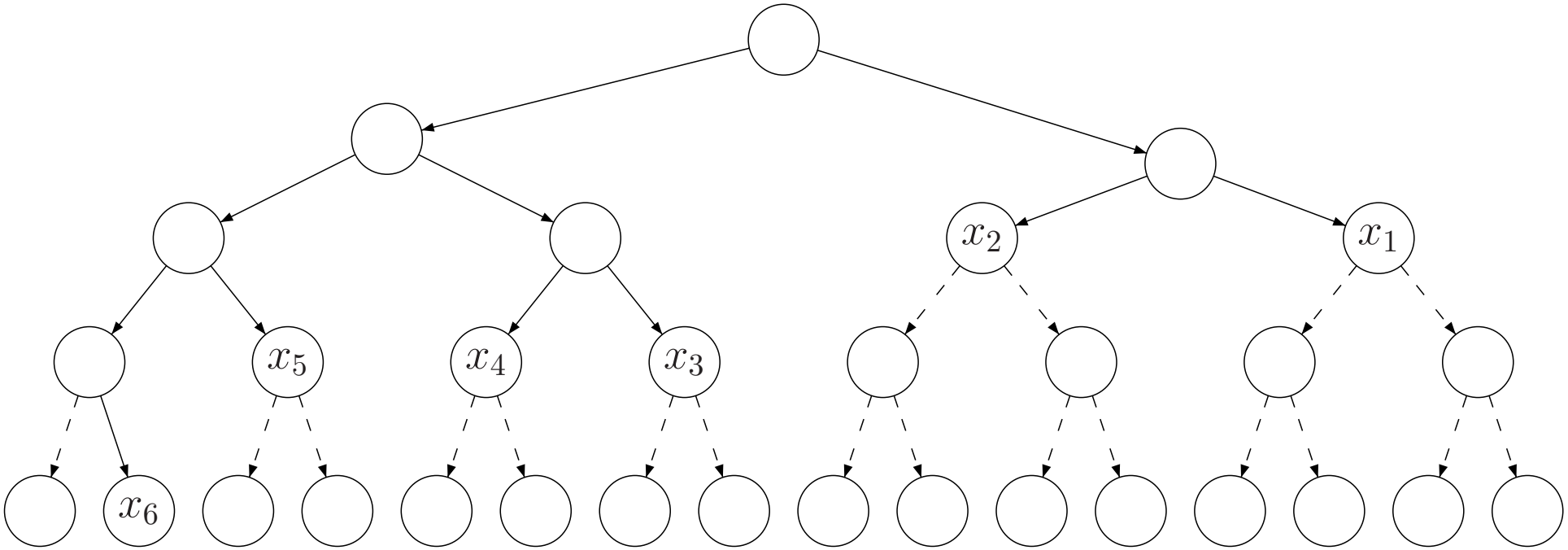
  Gilt $\sum_{i=1}^{k} d^{-n_i} \leq 1$ für natürliche Zahlen $(n_1, n_2, \ldots, n_k)$, so existiert ein präfixfreier (also eindeutig dekodierbarer) Kode mit Kodewortlängen $(n_1, n_2, \ldots, n_k)$

Beispiel zum Satz von Kraft

Binärkodierung der Symbole aus $\Sigma = \{x_1, x_2, \ldots, x_6\}$

mit Kodewortlängen $n_1 = n_2 = 2, n_3 = n_4 = n_5 = 3, n_6 = 4$

$$\sum_{i=1}^{6} 2^{-n_i} = \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} = \frac{15}{16} \leq 1$$



Satz von Shannon (1948): Noiseless Coding Theorem

Sei $A = (a_1, a_2, \ldots, a_n)$ ein Wort über $\Sigma = \{s_1, s_2, \ldots, s_k\}$. Die Länge einer optimalen Kodierung über einem Kodealphabet mit d Symbolen von A hat eine durchschnittliche Kodewortlänge von

$$H(A) \leq \sum_{i=1}^{k} p_i \cdot l_i < H(A) + 1$$

wobei p_i, $i = 1 \ldots k$ die relative Häufigkeit des Symbols s_i in A und l_i für $i = 1 \ldots k$ die Länge des Kodewortes für s_i ist

Beweis

$$H(A) - \sum_{i=1}^{k} l_i \cdot p_i = \frac{1}{\ln d} \cdot \sum_{i=1}^{k} p_i \cdot \ln\left(\frac{d^{-l_i}}{p_i}\right) \leq \frac{1}{ln(d)} \cdot \sum_{i=1}^{k} p_i \left(\frac{d^{-l_i}}{p_i} - 1\right)$$

$$= \frac{1}{\ln d} \cdot \left(\left(\sum_{i=1}^{k} d^{-l_i}\right) - \left(\sum_{i=1}^{k} p_i\right)\right) \leq 0$$

weil nach Satz von McMillan $\sum_{i=1}^{k} d^{-l_i} \leq 1$ gilt

Wähle l_i: $-\log_d(p_i) \leq l_i < -\log_d(p_i) + 1 \Leftrightarrow d^{-l_i} \leq p_i < d^{-l_i+1}$

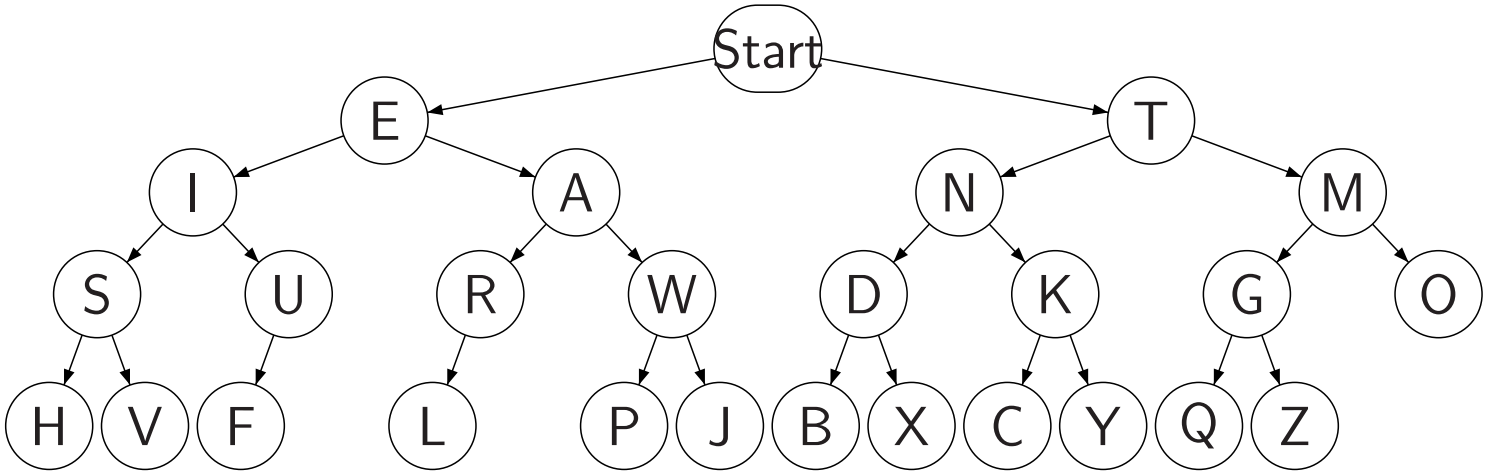
$\Rightarrow \sum_{i=1}^{k} d^{-l_i} \leq \sum_{i=1}^{k} p_i = 1$, also ex. nach Satz von Kraft ein solcher Kode

$$p_i < d^{-l_i+1} \Leftrightarrow \log_d(p_i) < (-l_i + 1) \Rightarrow \sum_{i=1}^{k} p_i \cdot \log_d(p_i) < \sum_{i=1}^{k} p_i \cdot (-l_i + 1)$$

$$\Leftrightarrow H(A) + 1 > \sum_{i=1}^{k} p_i \cdot l_i$$

Der Morse-Kode

- Mitte des 18. Jahrhunderts zum Telegraphieren entwickelt
- Nicht präfixfrei: Drei Zustände zur Beschreibung
  - *kurz:* linker Sohn
  - *lang:* rechter Sohn
  - *aus:* Pause zwischen *kurz* und *lang*
  - *aus aus aus:* Ende des Pfades



Darstellung des Morse-Kode mit 0 und 1

Es gibt zwei Möglichkeiten der Kodierung mit $\Sigma = \{0, 1\}$

1. 1:=kurz, 0:=lang: Vor jedes Bit 1 setzen, 0 zeigt Pfadende an
   also 11=kurz, 10 =lang, 0=Ende
2. 1:=an, 0=aus: 1=kurz, 11=lang, 0 beschreibt Ende
   also 10=kurz, 110=lang, 0=Ende

Kodierung des A:

1. 10 11 0: weitergehen links, weitergehen rechts, Ende
2. 10 110 0: (an,aus)=links, (an,an,aus)=rechts, (aus)=Ende

Funktionsweise der Shannon-Fano-Kodierung

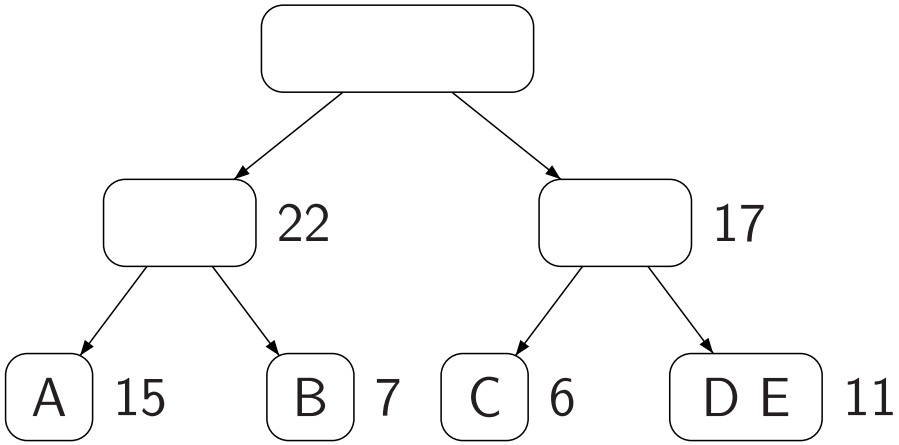
Gegeben ein Wort A über $\Sigma = \{s_1, s_2, \ldots, s_k\}$. Kodealphabet $\{0, 1\}$

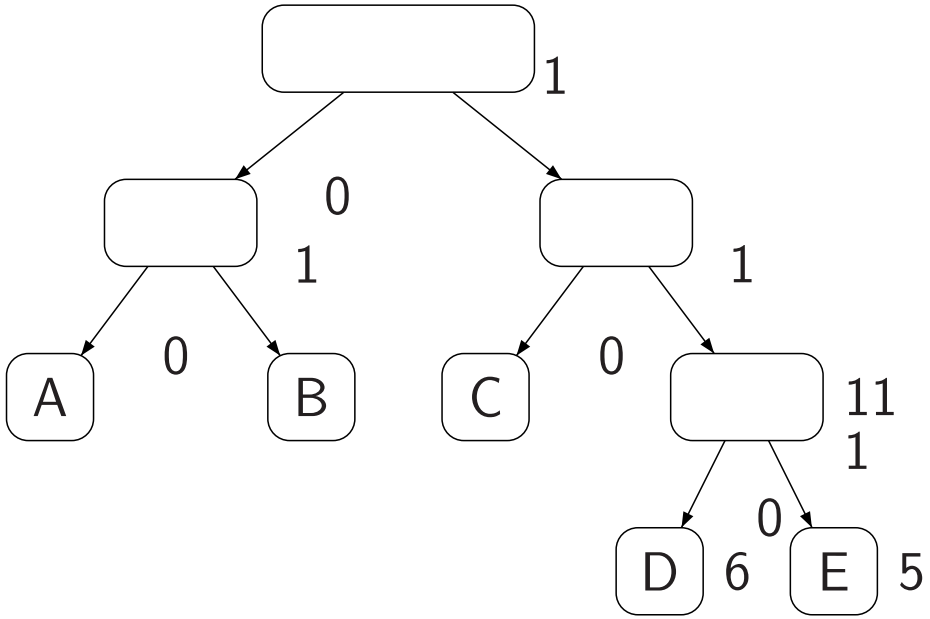
- Für jedes $s \in \Sigma$ bilde einen Knoten, speichere in ihm die Häufigkeit von s in A
- Sortiere die Knoten nach Häufigkeit
- Teile die Menge der Knoten in zwei Gruppen, deren addierte Häufigkeit möglichst gleich groß ist. Bilde einen neuen Knoten, die erste Gruppe kommt in den linken Teilbaum, die zweite in den rechten Teilbaum
- Wiederhole den letzten Schritt mit den Gruppen in den Teilbäumen, bis die Gruppen ein-elementig sind
- linker Sohn = 0, rechter Sohn = 1. Die Kodierung ist der Weg zum Knoten

Beispiel zur Shannon-Fano-Kodierung

A: 15, B: 7, C: 6, D: 6, E: 5

Beispiel zur Shannon-Fano-Kodierung





A$\widehat{=}$00, B$\widehat{=}$01, C$\widehat{=}$10, D$\widehat{=}$110, E$\widehat{=}$111

Die durchschnittliche Kodewortlänge beträgt $2,28$ Bits.

Huffman-Kodierung

Funktionsweise der Huffman-Kodierung

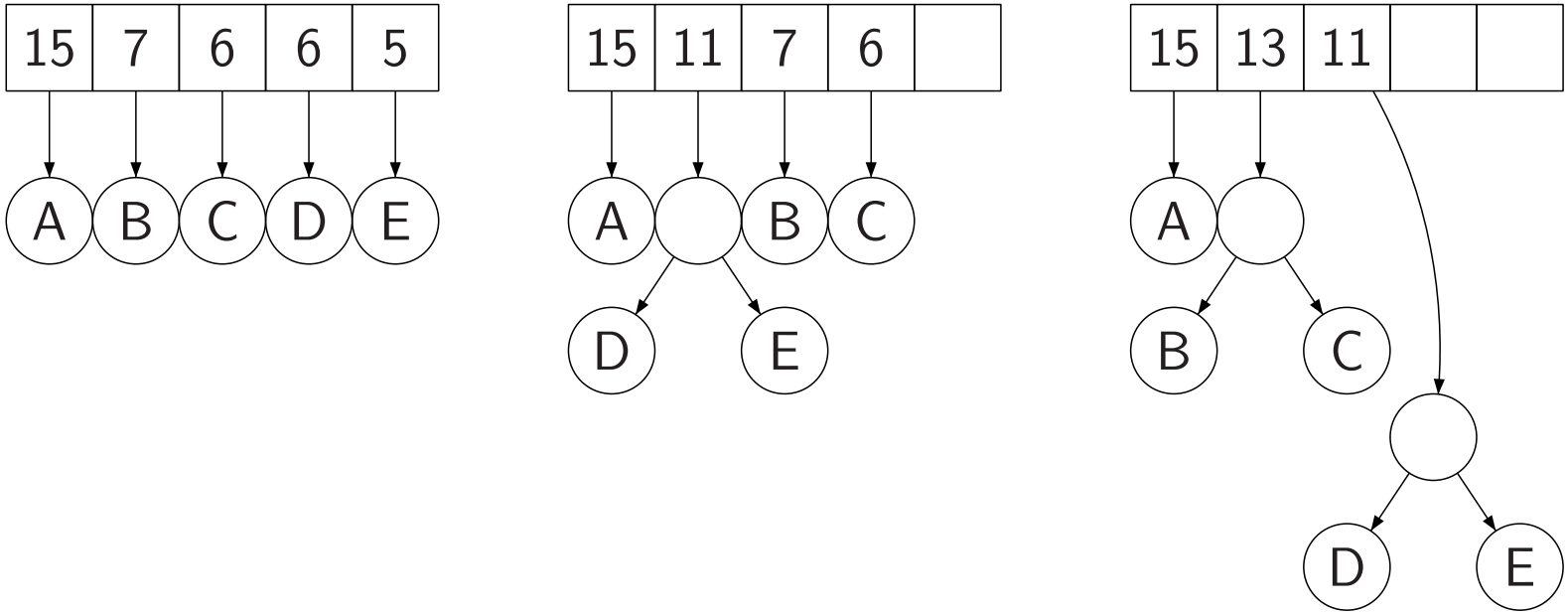
Gegeben ein Wort A über $\Sigma = \{s_1, s_2, \ldots, s_k\}$.

- Bilde ein Array der Länge k aus Knoten, die ein $s \in \Sigma$ und dessen Häufigkeit in A speichern
- Sortiere das Array nach der Häufigkeit, lösche Knoten mit Häufigkeit 0
- Erstelle einen neuen Knoten. Setze die zwei Knoten mit den geringsten Häufigkeiten als seine Söhne und sortiere den Knoten mit der addierten Häufigkeit seiner Söhne in das Array ein
- Wiederhole den letzten Schritt, bis nur noch ein Baum im Array existiert

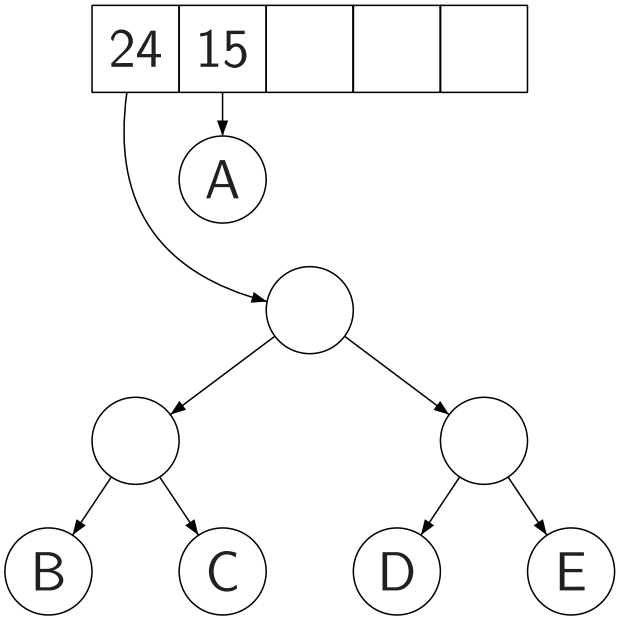
Bezeichne die Kanten mit den Symbolen aus dem Kodealphabet. Der Pfad zu einem Symbol ist seine Kodierung

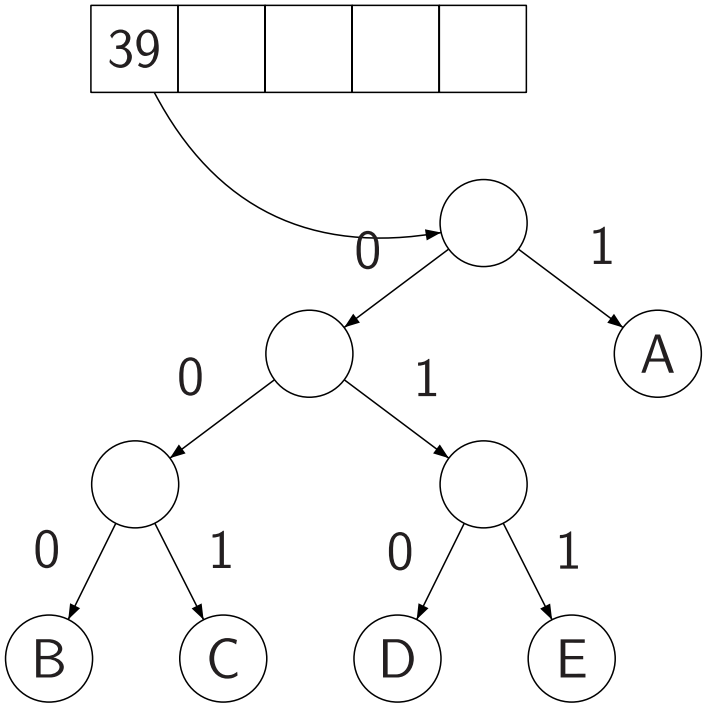
Beispiel zur Huffman-Kodierung

A: 15, B: 7, C: 6, D: 6, E: 5



Beispiel zur Huffman-Kodierung





A$\widehat{=}$1, B$\widehat{=}$000, C$\widehat{=}$001, D$\widehat{=}$010, E$\widehat{=}$011

Die durchschnittliche Kodewortlänge beträgt $2,23$ Bits.

Laufzeitkomplexität

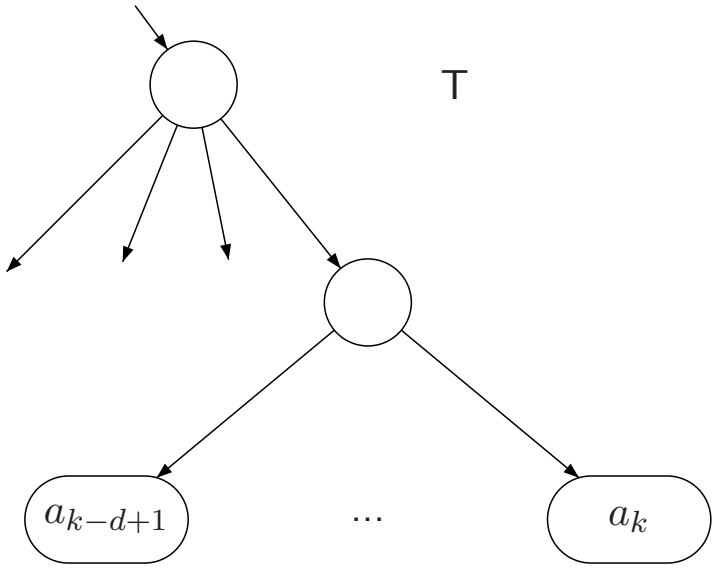
- Einlesen der Datei und bestimmen der Häufigkeit
- Sortieren des Array mit Merge-Sort
- Aufbau des Huffman-Baumes
- Auslesen des Baumes
- Ausgabe der Daten

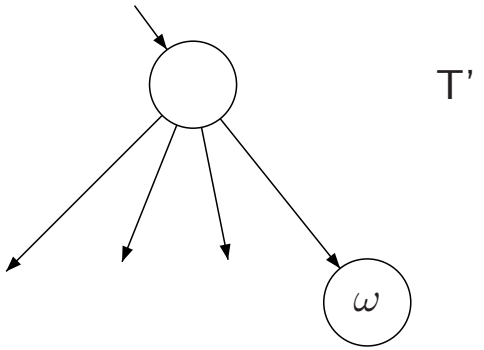
$$O\left(\underbrace{n}_{\text{einlesen}} + \underbrace{k \cdot \log_2(k)}_{\text{sortieren}} + \underbrace{\frac{k^2 + (2-d)\cdot k + d - 3}{2 \cdot d - 2}}_{\text{Baum generieren}} + \underbrace{k^2 + k + \frac{k-1}{d-1}}_{\text{Kode auslesen}} + \underbrace{n \cdot H(A) + n}_{\text{ausgeben}}\right)$$

$$= O(n + k^2)$$

Beweis der Optimalität: Zusätzliche Definitionen

Ist T ein Kodebaum eines Kodes für ein Wort A, so bezeichne T' den Baum, der entsteht indem die d seltensten Symbole zu einem neuen Symbol ω zusammengefügt werden. T' beschreibt eine Kodierung für ein modifiziertes Wort A', in dem alle $a_{k-d+1} \dots a_k$ durch ω ersetzt sind. Dann ist $P(\omega) = \sum_{i=k-d+1}^{k} p_i$





Beweis der Optimalität: Bemerkungen

$B(T) := \sum_{t \in T} l_t \cdot p_t$ ist die durchschnittliche Kodelänge der Kodierung, die der Baum T beschreibt

- Bemerkung 1: Ein Baum T ist genau dann Kodebaum eines optimalen Kodes für ein Wort A, wenn $B(T) = H(A)$ gilt
- Bemerkung 2:

$$B(T') = B(T) - \sum_{i=k-d+1}^{k} p_i \cdot l_i + \sum_{i=k-d+1}^{k} p_i \cdot (l_i - 1) = B(T) - P(\omega)$$

- Bemerkung 3: Aus $p_i \leq p_j$ folgt in einer optimalen Kodierung $l_i \geq l_j$. Daher können die Symbole $a_1 \ldots a_k$ so nummeriert werden, dass gilt:

$$p_1 \geq p_2 \geq \ldots \geq p_{k-1} \geq p_k \text{ und } l_1 \leq l_2 \leq \ldots \leq l_{k-1} \leq l_k$$

Beweis der Optimalität: Lemmata

Sind $a_{k-d+1} \dots a_k$ die Symbole mit der geringsten Häufigkeit, dann existiert ein optimaler Kodebaum T, in dem sie Brüder sind

Beweisidee: a_k ist niedrigstes Blatt, seine Brüder müssen ebenfalls Blätter sein. Ist ein a_i Bruder von a_k mit $p_i > p_{k-d+1}$ und p_j nicht Bruder von a_k aber $p_{k-d+1} \geq p_j \geq p_k$. Vertausche a_i und a_j, der resultierende Baum beschreibt eine bessere Kodierung

Ist T ein optimaler Baum für ein Wort A über einem Alphabet Σ, so ist T' ein optimaler Baum für A' über Σ'

Beweisidee: Wäre T' kein optimaler Baum für A', dann existiert ein Baum $T^{*'}$ mit $B(T^{*'}) < B(T')$. Wird ω in $T^{*'}$ durch $a_{k-d+1} \dots a_k$ ersetzt, so ergibt sich T^* mit

$$B(T^*) = B(T^{*'}) + P(\omega) < B(T') + P(\omega) = B(T) + P(\omega) - P(\omega)$$

$\Rightarrow B(T^*) < B(T)$ also ist T nicht optimal. Widerspruch

Beweis der Optimalität: Lemmata

Ist T ein Huffman-Baum für ein Wort A über einem Alphabet Σ, so ist T' ein Huffman-Baum für A' über Σ', weil der Algorithmus genau diese Ersetzung durchführt

Hat ein Kodebaum T eine Höhe h und hat ein innerer Knoten k auf einer Höhe $l_k < h - 1$ weniger als d Söhne, so beschreibt T keinen optimalen Kode

Beweisidee: Wird ein niedrigstes Blatt b von der Höhe h als Nachfolger von k gesetzt, entsteht T^* mit

$$B(T^*) = B(T) - P(b) \cdot h + P(b) \cdot (l_k + 1) = B(T) + P(b) \cdot \underbrace{(l_k + 1 - h)}_{<0}$$

$\Rightarrow B(T^*) < B(T)$ also ist T nicht optimal.

Beweis der Optimalität

Fall $k = |\Sigma| \in \{(d-1) \cdot \alpha + 1 \mid \alpha \in \mathbb{N}\}$

Induktionsanfang: $k = d$: trivial

Induktionsannahme: Ein Huffman-Baum für ein Wort über einem Alphabet mit k Knoten ist optimal

Induktionsschritt: $k \to k + d - 1$: Ist ein Huffmanbaum T mit $k + d - 1$ Blättern nicht optimal, existiert T^* mit $B(T^*) < B(T)$. Da für beide Bäume die Summe der Häufigkeiten der seltensten Symbole $(P(\omega))$ übereinstimmt, ist

$$B(T^{*'}) = B(T^*) - P(\omega) < B(T) - P(\omega) = B(T')$$

Also ist T' entgegen der Induktionsannahme ein Huffmanbaum mit k Knoten, der nicht optimal ist

Beweis der Optimalität

Fall $k = |\Sigma| \in \mathbb{N} \setminus \{(d-1) \cdot \alpha + 1 \mid \alpha \in \mathbb{N}\}$

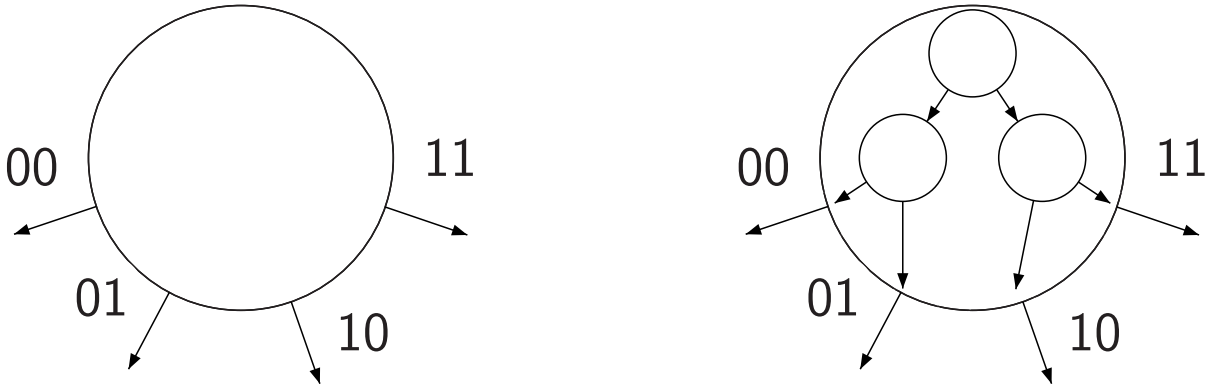
Es sei A ein Wort über einem Alphabet Σ.
Σ^* sei ein Alphabet, das aus Σ durch Ergänzung von Symbolen, die nicht in Σ vorkommen, entsteht, wobei $k^* = |\Sigma^*| \in (d-1) \cdot \alpha + 1$, $\alpha \in \mathbb{N}$ gelten soll.
T^* sei der Huffman-Baum für A über Σ^*.
T sei nun der Baum, der durch Entfernung der ergänzten $k^* - k$ Symbole aus T^* entsteht. Wie bewiesen ist T^* ein optimaler Kodebaum für A

$$B(T^*) = \sum_{i=1}^{k} p_i \cdot l_i + \sum_{i=k+1}^{k^*} \underbrace{p_i}_{=0} \cdot \underbrace{l_i}_{<\infty} = B(T)$$

Modifikationen

Verwendung nicht-binärer Bäume

- Verwendung eines Kodealphabets mit d Zeichen $\Leftrightarrow$ Verwendung von Knoten mit d Söhnen im Huffman-Baum
- d als Zweierpotenz wählen, um alle mit Bits kodierbaren Wege zu nutzen
- Die Nachfolger eines Knotens werden mit Binärbäumen im Knoten ausgewählt



Größere Wortlänge

- Einlesen der Datei in Blöcken der Länge q $\Rightarrow$ nur noch $\left\lceil \frac{n}{q} \right\rceil$ Kodewörter nötig
- Algorithmus muss nur zum Einlesen der Blöcke geändert werden
- Anzahl der verschiedenen Blöcke ist durch $\left\lfloor \frac{n}{q} \right\rfloor \leq k^q$ begrenzt
- Kodelänge wird für jedes q kürzer
- Im worst-case muss der gesamte Text im Kode-Baum gespeichert werden

Beispiel zur Größeren Wortlänge

Die Kodierung des Wortes

acadcadaaccadadcccbcddcbdcdddccbdcbabbccdbccbcddccbabcddcbaccbd
dccccdddcbaacddccabcadcacccbacccbdadccbadaccbcbdcbddadcddccbabd
bddadcddacddcaadacacbddadababdacdadcdaddacbddadadcbcddcbddccd
ddadddcdacbddaddaacdddddccddcbddadaddababdaccaacdddccbacddcbcb

ist nach zeichenweiser Huffman-Kodierung 569 Bits lang und der dazugehörige Kodebaum hat 4 Blätter für die Symbole und 3 innere Knoten.

Wird die Datei in Blöcken der Länge 2 eingelesen und kodiert, ist die Kodierung dagegen nur 532 Bits lang. Der Kodebaum hat 16 Blätter und 15 innere Knoten.

Adaptive Huffman-Kodierung

Huffman-Kodierung ist in der Praxis nicht immer einsetzbar, da die Häufigkeiten vor der Kodierung bekannt sein müssen
Problematisch zum Beispiel bei der Kodierung einer stetigen Datenübertragung

Zwei mögliche Modifikationen des Algorithmus zur Erstellung eines Baumes

- statische Huffman-Kodierung: Verwendung eines anhand von Vorüberlegungen oder Statistiken erstellten Baumes
- adaptive Huffman-Kodierung: Verwendung eines dynamischen Baumes, der nach jedem kodierten Zeichen geändert wird

- Kodierer und Dekodierer haben jeweils ihren eigenen Kode-Baum, die immer identisch sein müssen
- Wird ein Zeichen zum ersten mal kodiert, müssen die Bäume erweitert werden
- Wird ein bereits enthaltenes Zeichen kodiert, müssen die Bäume aktualisiert werden

Ordnung des Kode-Baumes

- Der rechte Sohn eines Knotens muss eine höhere Häufigkeit als der linke haben. Sonst werden die Söhne einschließlich ihrer Teilbäume miteinander getauscht
- Ein Knoten muss eine geringere Häufigkeit als die Knoten der höheren Ebenen haben. Sonst werden die Knoten einschließlich ihrer Teilbäume miteinander getauscht

Funktionsweise Adaptive Huffman-Kodierung

Beide Bäume haben zu Beginn nur eine Wurzel, den „NYT“-Knoten (Not Yet Transmitted). Soll nun ein Zeichen kodiert werden, sucht der Kodierer in seinen Baum nach dem Zeichen

Das Zeichen ist bereits im Baum enthalten

- Die Kodierung des Zeichens wird übertragen
- In beiden Bäumen wird die Häufigkeit auf den Pfaden zu dem Symbol um 1 erhöht
- Gegebenenfalls muss die Ordnung wiederhergestellt werden

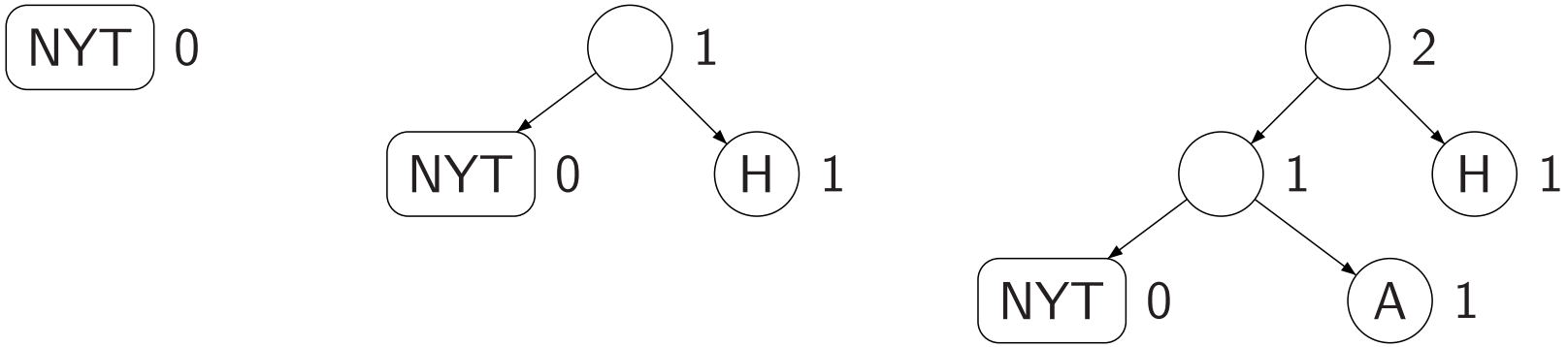
Funktionsweise Adaptive Huffman-Kodierung

Das Zeichen ist nicht im Baum enthalten

- Der Kode des „NYT“-Knoten wird übertragen, gefolgt vom unkodierten Zeichen
- Der „NYT“-Knoten wird in beiden Bäumen durch einen inneren Knoten ersetzt. Dessen linker Sohn wird der neue „NYT“-Knoten und der rechte speichert das neue Symbol
- Die Häufigkeiten auf dem Pfad zum neuen Symbol-Knoten werden um 1 erhöht
- Gegebenenfalls muss die Ordnung wiederhergestellt werden

Beispiel Adaptive Huffman-Kodierung

Die Zeichenfolge HALLO soll kodiert und übertragen werden

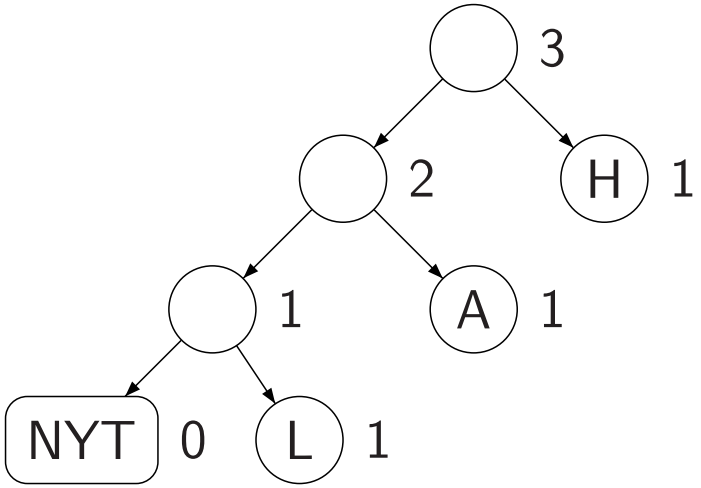


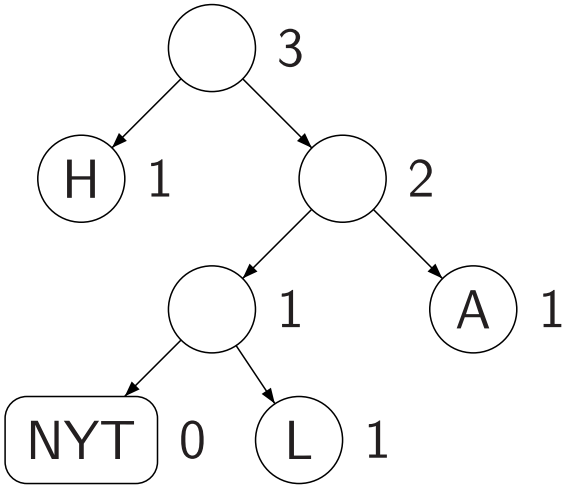
Übertragung: H

Übertragung: 0 A

Übertragung: 00 L

Beispiel Adaptive Huffman-Kodierung

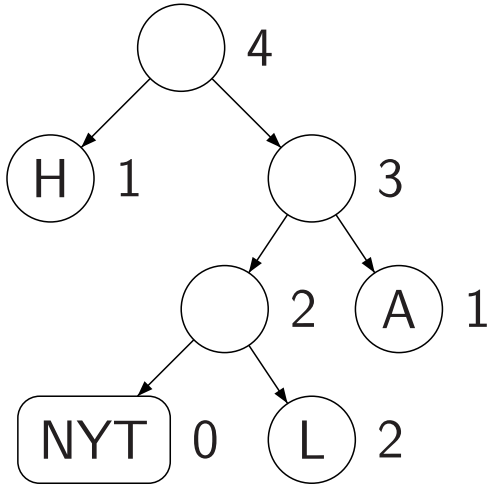


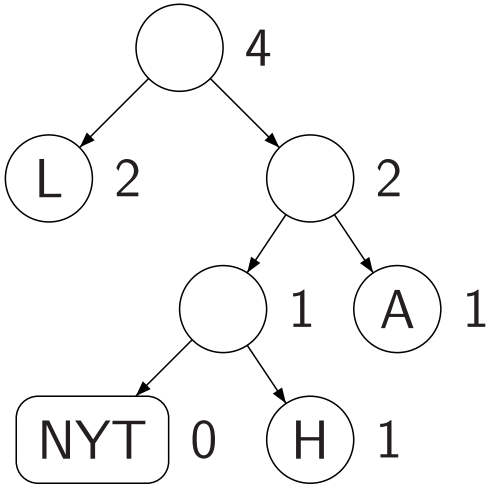


Übertragung: 101

Wiederherstellung der Ordnung: Söhne des Knotens mit der Häufigkeit 2 tauschen

Beispiel Adaptive Huffman-Kodierung

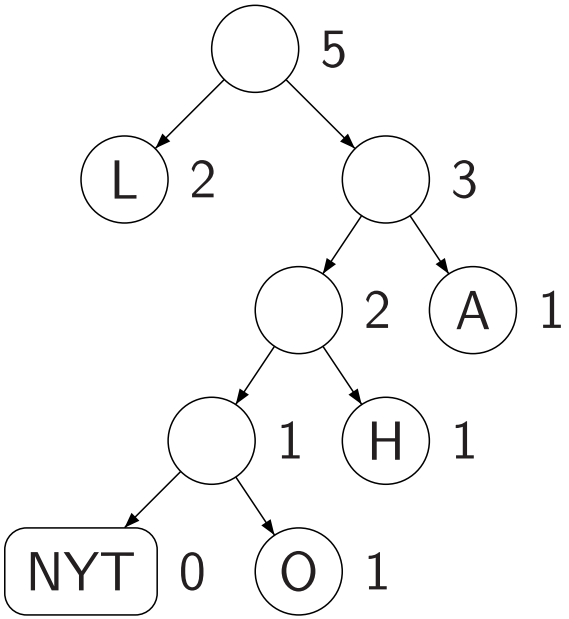


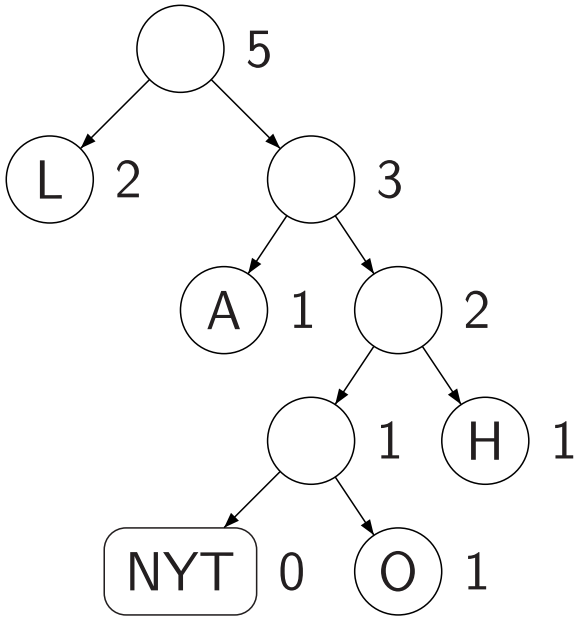


Übertragung: 100 O

Wiederherstellung der Ordnung: H und L tauschen

Beispiel Adaptive Huffman-Kodierung





Wiederherstellung der Ordnung: Söhne des Knotens mit der Häufigkeit 3 tauschen

Zusammenfassung

- Die Huffman-Kodierung ist eine präfixfreie, im Sinne der Entropie optimale Kodierung
- Sie baut, im Gegensatz zur Shannon-Fano-Kodierung, den Kodebaum von unten nach oben auf, indem die seltensten Knoten zu einem Baum zusammengefasst werden und dieser wieder einsortiert wird
- durchschnittliche Kodewortlänge in dem Beispiel: Shannon-Fano-Kodierung 2,28 Bits, Huffman-Kodierung 2,23 Bits
- Komplexität linear in der Länge der Eingabedatei und quadratisch in der Größe des Quellalphabets
- Varianten:
  - nicht-binäre Bäume
  - größere Wortlänge
  - adaptive Kodierung