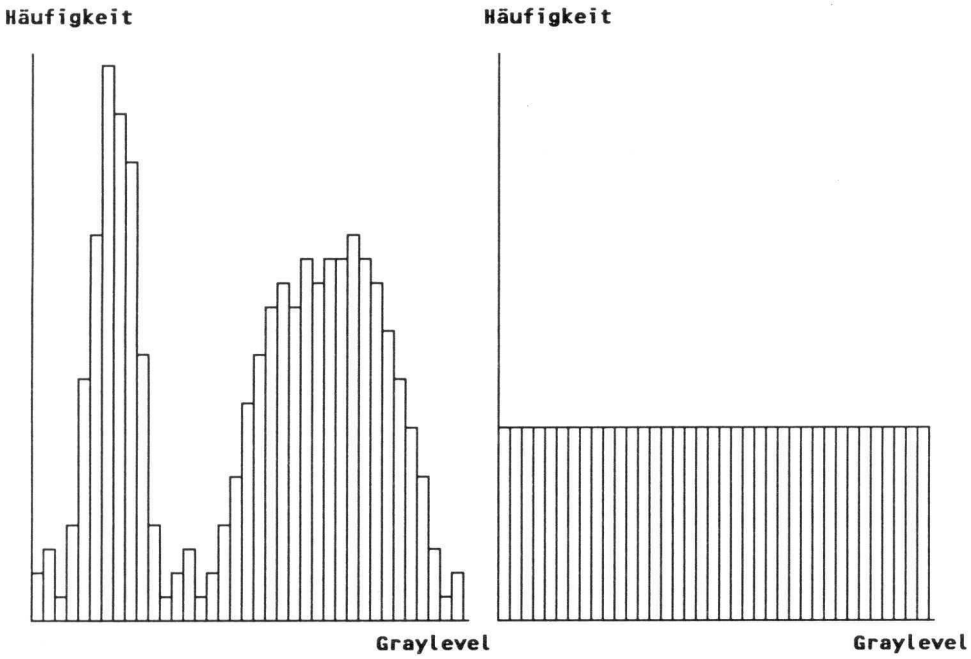
Einführung in die Bilddaten-kompression mittels arithmethischer Kodierung

H. Thomas

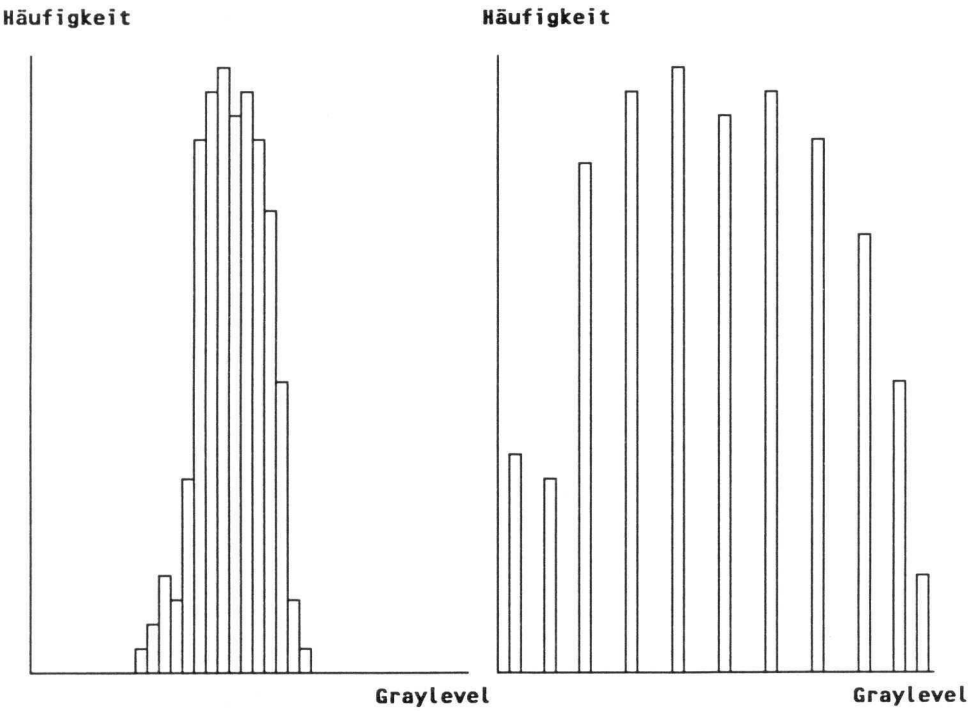
Inhalt

- **Summenhäufigkeit in der Bildverarbeitung**
- **Information und Wahrscheinlichkeit**
- **Arithmetische Kodierung**
- **Datenmodelle und Bildkompression**

Histogramm-Ausgleich

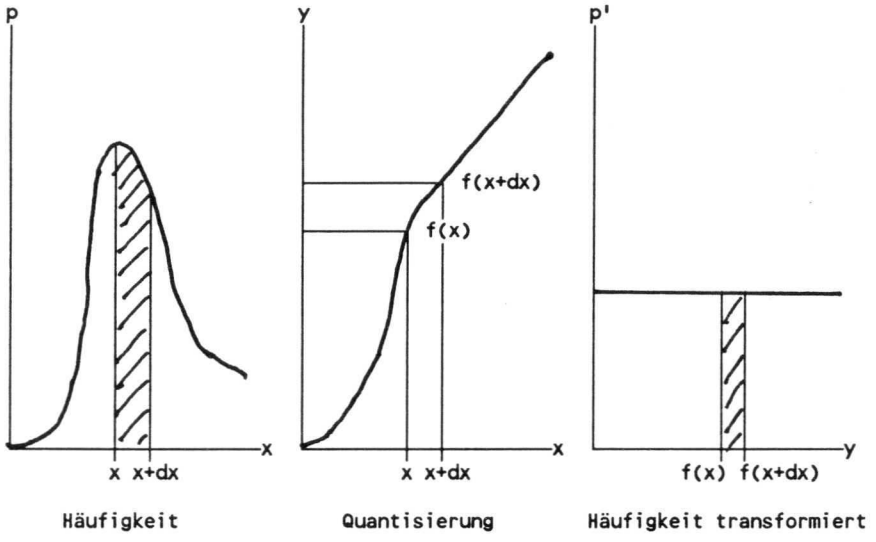


Maximale Ausnützung des Bereichs



konstante mittlere Dichte

Uniforme Quantisierung (stetig)



Die Häufigkeitsdichte zwischen x und $x+dx$ ist nach der Transformation gleich derjenigen zwischen $f(x)$ und $f(x+dx)$:

$$dx \cdot p(x) = (f(x+dx) - f(x)) \cdot p'(x) \quad .$$

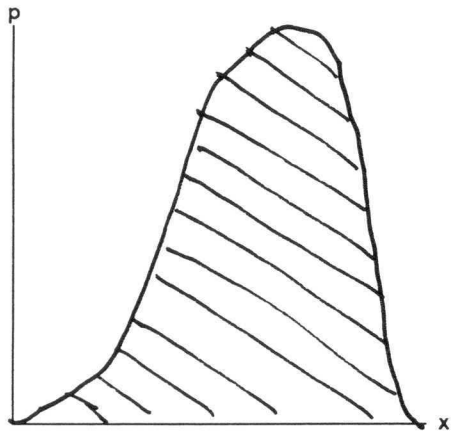
Daraus erhält man mit $p'(x) \equiv 1$ und $df = f(x+dx) - f(x)$:

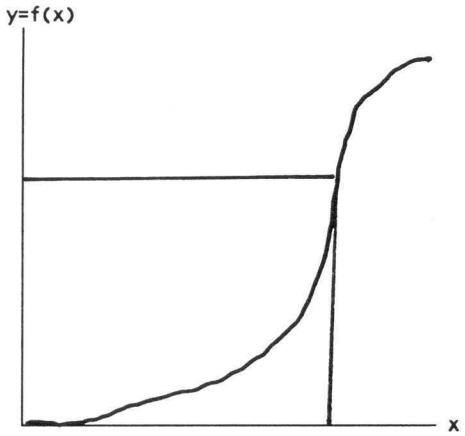
$$\frac{df}{dx} = p(x) \quad ,$$

woraus sich durch Integration ergibt, dass die Transformationsfunktion f gerade gleich der **Summenhäufigkeit** ist:

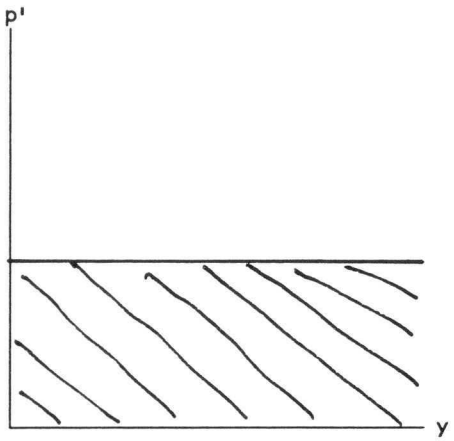
$$f(x) = \int_0^x p(t) \cdot dt \quad .$$

Summenhäufigkeit





x = = > y: Histogramm-Ausgleich
y = = > x: Zufallszahlen-Generator für beliebige Verteilungen



Palettenoptimierung

Problem: CLUT-Reduktion (CLUT = Color LookUp Table)

Reduktion von 24-bit-Farbbildern (Rot: 8 bit, Grün: 8 bit, Blau: 8 bit) auf 8-bit Paletten-Darstellung (für PC, Mac, . . .)

Idee: Dichtemodulierte Palette

Kein Bild nützt die 16 Millionen Farbschattierungen aus.
Wir wollen keine Paletteneinträge auf nicht verwendete Farbtöne verschwenden.
Die Paletten-Farbdichte im Farbwürfel soll proportional zur Häufigkeit der Farben im Bild gewählt werden.

Lösung: 2 x Summenhäufigkeit

Palettenerzeugung:

gleichverteilte Palette = = = > dichtemodulierte Palette

y - - - > x

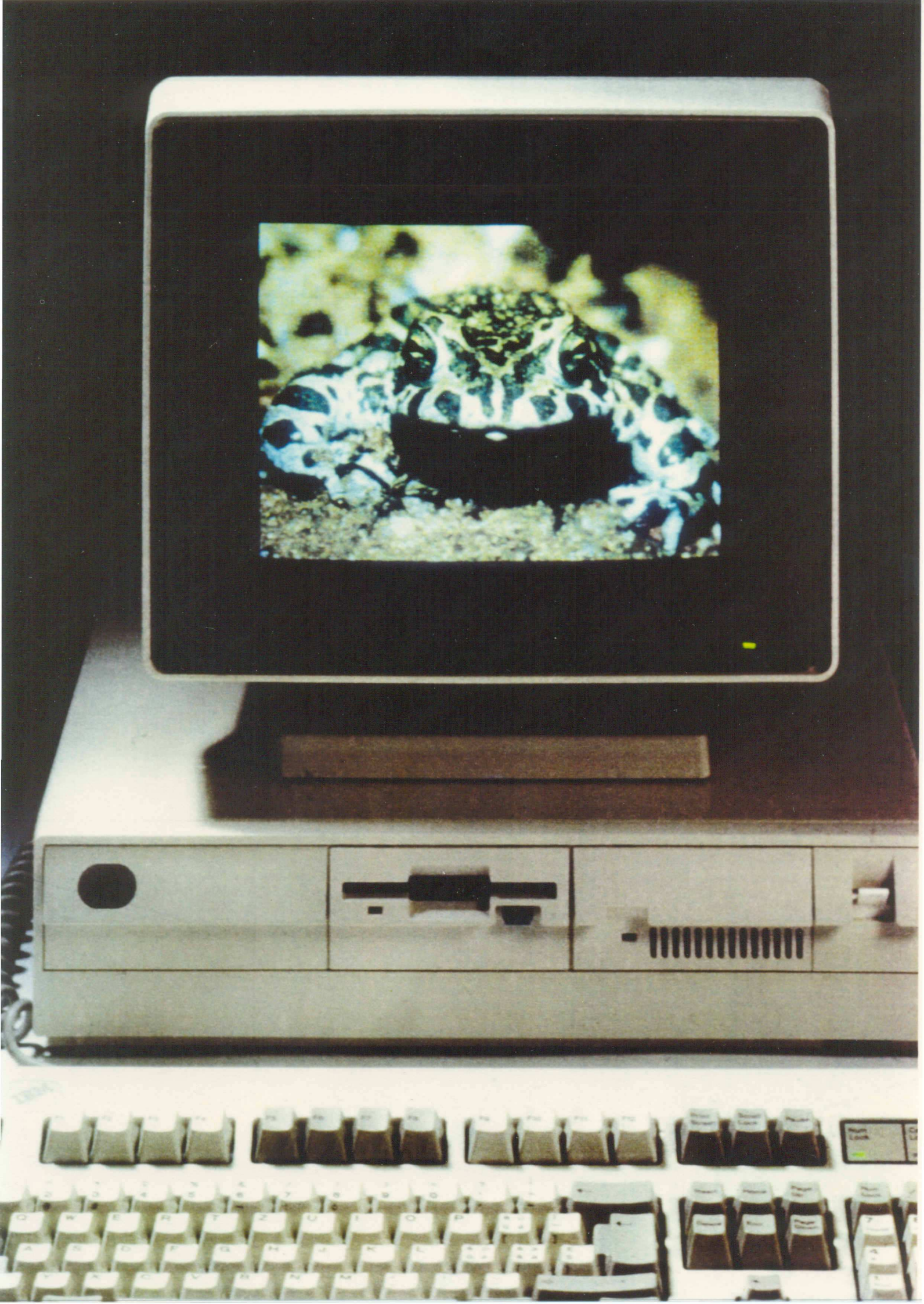
Color Lookup:

dichtemodulierter Farbton = = = > gleichverteilte Palette

x - - - > y

Dieser Algorithmus zur optimalen Farbreduktion von 24-bit Bildern auf 8-bit Bilder wurde gemeinsam entwickelt von

Furrer+Partner AG, Zürich,
Inventec Informatik AG, Zürich (Christian d'Heureuse),
Enter AG, Zürich (Hartwig Thomas)



Arithmetic Coding

Universelles

Verfahren

zur

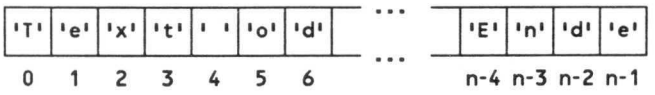
verlustlosen

Datenkompression

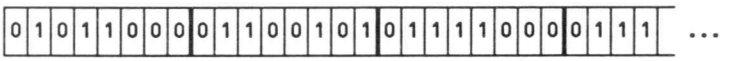
- C. E. Shannon, *A mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, 1948.
- J. Rissanen, *Arithmetic coding of strings*, IBM Research Report RJ 1591, June 3, 1975.
- R. Pasco, *Source Coding Algorithm for Fast Data Compression*, Ph.D. Thesis, Dept of E.E., Stanford University, Cal., 1976
- J. Rissanen, *Generalized Kraft inequality and arithmetic coding*, IBM J. Res. Dev., vol. 20, pp. 197-300, May 1976.
- J. Rissanen and G. G. Langdon, Jr., *Arithmetic Coding*, IBM J. Res. Dev., vol. 23, no. 2, pp. 149-162, Mar. 1979.
- F. Rubin, *Arithmetic Stream Coding Using Fixed Precision Registers*, IEEE Transactions on Information Theory, vol. 25, no. 6, pp. 672-675, June 1979.
- J. Rissanen, *Arithmetic coding as number representations*, Acta Polytech. Scandinavica, pp. 44-51, May 1979.
- M. Guazzo, *A General Minimum-Redundancy Source-Coding Algorithm*, IEEE Transactions on Information Theory, vol. 26, no. 1, pp. 15-25, Jan. 1980.
- G. G. Langdon Jr. and J. Rissanen, *Compression of Black-White Images with Arithmetic Coding*, IEEE Transactions on Communications, vol. 29, no. 1, pp. 12-23, 1981.
- J. Rissanen and G. G. Langdon Jr., *Universal Modeling and Coding*, IEEE Transactions on Information Theory, vol. IT-27, no. 1, Jan. 1981.
- C. B. Jones, *An efficient Coding System for Long Source Sequences*, IEEE Transactions on Information Theory, vol. 27, no. 3, pp. 280-291, Mar. 1981.
- I. H. Witten, R. M. Neal, J. G. Cleary, *Arithmetic Coding for Data Compression*, Communications of the ACM, vol. 30, no. 6, pp. 520-540, June 1987.
- R. N. Williams, *Adaptive Data Compression*, Kluver Academic Publishers, Boston, 1991.

Strings und dyadische Zahlen

- **Jeder zu komprimierende Datenstrom (Meldung, Text, Datei) ist auf dem Computer dargestellt als Folge von Bytes.**



- **Jede Folge von Bytes ist als eine Sequenz von Bits kodiert.**



- **Binäre (dyadische) Entwicklung einer reellen Zahl im Intervall [0,1):**

```
0.0101100001100101011110000 0111 ...
```

> Jedem Datenstrom entspricht
> genau eine Zahl im Intervall [0,1)

Das Kodierverfahren heisst *arithmetisch*, weil es mit den Daten rechnet wie mit Zahlen.

Erzielung der Gleichverteilung

- **Das Intervall [0,1) soll monoton auf sich selbst abgebildet werden.**

- **Schrittweises Vorgehen:**
  Von der zu komprimierenden Zahl ist erst ein Anfangsstück bekannt, ein Teilintervall von [0,1), in welchem sie liegt.
  Von der komprimierten Zahl ist erst ein Teilintervall I_n von [0,1) bekannt, in dem sie liegen wird.

- **0. Schritt:**
  Initialisierung:
  $I_0 = [0,1)$
  Modellwahrscheinlichkeit für das erste Bit:
  $p_0 = Pr(b_0 = 0)$.
  Kodierschritt:
  $b_0 = 0$: $I_1 = [0, p_0)$; $b_0 = 1$: $I_1 = [p_0, 1)$.

- **k. Schritt:**
  Schon berechnet:
  $I_k = [l_k, h_k)$.
  Modellwahrscheinlichkeit für das nächste Bit:
  $p_k = Pr(b_k = 0)$.
  Unterteilungspunkt:
  $c = p \bullet l_k + (1-p) \bullet h_k$
  Kodierschritt:
  $b_k = 0$: $I_{k+1} = [l_0, c)$; $b_k = 1$: $I_{k+1} = [c, h_k)$.

- **letzter Schritt:**
  Schon berechnet:
  $I_n = [l_n, h_n)$.
  Output:
  Irgendeine Zahl in diesem Intervall, welche mit einer möglichst kurzen dyadischen Entwicklung dargestellt werden kann.

Implementation der Arithmetik

- Die dyadischen Ziffern, welche der oberen unteren Grenze des Intervalls gemeinsam sind, werden nicht mehr benötigt und können schon als Output abgespeichert werden.

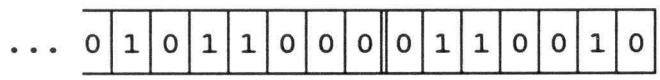
- Die Ziffern, wo sich obere und untere Grenze unterscheiden, können als vorzeichenlose Integers in 16- oder 32-bit Wörtern gespeichert werden. Die Unterteilung des Intervalls läuft auf Integer-Arithmetik hinaus.

- Wenn eine dyadische Ziffer der oberen Grenze des Intervalls trotz immer kleiner werdendem Intervall lange verschieden bleibt von derjenigen des unteren, so handelt es sich um eine 1, die von vielen 0 gefolgt ist, während die untere Grenze dort eine Null hat, die von vielen 1 gefolgt ist.

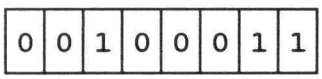
Darstellung des Intervalls

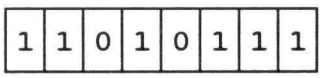
Das Intervall von
0. ... 010110000110010 01110100011 0000000000000... (Minimum)
bis
0. ... 010110000110010 10001010111 1111111111111... (Supremum)
wird dargestellt durch

1) gemeinsame dyadische Ziffern auf Ausgabedatei geshiftet:

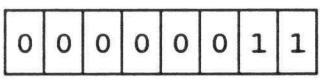


2) obere und untere Grenze (Computerwörter):





3) Anzahl „geschuldete“ 0 in der oberen Grenze, bzw. 1 in der unteren Grenze (Computerwort):



Optimalität von AC

- **Intervallänge nach dem letzten Schritt ≈ Wahrscheinlichkeit gemäss Modell *P***
- **Anzahl dyadische Stellen für Zahl in diesem Intervall: -log(*P*) = idealer Informationsgehalt gemäss Modell**

Implementationsverluste:

- **Speicherung der Länge (4 Byte)**
- **Runden bei der Intervallunterteilung**

Erfahrungstatsache:

Die erzielte Länge des Komprimats weicht bei der Arithmetischen Kodierung weniger als 2% ab vom idealen Informationsgehalt gemäss Wahrscheinlichkeitsmodell.

Vergleich mit anderen Verfahren

- **Mit arithmetischer Kodierung kann man jedes andere Kompressionsverfahren durch geeignete Wahl des Wahrscheinlichkeitsmodells emulieren.**

- **Die Arithmetische Kodierung kodiert Zeichen in Bruchteilen von Bits.**

- **Die Arithmetische Kodierung trennt Kompressionsalgorithmus und datentyp-spezifisches Modell explizit.**

- ***Die Arithmetische Kodierung erlaubt freie Wahl des Wahrscheinlichkeitsmodells.***

Eigenschaften der Arithmetischen Kodierung

- universell
- reversibel
- relativ langsam

Modelle und Reihenfolge

Beschränktheit der implementierbaren Modelle:

- **Markov-Modelle tiefer Ordnung**

Wichtigkeit der Durchmusterungsreihenfolge:

- **Man rearrangiere eine Textdatei mit den geraden Bytes von hinten nach vorn.**

Abhängigkeit von Datentyp:

- **Man komprimiere Texte und 8-bit-Bilder mit Huffmann und LZW. Huffmann ist besser bei Bildern, LZW bei Texten.**

```
Welche Reihenfolgen kommen beim Datentyp
"Mehrwertiges Realbild" in Betracht?
```


Pixelpyramiden