ETH Zürich
Dept. Informatik

Prof. Markus Gross
WS 05/06

Informationstheorie

Lösung 9

9.1 Intervalllängen-Codierung

a) Die Sequenz $s = \langle a, c, c, b, b \rangle$ wird in die Sequenz $z = \langle 3, 2, 1, 5, 1 \rangle$ codiert.

b) Wäre a das nächste Zeichen, würde es zur 5, entsprechend b zur 1 und c zur 3 codiert werden.

c) Eine bessere Codierung bekommt man, wenn jedes Zeichen in die Anzahl *verschiedener* Zeichen bis hin zum letzten Auftreten codiert wird. Somit wird b zur 1, c zur 2 und a zur 3 codiert. Dies ist die sogenannte *Move-To-Front-Codierung*.

d) In der Intervalllängen-Codierung merkt sich der Sender die letzte Position jedes Symbols in einem Array. Um das n-te Zeichen zu codieren, sucht man die letzte Position des Zeichens, aktualisiert diesen Wert und gibt dann die Differenz zu n aus. Dies geht in $O(1)$. Der Empfänger arbeitet analog.

In der Move-To-Front-Codierung kann der Sender zum Beispiel eine lineare Liste der Länge $|\mathcal{X}|$ verwalten, in der die Symbole gemäss ihrem letzten Auftreten geordnet sind. Um ein Zeichen zu codieren, gibt man die Position des Zeichens in der Liste aus und verschiebt das Zeichen dann an den Anfang der Liste. Das Verschieben geht schnell, das Suchen das Zeichens dauert aber $O(|\mathcal{X}|)$. Der Empfänger arbeitet analog.

9.2 Lempel-Ziv

a) Der Input wird wie folgt in Teilstücke zerlegt:

`0|01|00|1|11|010|011|0101`

Der entstehende Code ist dann:

`000,0|001,1|001,0|000,1|100,1|010,0|010,1|110,1`

b) Wenn das Präfix mit 3 bit codiert worden ist, wird der Code wie folgt unterteilt:

`000,1|001,1|001,0|000,0|011,0|101,1|100,0|100,1`

Daraus ergibt sich der Klartext:

`1|11|10|0|100|1001|00|01`

c) Ein modifiziertes Verfahren benutzt immer k = ceil(log(x)) Bits zum Codieren des Präfix, wobei x die aktuelle Anzahl von Einträgen im Wörterbuch ist. Der Decoder kennt an jeder Position im String die aktuelle Länge des Wörterbuchs, kann also auch k berechnen. Somit ist kein Präfixfreier Code zur Codierung der Präfixindexlänge notwendig, die Indizes können einfach in binärer Form gespeichert werden.

9.3 Vergleich von Codierungsverfahren

- Huffmann: Codiert Symbole aus einer Quelle über einem beliebigen Alphabet. Die Menge der Codewörter ist präfixfrei, und damit eindeutig decodierbar. Huffmann-Codes sind optimal. Um einen Huffmann-Code zu erzeugen, muss die Quellenstatistik, d.h. die Wahrscheinlichkeiten der einzelnen Elemente des Alphabets bekannt sein. Huffmann-Codes werden in der Praxis haeufig zur Kompression eingesetzt, wenn die Quellenstatistik bekannt ist oder geschätzt werden kann.

- Shannon-Fano: Codiert Symbole aus einer Quelle über einem beliebigen Alphabet. Die Menge der Codewörter ist präfixfrei, und damit eindeutig decodierbar. Shannon-Fano-Codierung ist nicht notwendigerweise optimal. Um einen Shannon-Fano-Code zu erzeugen, muss die Quellenstatistik bekannt sein. Shannon-Fano-Codes könnten ähnlich wie Huffmann Codes zur Kompression eingesetzt werden, in der Praxis wird Shannon-Fano allerdings praktisch nie benutzt, da Huffmann dieselben Voraussetzungen hat und bessere Codes erzeugt.

- Codierung der ganzen Zahlen: Codiert ganze Zahlen. Die Menge der Codewoerter ist präfixfrei, und damit eindeutig decodierbar. Man kann jeden endlichen String als ganze Zahl auffassen und so codieren, ohne weitere Annahmen zu machen. Die Codierung der ganzen Zahlen wird benötigt, um Zahlen präfixfrei abzuspeichern.

- Arithmetische Codierung: Codiert einen String über einem beliebigen Alphabet. Der sich ergebende Code repräsentiert den String, aber den Elemente des Alphabets werden keine Codewörter zugewiesen. Die Frage nach eindeutiger Decodierbarkeit der Codewortmenge stellt sich daher nicht. Ein mittels Arithmetischer Codierung erzeugter Codestring ist allerdings nur dann decodierbar, wenn die Länge des Eingabestrings bekannt ist (oder durch spezielle konventionen gekennzeichnet). Arithmetische Codierung ist in dem Sinne optimal, dass sie asymptotisch auf die Entropie codiert. Da es keine Codewörter gibt, kann das Kriterium der mittleren Codewortlänge nicht angewandt werden. Die Quellenstatistik des zu codierenden Strings muss bekannt sein. Arithmetische Codierung wird zur Kompression von Strings mit bekannter Länge bei bekannter Quellenstatistik eingesetzt.

- Intervalllängencodierung: Codiert Symbole aus einer Quelle mit beliebigem Alphabet. Auch bei der Intervalllängencodierung wird den Elementen des Quellalphabets keine Codewörter zugeordnet. Die erzeugten Codestrings sind allerdings ohne weitere Information decodierbar. Die Intervallängencodierung komprimiert Quellen mit endlichem Alphabet nicht auf die Entropie, ist also in diesem Sinne nicht optimal. Durch Einführen von Blockcodes kann dieses Problem behoben werden. Die Quellenstatistik muss weder zur Codierung noch zur decodierung bekannt sein. Die Intervalllängencodierung wurde und wird zur Codierung von Information auf physikalischen Datenträgern verwendet, z.B. zur Darstellung von “sparse files” im ext2/3-Dateisystem.

- Lempel-Ziv: Codiert Strings über einem beliebigem Alphabet. Den einzelnen Elementen des Alphabets werden keine Codewörter zugeordnet. Der Code ist jedoch ohne weiteres Wissen eindeutig decodierbar. Lempel-Ziv codiert asymptotisch auf die Entropie, ist also in diesem Sinne optimal. Die Quellenstatistik wird nicht benötigt. Das Lempel-Ziv Verfahren ist ein klassisches Kompressionsverfahren, das ohne Vorwissen eingesetzt

werden kann. Das Programm “compress” benutzt dieses Verfahren, Varianten davon sind in allen gängigen Kompressionsprogrammen zu finden.