1.3 Kompression unter der Lupe

Wenn Sie das Vorkapitel durchgelesen haben, dann wissen Sie jetzt, daß man Dateien verkleinern kann. Sie kennen nun auch einige simple Methoden, um die Datenvielfalt zu verkleinern. Allerdings werden Sie sicherlich zugeben, daß die Alltagspraxis einerseits nicht nur Packraten jenseits von 12,5 % verlangt, sondern andererseits auch Programme zu komprimieren sucht.
Der Grund, warum ich im Vorkapitel so sehr auf den Text- oder ASCII-Dateien herumgeritten bin, liegt in der Tatsache, daß die vorgestellte Methode nicht auf Programmdateien paßt. Charakteristisch für Programmdateien ist nämlich, daß man nicht mehr voraussetzen kann, daß bei den Bytes der Daten das achte Bit immer 0 ist, wie bei den Textdateien. Also muß man hier komplett anders vorgehen.
Wie könnte das funktionieren?
Also nochmal: Der PC hat einen Zeichenvorrat von 256 Zeichen. Diese 256 Zeichen werden durch einen Code von einem Byte Länge repräsentiert. Ein Byte ist dabei eine Kombination aus acht Bits, die jeweils den Zustand Eins oder Null haben können.
Das ist bereits ein alter Hut. Jetzt kommt aber die Krux: Wenn eine Datei alle 256 Zeichen enthält, wie kann man dann diese Informationsflut verkleinern? Schließlich braucht man alle acht Bits, um das Zeichen gemäß seinem ASCII-Code zu speichern. Würde irgendwo ein Bit fehlen, dann ist das Zeichen nicht mehr zu rekonstruieren.
Wenn Sie einmal gutes Geld in Aktien gesteckt haben, kommen Sie vielleicht auf des Rätsels Lösung. Nehmen wir einmal an, Sie haben von dem Papier der PC-AG 100 Stück zu einem Preis von 200 DM gekauft. Leider berichtet die einschlägige Wirtschaftspresse am Tag nach dem Kauf, daß die PC-AG mit ihrem veralteten Modell BS/2 keine Kunden mehr findet und riesige Verluste zu erwarten hat. Der Kurs fällt von einem Tag zum anderen auf bodenlose 21 DM
Wenn Sie noch einige Mark auf dem Konto haben, sollten Sie jetzt nicht auf der allgemeinen Welle schwimmen und Ihre Stücke ebenfalls verkaufen. Statt dessen spielen Sie "Trendbrecher" und kaufen sich noch mal 100 Stück für 21 DM und haben jetzt 200 Aktien zu einem Durchschnittskurs von 110,50 DM.
"Wir wissen nicht, ob Sie in diesem Jahr nochmal mit Ihren Aktien in die Gewinnzone rücken, aber eine Lehre können wir für die Datenkompression aus der finanziellen Tragödie ziehen: Von den billigen sollte man viel kaufen und von den teuren möglichst wenig. Übertragen auf die Packer-Welt heißt das: Man versehe häufige Zeichen mit einem kurzen Code und seltene Zeichen mit einem längeren.
Jetzt steht es PC-Anwendern aber nicht zu, den Duden durchzuarbeiten, Buchstaben zu zählen und nach den statistischen Daten die ASCII-Codes nach eigenem Gutdünken zu verändern. Oder etwa doch???
Die Antwort ist ein klares JEIN! Natürlich wird der ASCII-Code nicht verändert, aber für die Komprimierung kann man sich durchaus andere Bitkombinationen ausdenken, um verschiedene Zeichen neu zu kodieren.

Die Huffman-Methode

Diese Methode der Datenkompression geht genauso vor. Durch die Zuordnung kurzer Bitmuster an häufige Zeichen und langer Bitmuster an seltene Zeichen wird im Durchschnitt die Länge der Zeichen unter die Vorgabe von 8 Bits gedrückt. Das Erstaunliche an der Sache ist, daß es keine moderne Erfindung eines zwölfjährigen Computerfreaks ist, sondern eine mathematisch fundierte Analyse aus den fünfziger Jahren! Die Huffman-Methode ist so sicher, daß auch alle Telefaxgeräte ihre Informationen mit einer auf dem Huffman-Algorithmus basierenden Methode komprimieren und durch den Draht schicken.
Auch die meisten Packer arbeiten mit einem (meist etwas modifizierten) Huffman-Algorithmus. Diese Methoden werden dann in der Regel als Squeezing bezeichnet. Sie kümmert sich nicht um die Art der Codes, ob sie aus dem gesamten ASCII-Spektrum kommen, und ob die Zeichen überhaupt ASCII-Zeichen sind. Das A und O ist die Ermittlung der Häufigkeit der zu kodierenden Zeichen. Aus dem Ergebnis dieses Tests bauen sich im Folgeschritt die neuen Bitmuster auf. Der Huffman-Algorithmus arbeitet also in zwei Schritten:

1. Ermittlung der Anzahl eines jeden Zeichens einer Datei.
2. Erstellung der neuen Codes auf der Grundlage deren Häufigkeit.

Natürlich möchte ich Ihnen den Huffman-Gedanken etwas näherbringen. Wenn Sie ein wenig neugierig sind, dann sind die weiteren Seiten sicherlich interessant für Sie, weil auch einige tiefere theoretische Aspekte unkompliziert (und beinahe unbemerkt) mit einfließen.
Für das Beipiel nehmen wir eine beliebige Datei beliebiger Größe mit beliebigem Inhalt. Es könnte damit das Programm WORD.EXE mit seinen mehr als 600000 Bytes Länge genauso herhalten wie auch COMMAND.COM oder die Datei FRITZ.DAT, die Sie sich nachfolgend näher anschauen können. Mit DIR FRITZ.DAT untersucht, gibt die Datei folgendes her:

```
FRITZ    DAT       160   04.03.92  10:12
```

FRITZ.DAT ist also 160 Bytes lang. Für die Demonstration ist dies lang genug, es könnte aber auch jede andere Größe sein.
Interessanter als die Größe ist für den Huffman-Algorithmus der Inhalt der Datei. So besteht als erstes die Aufgabe, alle vorkommenden Zeichen in der Datei festzustellen und zu zählen. Nach dem ersten Prüflauf durch die Datei steht also fest, daß nur die Buchstaben T, W, X, C, B, O und A mit der angegebenen Häufigkeit enthalten sind. Wer immer diese Datei auch erzeugt hat, der Sinn und Zweck wird für immer sein Geheimnis bleiben (oder können Sie aus den angegebenen Buchstaben einen tieferen Sinn zusammenbasteln?).
Die Reihenfolge dieser Zeichen in der Datei ist aus dieser Häufigkeitstabelle nicht zu ersehen - sie spielt an dieser Stelle auch gar keine Rolle. Auch die Anordnung der Buchstaben in der Abbildung ist völlig unerheblich. Ich habe sie nur deshalb so plaziert, damit sich die Ermittlung der Codes in den Abbildungen etwas übersichtlicher aufbaut.