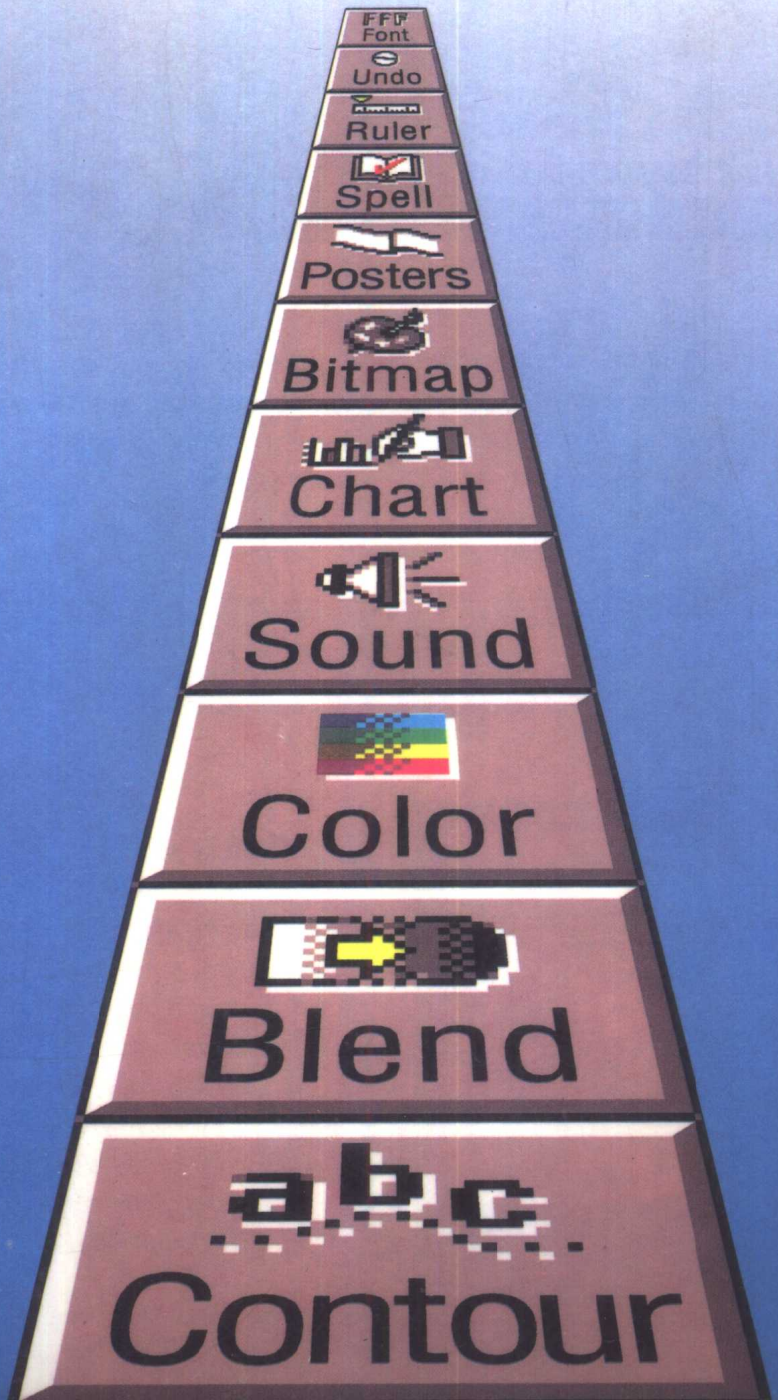
计算机应用软件系列丛书

数据压缩技术经典

钱国祥
孙　宏
彭振云　　编著
詹榜华
徐广方



希望

学苑出版社

计算机应用软件系列丛书

数据压缩技术经典

钱国祥　　孙　宏
彭振云　詹榜华　徐广方　　编著
侯业勤　　审校

学苑出版社

(京)新登字 151 号

内 容 提 要

本书详细介绍了信息论中的数据压缩归档技术及通用的压缩软件。包括统计模型、字典压缩、有损压缩以及通用压缩软件的用法等等。并提供了包括著名的 LZW 算法在内的各种压缩算法的 C 语言源程序。本书对从事数据压缩工作或程序员有很大的参考价值。是一本不可多得的好书。

版 权 声 明

计算机应用软件系列丛书

数据压缩技术经典

编　　著：钱国祥　孙　宏　彭振云　詹榜华　徐广方
审　　校：侯业勤
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：27.875　　**字　数**：653 千字
印　　数：1～5000 册
版　　次：1994 年 8 月北京第 1 版第 1 次
ISBN7－5077－0974－4/TP・33
本册定价：46.00 元

学苑版图书印、装错误可随时退换

前　言

数据压缩的目的是，通过剔除冗余信息以减少存储和传递数据所需的比特数。80 年代中期以来，由于 VLSI、通信、计算机等技术的高速发展，数据压缩的研究和应用进入了一个空前繁荣的时期。CCITT、ISO 和 CCIR 等标准算法大都需用专门的硬件实现，技术难度很大，而且价格昂贵，一般用户是不敢问津的。

桌面计算机领域对数据压缩的要求是：算法简单，易用软件实现。由于 CPU 和存储器的价格不断降低，性能越来越卓越，而且一些廉价的信号处理芯片已开始成为微机的标准组成部件。这使得越来越多的高级压缩算法能在桌面计算机上实现。于是就涌现出了 PKZIP、ARJ、LHARC 等大量以公用软件方式发行的压缩/归档软件。今天的程序员恐怕很少有人会说从未用过这些程序。但是，同样也很少有人会说对这些软件的工作原理有全面、深入的了解，当然编写这方面程序的人更少。

本书的目的就是要为那些感兴趣的读者消除这种遗憾。

这本书将系统地介绍桌面计算机上采用的各种压缩技术，描述典型压缩算法的原理及算法，并给出所有算法的 C 语言实现细节！我们不仅希望这本书是一本经典性的数据压缩手册，更希望它是一本压缩/归档软件的编程指南。

我们渴望为读者提供这样的境界：让略知 C 语言编程的人读完这本书后也敢尝试编写像 PKZIP 一样的压缩/归档软件！

全书共分五个部分十二章。第一部分引论介绍数据压缩的基本概念和发展简史，接下来三部分讲解三种类型的压缩技术：基于统计模型的数据压缩，基于字典的数据压缩和有损压缩。最后一部分讲解数据压缩/还原的孪生物－文件归档的所有细节。此外，为方便读者，本书每章的最后附上了书中讲到的完整的源程序。这些程序都经过严格调试，供读者参考或直接引用。

为考虑各个层次的读者的不同需要，我们在本书的附录中还提供了以下一些有用的资料：附录 A 主要是本书第二、三部分介绍的各种压缩算法的简要概括；附录 B 详细介绍了 PKZIP、LHA、ARJ、PKLITE、LZEXE 等压缩软件的用法；附录 C 着重讲述 ZIP 文件格式；附录 D 是两个压缩/归档软件的 C 语言源程序，供读者参考。

由于我们急于向读者展示这个迷人领域的一切，编写时间仓促，缺点错误在所难免，恳请读者不吝赐教。

编　者

1994 年 4 月于北京

目　录

第一部分　引　论

第二部分　基于统计模型的数据压缩

第三部分 基于字典的数据压缩

第四部分　有损数据压缩

第五部分　文件归档原理

第一部分　引　论

本部分由两章组成：

● 引言

介绍本书的特点和组成。特别阐明为什么数据压缩算法适合用C语言实现，并讨论了用C语言编写压缩程序时需注意的若干方面。

● 数据压缩词汇和历史

简要介绍数据压缩的基本概念和发展历史，综述本书涉及的压缩算法。数据压缩分成有损压缩和无损压缩两大类。前者用于要求压缩数据能精确恢复的场合，如程序和文本文件压缩。后者用于语音、图象等能够允许一定失真的数据。无损压缩又有统计编码压缩和字典压缩两种方法。后面的篇章按统计编码、字典压缩和有损压缩的顺序详细介绍各种算法。

• MEMO •

第一章 引 言

本书的主要目的是用C语言讲解各种数据压缩技术。数据压缩的目的是减少用于存储或传递信息的比特数。它包含非常多的软件和硬件压缩技术，这些技术各不相同，除了都是压缩数据这一点外它们极少有共性。例如：用于Compuserve GIF规范的LZW算法与用于压缩数字电话语音的CCITT G.721规范就没有任何共同之处。

在最近的二十五年间，数据压缩领域获得了高速的发展，这使我们不可能对每一种数据压缩技术都给予介绍。本书所覆盖的是通常应用于个人计算机和中型计算机上的各种类型的数据压缩。包括二进制程序、数据文件、声音和图像文件的压缩。

此外，对于那些在实际使用中依赖于硬件或要求硬件设备的数据压缩技术，本书忽略或只是少量提及。当前许多语音压缩方案是为世界性的固定带宽的数字化电信网络设计的。其实这些压缩方案是令人感兴趣的，但他们要求特殊的硬件设备以便调整到信道的固有波段上来。PC机上用另外一些没有这种要求的算法压缩数字化语音，而且一般来讲性能更好，所以我们主要针对后者。

然而当前数据压缩中某些最令人感兴趣的领域是用新的、更加强有力的硬件使压缩技术得以实现。例如：有损图像压缩(如：多媒体系统中所用的)，现在已能够在标准的台式平台上实现。本书将涉及应用这些技术的某些算法及其实现方法。

1.1 读者

本书的读者需要基本的编程技巧以便正确理解数据压缩程序，要求具有阅读诸如C语言或Pascal语言程序的能力。另外，还要求了解计算机结构，足以理解诸如移位、逻辑或、逻辑与等位运算。

这并不是说你要为阅读本书而花费时间去精通C语言，你甚至不必是一个程序员，但是由于书中描述的概念都是以简易的C语言程序来说明的，因此有阅读程序的能力是基本的要求。

我们尽可能使书中的C语言程序简单明了，以使得C语言的初学者也能够读懂。我们还将避免使用那些较为深奥的C语言结构，但程序是用C语言而不是伪代码或自然语言(如：英语)来编写的。

1.2 为什么要用C语言

使用C语言说明数据压缩算法可能会引起某些人的不满，写作本书的更为传统的方法是使用伪代码概略地叙述算法。但是由于伪代码没有严格定义，因此常常满篇都是" PROCESS FILE UNTIL OUT OF DATA "这样一些不完整的程序行。这样做的缺点是代码容易阅读，但不那么容易改写成实用程序。

如果说伪代码是不适宜的，那最好的选择就是使用一种常规的编程语言。对中国的大多数程序员来说，C 语言无疑是一种较好的选择。

首先，在许多方面 C 语言已经成为程序设计者的公用语言。C 语言编译器支持的计算机范围很广：从低级的 8051 微控制器到具有 100MIPS 的高级计算机。这并不意味着所有的程序都选择 C 语言来编写，它只意味着大多数程序员会有适用于他们机器的 C 语言编译器。而大多数程序员可能经常接触 C 语言程序，因此，许多使用其它语言的程序员也能够处理用 C 语言书写的程序，至少是可以读懂 C 语言程序。

其次，C 语言是一种没有太多生僻的语言。作为基本语言元素的几个结构能够容易地移植到其它语言。因此，一个用 C 语言构造的数据压缩程序可以通过一个相对简单的移植过程转变为 Pascal 程序，就连汇编语言程序员也会觉得这一过程很容易接受。

也许使用 C 语言最重要的理由是效率。C 语言通常被看作是高级汇编语言，因此，它允许程序员与硬件有密切的联系，尽管新的 C 语言编译器增强了优化处理，但仍不可能达到手工编制的汇编语言程序所具有的速度和大小。不过 C 语言程序能够很容易地移植到另一种机器上去，这种优点可以弥补这种不足。因此，在本书这类文献中使用 C 语言或许是最佳选择。

1.3 选用哪种 C 语言：移植性问题

尽管 C 语言是一种“可移植”的语言，但在一台机器上编译和执行的 C 语言程序不能保证在其它任何一台机器上都能运行，甚至在同一台机器上使用不同的编译器都不能编译。要记住，重要的不是 C 是可移植的，而是它是能够移植的。本书的程序已写成了可以移植的，而且它的编译和运行是在明确规定的几个编译器和环境条件下进行。这些编译器/环境包含如下：

1. Microsoft C/C++
2. Borland C++
3. Zortech C++
4. 具有可移植 C 编译器的 Interactive Unix System 3.2

一个重要的可移植性问题是库函数调用。虽然 C 语言在 K&R 的原始教程中（Brian W. Kernighan and Dennis M. Ritchie. The C Programming Language[Englewood Clihs, N. J.: Prentice-Hall, 1978]）作了全面的定义，但是运行库的实现已经变成了实现者的任意行为。

美国国家标准协会在 1990 年制定了 C 语言标准，并将其结果作为 ANSI 标准 X3J11.34 予以发表。这一标准不仅对最初的 K&R 语言规范进行了扩展和定型，而且它也承担了对标准 C 语言运行库的定义。这使得编写在不同机器上工作的程序代码更加简易。本书的程序力图只使用 ANSI-C 库调用，尽量避免使用依赖于编译器的语言和库扩展。

给定了标准化的库，剩下的可移植性问题围绕两件事：基本数据类型的大小和不相容编译器的处理。当 16 位机和 32 位机切换时，大多数数据类型冲突将会出现。

16 位和 32 位机的切换处理是很容易的，虽然基本整数类型有 16 位和 32 位，但两种机器都有一个 16 位的短整数类型，而且又都有一个一般为 32 位的长整数类型。因此，在那些整数大小清楚的情况下使用适当的说明能够使它固定为 16 位或 32 位。

在现今世界的绝大多数机器上，C 语言编译器处理的 char 型数据是 8 位的。本书中，我们将不考虑其它长度存在的可能性，而只处理 8 位的 char。一般来说，把书中出现的程序转到一

台具有非常规字符长度的机器上去是不太困难的，但花费太多的时间在这上面将会使程序的重要之处－数据压缩不能得以突出。

编写可移植程序时要处理的最后一个问题是编译器不一致的问题。在MS－DOS领域中，大多数C语言编译器每两年左右都要发行和升级一次。这意味着大多数编译器的卖主能够发行与ANSI－C标准紧密一致的新的编译器版本。但对于许多其它操作系统的用户来说情况就不是这样了，尤其是UNIX用户，他们经常使用与系统同时获得的符合K&R语言定义的C语言编译器。尽管ANSI－C委员会做了很大的努力使ANSI－C与K&R－C向上兼容，我们还要注意几个问题：

第一个问题出现在函数原型的使用上。在K&R－C中通常只有在必须的情况下才使用函数原形(function propotype)。编译器假定任何隐含的函数返回一个整数并且无条件地接受它。如果一个函数返回某一非常规的东西，如一个指针或一个长函数，程序员要写一个函数原型通知编译器：

```
long locate_string();
```

这里，原型通知编译器生成返回长整数而不是整型数的函数代码。函数原型的使用仅此而已。因此工作在K&R体制下的程序员很少或没有使用过函数原型，而且它们在程序中显得不好理解。

虽然ANSI－C委员会并没有企图改变C语言的基本特性，但是，他们也没有放过对语言可能的改进，通过扩展原型的功能可以实现这种改进。在ANSI－C中函数原型不仅定义了函数的返回类型，而且也定义了所有的参数类型。例如前面给出的函数在ANSI－C编译器中可能具有下面的原型：

```
long locate_string( FILE *input_file, char *string )
```

这使得编译器为返回类型产生正确的代码并同时检查参数类型和个数的正确性。由于允许函数的参数类型和个数有错是C语言程序员的一个主要错误源，因此要求全面地实现这些形式的类型检查，使C语言前进了一步。

在许多ANSI－C编译器中，非常支持使用完整的ANSI－C函数原型，事实上，当一个函数被使用而没有确切地遇到一个原型时，许多编译器产生警告信息。还好，在UNIX的一个可移植C语言编译器上，同样的函数原型是不起作用的。

这一解决问题的方法并不漂亮，但它有效。在ANSI－C中总是定义预定义宏__STDC__来表明程序是通过一个可能的ANSI编译器来编译的。根据所用编译器是否符合ANSI规范，我们可以让预处理器打开或关闭头文件的某些部分，对于一个面向位(bit－oriented)的软件包，其含有原型的头文件可能其形式如下例所示：

```
#ifdef __STDC__
    .
    .
    .
#endif
```

预处理命令不会对程序的形式产生太多的影响，但它们是编写可移植程序的必需部分。由于本书中的程序假设是由具有最大可能警告程度的编译器来编译的。因此包括一些#ifdef语句。

K&R 编译器家族的第二问题在于函数体，在 K&R－C 中一个特定的函数可能有如下的定义：

```
int foo( c )
char c;
{
/ * Function body * / }
```

同一个函数，如果用 ANSI－C 来编写，函数体如下：

```
int foo( char c )
{
/ * Function body * / }
```

这两个函数可能看起来相同，但 ANSI－C 的规则要求对它们进行不同的处理。K&R－C 要求字符参数在函数体中被使用前转化为整型值，而 ANSI－C 保留其为字符型。将一种整型量转化成另一种会在一个看起来很好的程序中潜伏一些问题。当严格的编译器发现这种性质的问题时它将发出警告。

由于 K&R 编译器不接受第二种形式的函数体，因此对函数定义字符变量时要小心。不幸的是问题的解决方法又是要么不用字符变量，要么借助于更多的令人讨厌的"＃ifdef"预处理器包。

本书所附的程序都在 Borland C/C＋＋ 3.1 上调试通过，读者如使用其他的 C 语言环境，请注意该语言环境的特殊性，必要时请查阅语言手册。

1.4 现有的记录

贯穿全书，都与压缩比、压缩统计有关。压缩统计总是与样本压缩文件相关连。为了使各种形式的压缩有一个相对合理的比较，本书使用的样本压缩文件分为三类：第一类是文本，由手稿、程序、便函和其它可读文件组成；第二类由二进制数据组成，包括数据库文件、可执行文件和待译数据；第三类由以原始的屏幕转储格式存储的图像文件及各种格式的声音文件组成。

对本书中建立和讨论的程序将以三种粗略的性能量度予以评价。第一是压缩过程中程序所需要的存储器数量，这个数量与它能够做到的相近似；第二是程序在压缩全部 Dr. Dobb's 数据组时所占用的时间量；第三是整个数据的压缩比。

不同的人使用不同的公式计算压缩比。某些人偏爱 bits/byte，而另一些人使用比率如 2：1 或 3:1(做广告的人似乎喜欢这种形式)。在本书中我们将使用一个简单的压缩百分率公式：

(1 － compressed_size / raw_size) * 100%

这意味着一个在压缩后完全没有改变的文件具有百分之零的压缩比；一个压缩到原长度的 1/3 的文件具有 60～70%的压缩比；一个收缩为 0 字节的文件据有 100%的压缩比。

这种测量压缩的方法可能不是完美的，但它表现了理想的百分之百和完全失败的百分之零。事实上，如果一个文件经过压缩后反而变大则将得出负的压缩比。

1.5 本书结构

本书由十二章和一张软盘组成，结构大致上与数据压缩的历史进展相平行，以大约 1950

年的“萌芽时期”为始，直到现在最新的数据压缩技术。

第二章是一个参考，试图建立基本的数据压缩词汇集。它描述了信息理论的诞生，介绍了本书其余部分反复使用的一系列概念、术语、流行语和理论。哪怕你是一个数据压缩方面的初学者，对于第二章的掌握也会把你送到很高水平的层次上去。这意味着即使你没有搞懂数据压缩的内涵，也能以一个行家的口吻谈论数据压缩。

第三章以可变长度比特编码开始，讨论数据压缩的诞生。Shannon－Fano 编码和 Huffman 编码的发展代表了数据压缩和信息论两者的诞生。今天这些编码方法仍然广泛地使用。另外，第三章评述了模型和编码之间的不同—它们是数据压缩的两个方面。

在用于高性能数据压缩时，标准的 Huffman 编码遇到了明显的问题。压缩程序必须传递一个完整的 Huffman 编码统计数字的拷贝给还原程序。当压缩程序收集更多的统计数字，以增加压缩比时，统计数字占去了更多的空间，抵消了所增加的压缩效率。

第四章讨论了解决这一困境的一种方法：自适应 Huffman 编码。这是一个较新的革新，它归功于 CPU 和存储设备。自适应编码极大提高了 Huffman 编码水平，使压缩比得以巨大地提高。

Huffman 编码对每一码字必须使用整数个比特，它通常略少于最理想的情况。更近期的新方法—算术编码—对每一码字使用分数个比特，允许它逐渐改善压缩性能。

第五章说明了这一新近的方法是如何工作的，并指出了一个算术编码和一个统计模型是如何结合在一起的。

第六章讨论了统计模型，无论使用 Huffman 编码、自适应 Huffman 编码或者算术编码都必须有一个统计模型来驱动编码器。本章介绍一些建立强有力模型的重要技术，它们只使用有限的存储资源。

第七章提供了字典压缩方法的概况，该压缩方法采取了与前四章所描述的完全不同的方法来进行压缩。它采用字符串压缩而不是单代码压缩。事实上字典方法已经成为小型计算机的通用压缩标准。因为该方法性能卓越而且对内存需求合理。字典压缩之父是 Ziv 和 Lempel，他们在 1977 年发表的一篇论文上提出了一种可调整的字典数据压缩方法，现在已是众所周知。

第八章着眼于通用的压缩/归档程序（如：PKZIP）当中的 LZ77 压缩方法的新改进。

第九章详细地考查了最为普及的基于字典的压缩方法之一：LZW 压缩。LZW 压缩算法用于 UNIX COMPRESS 程序和 MS－DOS ARC 程序的早期版本中。本章还考查了 LZW 压缩的基本原则，是由 Ziv 和 Lempel 于 1978 年发表的。

所有在前九章中讨论的压缩技术都是所谓的“无损压缩”，无损压缩适用于前面提到的样本压缩文件的前两类：文本和二进制文件。第三类的语音和图像文件压缩需要用到第十、十一章的“有损压缩”技术，他们具有显著提高压缩比的能力。

第十章告诉你为什么利用线性预编码和自适应 PCM 这类的技术的有损压缩可用于数字化的语音数据。

第十一章讨论了有损压缩技术对于计算机图像的应用。JPEG 标准的制定正快速地进行，在这一章中将反映用于 JPEG 标准中的技术。

第十二章评述了如何将这一切放入一个归档程序中，通用的归档程序应能够压缩和还原文件，同时记录文件的名称、日期、属性、压缩比和压缩方法。一个归档格式应该完全能够移植

到不同类型的计算机上去。开发一个归档程序的实例就是应用前述各章中的技术构成一个完整的程序。

本书的最后部分是附录。附录 A 把本书前面几章的数据压缩算法作了一个全面的总结；附录 B 详细介绍了目前市场上流行的通用压缩软件(如 PKZIP、LHA、ARJ、PKLITE 等的使用)；附录 C 向读者介绍了通用的. ZIP 文件格式；附录 D 是两个压缩归档软件的 C 语言源码作为我们的实例。

第二章　数据压缩词汇和历史

和其它科学或工程领域一样，数据压缩领域也有它的专用词汇。对于一个初学者来说，第一次接触这些词汇会感到非常陌生。类似 Lempel－Ziv 压缩、算术编码、统计模型这样的术语大量使用，随处可见。

尽管列出了很多的词汇，但掌握它们并不象初看时那么可怕，只要花费少许的学习和注意，任何一个程序员都能口若悬河地谈论数据压缩技术。

2.1　两个领域

数据压缩技术可分为两个主要的分支：有损的和无损的。有损压缩允许以一定的精度损失换取高的压缩效率。已经证明有损压缩用于图像和数字化语音方面是有效的。模拟量的数字化表达原本就不是精确的，正是由于这一特征，使得输出和输入的不完全对应可以接受。大多数有损压缩技术可以设计成不同的质量等级，获取高精度就要以损失压缩效率为代价。到目前为止仍然采用最初的以硬件为代价的方法实现有损压缩。近年来，很多有损压缩程序已经进入桌面 CPU 中，然而即使如此，这一领域仍然受着硬件设备的支配。

要求经过一个压缩/还原周期之后能够产生与输入完全一致的压缩技术称为无损压缩，这类压缩用于存储数据库记录或字处理文件。在这些应用中，即使一个比特的损失都可能是灾难性的，本书所讨论的大多数技术是无损的。

2.2　数据压缩＝建模＋编码（Modeling＋Coding）

一般来说，数据压缩由读取符号串和把它转换为码字这两者组成。若压缩有效率，那么结果码字串会小于原始符号串。对于一给定的符号或一组符号，所输出的结果取决于模型。简单地说，模型是一个数据和规则的集合。这些数据和规则用于处理输入符号和决定输出哪些码字。程序使用模型为每一符号精确地定义一个概率，而编码器基于这些概率产生一个适当的代码。

建模和编码是截然不同的两件事，人们频繁地使用术语 Coding 来表示整个数据压缩过程而不是指这个过程的一部分。你将看到 Huffman 编码、Run Length 编码这类描述数据压缩技术的词汇。事实上，它们所指的是与数据压缩模型同时使用的编码方法。

以 Huffman 编码为例，压缩过程可分解如下：

在 Huffman 编码中，编码器的实际输出由一组概率值来决定。使用这类编码器时，对出现概率非常高的符号生成一个很少比特数的码字，而对出现概率低的符号生成一个多比特数码字。模型（model）与程序的编码过程是不同的。这是因为与模型数据相对应的各种方法都可以使用相同的编码过程来产生他们的输出。比如，一个简单的 Huffman 编码程序可以使用一个模型。在这一模型中，设定符号在输入流中出现的大致的恒定的概率。而一个更为精致的程序

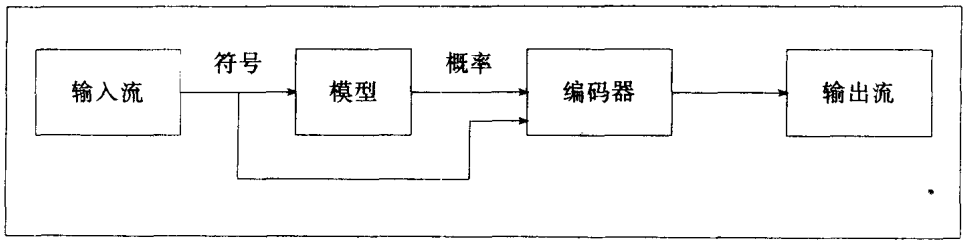


图2－1 huffman编码器的模型

可以是依据输入流中最近的10个符号来计算概率。虽然两个程序都使用Huffman编码来产生输出，但他们的压缩比很可能是截然不同的。

因此，当你再谈论这个话题时，对于“Huffman编码方法一般不产生很高的压缩比”这类问题一定要注意，你应该回答说：“我总是认为一个人跑的快与慢，更多地取决于他的腿部力量，而不是取决于他所穿的运动鞋”。

2.3 萌芽时期

或许数据压缩是信息论的基本表现。信息论是数学的一个分支，四十年代后期由Bell实验室的Clande Shannon始创。信息论涉及各种各样的有关信息方面的问题，其中包括信息存储和通信的各种方式。

数据压缩进入信息论领域是因为它涉及了冗余问题。消息中的冗余部分对于编码来说要占有额外的信息位，如果去掉这些额外的信息，就能够减少消息所占的空间。

信息论用术语熵(entropy)作为消息中被编码的数量衡量参数，熵这个词借助于热力学，并具有类似的含义。消息的熵越高，所含的信息量就越多。一个符号的熵定义为它的概率的对数的相反数。为要确定一个二进制消息中的信息容量，我们用以2为底的对数来表示熵。

Number of bits＝－log2(probability)

全部消息的熵，简单地说，就是全部符号的熵加在一起。

熵适用于数据压缩是因为它能够确定表示一个消息实际所需的比特数。举例来说：设出现在本书中的字符"e"的概率为1/16，则这一字符的信息容量是4比特，这样一来字符串"eeee"总的含量是20比特。如果我们用标准的8比特ASCII字符为这一信息编码的话，则我们实际使用40比特。20比特的熵与编码的40比特之间的差就是数据压缩的潜力所在。

关于熵要注意的一个事实就是：与热力学中熵的度量方法不同，我们不能使用给定信息的绝对信息容量来度量其熵。问题是当我们计算熵时，我们使用的是一个指出给定符号概率的数，这一概率数实际上是对应于给定模型的概率，而不是绝对数。如果我们改变模型这个概率也就随之改变。

对一个统计模型使用不同的阶时，可以清楚地看出概率的变化情况，统计模型依据先期出现在输入流中的符号来跟踪计算一个符号的概率。模型的阶决定了计入统计的先期出现的符号数目。比如：0阶模型不检查先期字符，1阶模型查看1个先期字符，等等。

对于同一字符，不同阶的模型可能产生极端不同的概率。比如，在0阶模型下，字母u可能只有1%的出现概率，而在1阶模型下，如前一字符为q，则u可能有95%的概率。