ICS 35.040
L 71

中华人民共和国国家标准

GB/T 16686—1996
idt ISO/IEC 11558:1992

信息技术　信息交换用数据压缩　具有嵌入字典的自适应编码　DCLZ 算法

**Information technology—
Data compression for information interchange—
Adaptive coding with embedded dictionary—DCLZ algorithm**

1996-12-18 发布　　1997-07-01 实施

国家技术监督局　发布

中 华 人 民 共 和 国
国 家 标 准
信息技术 信息交换用数据压缩
具有嵌入字典的自适应编码 DCLZ 算法
GB/T 16686—1996
*
中国标准出版社出版
北京复兴门外三里河北街 16 号
邮政编码:100045
电 话:68522112
中国标准出版社秦皇岛印刷厂印刷
新华书店北京发行所发行 各地新华书店经售

*
开本 880×1230 1/16 印张 1 字数 22 千字
1997 年 6 月第一版 1997 年 6 月第一次印刷
印数 1—1000
*
书号:155066·1-13942 定价 10.00 元
*
标 目 312—060

前　言

本标准等同采用国际标准 ISO/IEC 11558:1992《信息技术　信息交换用数据压缩　具有嵌入字典的自适应编码　DCLZ 算法》。

为适应信息交换，本标准规定了无损的压缩算法，以减少用编码形式表示的数据的位数。本标准无论在技术内容上，还是在编排格式上均与国际标准保持一致。

附录 A、附录 B 和附录 C 均是提示的附录。

本标准由中华人民共和国电子工业部提出。

本标准由电子工业部标准化研究所归口。

本标准起草单位：电子工业部标准化研究所。

本标准主要起草人：王宝艾、杨霖、郑洪仁。

ISO/IEC 前言

ISO(国际标准化组织)和 IEC(国际电工委员会)是世界性的标准化专门机构。国家成员体(它们都是 ISO 或 IEC 的成员国)通过国际组织建立的各个技术委员会参与制定针对特定技术范围的国际标准。ISO 和 IEC 的各技术委员会在共同感兴趣的领域内进行合作。与 ISO 和 IEC 有联系的其他官方和非官方国际组织也可参与国际标准的制定工作。

对于信息技术,ISO 和 IEC 建立了一个联合技术委员会,即 ISO/IEC JTC1。由联合技术委员会提出的国际标准草案需分发给国家成员体进行表决。发布一项国际标准,至少需要 75%的参与表决的国家成员体投标赞成。

国际标准 ISO/IEC 11558 是由欧洲计算机制造商协会(标准 ECMA—151)编制的,在特定的"快速跟踪程序"下,被 ISO/IEC JTC1 所采纳,同时被 ISO 和 IEC 国际组织通过。

附录 A、附录 B 和附录 C 仅供参考。

引　　言

在过去的十年里,ISO/IEC 颁布了许多有关磁带、盒式磁带和卡式磁带以及盒式光盘的国际标准。最近开发的这些媒体具有高的物理记录密度。为了最佳利用最终的数据容量,设计了多种压缩算法以减少用编码形式表示的用户数据的位数。

将来,这些压缩算法将由 ISO/IEC 建立的国际登记机构登记。登记将对每一个已登记的算法分配一数字的标识符。对于记录媒体,该标识符应包含在记录格式中以指明所使用的是哪种(哪些)压缩算法。

该国际标准是第一个有关压缩算法的国际标准。

目　　次

中华人民共和国国家标准

信息技术　信息交换用数据压缩　具有嵌入字典的自适应编码　DCLZ 算法

GB/T 16686—1996
idt ISO/IEC 11558:1992

Information technology—
Data compression for information interchange—
Adaptive coding with embedded dictionary—DCLZ algorithm

1　范围

本标准规定了无损的压缩算法，以减小用 8 位字节编码表达信息所要求的位数。此算法称为 DCLZ（根据 Lempel 和 Ziv 的数据压缩）。

本标准既不规定重置字典的策略，也不规定冻结字典的策略，因为它们是依赖于实现的。

当信息必须记录在可互换的媒体上时，此算法特别有用。它的使用并不局限于这种应用。

2　一致性

如果一个压缩算法的输出数据流满足第 7 章的要求，则认为与本标准一致。

3　引用标准

下列标准所包含的条文，通过在本标准中引用而构成为本标准的条文。本标准出版时，所示版本均为有效。所有标准都会被修订，使用本标准的各方应探讨使用下列标准最新版本的可能性。

ISO/IEC 11576:1994　信息技术　无损的数据压缩算法的登记规程

4　定义

4.1　代码值　code value

一个从 0 到 4095 变化的整数，它由压缩算法产生。

4.2　代码字　codeword

在以二进制表达代码值的输出流中，9、10、11 或 12 个连续位的集合。

4.3　压缩比　compression ratio

压缩算法的输入流中的位数除以压缩算法的输出流中的位数。

4.4　字典　dictionary

由 3832 项组成的一个表，它用于保留输入流中选择的字节串。每一项由大于 263 的唯一代码值标识。

4.5　空状态　empty state

字典中无数据的状态。

4.6　冻结状态　frozen state

不再有数据加入字典的状态。

国家技术监督局 1996-12-18 批准　　1997-07-01 实施

5 记法和同义词

——本标准中的数用十进制表示。

——EOR：记录结束。

6 算法标识符

本算法在国际登记组织登记的数字标识符是 32。

7 DCLZ 压缩算法

7.1 概述

DCLZ 压缩算法应以 8 位数据字节流的形式接受信息输入，并以新组织成 8 位字节的位流形式输出代码字。本算法应识别输入流中字节串的重复并应从输出流中排除这样的冗余。

随着由电子信息处理系统和设备产生、发送或记录的信息类型的多样化，数据重复度足够高以允许输出流比输入流有效地包含更少的位数。但是，在变态环境下，输出流可能比输入流包含更多的位数。实际上达到的压缩比依赖于具体的输入数据流的特征。

本算法的压缩是无损的，即它可能使用互补的解压缩算法完全恢复数据的原始表达。

本算法包含一些特征，这些特征帮助算法实现数据存储和检索设备在顺序方式下处理可变长度的数据记录。

7.2 运算原则

运算的基本原则是对出现在输入流中的字节串的一个字典进行编译，使用该字典检测重复串，并为每个重复串产生一个代码字。这个代码字表达一个代码值，它是对应重复串而被引用的字典项。

7.2.1 字典的编译

本算法开始运算之前，字典应设置成空状态(见 7.3.1.2)。

本算法应检测输入流并应查找第一个出现的唯一对或唯一串。唯一对是一个还没有被分配字典项的 2 字节串。n 个字节($n>2$)的唯一串是一个还没有被分配字典项的字节串；但是前面的 $n-1$ 个字节应已被分配字典项。字典项能分配给串的最大长度是 128 个字节。

当遇到唯一对时，本算法应输出一个代码字以表达该对的第一个字节的代码值。当遇到 n 字节的唯一串时，本算法应输出一个代码字以表达该串前面 $n-1$ 个字节的代码值。

然后，如果字典未被冻结(见 7.2.2)且 n 不超过 128，则它应把唯一对或唯一串输进字典并分配下一个未使用的代码值给该项。

从当前唯一对的第二个字节或从当前唯一串的最后一个字节开始，则本算法应继续检测输入流并查找下一个唯一对或唯一串。

7.2.2 冻结字典

当出现下列情况时，应认为字典处在冻结状态：

——所有有用的代码值都已被分配；

——本算法的执行程序已决定不把唯一对或唯一串输入字典，例如因为在字典中查找未占用空间所耗费的时间太多。

改掉字典冻结状态的唯一方法是重置空状态(见 7.3.1.1)。

7.2.3 重置字典为空状态

如果已输入算法的所有字节已由代码字表达，本算法允许在任何时候重置字典为空状态。

例如，如果因为当前字典项没有充分反映输入流当前的重复特征而使当前压缩程度不够，算法可以选择重置字典。

7.2.4 边界

在输入流中，自然边界可以存在于字节集之间。例如，输入流可以由记录的序列组成，每个记录包含一个或多个字节；这种情况下，自然边界存在于记录之间。本算法应提供在输出流中识别这种边界的方法，以致于它们可以被解压缩算法认识和重组。

这种标识应由 EOR 代码字输出完成(见 7.3.1.4)，后跟表达单个字节、字节对或字节串的代码值的代码字，这些代码字是为检测输入流的唯一对或唯一串的目的而暂时保存的。然后，输入流的检测应从下一记录的第一个字节继续开始。这个结果是输入流中边界之间的数据由输出流中相应边界之间的代码字完全表达。输出流中这样的边界被认为是定位于跟在紧随 EOR 代码字后的代码字的填充位的末端。

7.2.5 字典的重建

字典本身不作为特征项包含在输出流中。任何适用的解压缩算法都将重建字典并存储来自压缩算法输出流数据的原始表达式。

7.3 代码值

代码值 0 到 7 分配给控制代码。见 7.3.1。

代码值 8 到 263 分配给编码字节。见 7.3.2。

代码值 264 到 4095 分配给字典代码。见 7.3.3。

7.3.1 控制代码

定义 4 个控制代码，如下所述的代码值 0、1、2、3。4 到 7 的值不用。

7.3.1.1 字典冻结

该控制代码的代码值应为 0。它指示字典已经冻结，不强制算法输出该代码值。

只要输入至算法的所有字节已由代码字表达，就可以在算法决定冻结字典之后的任何时候输出它。

7.3.1.2 字典重置

该控制代码的代码值应为 1。它应是字典重置为空状态后由本算法输出的第一个代码值。在其他任何时候都不应当输出它。

必要时，在此输出流中，包含此代码值的代码字应由足够数量的用来填充下一个 8 位字节的且被置成 0 的比特跟随着。

7.3.1.3 增加代码字大小

该控制代码的代码值应为 2。它应该表示所有随后代码字(若有的话，直到下一个增加代码字大小的控制代码为止)的位数目比包含此代码值的代码字的位数目多 1。

7.3.1.4 记录结束(EOR)

该控制代码的代码值应为 3。它应该指示在输入流中一个记录边界存在于用下一个代码字表达的代码值表示的字节、对或串之后。

必要时，在此输出流中，包含该 EOR 代码值的代码字应由足够数量的用来填充下一个 8 位字节的且被置成 0 的比特跟随着。此输出流中的下一个代码字应由足够数量的用来填充下一个 8 位字节的且被置成 0 的比特跟随着。后者要求是为确保表示输入流中的一个记录的一组代码字从 8 位字节开始，在连续的 8 位字节结束。

7.3.2 编码字节

一个已编码字节表示输入流中的单个字节。一个已编码字节的完整集合表示单个字节的可能值的完整集合，即 0 到 255。一个已编码字节应这样计算：将此字节值加 8 来进行编码。

7.3.3 字典代码

一个字典代码识别一个字节对或一个字节串的字典项。

7.4 代码字

当字典是空状态时，代码字的大小是 9 位。如有必要，增加代码字大小，使之能够表达因太大而不能被当前代码字大小表达的代码值。见 7.3.1.3。

本算法也可以在其他任何时候增加代码字大小。如果当前代码字的大小大于表达要求的代码值所必须的大小，冗余位应比需要位在更高的位置，并置为0。

减少代码字大小的规则应是：

——本算法置字典为空状态；

——应输出表达字典重置控制代码的代码字，该代码字的大小应是如果存在的最后一个增加代码字大小控制代码所要求的大小；

——如果有必要，应将填充位置成0，直到下一个字节；

——下一个代码字应是9位的代码字。

输出时，代码字应从最低有效位开始，顺序输入代码字的位，从而构造成8位字节；从最右位开始，从右至左外理，构造成8位字节的连续位。当字节中所有位置都被使用，应输出此字节，随后代码字的位进入输出流的下一字节。

附 录 A
（提示的附录）
通常的 DCLZ 算法示例

下面描述了通常的 DCLZ 压缩算法。表述使用结构化的英语，也称为伪代码。该语言使用普通英语的词汇、语法和语义，加上特殊的单词(ENDIF)和格式约定。它表述指令或者是无条件执行，或者是按存在的特殊条件(或者多个条件的组合)执行。它也表述这样的条件。另外，它还提供注释；其目的是有助于读者理解本算法。

按照有条件执行或重复执行，将指令组成集合以文本缩排分层来表示。一个指令集合有相同等级的或不同等级的缩排。

除非要在哪里重复执行或有条件执行已被表明外，指令应从页上方向下顺序执行。注释被括在花括号内，它并不执行。

算法的具体实现可互不相同，例如决定冻结字典或重置字典为空状态的实现策略。

算法包括两部分。第一部分处理输入流并产生代码值，第二部分处理代码值并产生代码字。在输入流中，如果记录边界跟有在第一部分产生的特殊代码值表示的串的最后字节，则给该代码值添加一个指示符标志。出现这样一个标志就命令第二部分产生包含 EOR 代码值的代码字。

A1 代码值产生器

本算法的运算如表 A1 所示。术语 'Pop' 意为"输出一个代码值"。术语 'Pop&flag' 意为"输出一个带添加指示符标志的代码值"。被输出的此代码值括在圆括号内。

本算法的一个基本成分是串，称为 Current _string。它用于匹配输入流与相应的字典项。它可以为空。如果非空，它包含的字节数应少于 130 个。字典项是长度在 2 到 128 字节之间的串。因而，在字典中查找一个 129 字节的串将会失败。对待输入流中的最后字节就如同它后面跟有记录边界一样。

以下是对表 A1 所示算法的基本特征的总体描述。

A1.1 外部层

初始化字典为空状态，并输出字典重置控制代码。重复执行 A1.2 的指令，每重复一次便处理一个记录，直到整个输入流字节被处理完为止。

A1.2 处理一个记录

从输入流取得记录的第一个字节。如果该记录仅包含这个字节，则输出该字节的编码字节，并确保代码字产生器产生 EOR 代码字和适当的填充位。否则，重复执行 A1.3 中的指令，直到此记录的所有字节被处理完为止。

A1.3 处理一个字节对

从输入流取得下一个字节，从而形成对。如果在字典中有这个对，那么执行 A1.4 中的指令。否则，如果可能，把这个对加到字典中，或者如果不可能，则冻结字典；输出对的第一个字节的编码字节，并抛弃第一个字节。如果剩余的字节是此记录的最后字节，那么输出该字节的编码字节，并确保代码字产生器产生 EOR 代码字和适当的填充位。

A1.4 处理一个字节串

重复执行 A1.5 中的指令，把此对扩充到增加长度的串中，直到到达记录末尾或在字典中不存在该串。在前一种情况下，输出该串的字典代码，并确保代码字产生器产生 EOR 代码字和适当的填充位。在后一种情况下，如果可能，把这个串加到字典中，或者如果不可能，则冻结字典；输出此串的所有字节(除了最后字节外)的字典代码，抛弃这些字节。如果剩余的字节是此记录的最后字节，那么，输出该字节的编码字节，并确保代码字产生器产生 EOR 代码字和适当的填充位。

A1.5 扩充对或串

除非此对或串的当前最后字节是此记录的最后字节，从输入流取得下一字节，把它添加到当前对或串并为新形成的串查找字典。

表 A1 代码值产生器

```
Reset dictionary to empty state
Pop(Dictionary Reset)
Regard dictionary as not frozen
REPEAT{processing one record}
  Initialize Current _ string to next byte from input stream
  IF a record boundary follows that byte{i. e. record is only 1 byte}
    THEN Pop&flag(Encoded Byte for that byte)
    ELSE REPEAT{processing pairs and strings}
              Append next byte from input stream to Current _ string
              Search dictionary for Current _ string
              IF search failed{i. e. if a unique pair is found}
                  THEN IF dictionary is not frozen
                              THEN Attempt to add Current _ string to dictionary
                                        IF not successful
                                              THEN Regard dictionary as frozen
                                        ENDIF
                       ENDIF
                              Pop(Encoded Byte for 1st byte of Current _ string)
                              Remove 1st byte from Current _ string
                              IF record boundary follows remaining byte in Current _ string
                              THEN Pop&flag(Encoded Byte for that byte)
                                        Set Current _ string to null
                              ENDIF
                    ELSE REPEAT {a unique pair has not been found,so continue examining
                                        the input stream,looking for a unique string or
                                        a record boundary}
                          IF record boundary follows last byte of Current _ string
                              THEN Pop&flag(Dictionary Code for Current _ string
                                        set Current _ string to null
                              ELSE Append next byte from input stream to Current _ string
                                        Search dictionary for Current _ string
```

表 A1 （完）

```
            ENDIF
        UNTIL Current _ string is null or dictionary search fails
        IF search failed{i. e. if the string is a unique string}
            THEN IF dictionary is not frozen and
                    Current _ string length<129 bytes
                    THEN Attemptto add Current _ string to dictionary
                        IF not successful
                            THEN Regard dictionary as frozen
                        ENDIF
                    ENDIF
                Pop(Dictionary Code for entry of all but last byte of Current _ string)
                IF record boundary follows last byte of Current _ string
                        THEN Pop&flag(Encoded Byte for last byte)
                                Set Current _ string to null
                        ELSE Remove all bytes but last of Current _ string
                ENDIF
        ENDIF
    ENDIF
  UNTILCurrent _ string is null{i. e. processing of this record is complete}
ENDIF
UNTIL input stream is exhausted{i. e. processing of all records is complete}
```

A2 代码字产生器 （Codeword Generator）

本算法的本部分从代码值产生代码字。同时按需要在输出流中填充字节边界。

本算法的运算如表 A2 所示。要输出的代码字的内容括在圆括号内。

表 A2 代码字产生器

```
Set Codeword size to 9 bits
REPEAT{process all Code Values,one per cycle}
    Fetch next Code Value
    IF Code Value is Dictionary Reset
        THEN Output Codeword(Dictionary Reset)
            IF last bit of Codeword is not at byte boundary in output stream
                THEN Output ZERO bits to next byte boundary
            ENDIF
            Set Codeword size to 9 bits
        ELSEIF Codeword size is too small to express Code Value
                    THEN REPEAT
                                Output Codeword(Increment Codeword Size)
                                Increment Codeword size by one bit
                            UNTIL Codeword size is sufficient to express Code Value
                ENDIF
                IF Code Value is flagged
                        THEN Output Codeword(EOR)
                            IF last bit of Codeword is not at byte boundary in output stream
                                    THEN Output ZERO bits to next byte boundary
                                ENDIF
                ENDIF
                Output Codeword (Code Value)
                IF Code Value is flagged
                        THEN Output ZERO bits to next byte boundary
                ENDIF
    ENDIF
UNTIL Code Value stream is exhausted
```

附 录 B
（提示的附录）
对给定的输入流输出代码值的示例

下例中，自上而下运行。

表 B1

输入流：abcdabcdabcdabcdabcdaabcdxyz				
输入字节	字典代码	字典项	代码值输出	代码值含义
			1	字典重置
a				
b	264	ab	105	a 的编码字节
c	265	bc	106	b 的编码字节
d	266	cd	107	c 的编码字节
a	267	da	108	d 的编码字节
b				

表 B1 （完）

输入流：abcdabcdabcdabcdabcdaabcdxyz				
输入字节	字典代码	字典项	代码值输出	代码值含义
c	268	abc	264	ab 的字典代码
d				
a	269	cda	266	cd 的字典代码
b				
c				
d	270	abcd	268	abc 的字典代码
a				
b	271	dab	267	da 的字典代码
c				
d	272	bcd	265	bc 的字典代码
a				
b				
c	273	dabc	271	dab 的字典代码
d				
a				
a	274	cdaa	269	cda 的字典代码
b				
c				
d				
x	275	abcdx	270	abcd 的字典代码
y	276	xy	128	x 的编码字节
z	277	yz	129	y 的编码字节
			3	EOR
			130	z 的编码字节

在这个例子中，用于表示数据的位数目从 224 减小到 168。这是 1.333 的压缩比。对 x 更长的输入流压缩比更大，这是因为在输入流中重复的串的事例将更多而将做出更多的字典项。压缩比的典型值是从 2 到 4 的范围。

附 录 C
（提示的附录）
参 考 文 献

Mark J. Bianchi 等，《1/2inch 盘式磁带驱动器数据压缩》，惠普月刊，第 40 卷，No. 3 1989 年 6 月，第 26～31 页。