INTERNATIONAL STANDARD

ISO/IEC 11558

First edition
1992-09-01

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

Technologies de l'information — Compression de données pour l'échange d'information — Codage adaptif avec un dictionnaire incorporé — Algorithme DCLZ

Reference number
ISO/IEC 11558:1992(E)

Contents

	Page
1 Scope	1
2 Conformance	1
3 Normative references	1
4 Definitions	1
4.1 Code Value	1
4.2 Codeword	1
4.3 compression ratio	1
4.4 dictionary	1
4.5 empty state	1
4.6 frozen state	1
5 Conventions and notations	1
6 Algorithm identifier	1
7 DCLZ compression algorithm	2
7.1 Overview	2
7.2 Principle of operation	2
7.2.1 Compilation of the dictionary	2
7.2.2 Frozen dictionary	2
7.2.3 Resetting the dictionary to the empty state	3
7.2.4 Boundaries	3
7.2.5 Re-creation of the dictionary	3
7.3 Code Values	3
7.3.1 Control Codes	3
7.3.2 Encoded Bytes	4
7.3.3 Dictionary Codes	4
7.4 Codewords	4
Annexes	
A - Example of a generic DCLZ algorithm	5
B - Example of Code Values output for a given input stream	9
C - Bibliography	10

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11558 was prepared by the European Computer Manufacturers Association (as Standard ECMA-151) and was adopted, under a special "fast-track procedure", by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annexes A to C of this International Standard are for information only.

Patents

During the preparation of the ECMA standard, information was gathered on patents upon which application of the standard might depend. Relevant patents were identified as belonging to Hewlett Packard Limited. However, neither ECMA nor ISO/IEC can give authoritative or comprehensive information about evidence, validity or scope of patent and like rights. The patent holders have stated that licences will be granted under reasonable and non-discriminatory terms. Communications on this subject should be addressed to

Hewlett Packard Limited
Computer Peripherals Bristol
Filton road
Stoke Gifford
Bristol BS12 6QZ
United Kingdom

Introduction

In the past decades ISO/IEC have published numerous International Standards for magnetic tapes, magnetic tape cassettes and cartridges, as well as for optical disk cartridges. Those media developed recently have a very high physical recording density. In order to make an optimal use of the resulting data capacity, compression algorithms have been designed which allow a reduction of the number of bits required for the representation of user data in coded form.

In future, these compression algorithms will be registered by an International Registration Authority to be established by ISO/IEC. The registration will consist in allocating to each registered algorithm a numerical identifier. For a recorded medium this should be included in the recorded format to indicate which compression algorithm(s) has been used.

This International Standard is the first of a series of International Standards for compression algorithms.

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

1 Scope

This International Standard specifies a lossless compression algorithm to reduce the number of bits required to represent information coded by means of 8-bit bytes. This algorithm is known as DCLZ (Data Compression according to Lempel and Ziv).

This International Standard specifies neither the strategy for resetting the dictionary nor that for freezing it, as these are implementation-dependent.

This algorithm is particularly useful when information has to be recorded on an inter-changeable medium. Its use is not limited to this application.

2 Conformance

A compression algorithm shall be in conformance with this International Standard if its output data stream satisfies the requirements of clause 7.

3 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the standard listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 11576, *Information technology - Procedure for the registration of algorithms for the lossless compression of data*

4 Definitions

4.1 **Code Value:** An integer in the range 0 to 4 095 that is generated by the compression algorithm.

4.2 **Codeword:** A set of 9, 10, 11 or 12 consecutive bits in the output stream to express a Code Value in binary form.

4.3 **compression ratio:** The number of bits in the input stream of the compression algorithm divided by the number of bits in the output stream of the compression algorithm.

4.4 **dictionary:** A table, comprising 3 832 entries, that is used to retain strings of bytes selected from the input stream. Each entry is identified by a unique Code Value that is greater than 263.

4.5 **empty state:** The state in which no data is in the dictionary.

4.6 **frozen state:** The state in which no further data shall be added to the dictionary.

5 Notations and acronyms

- Numbers in this International Standard are expressed in decimal notation.
- EOR: end of record.

6 Algorithm identifier

The numeric identifier of this algorithm in the International Register is 32.

7 DCLZ compression algorithm

7.1 Overview

The DCLZ compression algorithm shall accept information input, in the form of a stream of 8-bit data bytes, and shall output Codewords, in the form of a stream of bits which are organised into 8-bit bytes. The algorithm shall identify repetition of byte strings in the input stream and shall exclude such redundancy from the output stream.

With many types of information generated, transmitted or recorded by electronic information processing systems and equipment, the degree of repetition in data is sufficiently high to permit the output stream to contain significantly fewer bits than the input stream. Under degenerate circumstances, however, the output stream may contain more bits than the input stream. The compression ratio achieved in practice is dependent on the characteristics of the actual input data stream.

Compression by this algorithm is lossless, i.e. it is possible to restore exactly the original representation of data by means of a complementary decompression algorithm.

The algorithm contains features which aid its implementation in data storage and retrieval equipment which handles, in a sequential manner, data records of varying length.

7.2 Principle of operation

The fundamental principle of operation is the compilation of a dictionary of strings of bytes which occur in the input stream, the use of that dictionary to detect repetition, and the generation of a Codeword for each repeated string. The Codeword expresses a Code Value which is the reference to the dictionary entry for the repeated string.

7.2.1 Compilation of the dictionary

Prior to the commencement of operation of the algorithm, the dictionary shall be reset to the empty state (see also 7.3.1.2).

The algorithm shall examine the input stream and shall search for the first occurrence of a unique pair or a unique string. A unique pair is a 2-byte string which has not yet been allocated a dictionary entry. A unique string of n bytes ($n > 2$) is one which has not yet been allocated a dictionary entry; however, the first n-1 bytes shall have been already allocated a dictionary entry. The maximum length of a string for which a dictionary entry can be allocated shall be 128 bytes.

Upon encountering a unique pair, the algorithm shall output a Codeword which expresses the Code Value for the first byte of the pair. Upon encountering a unique string of n bytes, the algorithm shall output a Codeword which expresses the Code Value for the first n-1 bytes of the string.

It shall then enter the unique pair or unique string into the dictionary and assign the next unused Code Value to the entry, provided that the dictionary is not frozen (see 7.2.2) and that n does not exceed 128.

Starting with the 2nd byte of the current unique pair or the last byte of the current unique string, the algorithm shall then continue to examine the input stream and search for the next unique pair or unique string.

7.2.2 Frozen dictionary

The dictionary shall be considered to be in the frozen state in the following cases:

- all available Code Values have been assigned;
- the implementation of the algorithm has decided not to enter a unique pair or a unique string into the dictionary, for example because the search for free space in the dictionary takes too much time.

The only means by which the dictionary may be removed from the frozen state is by being reset to the empty state (see also 7.3.1.1).

7.2.3 Resetting the dictionary to the empty state

The algorithm is permitted to reset the dictionary to the empty state at any time, provided that all bytes which have been input to the algorithm have been expressed by Codewords.

The algorithm may, for example, choose to reset the dictionary if the current degree of compression is not adequate because the current dictionary entries do not reflect the current repetition characteristics of the input stream to a sufficient extent.

7.2.4 Boundaries

Within the input stream, natural boundaries may exist between collections of bytes. For example, the stream may consist of a sequence of records, each comprising one or more bytes; in such a case, a natural boundary exists between records. The algorithm shall provide a means for identifying such boundaries in the output stream, so that they are recognized and re-constituted by a decompression algorithm.

Such identification shall be achieved by the output of the EOR Codeword, (see 7.3.1.4), followed by a Codeword which expresses the Code Value for the single byte, pair of bytes or string of bytes which is being held temporarily for the purpose of examining the input stream for a unique pair or a unique string. Examination of the input stream shall then continue from the first byte of the next record. The result is that the data between boundaries in the input stream is wholly represented by Codewords between corresponding boundaries in the output stream. Such a boundary in the output stream is considered to be located at the end of the pad bits that follow the Codeword that follows the EOR Codeword.

7.2.5 Re-creation of the dictionary

The dictionary is not itself included in the output stream as a distinct item. Any appropriate decompression algorithm will re-create the dictionary and restore the original representation of the data from the output stream of the compression algorithm.

7.3 Code Values

Code Values in the range 0 to 7 shall designate Control Codes. See 7.3.1.

Code Values in the range 8 to 263 shall designate Encoded Bytes. See 7.3.2.

Code Values in the range 264 to 4 095 shall designate Dictionary Codes. See 7.3.3

7.3.1 Control Codes

Four Control Codes are defined, with Code Values 0, 1, 2 and 3 as described below. Values in the range 4 to 7 shall not be used.

7.3.1.1 Dictionary Frozen

This Control Code shall have the Code Value 0. It shall indicate that the dictionary has been frozen. It is not mandatory for the algorithm to output this Code Value.

It may be output at any time after the algorithm has decided to freeze the dictionary, provided that all bytes which have been input to the algorithm have been expressed by Codewords.

7.3.1.2 Dictionary reset

This Control Code shall have the Code Value 1. It shall be the first Code Value which is output by the algorithm after the dictionary is reset to its empty state. It shall not be output at any other time.

The Codeword which contains this Code Value shall be followed in the output stream, if necessary, by a sufficient number of bits set to ZERO to pad to the next 8-bit byte.

7.3.1.3 Increment Codeword Size

This Control Code shall have the Code Value 2. It shall indicate that the number of bits in all subsequent Codewords (until after the next Increment Codeword Size Control Code, if any) is greater by 1 than the number of bits in the Codeword which contains this Code Value.

7.3.1.4 End of Record (EOR)

This Control Code shall have the Code Value 3. It shall indicate that, in the input stream, a record boundary exists after the byte, pair or string which is represented by the Code Value expressed by the next Codeword.

The Codeword which contains this EOR Code Value shall be followed in the output stream, if necessary, by a sufficient number of bits set to ZERO to pad to the next 8-bit byte. The next Codeword in the output stream shall be followed by a sufficient number of bits set to ZERO to pad to the next 8-bit byte. This latter requirement ensures that the set of Codewords which represents a record in the input stream begins with an 8-bit byte and ends with a subsequent 8-bit byte.

7.3.2 Encoded Bytes

An Encoded Byte represents a single byte in the input stream. The complete set of Encoded Bytes represents the complete set of possible values of a single byte, i.e. 0 to 255. An Encoded Byte shall be computed by adding 8 to the value of the byte to be encoded.

7.3.3 Dictionary Codes

A Dictionary Code identifies a dictionary entry for a pair or a string of bytes.

7.4 Codewords

When the dictionary is in the empty state, the Codeword size shall be 9 bits. The Codeword size shall be increased, as necessary, to be capable of expressing Code Values which are too large to be expressed by the current Codeword size. See 7.3.1.3.

The Codeword size may also be increased by the algorithm at any other time. If the current Codeword size is greater than that which is necessary to express the desired Code Value, the redundant bits shall be in the more significant positions than the required bits, and shall be set to ZERO.

The only procedure for decreasing the Codeword size shall be

- the algorithm shall reset the dictionary to the empty state;
- it shall output a Codeword expressing the Dictionary Reset Control Code; this Codeword shall have the size required by the last Increment Codeword Size Control Code, if any;
- pad bits, set to ZERO, if any are required, shall follow to the next byte;
- the next Codeword shall be a 9-bit Codeword.

Upon being output, Codewords shall be organised into 8-bit bytes by entering Codeword bits in sequence, starting with the least significant bit, into successive bits of an 8-bit byte, starting with the rightmost bit and proceeding from right to left. When all positions in a byte have been used, the byte shall be output, and subsequent Codeword bits shall be entered into the next byte of the output stream.

Annex A

(informative)

Example of a generic DCLZ algorithm

The following is a description of a generic DCLZ compression algorithm. The description is in structured English, also known as pseudo-code. The language utilises normal English vocabulary, syntax and semantics, plus a special word (ENDIF) and a style convention. It expresses instructions which are either to be executed unconditionally, or to be executed if a particular condition (or combination of conditions) exists. It also expresses such conditions. Additionally, it provides for annotations; these are intended to aid the reader in understanding the algorithm.

The grouping of instructions into sets which are subject to conditional or repeated execution is denoted by a hierarchy of text indentation. A set comprises all instructions which have the same or a greater degree of indentation.

Except where repetition or conditional execution is expressed, instructions are executed in sequence from the top of the page. Comments are enclosed in curly brackets and are not instructions.

Specific implementations of the algorithm may differ from each other, for example in their strategies for deciding to freeze the dictionary or reset it to an empty state.

The algorithm is in two parts. The first part processes the input stream and generates Code Values. The second part processes Code Values and generates Codewords. If, in the input stream, a record boundary follows the last byte of the string represented by a particular Code Value that is generated by the first part, an indicator flag is attached to that Code Value. The presence of such a flag instructs the second part to generate a Codeword which contains the EOR Code Value.

A.1 Code Value Generator

The operation of the algorithm is shown in table A.1. The term 'Pop' is used to mean 'output a Code Value'. The term 'Pop&flag' is used to mean 'output a Code Value with an indicator flag attached'. The Code Value to be output is enclosed in parentheses.

An essential component of this algorithm is a string, named Current_string. It is used for matching the input stream against dictionary entries. It may be null; if non-null, it contains less than 130 bytes. Dictionary entries are strings of between 2 and 128 bytes in length. Therefore, a search for a 129-byte string in the dictionary will fail. The final byte in the input stream is treated as if it is followed by a record boundary.

The following is a general description of the essential features of the algorithm shown in table A.1.

A.1.1 Outer level

Initialize the dictionary to the empty state and output a Dictionary Reset Control Code. Execute repeatedly the instructions of A.1.2, processing one record during each iteration, until all input stream bytes have been processed.

A.1.2 Process one Record

Get the first byte of the record from the input string. If the record comprises only this byte, then output the Encoded Byte for this byte, and ensure that the Codeword Generator will generate the EOR Codeword and appropriate pad bits. Otherwise, execute repeatedly the instructions in A.1.3 until all bytes of this record have been processed.

A.1.3 Process a Pair of Bytes

Get the next byte from the input stream, thus forming a pair. If this pair is in the dictionary, then execute the instructions in A.1.4. Otherwise, add this pair to the dictionary if possible or freeze the dictionary if it is not possible, output the Encoded Byte for the first byte of the pair, discard the first byte, and, if the remaining byte is the last byte of the record, output the Encoded Byte for this byte and ensure that the Codeword Generator will generate the EOR Codeword and appropriate pad bits.

A.1.4 Process a String of Bytes

Execute repeatedly the instructions in A.1.5 to extend the pair into a string of increasing length until the end of the record is reached or the string is not in the dictionary. In the former case, output the Dictionary Code for the string and ensure that the Codeword Generator will generate the EOR Codeword and appropriate pad bits. In the latter case, add the string to the dictionary if possible or freeze the dictionary if it is not possible, output the Dictionary Code for all bytes of the string except the last, discard those bytes, and, if the remaining byte is the last byte of the record, output the Encoded Byte for this byte and ensure that the Codeword Generator will generate the EOR Codeword and appropriate pad bits.

A.1.5 Extend the Pair or String

Unless the current last byte of the pair or string is the last byte of the record, get the next byte from the input stream, append it to the current pair or string and search the dictionary for the newly-formed string.

Table A.1 - Code Value Generator

```
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
                    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 Attempt to 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
         UNTIL Current_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}
```

A.2 Codeword Generator

This part of the algorithm generates Codewords from Code Values. Padding to the byte boundaries in the output stream shall also be performed as necessary.

The operation of the algorithm shall be as shown in table A.2. The content of the Codeword to be output is enclosed in parentheses.

Table A.2 - Codeword Generator

```
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
      ELSE IF 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
```