LZW

Sylvester L. Tuohy
Sylvester L. Tuohy is Professor of Computer Science at Pace University in Westchester where he teaches a wide range of courses at both the undergraduate and graduate levels. These include computer architecture; data communications and networks; computer networks and the Internet; operating systems; graphics; data structures and algorithms; beginning and advanced courses in Java programming; the new graduate survey course, Computer Systems and Concepts; and even Introduction to Computing.

Dr. Tuohy holds a B.S. in physics from Iona College, an M.S. in mathematics from Seton Hall University, an M.S. in computer science from the Polytechnic Institute of New York, and a Ph.D. in mathematics from Stevens Institute of Technology. His doctoral dissertation was on multitime expansions of perturbed differential equations.
LZW

Sylvester L. Tuohy

Computer Science Department
Pace University
Pleasantville, N.Y. 10570
stuohy@pace.edu

Introduction

LZW is a lossless compression algorithm named after Abraham Lempel, Jacob Ziv and Terry Welch. It is used to compress GIF and TIFF images and it is the first stage of PKZIP (which is then followed by Shannon-Faro trees algorithm - a "probabilistic compression" method- popular letters get the shortest code).

The beauty of the algorithm is its simplicity and the fact that it can compress on one pass through the data. LZW can be a lesson in Algorithms, Data Communications, Data Mining, Graphics or any class that involves compressing data. It is an algorithm that is worth learning about.

In this paper we will take an alphabet of 3 characters in order to create repetition in a text stream. First, an input stream is compressed and a dictionary created. Second, the compressed output stream is decompressed by using a newly created dictionary. In conclusion, we have some thoughts about the algorithm.

Compression

The Algorithm:
outputString= null;

Fill the dictionary with each character of the alphabet.(1.X 2.Y 3.Z 4. . .)

Repeat {

Parse the inputStream creating new unique dictionary entries that are 1 character more than a previous dictionary entry. (8. YXY . . . 11. YXXY or 11. 8Y )

Send the previous code to the outputString (8 to outputString)

The last character of the last created dictionary entry is the first character of the next dictionary entry. (12. YXYYZ 13. Z-- )

An Example: Y Y X X Y X Y Y X Y Y X Y Y Z X Y Z - -

Dictionary    AltDictionary    outputString
1. X
2. Y
3. Z
4. YY 2Y 2
5. YX 2X 2
6. XX 1X 1
7. XY 1Y 1
8. YXY 5Y 5
9. YYX 4X 4
10. XYY 7Y 7
11. YXY 8Y 8
12. YXYYZ 11Z 11
13. ZX 3X 3
14. XYZ 7Z 7
15. Z-- 3- 3

DeCompress

The Algorithm:

deCompressedString = null;

inputCodes = Compressed file (2 2 1 1 5 4 7 8 11 3 7 3 - -)

Fill the dictionary with each character of the alphabet.(1.X 2.Y 3.Z 4. . . )

Get w =lookup(first compressedCode#) in the Dictionary (w=Y=lookup(2) )

Write this value to the next dictionary entry.(4.Y- or 2-) and output it to deCompressedString

Repeat {

Get w = lookup(next compressedCode#) in the Dictionary (w = Y = lookup(2))

Concatenate w[0] to the dictionary entry not yet completed. (4.Y+w[0] or YY or 2Y)

Start the next dictionary entry with the last letter of the entry just finished. (5.w[0]- or Y-)

Continue parsing w and entering into the dictionary using the rule: “New unique entries are 1
character more than a previous entry”.

(7. X- and w = lookup(5) = YX therefore we then have 7. XY or 1Y 8. YX-)

Write w to deCompressedString. }

Note: It may happen that while working on a not yet completed dictionary entry that
compressedCode# is next. (11. YXY- and w= lookup(11) = YXY- no problem the same rule
applies the w[0] = Y completes 11. ie. the first letter of 11. in the inputCode is the last letter of
11. in the dictionary)

An Example: 2 2 1 1 5 4 7 8 11 3 7 3

w=Lookup(compressedCode#) Dictionary AltDictionary deCompressedString

1. X
<table>
<thead>
<tr>
<th>Step</th>
<th>Code</th>
<th>Code</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Y-</td>
<td>2Y</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>Y-</td>
<td>2-</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>X-</td>
<td>1-</td>
<td></td>
</tr>
</tbody>
</table>

**Conclusion**

With text LZW compresses to 50% of the original; with graphics it can be 10% of the original. The dictionary in reality uses 12 bits (1.5 Bytes) or has 4K of entries. When the dictionary is full a LRU algorithm is used for replacements. Sometimes a special character is sent and the whole dictionary is flushed and we continue. Most of the time is spent with the dictionary — so hashing and sorting the dictionary by frequency is used.
Sperry received the patent (U.S. 4,558,302) for LZW Compression in December 1985. CompuServe, the first online service in the United States in the 1980's, had to pay royalties, $0.15 or 1.5%, of the sale price of any programs using their browser and using LZW for compressing GIF files. Until the patent expired in June 2003, this set off a great debate about ownership and use of the LZW algorithm.

If ESCC has a survey for the all time best algorithms my nominations are: Boyer-Moore, LZW, Knuth Morris Pratt String Search, Prim’s Minimum Spanning Tree (or Dijkstra’s Shortest Path), Warshall’s Transitive Closure (adapted to finding the diameter of a network), Rivest-Shamir-Adleman public key encryption and Hamming Code. Maybe a professor that teaches Data Structures or Graphics has a few favorite sorts, searches, tree, graph or connectivity algorithms to nominate.

References


Smith, S. http://www.dsptguide.com/ch27/5.htm


http://en.wikipedia.org/wiki/LZW
The Ivan G. Seidenberg
School of Computer Science and Information Systems
Pace University
Technical Report Series

EDITORIAL BOARD

Editor:
Allen Stix, Computer Science, Pace--Westchester

Associate Editors:
Constance A. Knapp, Information Systems, Pace--Westchester
Susan M. Merritt, Dean, The Ivan G. Seidenberg School of Computer Science and Information Systems --Pace

Members:
Howard S. Blum, Computer Science, Pace--New York
Mary F. Courtney, Computer Science, Pace--Westchester
Nicholas J. De Lillo, Mathematics and Computer Science, Manhattan College
Daniel Farkas, Information Systems, Pace--Westchester
Fred Grossman, Information Systems; Doctor of Professional Studies, Pace--New York and White Plains
Fran Goertzel Gustavson, Information Systems, Pace--Westchester
Joseph F. Malerba, Computer Science, Pace--Westchester
John S. Mallozzi, Computer Information Sciences, Iona College
John C. Molluzzo, Information Systems, Pace--New York
Pauline Mosley, Technology Systems, Pace--New York
Narayan S. Murthy, Computer Science, Pace--New York
Catherine Ricard, Computer Information Sciences, Iona College
Judith E. Sullivan, CSIS Alumna, MS in CS from Pace--Westchester
Sylvester Tuchy, Computer Science, Pace--Westchester

The Ivan G. Seidenberg School of Computer Science and Information Systems, through the Technical Report Series, provides members of the community an opportunity to disseminate the results of their research by publishing monographs, working papers, and tutorials. Technical Reports is a place where scholarly striving is respected.

All preprints and recent reprints are requested and accepted. New manuscripts are read by two members of the editorial board; the editor decides upon publication. Authors, please note that production is Xerographic from your submission. Statements of policy and mission may be found in issues #29 (April 1990) and #34 (September 1990).

Please direct submissions as well as requests for single copies to:

Allen Stix
The Ivan G. Seidenberg School of Computer Science and Information Systems
Goldstein Academic Center
Pace University
861 Bedford Road
Pleasantville, NY 10570-2799