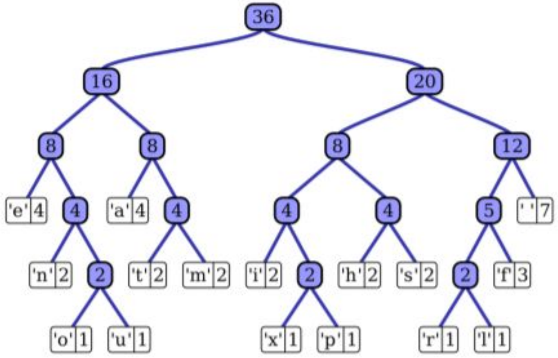
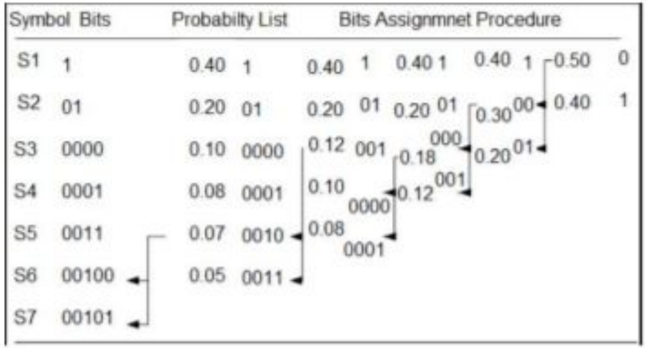
CO-203 Object Oriented Programming Innovative Project Report

COMPRESSION USING HUFFMAN CODING

GUI built on Qt5 using C++





Symbol	Bits	Probabilty List		Bits Assignmnet Procedure							
S1	1	0.40	1	0.40	1	0.40	1	0.40	1	0.50	0
S2	01	0.20	01	0.20	01	0.20	01	0.30	00	0.40	1
S3	0000	0.10	0000	0.12	001	0.18	000	0.20	01		
S4	0001	0.08	0001	0.10	0000	0.12	001				
S5	0011	0.07	0010	0.08	0001						
S6	00100	0.05	0011								
S7	00101										

Ishan Khanna
2K19/EE/120

COMPRESSION USING HUFFMAN CODING

Huffman code is a technique that is commonly used for lossless data compression. The process of such compression proceeds through **Huffman coding.** This algorithm was developed by David A. Huffman who at the time was a Sc.D. student at MIT, USA. It was published in the 1952 paper which was titled "A Method for the Construction of Minimum-Redundancy Codes".

The output obtained from Huffman's algorithm can be seen as a variable-length code table for encoding a source symbol (such as a character in a file). The algorithm created this table from the frequency of occurrence (also known as *weight) for* each possible value of the source symbol. However, even though it is among the most optimal methods for encoding symbols separately, it is not always optimal among all compression methods.

Huffman Encoder utilises Huffman coding trees to reduce the count of bits that are needed to represent any symbol. It is a variable length encoding.

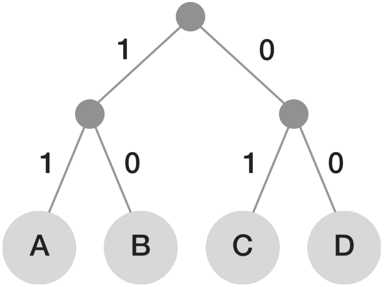
For Example :

- Message “CAB” is compressed to

011110.

- Otherwise the message would have been encoded using its ASCII representation as follows:

0010001100100001001000010.

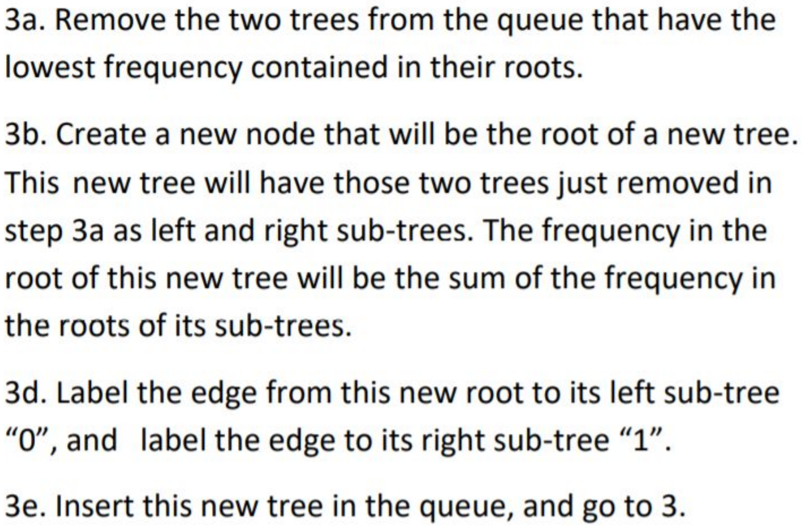


Algorithm

1) Find the frequency of each distinct symbol in the input.

2) Make a priority queue of single-node trees. Every node in the initial queue represents a distinct symbol from the set of possible symbols, and contains the frequency of that symbol in the message to be coded. Symbols with a frequency of zero are ignored.

3) Loop while there is more than 1 tree in the queue:



3a. Remove the two trees from the queue that have the lowest frequency contained in their roots.

3b. Create a new node that will be the root of a new tree. This new tree will have those two trees just removed in step 3a as left and right sub-trees. The frequency in the root of this new tree will be the sum of the frequency in the roots of its sub-trees.

3d. Label the edge from this new root to its left sub-tree “0”, and label the edge to its right sub-tree “1”.

3e. Insert this new tree in the queue, and go to 3.

4) Return the one tree in the queue as the Huffman code tree.

Time and space complexity analysis for Huffman Coding Algorithms

Time Complexity:

Since efficient priority queue data structures require $O(\log n)$ time complexity, and a tree that has n leaves must have $2n-1$ nodes. So, this algorithm operates in $O(n \log n)$ time, where n is the number of symbols.

Space Complexity:

If n is the count of distinct symbols in the input file, then we require $O(n)$ space for building the tree. Hence, space complexity of Huffman coding is $O(n)$.

Design and Implementation

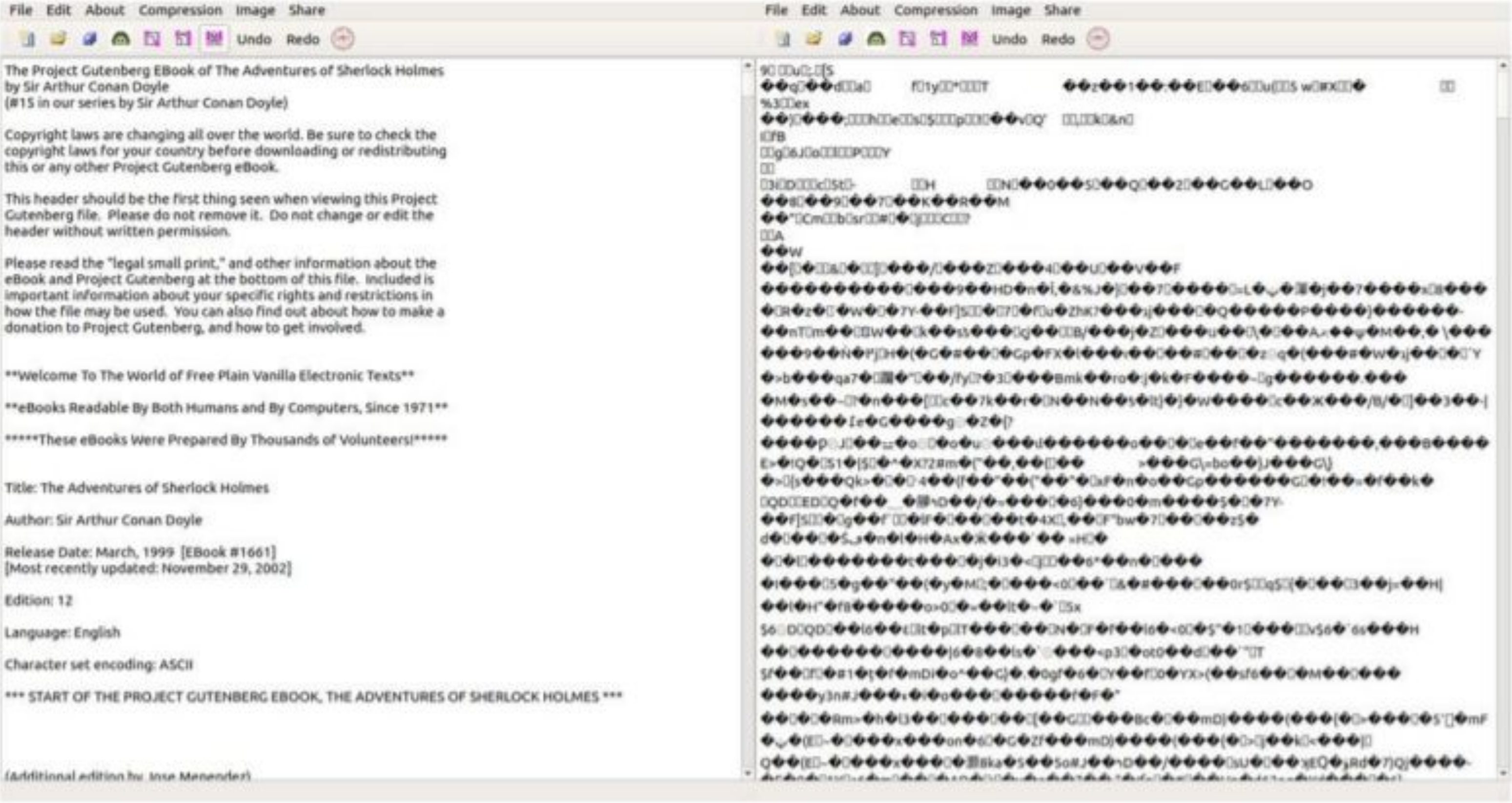
- I have built a GUI using Qt5. It includes a text view and an image view.
- The text view provides basic text editing features along with the options to compress and decompress the currently opened file.
- It also has provisions for sharing by e-mail option printing.
- The Image View includes an in-built image-display area and options to compress the file to JPEG.
- It also provides an option to print the image and to open an image for viewing.

- Features:

1. Text editor – Cut, Copy, Paste, Undo, Redo
2. Text Compression
3. Text Decompression
4. Image Viewer
5. Image Compression
6. Sharing by Mail
7. Print

Results

Text Compression



The Project Gutenberg EBook of The Adventures of Sherlock Holmes
by Sir Arthur Conan Doyle
(#15 in our series by Sir Arthur Conan Doyle)

Copyright laws are changing all over the world. Be sure to check the copyright laws for your country before downloading or redistributing this or any other Project Gutenberg eBook.

This header should be the first thing seen when viewing this Project Gutenberg file. Please do not remove it. Do not change or edit the header without written permission.

Please read the "legal small print," and other information about the eBook and Project Gutenberg at the bottom of this file. Included is important information about your specific rights and restrictions in how the file may be used. You can also find out about how to make a donation to Project Gutenberg, and how to get involved.

Welcome To The World of Free Plain Vanilla Electronic Texts

eBooks Readable By Both Humans and By Computers, Since 1971

*****These eBooks Were Prepared By Thousands of Volunteers!*****

Title: The Adventures of Sherlock Holmes

Author: Sir Arthur Conan Doyle

Release Date: March, 1999 [EBook #1661]
[Most recently updated: November 29, 2002]

Edition: 12

Language: English

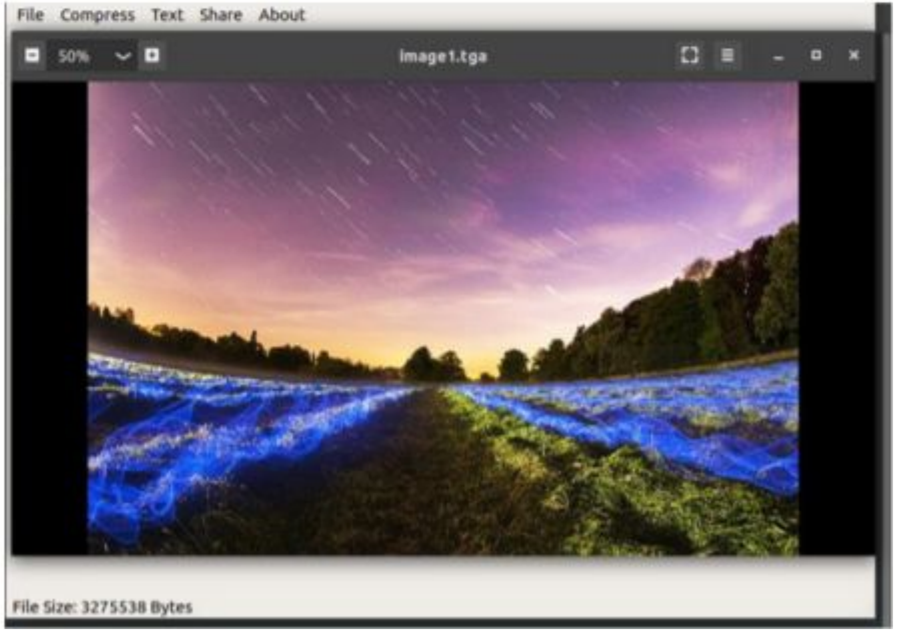
Character set encoding: ASCII

*** START OF THE PROJECT GUTENBERG EBOOK, THE ADVENTURES OF SHERLOCK HOLMES ***

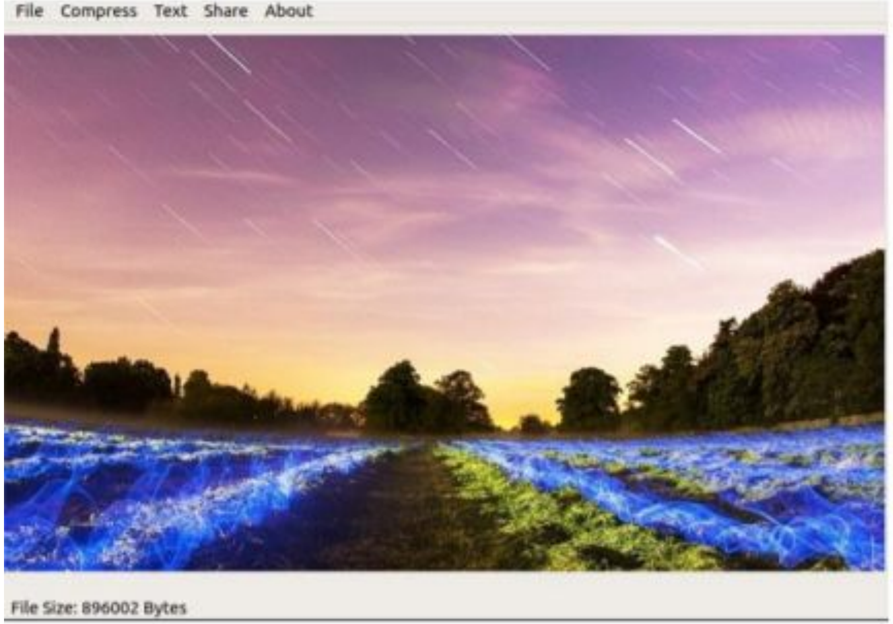
(Additional editing by Jose Menendez)

Results

Image Compression

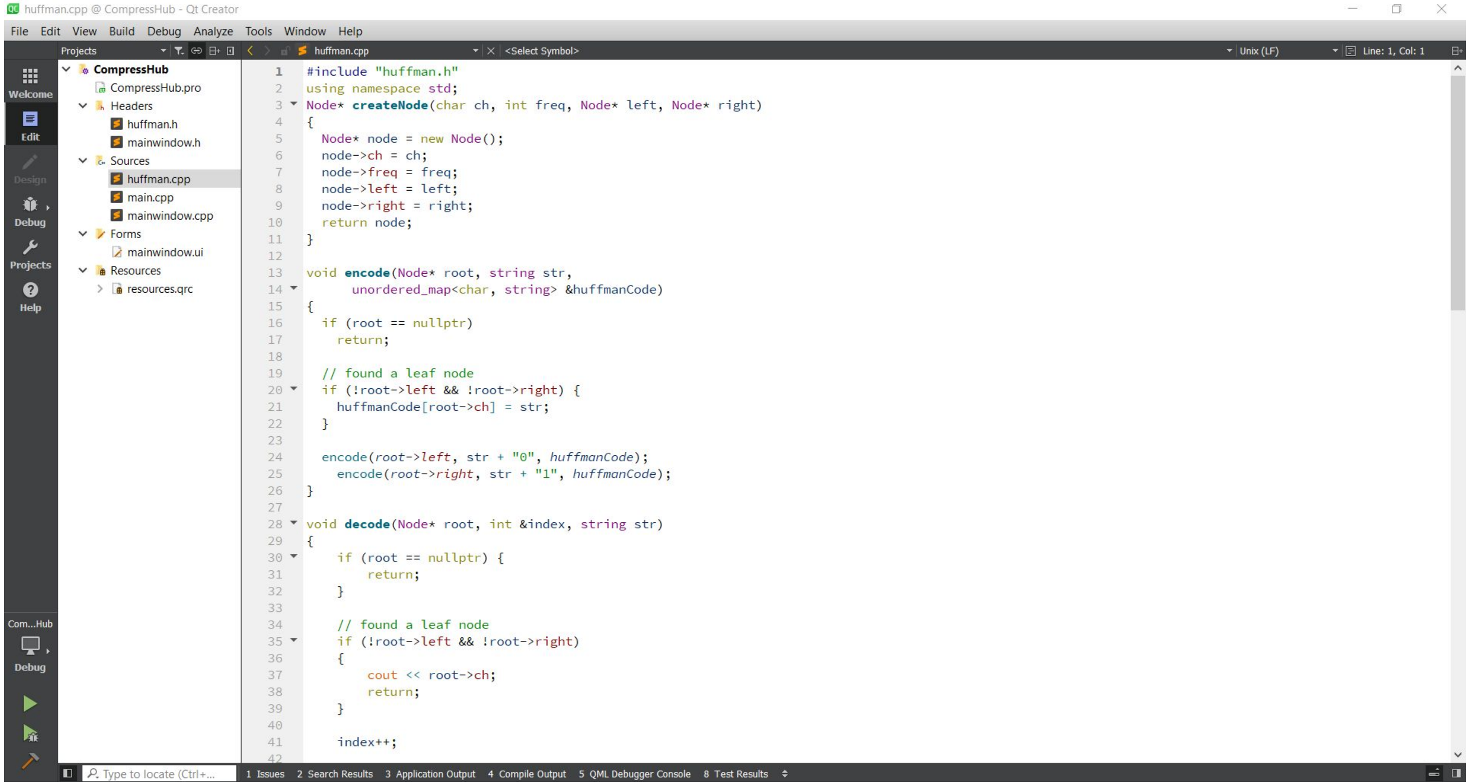


File Size: 3275538 Bytes



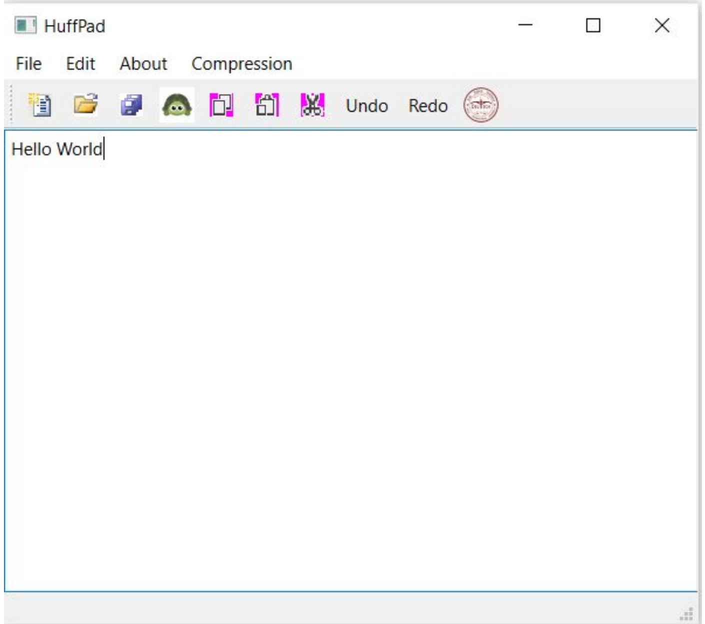
File Size: 896002 Bytes

Results

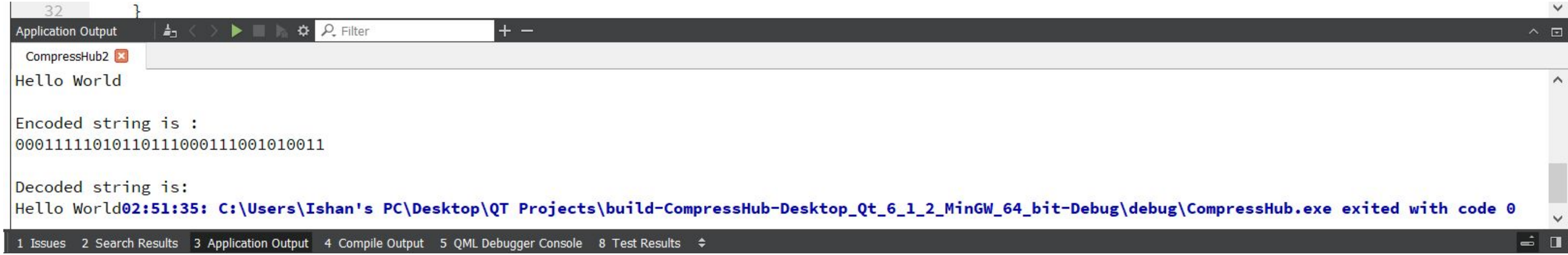


```cpp
#include "huffman.h"
using namespace std;
Node* createNode(char ch, int freq, Node* left, Node* right)
{
  Node* node = new Node();
  node->ch = ch;
  node->freq = freq;
  node->left = left;
  node->right = right;
  return node;
}

void encode(Node* root, string str,
      unordered_map<char, string> &huffmanCode)
{
  if (root == nullptr)
    return;

  // found a leaf node
  if (!root->left && !root->right) {
    huffmanCode[root->ch] = str;
  }

  encode(root->left, str + "0", huffmanCode);
    encode(root->right, str + "1", huffmanCode);
}

void decode(Node* root, int &index, string str)
{
    if (root == nullptr) {
        return;
    }

    // found a leaf node
    if (!root->left && !root->right)
    {
        cout << root->ch;
        return;
    }

    index++;
```

Results



Hello World



```
Hello World

Encoded string is :
00011111010110111000111001010011

Decoded string is:
Hello World02:51:35: C:\Users\Ishan's PC\Desktop\QT Projects\build-CompressHub-Desktop_Qt_6_1_2_MinGW_64_bit-Debug\debug\CompressHub.exe exited with code 0
```

References

1. *Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy Codes" (PDF). Proceedings of the IRE.* **40** *(9): 1098–1101. doi:10.1109/JRPROC.1952.273898.*
2. *Van Leeuwen, Jan (1976). "On the construction of Huffman trees" (PDF). ICALP: 382–410. Retrieved 2014-02-20.*
3. *Ze-Nian Li; Mark S. Drew; Jiangchuan Liu (2014-04-09). Fundamentals of Multimedia. Springer Science & Business Media. ISBN 978-3-319-05290-8.*
4. J. Duda, K. Tahboub, N. J. Gadil, E. J. Delp, *The use of asymmetric numeral systems as an accurate replacement for Huffman coding*, Picture Coding Symposium, 2015.
5. *Huffman, Ken (1991). "Profile: David A. Huffman: Encoding the "Neatness" of Ones and Zeroes". Scientific American: 54–58.*
6. *Gribov, Alexander (2017-04-10). "Optimal Compression of a Polyline with Segments and Arcs". arXiv:1604.07476 [cs.CG].*
7. *Gallager, R.G.; van Voorhis, D.C. (1975). "Optimal source codes for geometrically distributed integer alphabets". IEEE Transactions on Information Theory.* **21** *(2): 228–230. doi:10.1109/TIT.1975.1055357.*
8. *Abrahams, J. (1997-06-11). Written at Arlington, VA, USA. Division of Mathematics, Computer & Information Sciences, Office of Naval Research (ONR). "Code and Parse Trees for Lossless Source Encoding". Compression and Complexity of Sequences 1997 Proceedings. Salerno: IEEE: 145–171. CiteSeerX 10.1.1.589.4726. doi:10.1109/SEQUEN.1997.666911. ISBN 0-8186-8132-2. S2CID 124587565.*
9. *Hu, T. C.; Tucker, A. C. (1971). "Optimal Computer Search Trees and Variable-Length Alphabetical Codes". SIAM Journal on Applied Mathematics.* **21** *(4): 514. doi:10.1137/0121057. JSTOR 2099603.*
10. *Knuth, Donald E. (1998), "Algorithm G (Garsia–Wachs algorithm for optimum binary trees)", The Art of Computer Programming, Vol. 3: Sorting and Searching (2nd ed.), Addison–Wesley, pp. 451–453*. See also History and bibliography, pp. 453–454.

Bibliography

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 16.3, pp. 385–392.