IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010
ISSN (Online): 1694-0814
www.IJCSI.org

Ternary Tree and Clustering Based Huffman Coding Algorithm

[1]Pushpa R. Suri and [2]Madhu Goel

**[1]Department of Computer Science and Applications,
Kurukshetra University, Haryana, India**

**[2]Department of Computer Science and Engineering,
Kurukshetra Institute of Technology and Management, KUK**

Abstract

In this study, the focus was on the use of ternary tree over binary tree. Here, a new two pass Algorithm for encoding Huffman ternary tree codes was implemented. In this algorithm we tried to find out the codeword length of the symbol. Here I used the concept of Huffman encoding. Huffman encoding was a two pass problem. Here the first pass was to collect the letter frequencies. You need to use that information to create the Huffman tree. Note that char values range from -128 to 127, so you will need to cast them. I stored the data as unsigned chars to solve this problem, and then the range is 0 to 255. Open the output file and write the frequency table to it. Open the input file, read characters from it, gets the codes, and writes the encoding into the output file. Once a Huffman code has been generated, data may be encoded simply by replacing each symbol with its code. To reduce the memory size and fasten the process of finding the codeword length for a symbol in a Huffman tree, we proposed a memory efficient data structure to represent the codeword length of Huffman ternary tree. In this algorithm we tried to find out the length of the code of the symbols used in the tree.

Keywords: ***Ternary tree, Huffman's algorithm, Huffman encoding, prefix codes, code word length***

1. Introduction

Ternary tree [12] or 3-ary tree is a tree in which each node has either 0 or 3 children (labeled as LEFT child, MID child, RIGHT child). Here for constructing codes for ternary Huffman tree we use 00 for left child, 01 for mid child and 10 for right child.
Generation of Huffman codes for a set of symbols is based on the probability of occurrence of the source symbols. Typically, the construction of a ternary tree, describes the process this way:

- List all possible symbols with their probabilities;
- Find the three symbols with the smallest probabilities;
- Replace these by a single set containing all three symbols, whose probability is the sum of the individual probabilities;
- Repeat until the list contains only one member.

This procedure produces a recursively structured set of sets, each of which contains exactly three members. It, therefore, may be represented as a ternary tree ("Huffman Tree") with the symbols as the "leaves." Then to form the code ("Huffman Code") for any particular symbol: traverse the ternary tree from the root to that symbol, recording "00" for a left branch and "01" for a mid branch and "10" for a right branch. One issue, however, for this procedure is that the resultant Huffman tree is not unique.

One example of an application of such codes is text compression, such as GZIP. GZIP is a text compression utility, developed under the GNU (Gnu's Not Unix) project, a project with a goal of developing a "free" or freely available UNIX-like operation system, for replacing the "compress" text compression utility on a UNIX operation system.

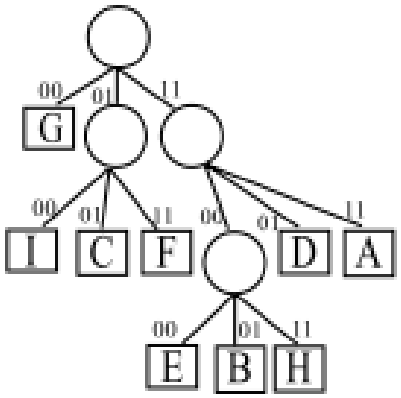
As is well-known, the resulting Huffman codes are prefix codes and the more frequently appearing symbols are assigned a smaller number of bits to form the variable length Huffman code. As a result, the average code length is ultimately reduced from taking advantage of the frequency of occurrence of the symbols

Huffman encoding [13] of numerical data, or more broadly variable bit length encoding of numerical data, is an important part of many data compression algorithms in the field of video processing. Huffman encoding is effective to compress numerical data by taking advantage of the fact that most data sets contain non-uniform probability distributions of data values. When using the Huffman method the data values are encoded using codes of different bit lengths. The more frequently occurring data values are assigned shorter codes and the less frequently occurring data values are assigned longer codes. Thus, the average code length for the data values in the data set is minimized. For typically skewed probability distributions

of data values the use of Huffman encoding may achieve a compression factor of between one and one-half and two times.

In Huffman Coding [6] the main work is to label the edges. Huffman Coding uses a specific method for choosing the representation for each symbol, resulting in a prefix - free code (some times called "Prefix Codes") i.e. the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol that expresses the most common characters using shorter strings of bits that are used for less common source symbols. The assignment entails labeling the edge from each parent to its left child with the digit 00, and the edge to the mid child with 01 and edge to the right child with 11. The code word for each source letter is the sequence of labels among the path from the root to the leaf node representing that letter. Only Huffman Coding is able to design efficient compression method of this type. Huffman Coding is such a widespread method for creating prefix-free codes that the term "Huffman Code" is widely used as synonym for "Prefix Free Code".

Now, for example, we will give a coding using variable length strings that is based on the Huffman Tree for weighted data item as follows: -



The Huffman Code for Ternary Tree assigns to each external node the sequence of bits from the root to the node. Thus the above Tree determines the code for the external nodes: -

G: 00	I: 0100	C: 0101
F: 0111	D: 1101	A: 1111
E: 110000	B: 110001	H: 110011

This code has "Prefix Property" i.e. the code of any item is not an initial sub string of the code of any other item. This means that there cannot be any ambiguity in decoding any message using a Huffman Code.

If we will represent the same data item with same weights in Binary Tree as well as in Ternary Tree[11] then we can easily point out the comparison between two representation as follows: -

In Ternary Tree: -

Memory used using Sequential Representation = 34

Memory used using Linked List Representation = 13

Number of Internal Nodes = 4

Path length = 199

Height of the tree = 4

Total Number of Nodes (Internal + External) = 13

Searching on Node is fast

Length of External Node (L_E)= $2L_I + 3n$

Here Labeling the left edge by 00, mid edge by 01 and right edge by 11 satisfies prefix Property

While In Binary Tree: -

Memory used using Sequential Representation = 51

Memory used using Linked List Representation = 17

Number of Internal Nodes = 8

Path length = 306

Height of the tree = 6

Total Number of Nodes (Internal + External) = 17

Searching on Node is slow

Length of External Node (L_E)= $= L_I + 2n$

Here Labeling the left edge by 0 and right edge by 1 satisfies prefix Property.

One may generate the length information for the Huffman codes by constructing the corresponding Huffman tree. However, as previously indicated, Huffman codes may not be unique when generated in this fashion. Nonetheless, it may be shown that by imposing two restrictions, the Huffman code produced by employing the Huffman tree may be assured of being unique. These restrictions are:

1. All codes of a given bit length have lexicographically consecutive values, in the same order as the symbols they represent; and

2. Shorter codes lexicographically precede longer codes.

2. Materials and Methods

Ordering and clustering based Huffman Coding groups the code words (tree nodes) within specified codeword lengths

Characteristics of the proposed coding scheme:

1. The search time for more frequent symbols (shorter codes) is substantially reduced compare to less frequent symbols, resulting in an overall faster response.
2. For long code words the search for the symbol is also speed up. This is achieved through a specific partitioning technique that groups the code bits in a codeword, and the search for a symbol is conducted by jumping over the groups of bits rather than going through the bit individually.
3. The growth of the Huffman tree is directed toward one side of the tree.

–Single side growing Huffman tree (SGH-tree)

Ex: H=(S, P)

S= {S1, S2,…, Sn}

P= {P1, P2,…, Pn}

Where p=No. of occurrence

S1	48	S1	48	S1	48
S2	31	S2	31	S2	31
S3	7	A1	8	A2	21
S4	6	S3	7		
S5	5	S4	6		
S6	2				
S7	1				

Table I

For a given source listing H, the table of codeword length uniquely groups the symbols into blocks, where each block is specified by its codeword length (CL).

TABLE II
TABLE OF CL-RECORDING

S_i	s_{i-1}	s_{i-2}	CL
S7	S6	S5	6
S4	S3	A1	4
A2	S2	S1	2

TABLE III
TABLE OF CODEWORD LENGTHS (TOCL)

CL	symbols
1	
2	S1,S2
3	
4	S3,S4
5	
6	S5,S6,S7

Each block of symbols, so defined, occupies one level in the associated Huffman tree.

1. Order the symbols according to their probabilities
   Alphabet set: S1, S2, …, SN
   Prob. of occurrence: P1, P2, , PN
   →The symbols are rearranged so that
   P1>=P2>=…>=PN

2. Apply a contraction process to the three symbols with the smallest probabilities
Replace symbols S_{N-2}, S_{N-1} and S_N by a “hypothetical” symbol, say H_{N-1} that has a `prob. of occurrence $P_{N-2}+P_{N-1}+P_N$
The new set of symbols has N-2 members:
   S1, S2, …, S_{N-3}, H_{N-2}

3. Set
   $S_{N-2}+S_{N-1}+S_N=A1$
Where A1 is inserted at proper location.

4. Repeat the steps 2 & 3 until we have last three symbols.
Table I shows the symbols used in Huffman tree.

5. Now construct the table of CL-Recording

6. First column of CL-Recording table is represented by Si where
Si= (entries of last row of every column) 0

Second row of CL-Recording is represented by S_{i-1} where
S_{i-1}= (entries of second last row of every column)

Third row of CL-Recording table is represented by S_{i-2} where
S_{i-2}= (entries of third lat row of every column)

Fourth row (last row) is used to calculate code word length.

7. To calculate the entries of codeword length.
CL (last row) =2
This designates the codeword length for entire row.
Now increment the last row
CL (last row) +2=4
Repeat steps to find the codeword length for entire symbols.

8. Therefore, table II (codeword length recording) indicate that each original symbol in the table has its CL (codeword length) specified.

9. Ordering the symbols according to their CL values gives table III (table of codeword length)

3. Result & discussion

If we will represent the same data item with same weights in Binary Tree as well as in Ternary Tree then we can easily point out the comparison between two representation as follows: -

In Ternary Tree: -

Number of Internal Nodes = 4

Path length = 199

Height of the tree = 4

Total Number of Nodes (Internal + External) = 13

Searching on Node is fast

Length of External Node (L_E)= $2L_I + 3n$

While in Binary Tree: -

Number of Internal Nodes = 8

Path length = 306

Height of the tree = 6

Total Number of Nodes (Internal + External) = 17

Searching on Node is slow

Length of External Node (L_E) =$L_I + 2n$

4. Conclusions

The main contribution of this study is exploiting the property implied in the Huffman tree to simplify the representation of the Huffman tree and the encoding procedure. Moreover our algorithm can also be parallelized easily. We already showed that representation of Huffman Tree using Ternary tree is more beneficial than representation of Huffman Tree using Binary tree in terms of path length, height, number of internal & external nodes and in error correcting & detecting codes

Acknowledgements

The author Madhu Goel would like to thank Kurukshetra University Kurukshetra for providing me University Research Scholarship & support of Kurukshetra Institute of Technology & Management (KITM) .

References

[1] BENTLEY, J. L., SLEATOR, D. D., TARJAN, R. E., AND WEI, V. K. A locally adaptive data compression scheme. Commun. ACM 29,4 (Apr. 1986), 320-330.

[2] DAVID A. HUFFMAN, Sept. 1991, profile Background story: Scientific American, pp. 54-58

[3] ELIAS, P. Interval and regency-rank source coding: Two online adaptive variable-length schemes. IEEE Trans. InJ Theory. To be published.

[4] FALLER, N. An adaptive system for data compression. In Record of the 7th Asilomar Conference on Circuits, Systems, and Computers. 1913, pp. 593-591.

[5] GALLAGER, R. G. Variations on a theme by Huffman. IEEE Trans. Inj Theory IT-24, 6 (Nov.1978), 668-674.

[6] Hashemain, “memory efficient and high-speed search Huffman Coding” IEEE Trans. Communication 43(1995) pp. 2576-2581.

[7] Hu, Y.C. and Chang, C.C., “A new losseless compression scheme based on Huffman coding scheme for image compression”,

[8] KNUTH, D. E, 1997. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd edition. Reading, MA: Addison-Wesley, pp. 402-406

[9] KNUTH, D. E. Dynamic Huffman coding. J. Algorithms 6 (1985), 163-180.

[10] MacKay, D.J.C., Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.

[11] MCMASTER, C. L. Documentation of the compact command. In UNIX User's Manual, 4.2 Berkeley Software Distribution, Virtual VAX- I Version, Univ. of California, Berkeley, Berkeley, Calif., Mar. 1984. ,

[12] PUSHPA R. SURI & MADHU GOEL, Ternary Tree & A Coding Technique, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.9, September 2008

[13] PUSHPA R. SURI & MADHU GOEL, Ternary Tree & FGK Huffman Coding Technique, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.1, January 2009

[14] PUSHPA R. SURI & MADHU GOEL, A NEW APPROACH TO HUFFMAN CODING, Journal of Computer Science. VOL.4 ISSUE 4 Feb. 2010 .

[15] ROLF KLEIN, DERICK WOOD, 1987, on the path length of Binary Trees, Albert-Lapwings University at Freeburg.

[16] ROLF KLEIN, DERICK WOOD, 1988, On the Maximum Path Length of AVL Trees, Proceedings of the 13th Colloquium on the Trees in Algebra and Programming, p. 16-27, March 21-24.

[17] SCHWARTZ, E. S. An Optimum Encoding with Minimum Longest Code and Total Number of Digits. If: Control 7, 1 (Mar. 1964), and 37-44.

[18] TATA MCGRAW HILL, 2002 theory and problems of data structures, Seymour lipshutz, tata McGraw hill edition, pp 249-255

[19] THOMAS H. CORMEN, 2001 Charles e. leiserson, Ronald l. rivest, and clifford stein.

[20] Thomas H.Cormen Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. Section 16.3, pp. 385–392.

.

Dr. Pushpa Suri is a reader in the department of computer science and applications at Kurukshetra University Haryana India. She has supervised a number of Ph.D. students. She has published a number of research papers in national and international journals and conference proceedings.

Mrs. Madhu Goel has Master's degree (University Topper) in Computer Science. At present, she is pursuing her Ph.D. and working as Lecturer in Kurukshetra Institute of Technology & Management (KITM), Kurukshetra University Kurukshetra. Her area of research is Algorithms and Data Structure where she is working on Ternary tree structures. . She has published a number of research papers in national and international journals.