TERNARY TREE & A CODING TECHNIQUE

Dr. Pushpa R.Suri † and Madhu Goel ††,

Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India

Summary

In this paper, the focus is on the use of Ternary Trees over Binary Tress. First of all, we give the memory representation for Ternary Trees. Huffman coding technique is developed using ternary trees, which benefits in computer implementation, efficient memory, compression, fast searching and error detecting & error correcting.

Keywords:

Memory Representation, path length, Huffman's Algorithm, coding, prefix codes, compression ratio, error detecting & correcting

1. Introduction:

Ternary Tree 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).

In Computer Science & Information Theory [7], Huffman coding is entropy Encoding algorithm used for loss less data compression. The term refers here to use of a variable length code table for encoding a source symbol (such as a character in a file). Where the variable length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the Source symbol. It was developed by David A Huffman [2] and published in the 1952 paper "A Method for the Construction of Minimum - Redundancy Code."

David A Huffman [1] discovered Huffman Concept. He uses Huffman Codes for binary tree. In this paper now we extend the Huffman algorithm for Ternary Tree, which benefits in computer implementation, compression, efficient memory, fast searching, error detecting & error correcting. Here, we discussed various different sections for describing it. First of all, computer implementation of Ternary Tree is presented. Then Huffman algorithm for Ternary Tree reducing path length [5] is described. After that coding technique using ternary tree is given, and at last we conclude the benefits of the application of ternary tree over binary tree.

Manuscript received October 5, 2008
Manuscript revised October 20, 2008

2. Computer Implementation of Ternary Tree:

2.1 Sequential List Representation of Ternary Tree.

2.2 Linked List Representation of Ternary Tree.

2.1 Sequential List Representation of Ternary Tree:

This representation uses only a singly linear array Tree as follows: -

2.1.1 The root R of Ternary tree is stored in Tree [1]

2.1.2 If a node N occupies TREE [K}, then its left child is stored in tree [3*k-1], Mid Child is stored in TREE [3*K] and rights child is stored in TREE [3*K+1]

EXAMPLE 2.1:

Suppose A, B, C, D, E, F, G, H and I are 9 data items and suppose they are assigned weights as follows:

Data Item	A	B	C	D	E	F	G	H	I
Weight	22	5	11	19	2	11	25	5	6

22	5	11	19	2	11	25	5	6
A	B	C	D	E	F	G	H	I

Table – 1

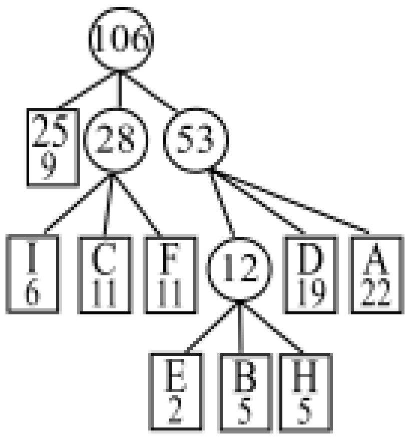


Fig. 1 The Sequential list representation for Example 2.1 can be represented as follows**: -**

	TREE
1	106
2	G-25
3	28
4	53
5	
6	
8	I-6
9	C-11
10	F-11
11	12
12	D-19
13	A-22
32	E-2
33	B-5
34	H-5

Table – 2

A Ternary Tree that is complete or nearly complete where efficient way of maintaining Ternary tree in memory called the sequential representation

2.2 Linked List Representation of Ternary Tree

Ternary Tree can be maintained in memory by means of Linked representation which uses four parallel arrays, INFO, LEFT, MID and RIGHT and pointer variable ROOT.

1. INFO (K) contains the data at Node N.

2. LEFT (K) contains the location of the left child

3. MID (K) contains the location of the mid child of node N.

4. RIGHT (k) contains the location of the right child of Node N.

The Linked list representation for Example 2.1 can be represented as follows**: -**

		INFO	LEFT	MID	RIGHT
	1.	25-G	0	0	0
	2.	28	7	8	9
	3.	53	10	11	12
ROOT	4.	E-2	0	0	0
5	5.	106	1	2	3
	6.	B-5	0	0	0
	7.	I-6	0	0	0
	8.	C-11	0	0	0
	9.	F-11	0	0	0
	10.	12	4	6	13
	11.	D-19	0	0	0
	12.	A-22	0	0	0
	13.	H-5	0	0	0

Table - 3

3 Coding Technique:

3.1 Static Huffman Coding Using Ternary Tree:

Static Huffman's Algorithm [3] uses binary tree, is extended to ternary tree as follows. Construct a full ternary Tree (A tree in which every node has either zero or three children's) whose leaves are labeled with the weights

HUFFMAN'S ALGORITHM: -.

Suppose W_1, W_2, and W_3 are three minimum weights among the n given weights $W_1, W_2, \dots W_n$. Find a tree T that gives a solution for (n-1) weights.

$$W_1 + W_2 + W_3, W_4, \dots W_n$$

Then in the tree T', replace the external node

$\boxed{W_1 + W_2 + W_3}$ by sub tree

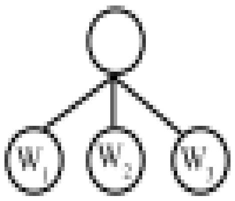


Fig. 2

The new Ternary Tree is the desired solution.

When the Huffman algorithm [8] is used to construct a code, the weights represent the probabilities associated with the source letters. At each step in the algorithm the tree corresponding to the three smallest weights W_i, W_j, & W_k are merged into a new tree whose weights is $W_i + W_j + W_k$ and whose root has three children that are the sub trees represented by W_i, W_j & W_k. The weights W_i, W_j and W_k are removed from the list and $W_i + W_j + W_k$ are inserted in the list. This process continues until the weight list contains the single value. If at any time, there is more than one way to choose a smallest pair of weights, any such pair may be chosen. In Huffman's proper the process begins with a non-increasing list of weights. Ternary Tree or 3-tree is Tree T in which each node has either 0 or 3 children. In any 3-Tree,

The number of External node is given by $N_E = 2N_I + 1$.

Accordingly, the running time of algorithm may depend on the length of the paths [4] in the Tree. The formula

$$\boxed{L_E = 2L_I + 3n}$$

is true for any ternary tree with n internal nodes. Where

L_E =Length of External node

L_I = Length of Internal node

n = Number of Internal Nodes.

The weighted path length p (External) of a ternary tree is defined to be the Sum of the weighted path length i.e.

$$P = W_1L_1 + W_2L_2 + \dots + W_nL_n$$

Where W_i and L_i denote respectively the weight and path length of an external node N_i.

By applying Huffman Algorithm, we have the Huffman tree for ternary tree is: -

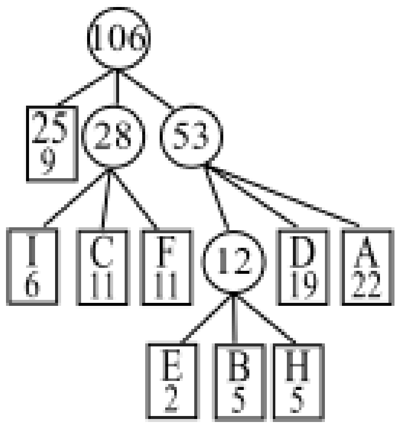


Fig. 3

3.2. Huffman Encoding:

Static Huffman's Algorithm [3] using binary tree is extended to ternary tree as follows: -

3.2.1. 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

3.2.2. Apply a contraction process to the three symbols with the smallest probabilities

Replace symbols SN-2,SN-1and SN by

a "hypothetical" symbol, say SN-1 that has a prob. of occurrence PN-2+PN-1+PN

The new set of symbols has N-1 members:

S1, S2, …, SN-2, SN-1

3.2.3. Repeat the step 2 until the final set has only one member.

3.3 Coding Technique For Ternary Tree:

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 No. 2.1, we will give a coding using variable length strings that is based on the Huffman Tree T for weighted data item as follows: -

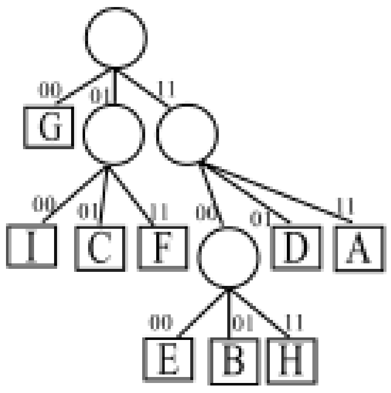


Fig. 4

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 T 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

Table - 4

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.

3.4 Compression Ratio (Fixed length code verses Huffman length code):

Average codeword length: -
Lave= l1p1+l2p2……………+lnpn
Lave= is a measure of the compression ratio.

In the above example,
9 symbols =4 bits (fixed length code representation)
Lave (Huffman) = 2.8982 bits
Compression ration = 4/2.8982 = 1.06

3.5 Error detecting & Error Correcting:

When this coding technique is applied in the message using ternary tree, then the number of transmitted bits is always even in number that is very beneficial in error detecting.

Error occurring during transmission is detected by following cases: -

Case 1: -Number of bits changed by addition or deletion of a bit.

Case 2: - Prefix property is violated

Case 3: - Sequence of bits does not exist as described in the labeling of edges in the coding technique.

If one of the cases occurs, accordingly can be corrected.

While In binary tree, the number of transmitted bits for a message can be either odd or even; therefore there is a difficulty in error detecting and in error correcting.

3.6 Properties Of Huffman Codes:

3.6.1. Fixed-length symbols ------→variable-length code words
: Error propagation

LAVE=AVERAGE CODE WORD LENGTH
LAVE= L1+P2L2+…………………..PNLN
H(s) <=lave < H(s)+2

3.6.2. The Huffman code-tree can be constructed both by
bottom-up method (in the above example)
top-down method

3.6.3. Huffman codes satisfy the prefix-condition: uniquely decodable: no codeword is a prefix of another codeword.

3.6.4. The complement of a Ternary Huffman code is also a valid Huffman code.

3.7 Benefits of Ternary Tree Over Binary Tree:

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: -

Memory used using Sequential Representation = 34

Memory used using Linked List Representation = 13

Number of Internal Nodes = 4

Path length = 199

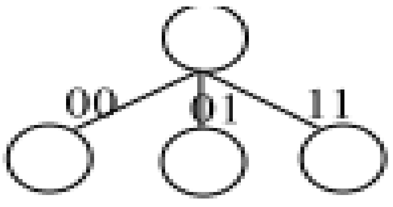
Height of the tree = 4

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

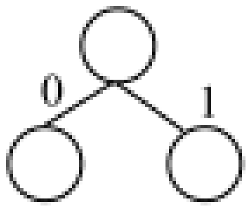
Height of the tree = 6

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.



4. Conclusion:

We can conclude that representation of Huffman Tree using Ternary Tree is more beneficial than representation of Huffman Tree using Binary Tree in terms of number of internal nodes, Path length, height of the tree, in memory representation, in fast searching and in error detection & error correction.

5. Acknowledgements:

The author Madhu Goel would like to thank Kurukshetra University Kurukshetra for providing me University Research Scholarship. .

6. References:

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

[2] HUFFMAN, D. A., 1952. "A Method for the Construction of Minimum-Redundancy Codes." *Proc. Inst. Radio Eng.* 40, pp 1098-1101

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

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

[5] 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.

[6] SCHWARZ, E. S., 1964. "An Optimum Encoding with Minimum Longest Code and Total Number of Digits." *Information and Control* 7, 37-44

[7] TATA MCGRAW HILL, 2002 theory and problems of data structures, seymour lipshutz, tata mcgraw hill edition, pp 249-255

[8] THOMAS H. CORMEN, 2001 charles e. leiserson, ronald l. rivest , and clifford stein. Introduction to algorithms, second edition. Mit press and mcgraw-hill. 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 PhD students. She has published a number of research papers in national and international journals and conference proceedings.

Mrs. Madhu Goel has Master's degrees(University Topper) in Computer Science. At present, She is pursuing her PhD As University Research Scholar in Computer Science. Her area of research is Algorithms and Data Structure where she is working on Ternary search tree structures.