A New Approach For Adaptive Huffman Coding

Pushpa Rani Suri[1] *Madhu Goel*[2]

Abstract

In this paper, the focus is on the use of ternary tree over binary tree in Huffman coding. First of all, we give the introduction of Huffman's coding. Then adaptive Huffman coding is discussed. Here, a one pass Algorithm developed by Vitter for constructing adaptive Huffman codes using binary tree is implemented to ternary tree. In this paper, it is shown that the use of Ternary tree results in minimizing numbers of nodes (internal) and path length, fast implementation, efficient memory, fast compression ratio and error detecting & error correcting.

Keywords : Ternary tree, Huffman's Algorithm, Adaptive Huffman coding, V algorithm, 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).

Data compression [15] or source coding is the process of encoding information using fewer bits or other information bearing units that an encoded representation would use through use of specific encoding schemes. As with any communication only works when both the sender and receiver understand that it is intended to be interpreted as characters representing the English language. Similarly compressed data can only be understood if the receiver knows the decoding method. Compression [1] is useful because it helps reduce the consumption of expensive resources such as hard disk or transmission bandwidth.

Huffman coding is an entropy-encoding algorithm used for loss less data compression. David A. Huffman [2] was given the concept of Huffman coding. Huffman coding uses a variable length [3] code table that has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix-free code that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique string of bits will produce a smaller average output size when the actual frequencies agree with those used to create the code.

Huffman coding is divided in to two categories:-

1. Static Huffman coding
2. Adaptive Huffman coding

Static Huffman coding suffers from the fact that the uncompressed need have some knowledge of the probabilities of the symbol in the compressed files. This can need more bits to encode the file. If this information

[1] Reader, Dept. of Computer Science & Applications, Kurukshetra University, Kurukshetra, Haryana, India.

[2] U.R.S, D.C.S.A, K.U.K, Haryana, India. Email : goelmadhu20@rediffmail.com, Mob.No : 9416182929

is unavailable compressing the file requires two passes. FIRST PASS finds the frequency of each symbol and constructs the Huffman tree. SECOND PASS is used to compress the file. We have already used the concept of static Huffman coding using ternary tree [10]. we conclude that representation of Static 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 [13], height of the tree, in memory representation, in fast searching and in error detection & error correction. Static Huffman coding methods have several disadvantages.

Therefore we go for adaptive Huffman coding. Adaptive Huffman coding calculates the frequencies dynamically based on recent actual frequencies in the source string. Adaptive Huffman coding is also called dynamic Huffman coding. It is based on building the code as the symbols are being transmitted that allows one-pass encoding and adaptation to changing conditions in data. The benefits of one-pass procedure is that the source can be encoded in real time, through it becomes more sensitive to transmission errors, since just a single loss ruins the whole code.

Implementations of adaptive Huffman coding: -

There are number of implementations of this method, the most notable are

1. FGK (Faller Gallager Knuth) Algorithm
2. Vitter Algorithm

We have already used the concept of FGK Huffman coding using ternary tree [11]. We conclude that representation of FGK 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 [12], height of the tree, in memory representation, in fast searching and in error detection & error correction.

Now here we try to use the concept of adaptive Huffman coding [4] using ternary tree with V Algorithm. All of these methods are defined- word schemes that determine the mapping from source messages to code- words on the basis of a running estimate of the source message probabilities. The code is adaptive, changing so as to remain optimal for the current estimates. In this way, the adaptive Huffman codes responds to locality, in essence, the encoder is learning the characteristics of the source. The decoder must learn along with the encoder by continually updating the Huffman tree so as to stay in synchronization with the encoder. Here we are given the concept of error detection and error correction. And the main point is that, this thing is only beneficial in TERNARY TREE neither in binary tree nor in other possible trees.

2. Why We Use Adaptive Huffman Coding

The key idea is to build a Huffman tree that is optimal for the part of the message already seen, and to recognize it when needed, to maintain its optimality. Adaptive Huffman [8] determines the mapping to code words using a running estimate of the source symbols probabilities.

1. It gives effective exploitation of locality. For example suppose a file starts out with the series of a character that are not repeated again in the file. In static Huffman coding that character will be low down on the tree because of its low overall count, thus taking lots of bits to encode. In adaptive Huffman coding, the character will be inserted at the highest leaf possible to be decoded, before eventually getting pushed down the tree by higher frequency characters.

2. Only one pass over the data.

3. Overhead, in static Huffman, we need to transmit someway the model used for compression that is the tree shape. This costs about 2n bits in a clever representation. As we will see, in adaptive schemes the overhead is nlogn.

3. Coding Technique

3.1 Algorithm Vitter using Ternary Tree

V algorithm in Adaptive Huffman coding [7] uses binary tree, is extended to ternary tree.

In this section we discuss the one-pass algorithm V using ternary tree. The two main disadvantages of static Huffman's algorithm are its two-pass nature and the overhead required to transmit the shape of the tree. In this paper we explore alternative one-pass methods, in which letters are encoded "on the fly". We do not use a static code based on a single ternary tree, since we are not allowed an initial pass to determine the letter frequencies necessary for computing an optimal tree. Instead the coding is based on a dynamically varying Huffman tree. That is, the tree used to process the t+1 st letter is a Huffman tree with respect to mt the sender encodes the t+1 st letter ai in the message by sequence 00, 01 and 11 that specifies the path from root to leaf. The receiver then recovers the original letter by the corresponding traversal of its copy of the tree. Both sender and receiver then modify their copies of the tree before the next letter is processed so that it becomes a Huffman tree μ (t+1).

$\mu t = a_{1t}, a_{2t} \ldots\ldots a_{it}$

The first t letters in the message

The adaptive Huffman algorithm of Vitter (Algorithm V) incorporates two improvements over algorithm FGK. It runs in real time and is optimum in our model of one-pass Huffman algorithms. There are two motivating factors in its design:-

1. The number of interchanges(↑s) should be bounded by some small number. The number of interchanges in which a node is moved upward is limited to one. This number is bounded in algorithm FGK only by L/2 where L is the length of the codeword for a t+1 when the recompilation begins.

2. The dynamic Huffman tree should be reconstructed to minimize not only $\Sigma wjlj$ but also Σlj and max {lj}, which intuitively has the effect of preventing a lengthy encoding of the next letter in the message.

Key Points Used in Vitter Algorithm

1. Swapping of nodes during encoding and decoding is onerous.

2. In FGK algorithm the number of swapping (considering a double cost for the updates that move a swapped node two levels higher) is bounded by [dt/2], where dt is the length of the added symbol in the old tree (this bound require some effort to be proved and is due to the work of Vitter)

3. In algorithm V, the number of swapping is bounded by 1.

These two objectives are reached through a new numbering scheme called implicit numbering.

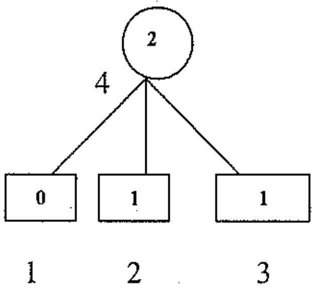
Implicit Numbering

- One of the key ideas of Algorithm v is the use of a numbering scheme for the nodes that is different from the one used by algorithm FGK. We use an implicit numbering in which the node numbering corresponds to the visual representation of the tree that is the nodes of the tree are numbered in increasing order by level; nodes on one level are numbered lower than the nodes on the next higher level.

- Nodes on the same level are numbered in increasing order from left to right.
- The node numbering used by algorithm FGK does not correspond to the implicit numbering

Invariant

- The key to minimize the other kind of interchanges is to maintain the following invariant
- For each weight w, all leaves of weight w precede (in the implicit numbering) all internal nodes of weight w.
- The interchanges, in the algorithm V are designed to restore implicit numbering, when a new symbol is read, and to preserve the invariant.

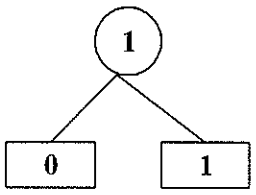


3.2 Algorithm V

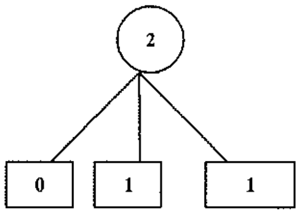
1. The nodes of the tree are numbered in increasing order by level; nodes on one level are numbered lower than the nodes on the next higher level.
2. Nodes on the same level are numbered in increasing order from left to right.
3. If this numbering is satisfied, certain types of updates cannot occur.
4. The key to minimize the other kind of interchanges is to maintain the following invariant for each weight w, all leaves of weight w proceed (in the implicit numbering) all internal nodes of weight w.
5. The interchanges, in the algorithms V, are designed to restore implicit numbering, when a new symbol is read, and to preserve the invariant.

Example 1

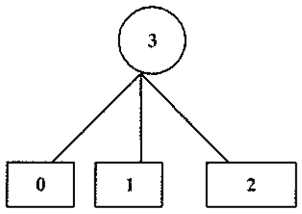
Construct the V tree for the message (e eae de eabe eae dcf) with ternary tree.



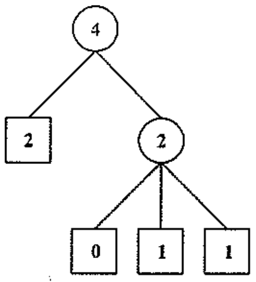
Step 1



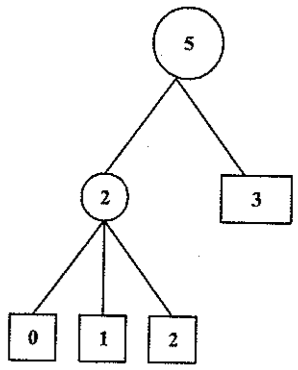
Step 2



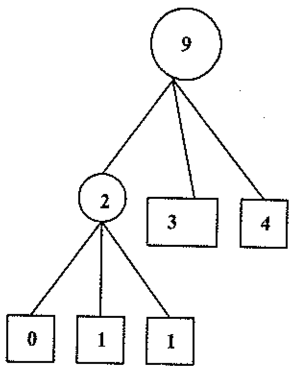
Step 3



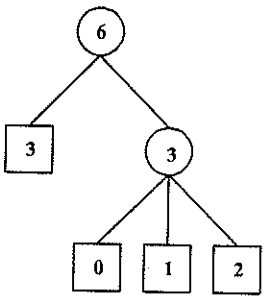
Step 4



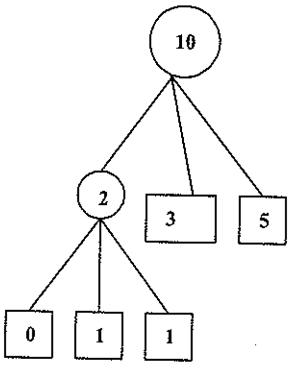
Step 5



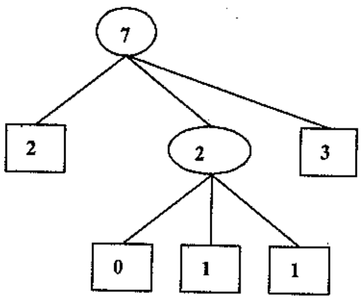
Step 9



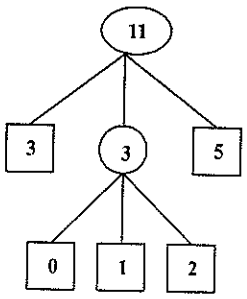
Step 6



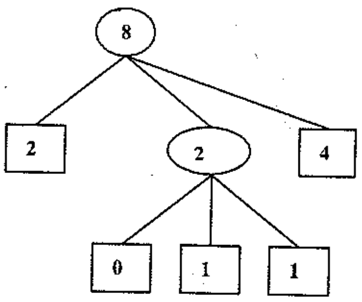
Step 10



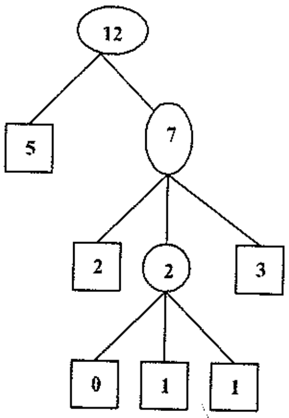
Step 7



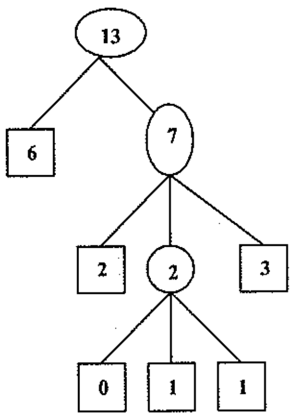
Step 11



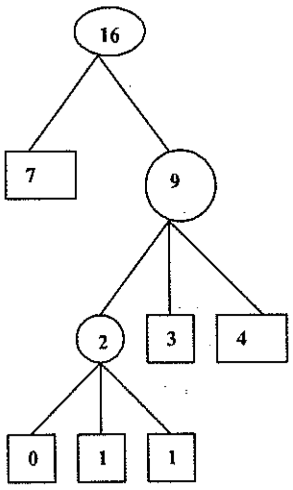
Step 8



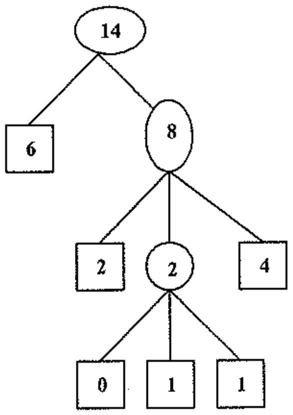
Step 12



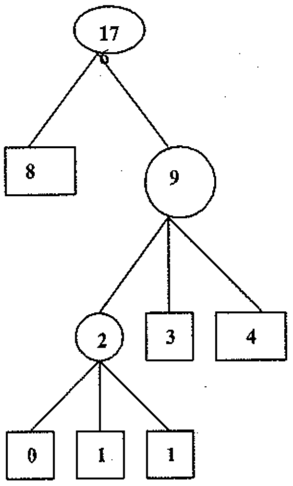
Step 13



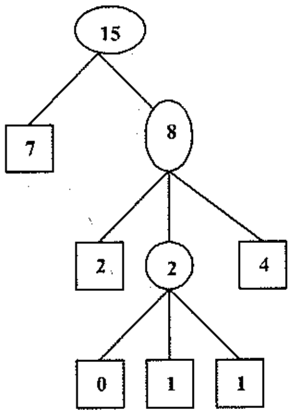
Step 16



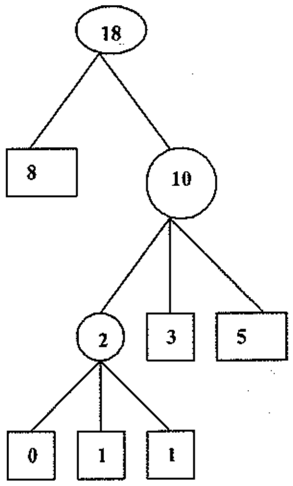
Step 14



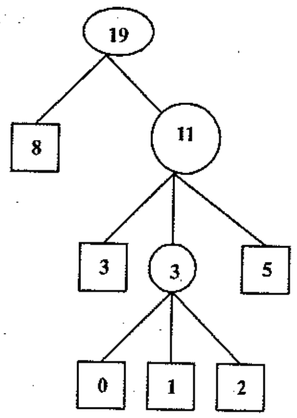
Step 17



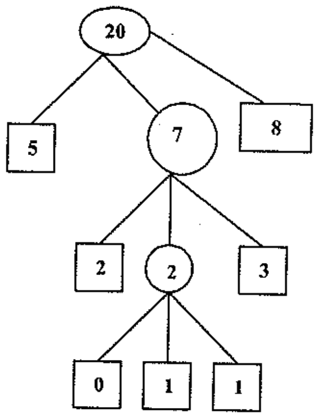
Step 15



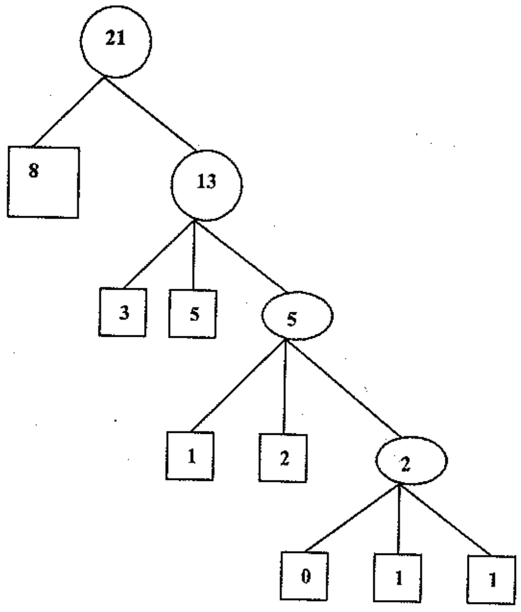
Step 18



Step 19



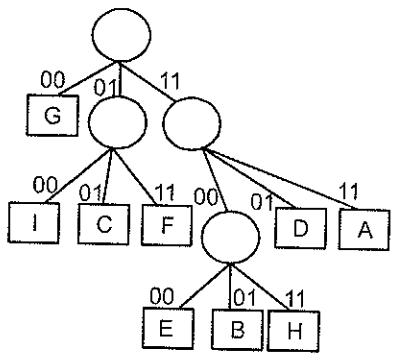
Step 20



Step 21

3.3 Coding Technique For Ternary Tree

In Huffman Coding [12] 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". We will give a coding using variable length strings that is based on the Huffman Tree T for weighted data item as follows: -



The Huffman Code [13] 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

Table : 1

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.

3.4 Compression Ratio (Fixed Length Code Cerses Huffman Length Code)

For example no. 1,

The number of fixed length code word bits= 4 bits (here in ternary tree, each symbol is represented by two bits, therefore for 7 symbols, number of fixed length code word bits are 4)

Average codeword length: -

Lave= l1p1+l2p2..............+lnpn

Lave= is a measure of the compression ratio.

Word	Probability		
e	8/21	=	.38095
a	3/21	=	.14285
sp	5/21	=	.23809
b	1/21	=	.04761
d	2/21	=	.09523
f	1/21	=	.04761
c	1/21	=	.04761

Lave=2 × .38095 + 4 × .147285 + 4 × .23809
+ 6 × .04761 + 6× .09523 + 8 × .04761
+ 8 × .04761

=. 7619+. 5891+. 9523+. 2856+. 5713+. 3808+. 3808
=3.9218

In the above example,

7 symbols =4 bits (fixed length code representation)

Lave (Huffman) = 3.9218 bits

Compression ration = 4/3.9218= 1.02

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.

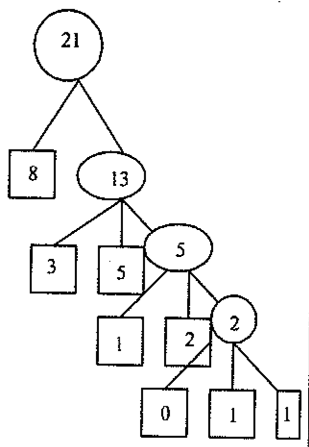
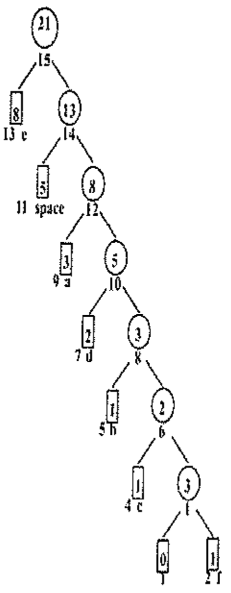
If one of the cases occurs, accordingly can be corrected.

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.

This thing is beneficial only in TERNARY TREE neither in binary tree nor in other possible trees.

3.6 Benefits Of Vitter Ternary Algorithm Over Vitter Binary Algorithm

Here, we are using message "e eae de eabe eae dcf" and then point out some comparison.

Ternary Adaptive Vitter	Binary Adaptive Vitter
The FGK tree for the message " e eae de eabe eae dcf " in ternary form is 	The FGK tree for the message " e eae de eabe eae dcf " in binary form is  21 15 8 13 e 13 14 5 11 space 8 12 3 9 a 5 10 2 7 d 3 8 1 5 b 2 6 1 4 c 3 0 1 2 f
Number of Nodes (Internal + External) in this → 12	Number of Nodes (Internal + External) in this → 15
Number of internal nodes in this → 4	Number of internal node nodes → 7
Path length → 45	Path length →57
Height of the tree → 5	Height of the tree →5

Memory space used using sequential representation less as compared to Ternary Representation	Memory space used using sequential representation more as compared to Binary Representation
Memory Space using linked list representation less as compared to Ternary Representation.	Memory Space using linked list representation more as compared to Binary Representation.
Here the number of transmitted bits is always even in number i.e. it is very beneficial in error detecting. This thing is beneficial only in TERNARY TREE neither in binary tree nor in other possible trees.	Here 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.
Searching fast	Searching slow.

4. Conclusion

We can conclude that representation of Huffman Tree using Adaptive Ternary FGK Algorithm is more beneficial than representation of Huffman Tree using Adaptive Binary FGK Algorithm 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.

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, 320-330, Apr. 1986.

[2] David A. Huffman, *"profile Background story: Scientific American"*, PP. 54-58, Sept. 1991.

[3] Elias. P, *"Interval and recency-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, PP. 593-591, 1913.

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

[6] Huffman.D.A, *"A method for the construction of minimum redundancy codes"*, In Proc. IRE 40, 1098-1101, 1951.

[7] Knuth. D. E, *"The Art of Computer Programming"*, Vol. 1: Fundamental Algorithms, 3rd edition, Reading, MA: Addison-Wesley, PP. 402-406, 1997.

[8] Knuth. D. E, *"Dynamic Huffman coding"*, J. Algorithms 6, 163-180, 1985.

[9] 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.

[10] Pushpa .R. Suri and Madhu Goel, *"Ternary Tree & A Coding Technique"*, IJCSNS International Journal of Computer Science and Network Security, Vol.8, No.9, October 2008.

[11] Pushpa R. Suri and Madhu Goel, *"Ternary Tree & FGK Huffman Coding Technique"*, IJCSNS International Journal of Computer Science and Network Security, Vol 9, No.1, January 2009.

[12] Rolf Klein, Derick Wood, *"on the path length of Binary Trees"*, Albert-Lapwings University at Freeburg, 1987.

[13] Rolf Klein, Derick Wood, *"On the Maximum Path Length of AVL Trees"*, Proceedings of the 13th Colloquium on the Trees in Algebra and Programming, PP. 16-27, March 21-24,1988.

[14] Schwartz. E. S, *"An Optimum Encoding with Minimum Longest Code and Total Number of Digits"*, If: Control 7, 1 , 37-44, Mar. 1964.

[15] Tata Mcgraw Hill, *"theory and problems of data structures"*, Seymour lipshutz, tata McGraw hill edition, PP. 249-255, 2002.

[16] Thomas H. Cormen, Charles e. leiserson, Ronald l. rivest, and Clifford stein, 2001.

Author's Biography

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.