Ternary Tree & FGK Huffman 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 tree over binary tree. First of all, we give the introduction of Huffman's coding. Then adaptive Huffman coding is discussed. Here, a one pass Algorithm developed by FGK (Fallar, Gallager, Knuth) for constructing adaptive Huffman codes for binary trees is implemented to ternary tree. In this paper, we are using the same set of symbols and try to draw Ternary tree which results in using minimum numbers of nodes (internal), minimizing path length, fast implementation, efficient memory, fast compression ratio, and in error detecting & error correcting.

Keywords
Ternary tree, Huffman's Algorithm, Adaptive Huffman coding, FGK 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).

In computer science & information theory, Huffman coding [2] is an entropy encoding used for loosless data compression. The term refers to the use of a 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. Huffman coding [14] uses a specific method for choosing the representation for each symbol, result in a prefix- free code (sometimes called “prefix codes”) that is the bit string representing some particular symbol is never a prefix of the bit representing any other symbol that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman coding suffers from the fact that the uncompressed need have some knowledge of the probabilities of the symbols in the compressed files.

Huffman coding basically divided in to two categories: -

1. Static Huffman coding
2. Adaptive/ dynamic 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 is unavailable compressing the file requires two passes. FIRST PASS find the frequency of each symbol and construct the Huffman tree. SECOND PASS is used to compress the file. [10] We already use the concept of static Huffman coding using ternary tree And 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 [11], 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 using ternary tree. Faller, Gallager, first conceived adaptive Huffman coding [3] independently. Knuth contributed improvements to the original algorithm, and the resulting algorithm is referred to as algorithm FGK. 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.

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

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 Adaptive Huffman Coding using Ternary Tree

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

In this section we discuss the one-pass algorithm FGK 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 μt 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 $\mu(t+1)$.

$\mu t = a_{1t}, a_{2t} \ldots\ldots a_{it}$
The first t letters in the message

KEY POINTS USED IN FGK ALGORITHM

1. Neither the tree nor its modification needs to be transmitted, because the sender and receiver use the same modification algorithm and thus always have equivalent copies of the tree.

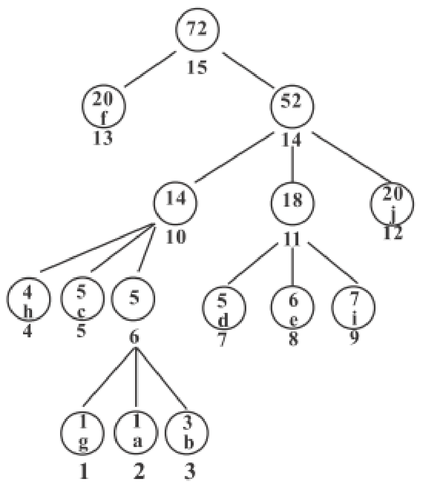
2. Dynamic Huffman codes [4] have one important property called sibling property. In sibling property, a ternary tree with p leaves of non negative weights is a Huffman tree if and only if The p leaves have non negative weights w1, w2,…………..wp and weight of each internal node is the sum of all its three children and The nodes can be numbered in non-decreasing order by weight, so that nodes 3j+1, 3j, 3j-1 are siblings and their common parent node is higher in the numbering. The node numbering corresponds to the order in which the nodes are combined by Huffman's algorithm: node 1,2 and 3 are combined first, node 4, 5 and 6 combined second and so on…

Algorithm FGK

The concept of FGK Algorithm binary tree is exerting to ternary tree. The Algorithm is as

1. The basis for algorithm FGK is the sibling property (Gallager 1978). A ternary code tree with non-negative weights has the sibling property if each node (except the root) has a sibling and sibling can be numbered in order of non decreasing weights the parent of a node is higher in the numbering.

2. A ternary prefix code is a Huffman code if and only if the code has the sibling property.

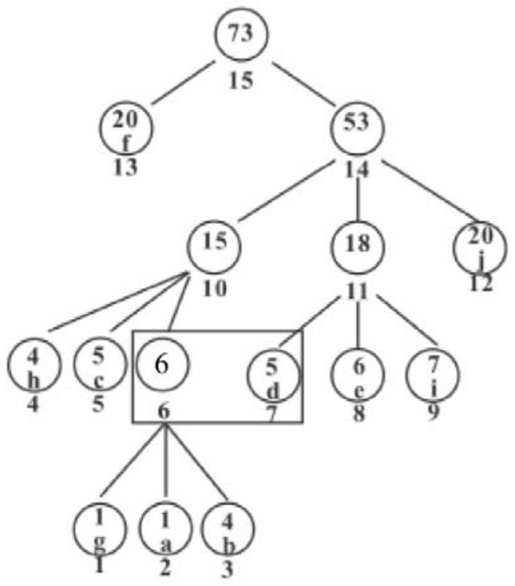


▪Note that node numbering corresponds to the order in which the nodes are combined by Huffman's algorithm, first nodes 1, 2 and 3 the nodes 4, 5, 6 …….

3. In Algorithm FGK, both encoder and decoder maintain dynamically changing Huffman code [6] trees. For each symbol the encoder sends the codeword for that symbol in current tree and then update the tree.

The problem is to change quickly the tree optimal after t symbols (not necessarily distinct) in to the tree optimal for t+1 symbols.

If we simply increment the weight of the t+1th symbols and of all its ancestors, the sibling property may not no longer be valid - we must rebuilt the tree



no more ordered by nondecreasing weight

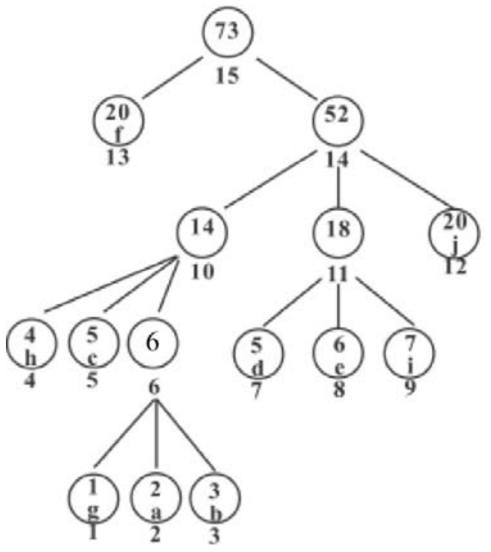
- Suppose next symbol is "b"
- if we update the weigths...
- ... sibling property is violated!!
- This is no more a Huffman tree

The solution can be described as a two-phase process.

First phase: Original tree is transformed in another valid Huffman tree for the first t symbols that has the property that simple increment process can be applied successfully.

Second phase: Increment process as described previously
-- The first phase starts at the leaf of the t+1th symbol
-- We swap this node and all its sub-tree, but not its numbering, with the highest numbered node of the same weight
-- New current node is the parent of this latter node.
■ The process is repeated until we reach the root

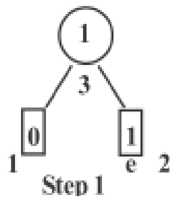
■ After the increasing process there is no node with previous weight that is higher numbered.

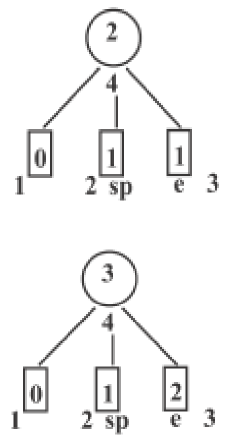


- **First phase**
- Node 2: nothing to be done
- Node 4: to be swapped with node 5
- Node 8: to be swapped with node 9
- Root reached: stop!

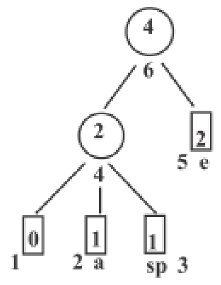
- **Second phase is applied**

EXAPMLE 1:
Construct the FGK tree for the message (e eae de eabe eae dcf) with ternary tree.

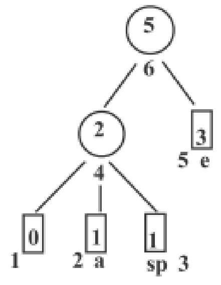




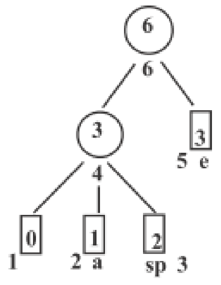
Step 3



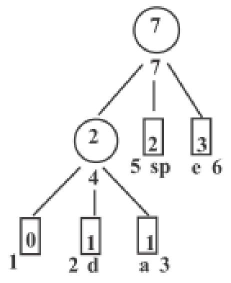
Step 4



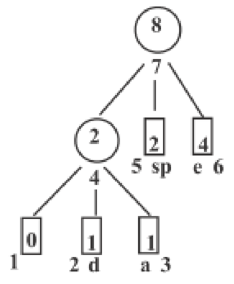
Step 5



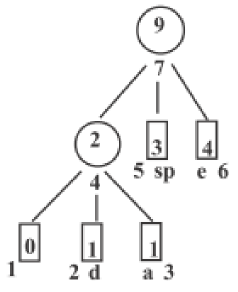
Step 6



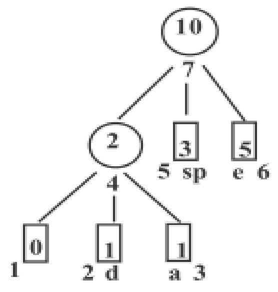
Step 7



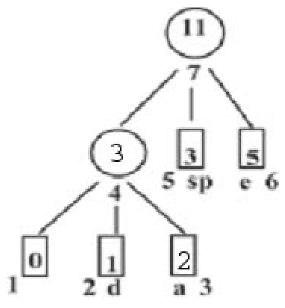
Step 8

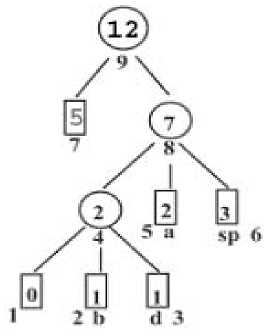


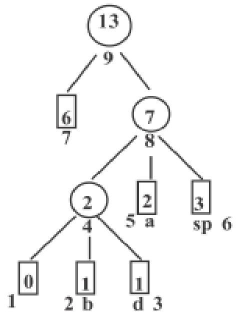
Step 9

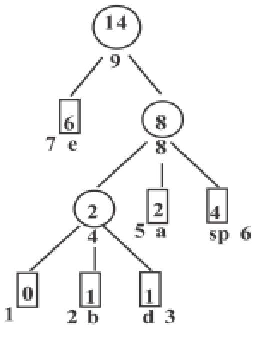


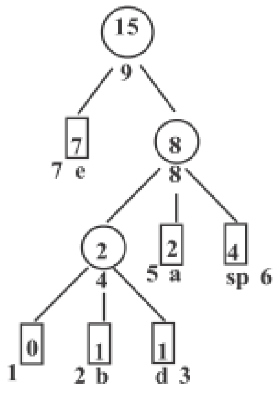
Step 10

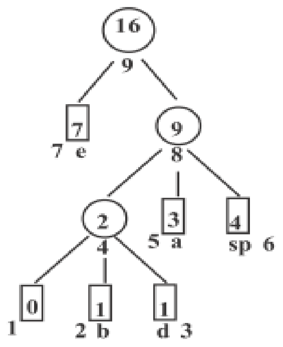

Step 11

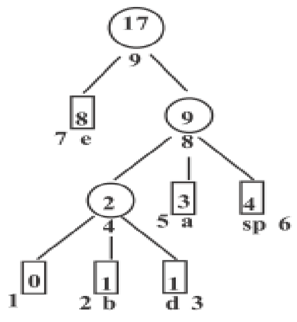

Step 12

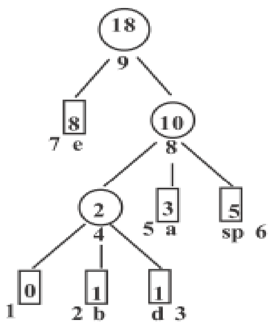

Step 13

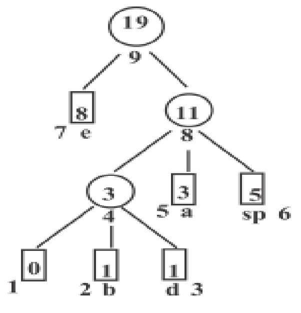

Step 14


Step 15

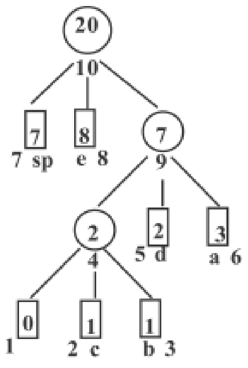

Step 16


Step 17

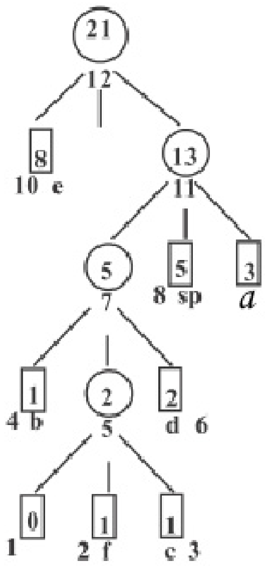

Step 18



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: -

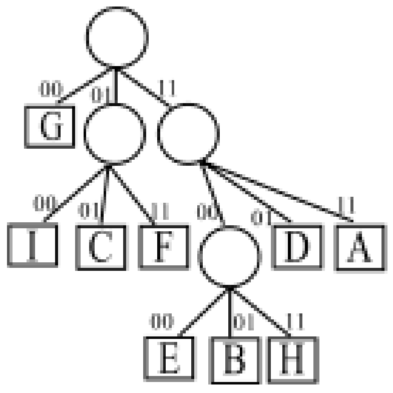


Fig1

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: -

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

Table - 1

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)

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.

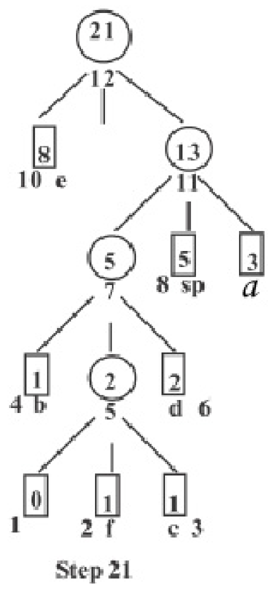
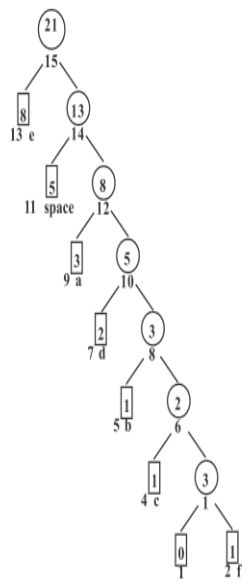
If one of the cases occurs, accordingly can be corrected.

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

3.6 Benefits Of FGK Ternary Algorithm Over FGK Binary Algorithm

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

	Ternary Adaptive FGK	Binary Adaptive FGK
1	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 

2	Numbers of nodes (Internal + External) in this → 12	Numbers of nodes (Internal + External) in this → 15
3	Number of internal node nodes → 4	Number of internal nodes in this → 6
4	Path length →45	Path length → 61
5	Height of the tree →5	Height of the tree → 8
6	Memory space used using sequential representation less as compared to Binary.	Memory space used using sequential representation more as compared to Ternary.
7	Memory Space using linked list representation less as compared to Binary.	Memory Space using linked list representation more as compared to Ternary.
8	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.

ACKNOWLEDGEMENTS

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

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 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. 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] HUFFMAN, D. A. A method for the construction of minimum redundancy codes. In Proc. IRE 40(1951), 1098-1101.

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

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

[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 & MADHU GOEL, Ternary Tree & A Coding Technique, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.9, September 2008 pp-

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

[12] 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.

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

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

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

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 degree (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.