iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** II **Month of publication:** February 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

Enhancement in File compression using Huffman approach

Suman (Student)
Delhi Institute of Technology and Management ,
Sonepat, Gannaur

Abstract: In this paper, we are showing how we can enhance the file compression by using Huffman approach. The main aim of the thesis is to utilize the concept of type casting and data normalization to show that it is good practice to this concept along with Huffman Coding. We are enhancing the concept of type casting and data normalization with Huffman Coding so as to increase the compression efficiency of Huffman Coding. Enhancement will be done in such a way that compressed data would be decompressed there would be no loss of data i.e. integrity of Huffman Coding would remain.

Keywords: Huffman approach, Compression, Decompression, Typecasting and Data normalization.

I.INTRODUCTION

In computer science and technology, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the 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 while he was a Ph.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".

1.2 Huffman Approach

The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols, n. A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the symbol itself, the weight (frequency of appearance) of the symbol and optionally, a link to a parent node which makes it easy to read the code (in reverse) starting from a leaf node. Internal nodes contain symbol weight, links to two child nodes and the optional link to a parent node. As a common convention, bit '0' represents following the left child and bit '1' represents following the right child. A finished tree has up to n leaf nodes and n − 1 internal nodes. A Huffman tree that omits unused symbols produces the most optimal code lengths.

The process essentially begins with the leaf nodes containing the probabilities of the symbol they represent, and then a new node whose children are the 2 nodes with smallest probability is created, such that the new node's probability is equal to the sum of the children's probability. With the previous 2 nodes merged into one node (thus not considering them anymore), and with the new node being now considered, the procedure is repeated until only one node remains, the Huffman tree.

Huffman`s procedure creates the optimal code for a set of symbols and probabilities' subject to the constraints that the symbols be coded one at a time . After the code has been created coding or Decoding is accomplished in a simple look up table manner. The code itself is an instantaneous uniquely decodable block code. It is called a block code because each source symbol is mapped into a fixed sequence of code symbols.

Procedure of Huffman Algorithm

ISSN: 2321-9653

The Huffman algorithm generates the most efficient binary code tree at given frequency distribution. Prerequisite is a table with all symbols and their frequency. Any symbol represents a leaf node within the tree.

Compression

The following general procedure has to be applied:

- search for the two nodes providing the lowest frequency, which are not yet assigned to a parent node
- couple these nodes together to a new interior node
- add both frequencies and assign this value to the new interior node

The procedure has to be repeated until all nodes are combined together in a root node.

Decompression

For decoding the Huffman tree is passed through with the encoded data step by step. Whenever a node not having a successor is reached, the assigned symbol will be written to the decoded data.

1.3 Lossless vs. Lossy Compression

Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely, but nevertheless perfectly. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small.

Another kind of compression, called lossy data compression, is possible if some loss of fidelity is acceptable. For example, a person viewing a picture or television video scene might not notice if some of its finest details are removed or not represented perfectly (i.e. may not even notice compression artifacts). Similarly, two clips of audio may be perceived as the same to a listener even though one is missing details found in the other. Lossy data compression algorithms introduce relatively minor differences and represent the picture, video, or audio using fewer bits.

1.4 Type Casting

Typecasting and coercion refer to different ways of, implicitly or explicitly, changing an entity of one data type into another. This is done to take advantage of certain features of type hierarchies or type representations. One example would be small integers, which can be stored in a compact format and converted to a larger representation when used in arithmetic computations. In object-oriented programming, type conversion allows programs to treat objects of one type as one of their ancestor types to simplify interacting with them.

Benefits of Using Type Casting

Typecast always returns the same number of bytes in the output Y as was in the input X. For example, casting the 16-bit integer 1000 to uint8 with typecast returns the full 16 bits in two 8-bit segments (3 and 232) thus keeping its original value (3*256 + 232 = 1000). The cast function, on the other hand, truncates the input value to 255.

1.5 Database Normalization

Data normalization is a process in which data attributes within a data model are organized to increase the cohesion of entity types. In other words, the goal of data normalization is to reduce and even eliminate data redundancy, an important consideration for application developers because it is incredibly difficult to stores objects in a relational database that maintains the same information in several places. Table 5.1 summarizes the three most common forms of normalization (First normal form (1NF), Second normal form (2NF), and Third normal form (3NF)) describing how to put entity types into a series of increasing levels of normalization. Higher levels of data normalization are beyond the scope of this article. With respect to terminology, a data schema is considered to be at the level of normalization of its least normalized entity type. For example, if all of your entity types are at second normal form (2NF) or higher then we say that your data schema is at 2NF.

Table 1.1 Data Normalization Rules.

Level	Rule
First normal form (1NF)	An entity type is in 1NF when it contains no repeating groups of data.

Second normal form (2NF)	An entity type is in 2NF when it is in 1NF and when all of its non-key attributes are fully dependent on its primary key.
Third normal form (3NF)	An entity type is in 3NF when it is in 2NF and when all of its attributes are directly dependent on the primary key.

II. Proposed work

In Matlab we have implemented Huffman coding in a two step procedure the first step is converting normal file to Huffman file and getting the compressed code outside. In next step or second step we convert that compressed code to back to original text code.

2.1 Proposed Compression Method

The proposed technique can be clearly understood by the block diagram below:

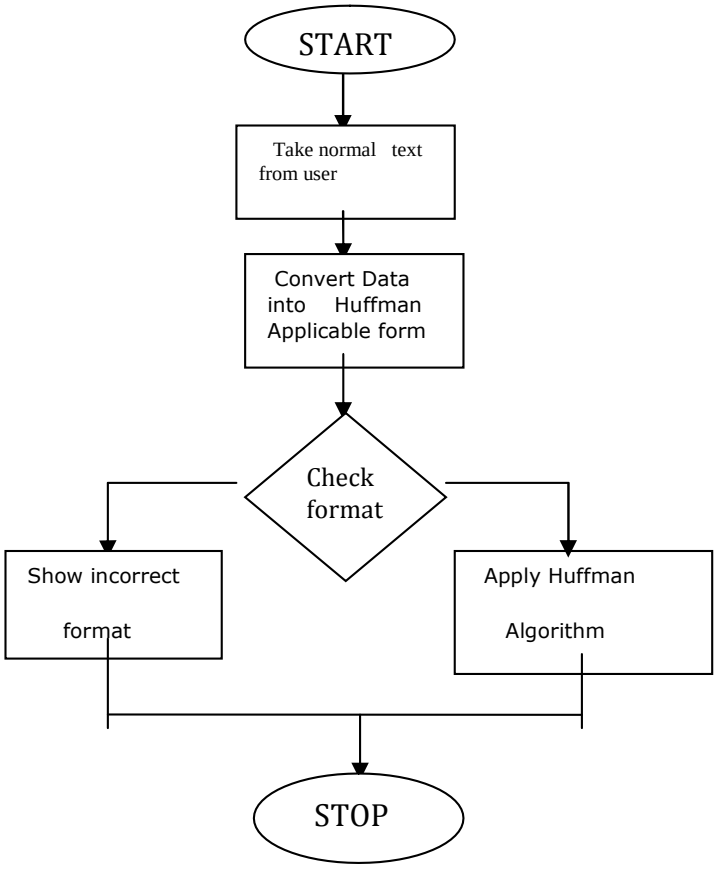


Figure 2.1 Proposed Compression Diagram

In the above algorithm we can see that first our Matlab code is taking data from the user. After that it is converting that data to UINT8 format which can be compressed using Matlab if by any chance data is not in UINT8 format then it cannot be compressed. After it is converted and we apply Huffman compression and get a compressed code.

Compression

The following general procedure has to be applied:

- search for the two nodes providing the lowest frequency, which are not yet assigned to a parent node
- couple these nodes together to a new interior node
- add both frequencies and assign this value to the new interior node

The procedure has to be repeated until all nodes are combined together in a root node.

Example: "abracadabra"

Symbol	Frequency
a	5
b	2
r	2
c	1
d	1

According to the outlined coding scheme the symbols "d" and "c" will be coupled together in a first step. The new interior node will get the frequency 2.

STEP 1:

Symbol	Frequency	Symbol	Frequency
a	5	a	5
b	2	b	2

r	2	r	2
c	1 ---------->	1	2
d	1		

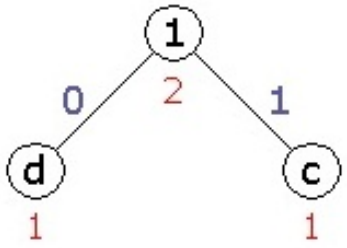


Figure 2.3 Code tree after the 1st step

STEP 2:

Symbol	Frequency	Symbol	Frequency
a	5	a	5
b	2	b	2
r	2 ---------->	2	4
1	2		

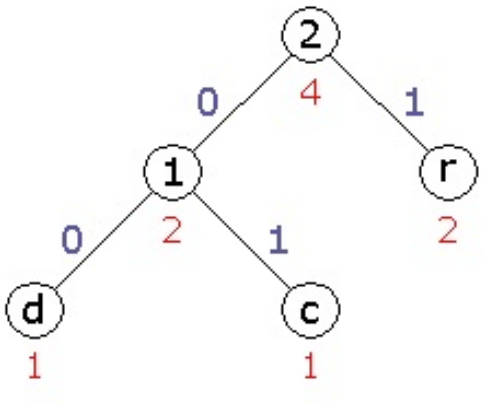


Figure 2.4 Code tree after the 2nd step:

STEP 3:

Symbol	Frequency	Symbol	Frequency
a	5	a	5
2	4 ---------->	3	6
b	2		

Code tree after the 3rd step:

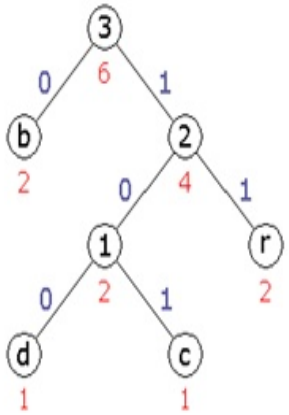


Fig. 2.5 Code tree after the 3rd step.

Step 4:

Symbol	Frequency	Symbol	Frequency
3	6 ---------->	4	11
a	5		

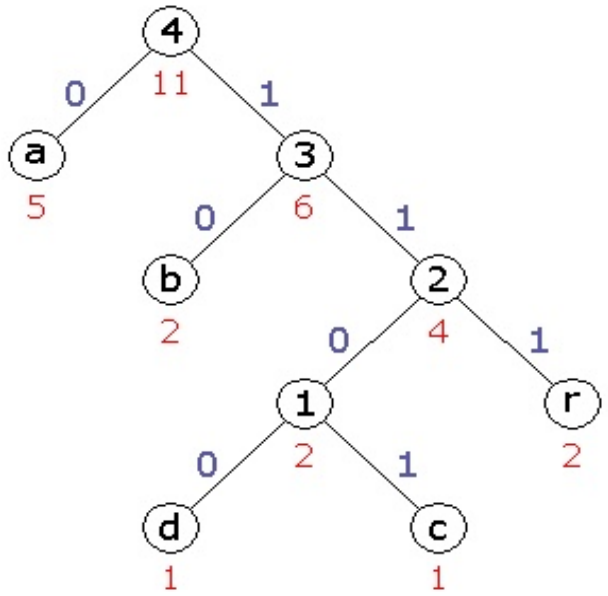


Fig.2.6 Code tree after the 4th step:

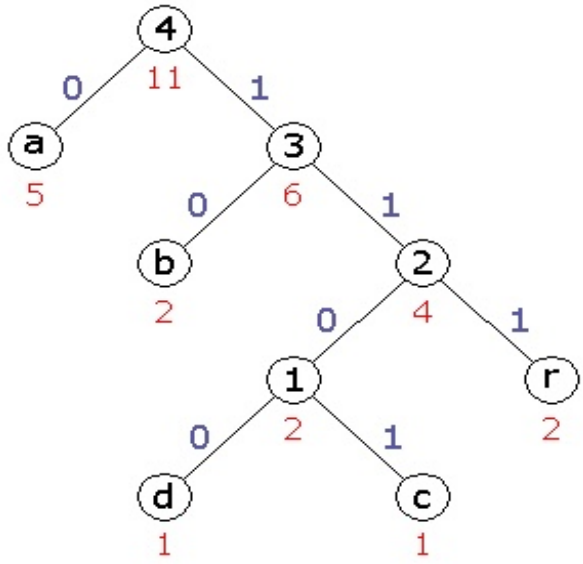


Figure 2.7 Complete huffman tree

Code Table

If only one single node is remaining within the table, it forms the root of the Huffman tree. The paths from the root node to the leaf nodes define the code word used for the corresponding symbol:

Symbol	Frequency	Code Word
a	5	0
b	2	10
r	2	111
c	1	1101
d	1	1100

Complete Huffman Tree:

Encoding:

The original data will be encoded with this code table as follows:

Symbol	Frequency	Code Word
a	5	0
b	2	10
r	2	111
c	1	1101
d	1	1100

a b r a c a d a b r a
0 10 111 0 1101 0 1100 0 10 111 0

Encoded data: 23 Bit

Original data: 33 Bit

2.2 Proposed Decompression Method

The proposed technique can be clearly understood by the block diagram below:

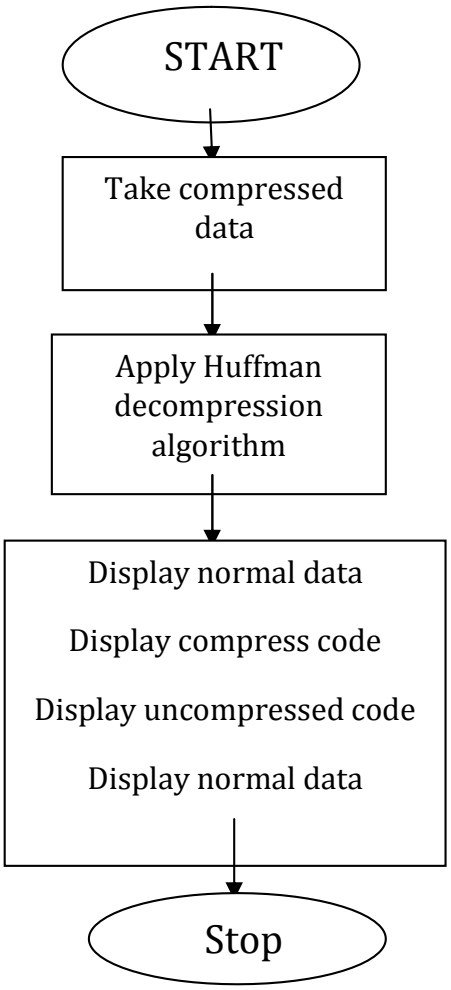


Figure2.2 Proposed Decompression Technique

Decompression

For decoding the Huffman tree is passed through with the encoded data step by step. Whenever a node not having a successor is reached, the assigned symbol will be written to the decoded data.

0101110110101100010111 0

Encoded	decoded
0	a
10	b
111	r
0	a
1101	c
0	a
1100	d
0	a
10	b
111	r
0	a

ACKNOWLEDGMENT

First of all I am thankful to almighty GOD to help me out in every odds and buts. There are numerous people without whom this dissertation might never have been completed. Among those who deserve credit and our heartfelt gratitude is the most important Mr. Rakesh Chawla for his valuable guidance and support. Without his this work would not have been possible. I am thankful for their constant encouragement and guidance. Last but not the least I would like to thank the faculty members of the Department of Computer Science Engineering for being very supportive to me throughout these years. I am also thankful to my friends and classmates for their valuable suggestions.

REFERENCES

1.Ternary Tree & A new Huffman Technique, IJCSNS International Journal of Computer Science and Network Security, Vol.10 N0.3, March 2011.

2. A Study and implementation of the Huffman Algorithm based on Condensed Huffman Table, 2008 International Conference on Computer Science and Software Engineering.

3. A Method for the Construction of Minimum-Redundancy Codes David A. Huffman, Associate, IRE.(1952).

4. Typecasting, Legitimation: A Formal Theory Greta Hsu Univ. Of California at Davis Michael t. Hannan Stanford University.

5. Database Normalisation: Korth Refrences:A CS2 assignment.

6.Database System Concepts By SILBERSCHATZ, KORTH, SUDARSHAN McGraw-Hill Higher Education, ISBN NO. 0-07-120413-X

7. A Method for the Construction of Minimum-Redundancy Codes DAVID A. HUFFMAN, ASSOCIATE,IRE.

8. A study and implementation of the Huffman Algorithm based on condensed Huffman table, 2010 International Conference on Computer Science and Software Engineering.

9. Owen L.Astrachan,2004,Huffman coding:A CS2 assignment.

10. Typecasting, Legitimation , and Form Emergence: A Formal Theory Greta Hsu Univ. of California at Davis Michael T. Hannan Stanford University László Pólos Durham University Running head: Typecasting, Legitimation, and Form Emergence March 24, 2010

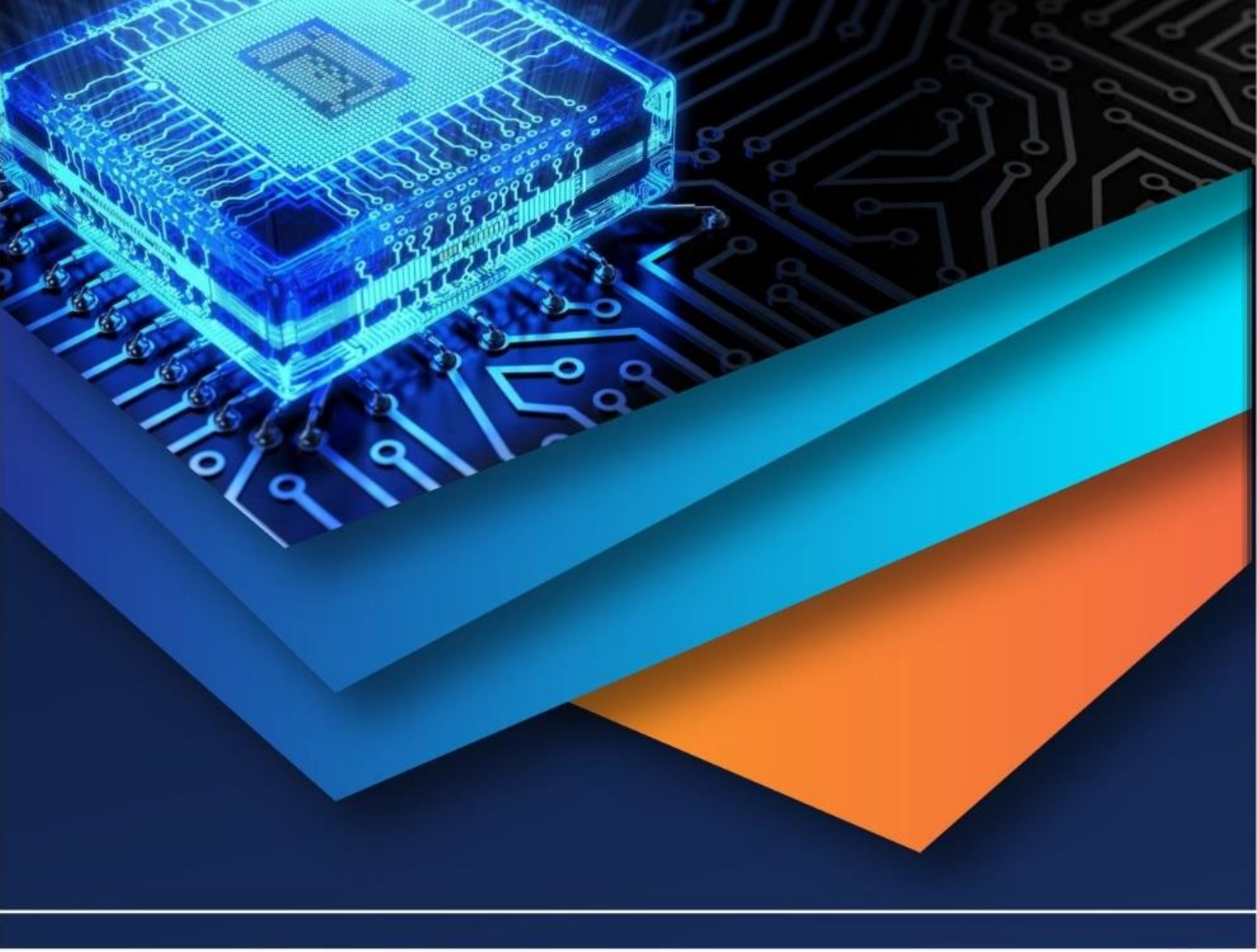
11. Introduction to Data Compression, Khalid Sayood.

12. D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102. Huffman's original article

13.http://en.wikipedia.org/wiki/Losslessdatacompression- Wiki

14. http://www.Huffman coding - Wikipedia, the free encyclopedia.htm

15.http://en.wikipedia.org/wiki/Database_normalization-Wikipedia



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)