ISSN No: - 2456 - 2165

Recursive Hexadecimal Prime Multiples Data Compression Algorithm

Gaurav J. Dhingani
Electronics & Communication Engineer
Government Engineering College
Rajkot, Gujarat, India

Abstract--The hexadecimal prime factorization compression methodology is presented. The method is suitable for random data format. The prime factorization algorithm contains a decimal notation technique to hexadecimal prime factors. That inverts the data format change by compression process and so, another compression layer is possible. Ergo, the algorithm is recursive and richer value of compression factor can be achieved.

Index Terms-: *4-bit data chunk method, decimal notation, hexadecimal notation, hexadecimal prime factorization, hexadecimal prime domain coding, recursive system, serial method.*

I. INTRODUCTION

Increased electronic logic density over computational circuitry has a physical limitation, but an optimized data compression algorithm can improve the transmission, storage and in some cases processing power of electronic devices.

Proposed algorithm is lossless and recursive. It uses hexadecimal prime factorization to encode data in prime domain and therefore achieves compression in each layer of encoding. Algorithm performs on bits (raw, physical and primary interpretation of data). In every random file format the fundamental data is binary i.e. 0's and 1's. It need not operate on patterns or highly repetitive data chunks; the compression technique takes only bits into account. Thus, operation on random file format is possible and no specified file structure is required.

The algorithm uses hexadecimal prime multiples to represent an ordinary but rather a big hexadecimal number obtained from any given binary sequence. It then represents prime factors of a number in a prime domain encoding and achieves a rather impressive compression ratio.

II. COMPRESSION ALGORITHM

Step-1: Bring in the binary sequence of nearly 200 bits or more (variable).

Step-2: Conversion to hexadecimal by every 4 bit data chunk, that will produce a number of digit scale of 50.

Step-3: Calculate hexadecimal prime factors of the number (factors are of hexadecimal system).

Step-4: Find the serial number associated with resultant hexadecimal prime factors (serial notations are in decimal system, not in hexadecimal).

Step-5: Now a compressor has a decimal serial number of hexadecimal prime factors. Arrangement:

1. (1^{st} hexadecimal prime number)^3, (5^{th} hexadecimal prime number)^4, (11^{th} hexadecimal prime number), (129^{th} hexadecimal prime number)
2. 1 ^ 3 , 5 ^ 4 , 11 , 129
3. 1 B 3 A 5 B 4 A 11 A 129

(Here, B=power of a previous number, A=separator)

Step-6: The code [1 B 3 A 5 B 4 A 11 A 129] is a decimal number with character separation but it can be interpreted as a hexadecimal notation. Convert the hexadecimal number in binary by 4 bit data chunk method (invers of step-2).

Step-7: A 2 byte flag is introduced at the end of compressed data segment. [Flag = 1111 1111 0000 0000]
Compressed file is produced.

III. DECOMPRESSION ALGORITHM

Step-1: Detect the compressed data segment via flagging pattern.

Step-2: Convert binary to hexadecimal via 4 bit data chunk method.

ISSN No: - 2456 - 2165

Step-3: Extract the prime factors to linear form.
E.g.

1. 1 B 3 A 5 B 4 A 11 A 129
2. 1 ^ 3 , 5 ^ 4 , 11 , 129
3. (1st hexadecimal prime number)^3, (5th hexadecimal prime number)^4, (11th hexadecimal prime number), (129th hexadecimal prime number)

Step-4: Convert this decimal serial notation of hexadecimal series to hexadecimal prime factors by respective notations.

Step-5: Multiply the hexadecimal prime factor to produce a hexadecimal number.

Step-6: Convert hexadecimal number to binary by 4 bit data chunk method.

The output file is recovered in its original form and in original size.

IV. CONCLUSION

The presented work contributes to data compression algorithms and data storage improvement. The conversions between numeric systems like hexadecimal and decimal are the essence for the recursive property of the compression platform. Theoretically, there is no limit in possible compression layers. The work presented here is in theory, but achieving it practically would result in a huge impact on the world of data compression, turning all the existing methods obsolete. If we practically succeed to achieve the content that is on paper, it will for sure cause a mini revolution.

REFERENCES

[1] A Novel Data Compression, Gordon Chalmers, Gordon@quartz.shango.com, physics/0510148v1, [physics.gen-ph] 16 Oct 2005.
[2] Efficient Data Compression Using Prime Numbers. by, Debasish Chakraborty, Snehasish Kar, Kalyan Guchait. ERCICA 2013, ISBN: 9789351071020.
[3] Data Compression With Prime Numbers, Gordon Chalmers, physics/0511145v1 [physics.gen-ph] 16 Nov 2005.
[4] Huffman coading, https://www.youtube.com/results?search_query=huffman+coding+algorithm&spfreload=1
[5] Lempel-Ziv-Welch compression algorithm, https://www.youtube.com/watch?v=j2HSd3HCpDs