***Huffman Encoding and construction of the Huffman tree***

***Name: - Mohamed Saleh Mahmoud***

***ID :- 201501343***

***Supervised by : - Dr.Mahmoud Abdelaziz***

***Table of Contents***

1. Introduction 3
2. Theory 3
3. Algorithm of codes 4
4. Results 5
5. Conclusion 7

***Introduction :-***

Huffman coding algorithm is basically based on sorting the probabilities in an ascending order and

then summing the worst two probabilities repetitively until we reach to a total of one and then by the

end we make like a back propagation adding either a 1 or 0 to code the symbol achieving an

optimization of making the least probable most lengthy and vice versa

***Theory :-***

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. Huffman coding uses a specific method for choosing the

representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes")

(that is, 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 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 strings of

bits will produce a smaller average output size when the actual symbol frequencies agree with those

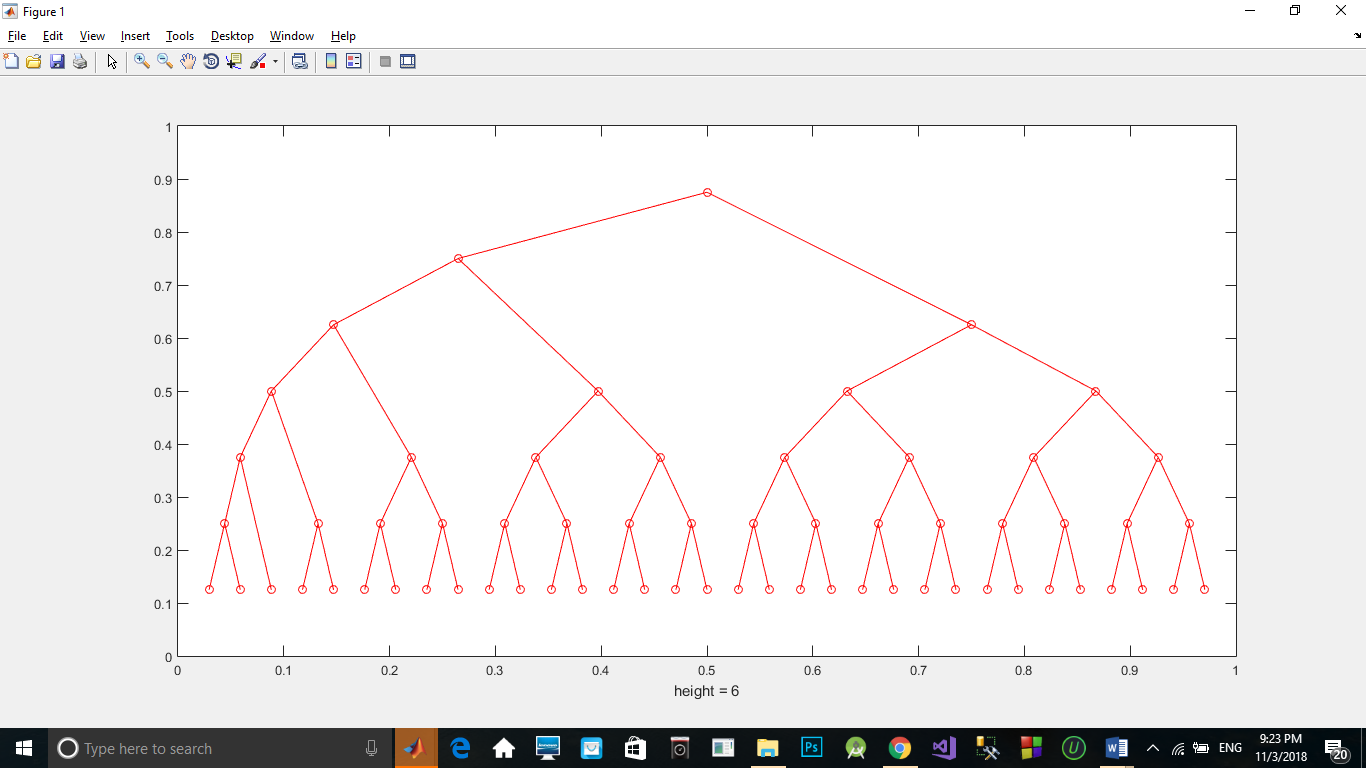
used to create the code.

**Algorithm of Codes :-**

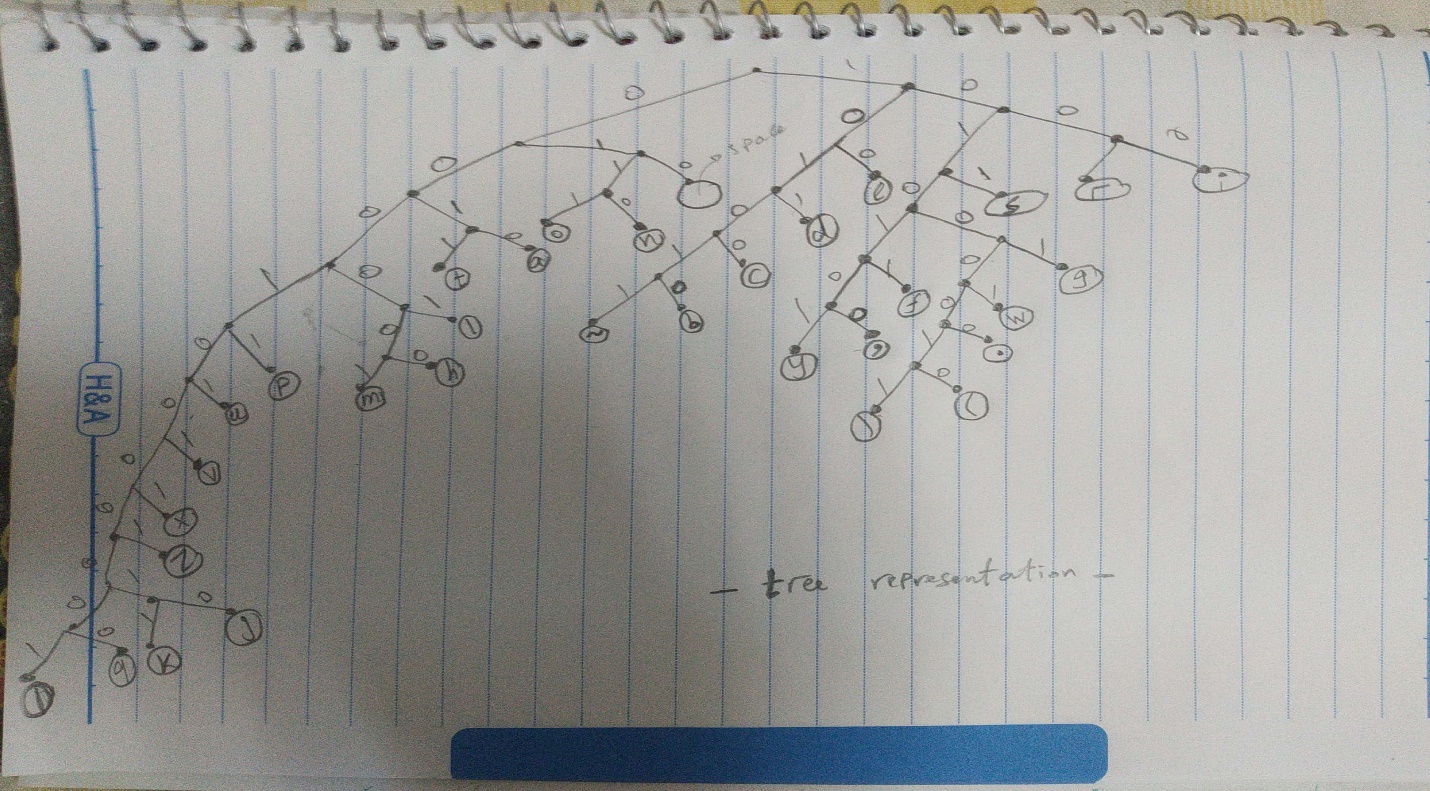
|  |  |
| --- | --- |
| **Function / Class / Script** | **Functionality** |
| CQueue | It’s a class to implement the Queue data  structure as I used it in calculating the parent  nodes of the tree. Further details can be obtained in the comments section of the code file |
| Main | The main script to run the algorithm. It reads at first the data from the file and then compute probabilities and then encode these symbols. |
| Calcparents | It calculates the parent nodes. It sorts the probabilities and then add to the parent vector two values of the variable counter so that it constructs by the end a binary tree representing tree that it needs to be then reshaped to produce the tree of probabilities which is included here in the report |
| Addnode | It adds a node of 0 or one to the existing code word in the function of encoder |
| HuffmanEncoder | It implements the Huffman algorithm returning two arrays one of the codes and the other for the corresponding symbols. It first sorts the probabilities and then finds codes. |

***Results :-***

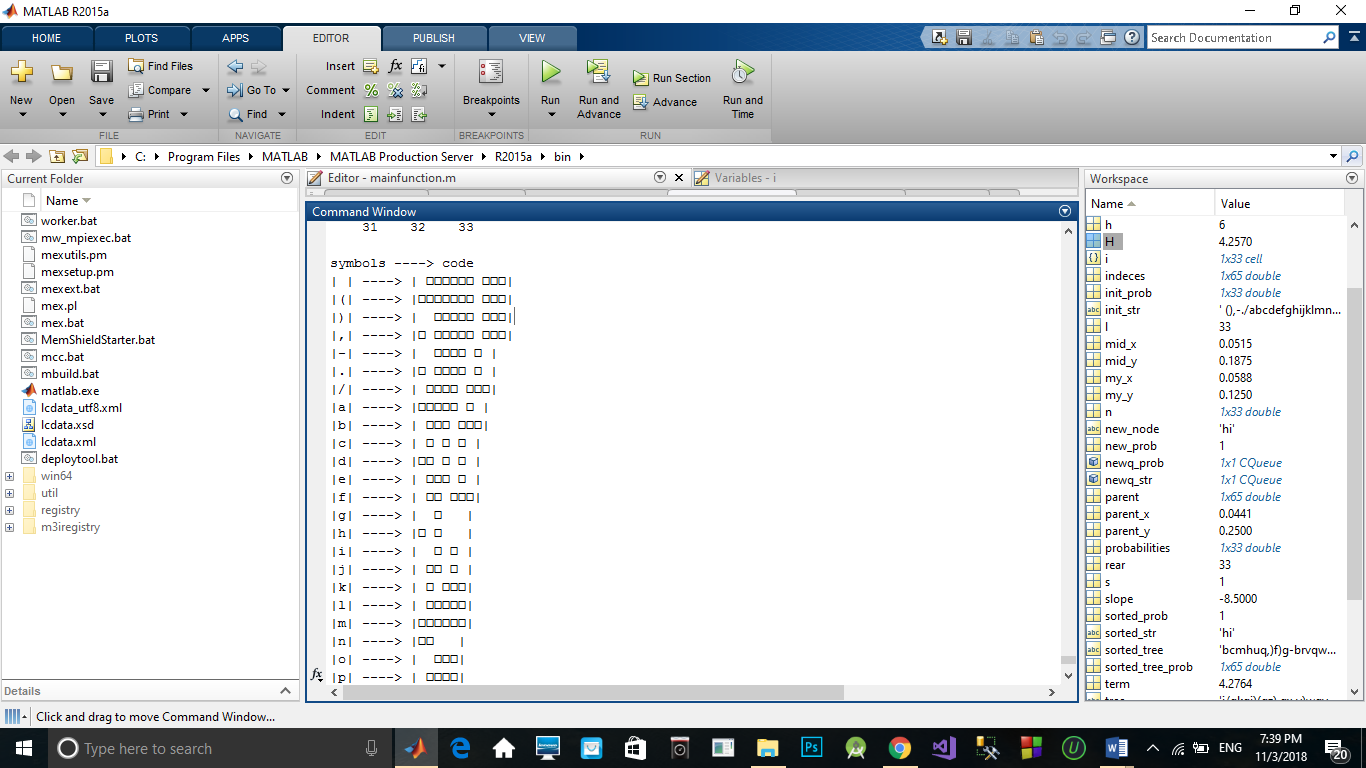
The Tree Produced by the codes which needs to be reshaped :-

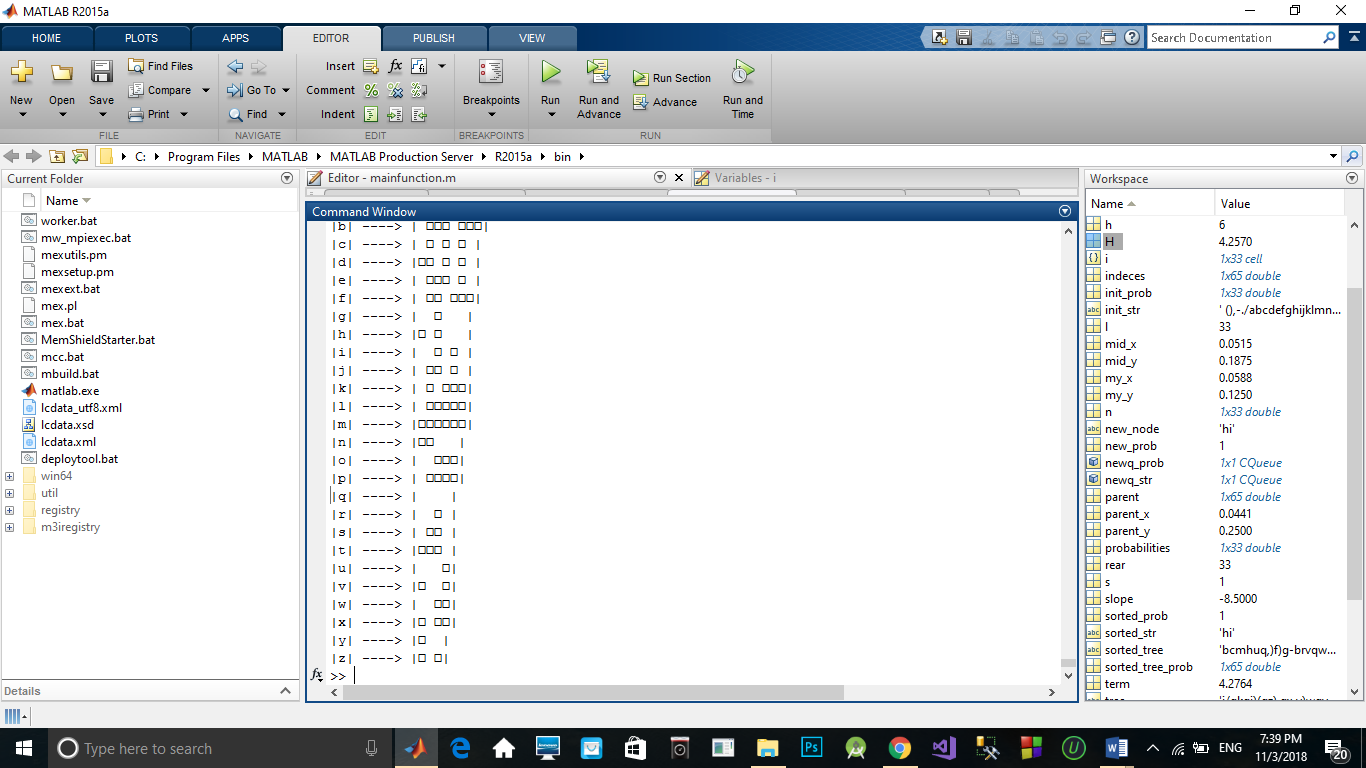


And the Final Tree is :-



The Symbols are :- (square = 1 , space = 0)





The Entropy Value = 4.2570

The Average Length = 4.2764

Efficiency = 99.55%

| 101 |<----------->|' '|, | 001 |<----------->|'e'|, | 1101 |<----------->|'t'|, | 1100 |<----------->|'a'| ,| 1001 |<----------->|'n'| ,| 1000 |<----------->|'o'| ,| 0111 |<----------->|'i'| ,| 0110 |<----------->|'r'| ,| 0100 |<----------->|'s'|, | 0000 |<----------->|'d'| ,| 11110 |<----------->|'l'| ,| 11100 |<----------->|'p'| ,| 00011 |<----------->|'c'|, | 111111 |<----------->|'h'| ,| 111110 |<----------->|'m'| ,| 111010 |<----------->|'u'|, | 010110 |<----------->|'g'| ,| 010100 |<----------->|'f'| ,| 000101 |<----------->|'b'| ,| 000100 |<----------->|'-'|, | 1110110 |<----------->|'v'| ,| 0101110 |<----------->|'w'| ,| 0101011 |<----------->|','|, | 0101010 |<----------->|'y'| ,| 11101110 |<----------->|'x'| ,| 01011111 |<----------->|'.'|, | 111011110 |<----------->|'z'| ,| 010111101 |<----------->|')'| ,| 010111100 |<----------->|'('| ,| 11101111101 |<----------->|'q'|, | 11101111100 |<----------->|'k'| ,| 11101111111 |<----------->|'j'|, | 11101111110 |<----------->|'/'|

*Conclusion :-*

The Huffman coding has optimized the compression of the symbols by producing the most lengthy codes for the least probable symbols and vice versa, therefore the average code length was almost equal to the original entropy which produced the minimum number of bits that can ever represent the symbols