Encoded string using Huffman tree:

1101100101000111011110001100011111101010110111010100100100010010101001111100111010001110111100000010011101111010101010110010011110000011001010100111111010001111011010100110010111100101111100101110010011101010101001101100101010110110100101010111011000101011110100110010010001001100010100111001011

Length: 297

Encoded string using normal ASCII encoding:

01101001 00100000 01100011 01110101 01110010 01110010 01100101 01101110 01110100
01101100 01111001 00100000 01101000 01100001 01110110 01100101 00100000 01100110
01101111 01110101 01110010 00100000 01110111 01101001 01101110 01100100 01101111
01110111 01110011 00100000 01101111 01110000 01100101 01101110 00100000 01110101
01110000 00101110 00101110 00101110 00100000 01100001 01101110 01100100 00100000
01101001 00100000 01100100 01101111 01101110 00100111 01110100 00100000 01101011
01101110 01101111 01110111 00100000 01110111 01101000 01111001 00101110

Length: 557

Calculation:
(297/100)*557 = 53.32 %

a)

The string encoded with the Huffman Tree is equivalent to 53.32% of the string encoded with ASCII in size. Almost half the size of the size of string has been reduced using Huffman encoding.

The reason why there is a significant reduction in string size is because the encoded string has characters that appeared frequently in the Jabberwock.txt.

The Huffman encoding works as the following; it constructs a binary tree where the path of frequently used characters is short and where path of less frequently used characters is long, instead of having a binary tree where every character has the same path length. Going left in the tree encodes ‘0’, while going right in the tree encodes ‘1’.

Had the encoded string been filled with non frequent characters in the Jabberwock.txt such as ‘x’, ‘;’ and ‘?’ (which only appeared once in the file), the Huffman encoded string’s size would be larger than the encoded string using ASCII.

long, instead of having a binary tree where every character has the same path length. Going left in the tree encodes ‘0’, while going right in the tree encodes ‘1’.