Huffman Coding

A **Huffman code** is a type of optimal prefix code that is used for compressing data. The Huffman encoding and decoding schema is also lossless, meaning that when compressing the data to make it smaller, there is no loss of information.

The Huffman algorithm works by assigning codes that correspond to the relative frequency of each character for each character. The Huffman code can be of any length and does not require a prefix; therefore, this binary code can be visualized on a binary tree with each encoded character being stored on leafs.

There are many types of pseudocode for this algorithm. At the basic core, it is comprised of building a Huffman tree, encoding the data, and, lastly, decoding the data.

Here is one type of pseudocode for this coding schema:

- Take a string and determine the relevant frequencies of the characters.
- Build and sort a list of tuples from lowest to highest frequencies.
- Build the Huffman Tree by assigning a binary code to each letter, using shorter codes for the more frequent letters. (This is the heart of the Huffman algorithm.)
- Trim the Huffman Tree (remove the frequencies from the previously built tree).
- Encode the text into its compressed form.
- Decode the text from its compressed form.

You then will need to create encoding, decoding, and sizing schemas.

For Example:

```
import sys

def huffman_encoding(data):
    pass

def huffman_decoding(data,tree):
    pass

if __name__ == "__main__":
    codes = {}

    a_great_sentence = "The bird is the word"

    print ("The size of the data is: {}\n".format(sys.getsizeof(a_great_senten
    print ("The content of the data is: {}\n".format(a_great_sentence))

    encoded_data, tree = huffman_encoding(a_great_sentence)

    print ("The size of the encoded data is: {}\n".format(sys.getsizeof(int(en
))))
    print ("The content of the encoded data is: {}\n".format(encoded_data))

    decoded_data = huffman_decoding(encoded_data, tree)

    print ("The size of the decoded data is: {}\n".format(sys.getsizeof(decode
    print ("The content of the encoded data is: {}\n".format(decoded_data))
```

Resources

Huffman Visualization!

Tree Generator

Additional Explanation