Problem 3 – Huffman Coding

Huffman Encoding:

Storing the frequencies takes O(n) time and space, as we have to traverse the array of data.

Sorting frequencies takes O(n*log(n)), and O(n) space.

The main part's goal is to create a tree, it takes a sorted list as an input. Under the hood it removes the elements until there's only one left, this operation is O(n), as we traverse through whole sorted list of length n.

Then, for the created tree `def encode()` generates the codes for each character in the tree – 0 for left side characters and 1 for right side characters. This one is O(n) in terms of both, time and space complexity as it traverses through whole list.

Huffman Decoding:

For decoding we are traversing through whole encoded string and the tree, then recursion is used. Then the string is returned that has the step by step procedure for finding desired value. This one takes also O(n), as the procedures are dependent upon length of the string.