Lecture 24 - Problems

Write pseudocode for each of the following parts of the process of building a Huffman coding tree. Feel free to make use of anything you've already implemented in this class (e.g., if you need to sort an array, you can use a sorting method from A1).

Try to make your pseudocode detailed enough that no design decisions need to be made to turn it into Java.

1. **Count frequencies**. Given an input `String`, Calculate the frequency (i.e., number of occurrences) of each character in a string.
2. **Build the tree.** Given the frequencies from part (1), build a Huffman tree. You'll likely need to define a tree and/or tree node class to represent it.
3. **Decode**. Given an encoded bitstring (assume it's a `String` containing only the characters `0` and `1`), decode it into the original input string.
4. **Build a coding dictionary**. To make encoding easier, build a dictionary (represent it using a hash table) that maps each possible input character with its binary encoding according to a given Huffman tree.
5. **Encode**. Given your coding dictionary and an input string, encode the string into its compressed binary representation (again, for simplicity we'll just represent it with a `String` containing `0`s and `1`s, although this is using a full byte for each binary digit and we would want to do better in a practical implementation).