Application of binary trees: Huffman coding

Characters when stored or transmitted have to be encoded using some number of bits. We are used to fixed-length encoding in which every character is encoded using the same number of bites.

Ex.

Asc II uses 8 bites for character.

8n bits are required for n characters.

We could use a variable-length encoding in which the number of bits varies by character. We can achieve an overall space savings by using fewer bits for frequently occurring characters and more bits for infrequently occurring characters.

One solution is o use prefix coding (also known as prefix-free coding). The encoding of any character is never a prefix of any other encoding.

We can view the encoding as a binary tree in which left branches are zeroes, right branches are ones, and characters are always leaves.

Q: Given the frequencies of the input characters, how do we build the tree that gives us the minimu number of bits for our input?

A: Huffman coding!

Huffman coding: each character starts as a single node/root. Repeatedly pick the roots with the lowest counts/frequencies and merge them under a new root