The construction of a Huffman tree can be illustrated with an example: If we are given the phrase **"There is no place like home"**, without double quotes, we can first construct the frequency table by counting each character occurrence in the phrase.

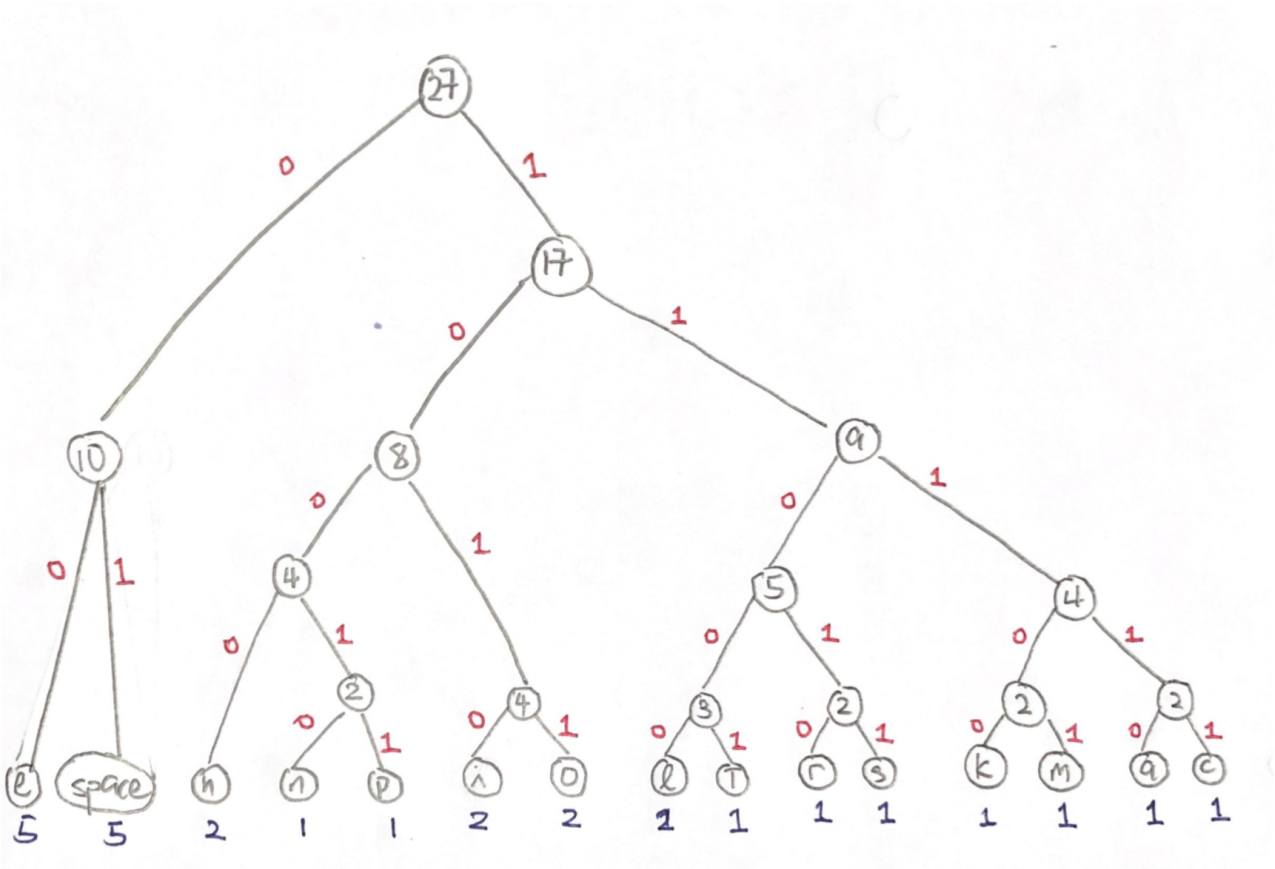
Frequency Table

Character	Frequency
e	5
space	5
h	2
i	2
o	2
l	2
T	1
r	1
s	1
n	1
p	1
a	1
c	1
k	1
m	1

Huffman Tree Construction

We can then construct the Huffman Tree by selecting the smallest two nodes and merging them, until there is one root left.

Note that: When we are picking the two smallest nodes, there might exist multiple nodes with the same weight. The Huffman algorithm does not specify the method for choosing the correct node in these kinds of cases. So, no matter which node we pick as long as **their weight are equal** it will still encode the message with the optimal number of bits among prefix-free codes.



Encoding Table

After constructing the tree, we can just create the codes for each character by tracing a path from root to leaf node (aggregating the bits along the way).

Character	Frequency	Encoding
e	5	00
space	5	01
h	2	1000
i	2	1010
o	2	1011
l	2	11000
T	1	11001
r	1	11010
s	1	11011
n	1	10010
p	1	10011
a	1	11110
c	1	11111
k	1	11100
m	1	11101

Average bits per character can be calculated as follows: ((5 * 2) + (3 * 2 * 4) + (2 * 5) + (9 * 5)) / 27 = 3.3 bits per character, which is better than 4 bits. We are taking advantage of the fact that each character has different probabilities (occurrences), so we can assign shorter codes to more frequent characters to save space, in contrast to RLE which uses a fix number of bits to represent the count.