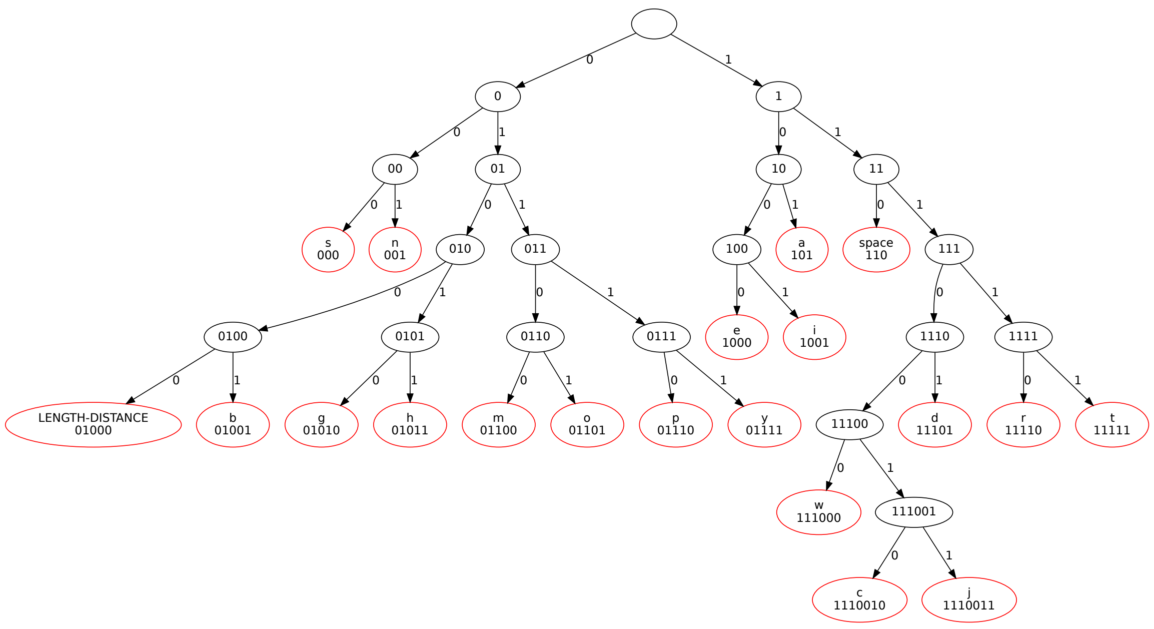
Day 6: Let’s sing a song!

You carry on working with P0P0, and manage to fix another 16 byte memory chip. He now has a total of 40 bytes of working memory, and as a token of his appreciation he would like to sing you a song. Unfortunately not many songs fit in 40 bytes, so he will need better compression.

P0P0 uses the same (“Huffman”) compression algorithm as before, but with a different decode tree:



What else has changed? This time there is a special *LENGTH-DISTANCE* marker that can appear in the message. When you find a *LENGTH-DISTANCE* marker, this means that some previous characters that you have already decoded should be repeated.

The *LENGTH-DISTANCE* marker will always be followed by two 8 bit integers: the first contains a number of characters (the “length”), the second says how many characters prior to the current position the repeated sequence started (the “distance”).

A message “I had recently had my tea” could be represented as “I had recently***<LENGTH-DISTANCE(5, 13)>***my tea”. In this example the LENGTH-DISTANCE(5, 13) means that we should repeat the 5 characters which started 13 characters earlier: “ had ”.

What are the words of the song that P0P0 wants to sing?

01001101 00110100 11010001 10100100 11100111 00111111 10011101 01100101

00011010 11111010 00110111 00100110 10110010 01001010 10110111 01011011

11000001 11011111 01011100 01100001 11111011 00111110 00011001 00000100

01100101 11010010 01110011 10010000 00111010 01011000 10111010 00100100

10101011 01111110 00111011 11010111 11100100 11000101 11110000

Bonus activities:

- How many ASCII characters did the song contain before compression?
- What percentage is the compressed 40 bytes of the original song size? (this is known as the “compression ratio”)
- How big would the message have been if we didn’t have the DISTANCE-LENGTH markers? How much does it improve compression?
- Read about LZ Compression on Wikipedia. This compression algorithm is used in GIF and ZIP files.