

Huffman Coding

Imagine the pages of a book. All those words.
With all of the letters.

Could we use RLE to compress this data?

Huffman Coding

RLE isn't very helpful when trying to compress the data here - look at how many single letters there are on this page.

Rather than m , we need to encode $1m$ - that's more bits to encode.

So using RLE to encode text like this is a bad idea because you rarely get long repeats of the same data item.

Huffman Coding

ASCII code uses 7 bits for each character.

So, in a word like:

banana

each character is stored using 7 bits.

total bits = number of characters x 7

Huffman Coding

Languages all have some letters which are much more common than others.

What if we could use that to reduce the number of characters needed to encode each character?

In

banana

for example, there are 3 as, 2ns and 1 b

Huffman Coding

Huffman coding takes the most common letters in a block of text and encodes them using fewer bits.

Less frequent characters are encoded using more bits.

This allows us to use less bits in total.

Huffman Coding

w o o l l o o m o o l o o

How often does each character appears?

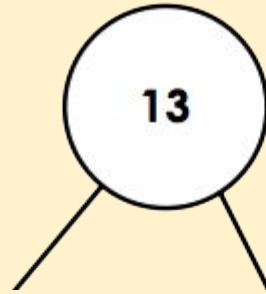
w	o	l	m

Huffman Coding

Once we know the frequency of each character we can build a Binary Tree to store each letter that appears in the text.

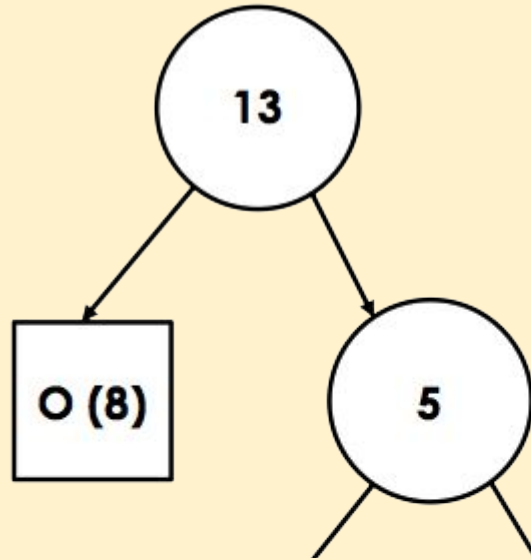
woolloomooloo

o	8
l	3
m	1
w	1



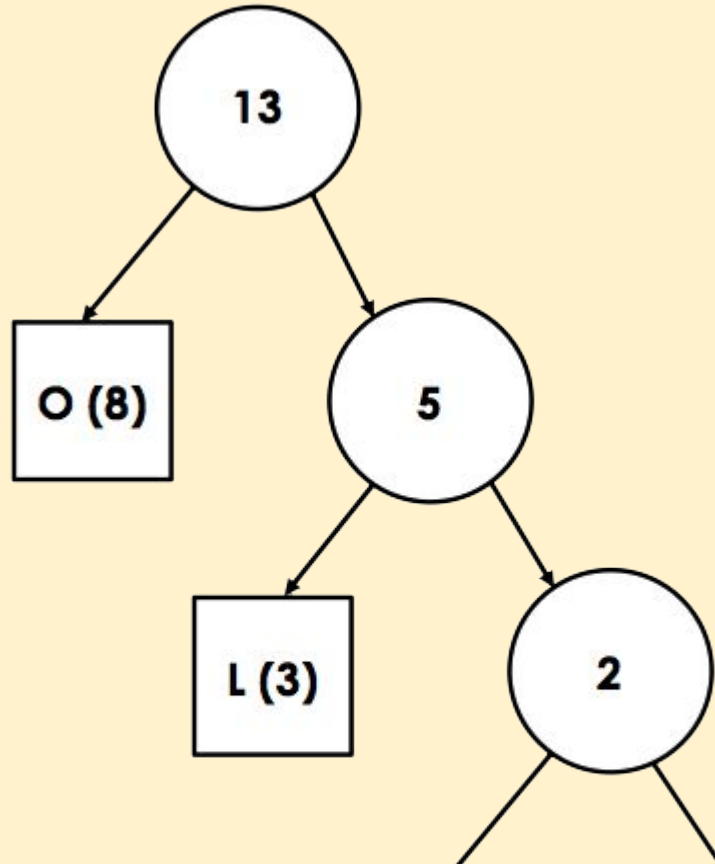
woolloomoo

o	8
l	3
m	1
w	1



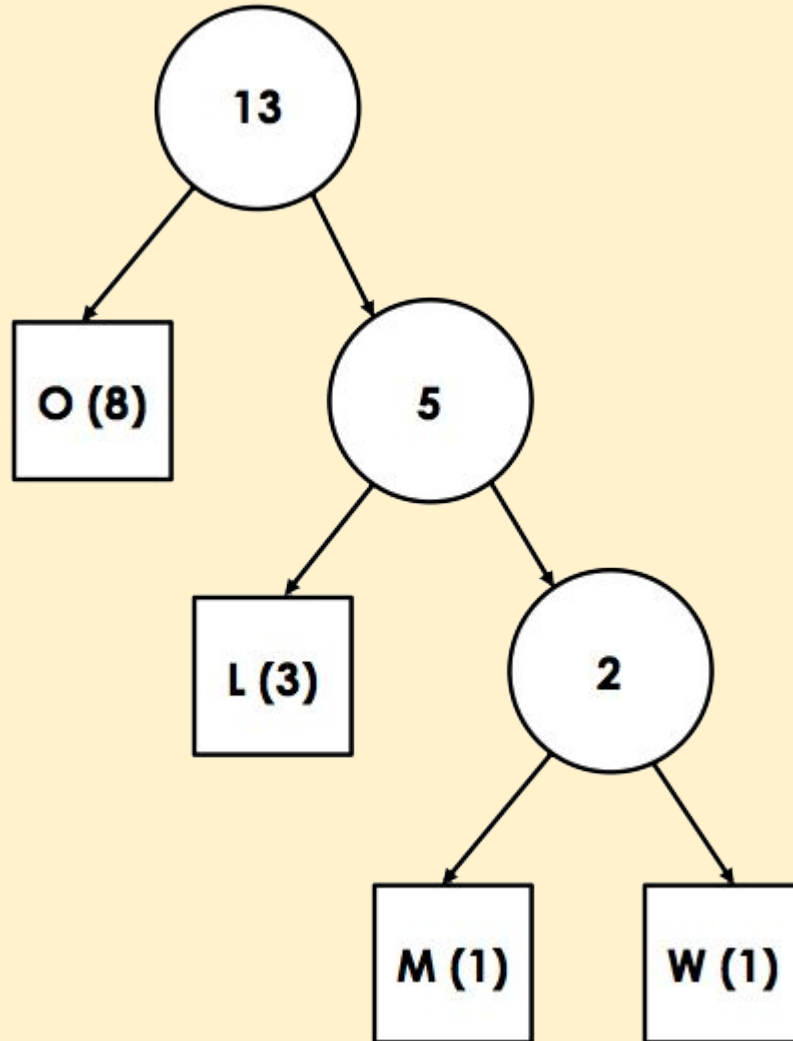
woolloomooloo

o	8
l	3
m	1
w	1



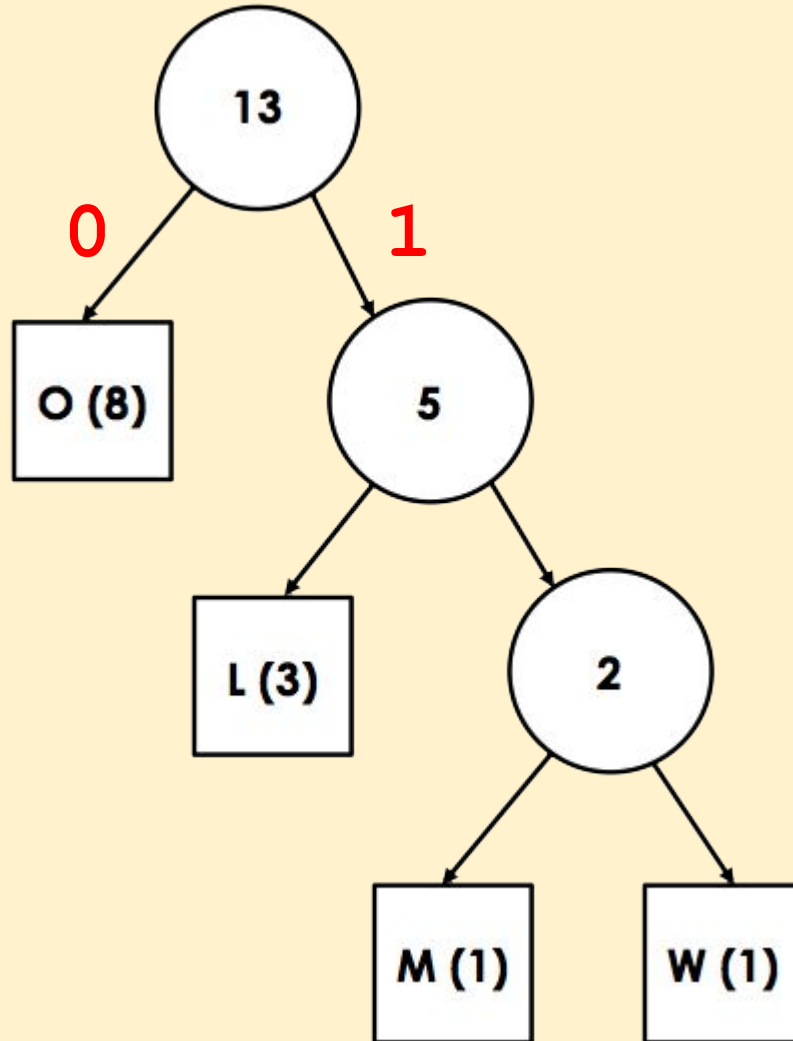
woolloomooloo

o	8
l	3
m	1
w	1



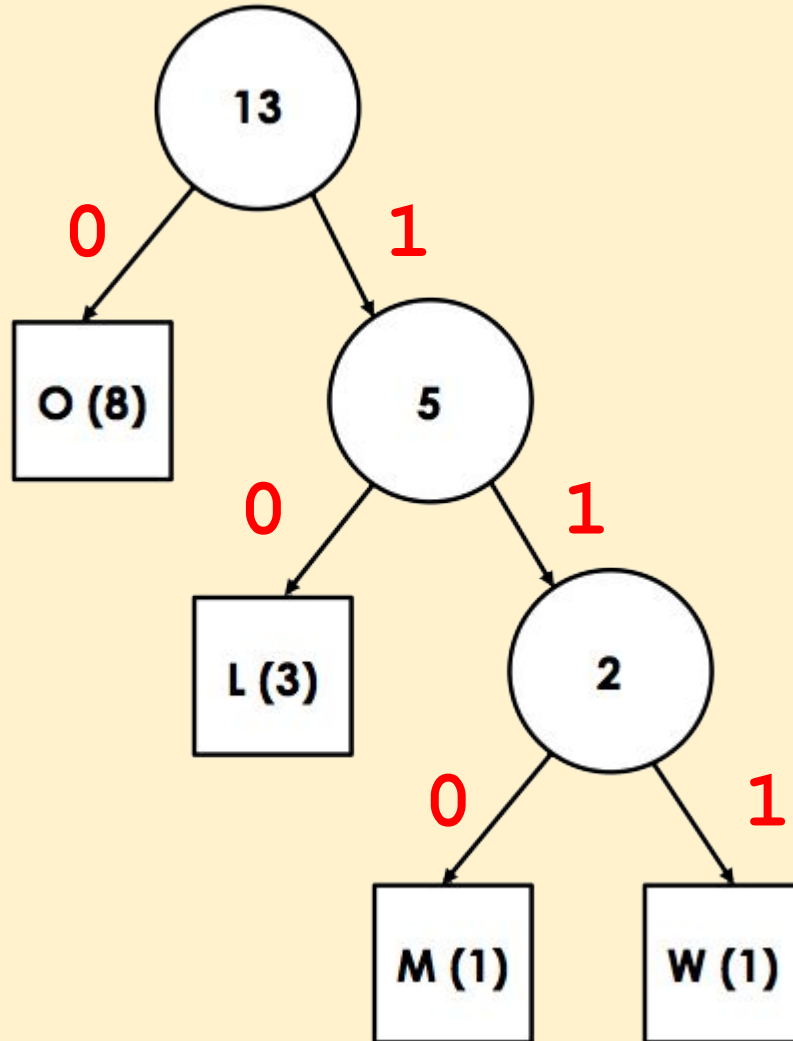
woolloomoo

o	8
l	3
m	1
w	1



woolloomooloo

o	8
l	3
m	1
w	1



Huffman Coding

The Huffman Tree gives us a way of encoding each character we used:

Character	Huffman Coding	Memory
o	0	1 bit
l	10	2 bits
m	110	3 bits
w	111	3 bits

Remember: ASCII is 7 bits per character

Huffman Coding

So, using the Huffman Tree

woollloomooloo

becomes

11100101000110001000

with gaps

111 0 0 10 10 0 0 110 0 0 10 0 0

Huffman Coding

Space needed

w o o l l o o m o o l o o

ASCII = 13 character x 7 bits = 91 bits

Huffman Tree:

11100101000110001000

= 20 bits

Space saving = 91 - 20 = 71 bits