3.3.8 Data compression (Huffman coding)

Lesson plan and printable activities

Materials needed

1. 3.3.8 (Huffman coding) Lesson.
2. Quiz 1, Quiz 2, Quiz 3.
3. Tables from worked example.

Lesson aims

1. To gain an overview of how the Huffman coding method may be used to encode plain ASCII text.
4. To see how the Huffman coding method compares in its application to related compression algorithms, eg Run Length Encoding (RLE)

Lesson objectives

1. Analyse a piece of plain ASCII text in order to determine the frequency of specific characters occurring.
5. Transform examples of uncompressed ASCII text into a compressed format using the Huffman coding method.
6. Convert a specific example of compressed text into ASCII characters using a given Huffman tree.
7. Describe conditions where the Huffman coding method of compression will provide optimal benefits.

Please note: This lesson can be split into two parts with slides 1–21 as lesson 1 and slides 22 onwards as lesson 2. To address: 'Students should be familiar with carrying out calculations to determine the number of bits saved by compressing a piece of data using Huffman coding.' This could be included as a homework.

Starter activity (10 minutes)

1. Re-cap what has been discussed in the previous lesson, emphasising that the focus continues to be on the use of data compression techniques. State that we will be looking specifically at Huffman coding in this lesson and that the technique is useful for encoding plain text.

   Use Quiz 1 to start the students thinking about the fundamental basis of how Huffman works, ie the fact that characters appear at different frequencies in text, and that there quickly become noticeable differences in the frequency patterns of specific characters as the size of the text increases.

Main activities (45 minutes)

1. **Slide 4:** Start by explaining that Huffman method is very useful but more complex for students to implement than the Run Length Encoding (RLE) method seen previously. Refer to the teacher notes in regard to the practical complexity of using the method in the classroom.

2. **Slide 5:** Demonstrate to students the frequency of different characters in a piece of text by using the search function on a Word or PDF document. Get students to suggest which the most frequent characters might be. Then get students to attempt Quiz 1.

3. **Slides 7–27:** There is a single worked example that provides sufficient complexity, as the text chosen has a number of characters. It is likely that not all students will follow the diagrams at an equal rate, it is strongly suggested that the accompanying tables are also provided as single-side printed documents in order to help students follow the explanation.

4. It might be useful for more able students to emphasise that the algorithm used is ‘lossless’ – a term not used on the GCSE specification, the rationale for this is that the encoded text may be reconstituted using the Huffman tree generated.

5. Complete Quiz 2.

Plenary activity

1. Complete Quiz 3**,** which asks students to consolidate their thinking after they have seen the algorithm in action.

Lesson

3.3 Fundamentals of data representation

3.3.8 Data compression (Huffman coding)

Lesson

Objectives

Be able to:

- Analyse a piece of plain ASCII text in order to determine the frequency of specific characters occurring.
- Transform examples of uncompressed ASCII text into a compressed format using the Huffman coding method.

Objectives

Be able to:

- Convert a specific example of compressed text into ASCII characters using a given Huffman tree.
- Describe conditions where the Huffman coding method of compression will provide optimal benefits.

Introduction

Continuing the theme of data compression…

In the **previous lesson,** we looked at why it is beneficial to be able to compress digital files.

We looked at one specific method of carrying out data compression:

RLE

Today, we are going to look at another method:

Huffman coding

How often do different characters occur in text?

This method works by recording the frequencies by which characters appear in text.

Some characters appear far more often than others.

Which do you think occur most frequently?

How about less frequently?

Try Quiz 1

Focus on Huffman coding

There are two steps when using Huffman's technique:

1. Create a Huffman tree – this gives each character used a unique code.
2. Encode the character sequence into a binary stream.

Huffman coding – worked example

Look at this sentence:

an easy example

What do you notice?

Huffman coding – worked example

Now create a table showing the frequency of occurrence for each character.

an easy example

Character	Frequency	Character	Frequency
a	Click to reveal	p	Click to reveal
e		s	
l		x	
m		y	
n		space	

Huffman coding – worked example

Now we need to create the graphical node representation used by the Huffman method.

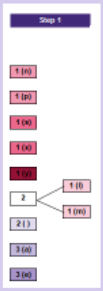
To begin this process, we have to start with the lowest frequencies at the top of the list – the actual alphabetical order does not matter.

The **Huffman tree** generated allows you to assign a **unique code** to **each character** used.

The wider the range of characters used, the bigger the Huffman tree!

Huffman coding – worked example

At the **start**, the characters are placed in order of occurrence, from top to bottom. Where the frequency is the same, the order does not matter. ( ) indicates the space character.



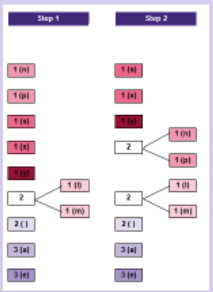
Step 1

The top two entries' frequencies are added together and a new node is inserted.

Huffman coding – worked example

Step 2

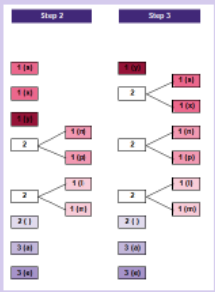
Same as **Step 1** with **1 (n)** and **1 (p)**.



Huffman coding – worked example

Step 3

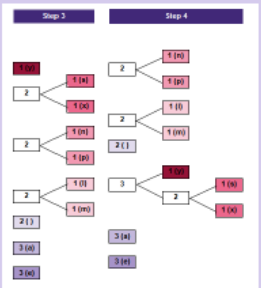
Same as **Step 2** with **1 (s)** and **1 (x)**.



Huffman coding – worked example

Step 4

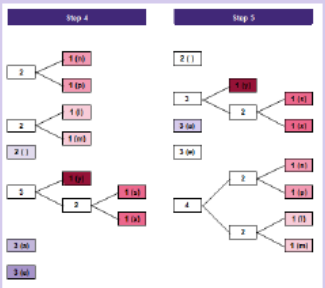
Look what happens to **1 (y)**.



Huffman coding – worked example

Step 5

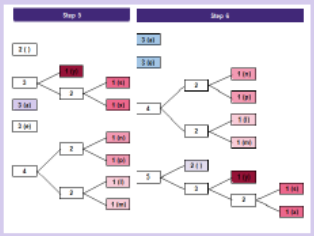
The top two nodes combine and move to the end of the list as to their combined value exceeds that of the other elements.



Huffman coding – worked example

Step 6

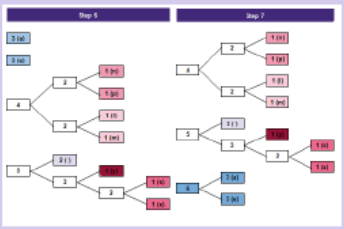
The pattern continues…



Huffman coding – worked example

Step 7

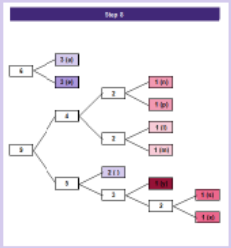
More of the same…



Huffman coding – worked example

Step 8

The objective is to end up with one single node.



Huffman coding – worked example

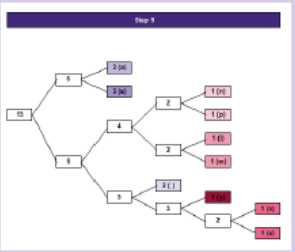
Step 9

We made it!

The value of the left-hand node should be equal to the number of characters in the original string.

an easy example = 15 characters

Success!



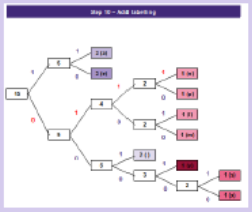
Huffman coding – worked example

Step 10

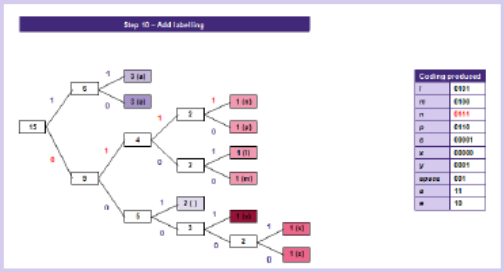
Put '1' on the upward directions and '0' on the downward directions.

How would we denote the 'n'?

Down one, up one, up one, up one = 0111



Huffman coding – worked example



Huffman coding – worked example

a	n		e	a	s	y		e	x	a	m	p	l	e
11				11						11				
			10					10						10
		001					001							
											0100			
												0110		
						0001								
													0101	
	0111													
					00001									
									00000					

Full binary representation =
11 0111 001 10 11 00001 0001 001 10 00000 11 0100 0110 0101 10

Decoding with the Huffman method – worked example

Our encoded text would be received as follows…

1101110011011000010001001100000011010001100101 10

You must be in possession of the Huffman tree to decode the text!

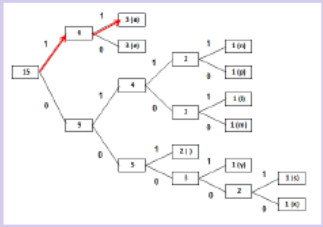
Decoding with the Huffman method – worked example

First character

Start at beginning of string and follow binary until it terminates:

11011100110110000100 01001100000011010001 10010110

Following '1' then '1' takes us to a.



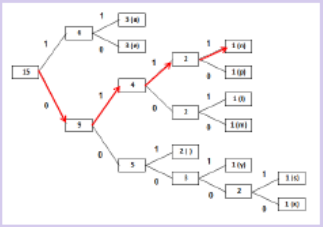
Decoding with the Huffman method – worked example

Second character

Move to the next set of digits:

~~11~~**0111001101100001000 10011000000011010001 10010110**

= an



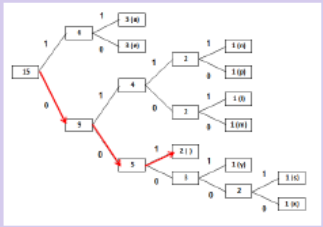
Decoding with the Huffman method – worked example

Third character

Continue the previous process – this time we find a space character…

~~11011100~~**1100001000 10011000000011010001 10010110**

= an



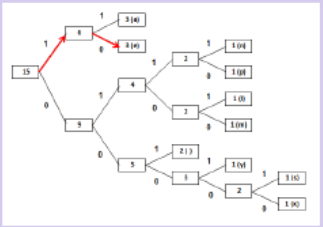
Decoding with the Huffman method – worked example

Fourth character

~~1101110011~~**0110000100 10011000000011010001 10010110**

= an e

Continue until no binary digits remain.



Huffman coding – conclusion

Look at the table.

You can see that the characters at the top generally:

- appear most frequently
- have short binary representations.

This is the advantage of Huffman coding.

If conventional ASCII uses eight bits to represent a character. Huffman coding can reduce that to as little as two bits.

l	0101
m	0100
n	0111
p	0110
s	00001
x	00000
y	0001
space	001
a	11
e	10

Huffman coding – conclusion

Note: Huffman coding method will incur some 'overhead' when used.

What do you think this is?

It is necessary to transmit a representation of the encoding 'tree' along with the encoded data. Each Huffman tree is unique as it is dependent upon the piece of text which it used to encode.

The Huffman method works best when there is a good distribution of frequently occurring 'common' characters.

To round things off…

Complete Quiz 2 and Quiz 3

Quiz 1 – Frequency analysis of characters in text

Question 1

Analyse the frequency of the occurrence of the characters in text using the example provided below.

Note: You should ignore the case of the characters and include any spaces.

The quick brown fox jumped over the lazy wolf.

Character	Frequency	Character	Frequency
a		***n***	
b		***o***	
c		***p***	
d		***q***	
e		***r***	
f		***s***	
g		***t***	
h		***u***	
i		***v***	
j		***w***	
k		***x***	
l		***y***	
m		***z***	
.		***,***	
space			

Question 2

Analyse the frequency of the occurrence of the characters in text using the example provided below.

Note: You should ignore the case of the characters and include any spaces.

We know the most commonly used passwords are numeric ones, so they are not secure, and people also use the same passwords for multiple sites. If one site gets hacked all the places that you use the same password get compromised, they are a big pain.

Character	Frequency	Character	Frequency
a		***n***	
b		***o***	
c		***p***	
d		***q***	
e		***r***	
f		***s***	
g		***t***	
h		***u***	
i		***v***	
j		***w***	
k		***x***	
l		***y***	
m		***z***	
.		***,***	
space			

Quiz 1 – Frequency analysis of characters in text – answers

Question 1			
Character	**Frequency**	**Character**	**Frequency**
a	1	***n***	1
b	1	***o***	3
c	1	***p***	1
d	1	***q***	1
e	4	***r***	1
f	2	***s***	-
g	-	***t***	-
h	-	***u***	2
i	1	***v***	1
j	1	***w***	2
k	1	***x***	-
l	1	***y***	1
m	1	***z***	1
.	1	***,***	-
space	7		

Question 2			
Character	**Frequency**	**Character**	**Frequency**
a	14	***n***	6
b	1	***o***	16
c	5	***p***	8
d	5	***q***	-
e	28	***r***	7
f	2	***s***	23
g	3	***t***	11
h	3	***u***	7
i	7	***v***	-
j	-	***w***	5
k	2	***x***	-
l	6	***y***	3
m	9	***z***	-
.	2	***,***	3
space	45		

Quiz 2 – Encode a piece of text using Huffman coding

Question 1
Create a Huffman tree for the following text sample. Your tree should indicate the binary code paths required to be followed in order to obtain each unique character. *here horse*

Quiz 2 – Encode a piece of text using Huffman coding – answers

Question 1
Create a Huffman tree for the following text sample. Your tree should indicate the binary code paths required to be followed in order to obtain each unique character. *here horse* Correct analysis as per table Correct tree L/H total = 10 Correct nodes Correct binary labelling

Huffman coding – character frequency record sheet

Character	Frequency	Character	Frequency	Character	Frequency
a	-	**n**	-	***space***	1
b	-	**o**	1	**.**	-
c	-	**p**	-	**,**	-
d	-	**q**	-		
e	3	**r**	2		
f	-	**s**	1		
g	-	**t**	-		
h	2	**u**	-		
i	-	**v**	-		
j	-	**w**	-		
k	-	**x**	-		
l	-	**y**	-		
m	-	**z**	-		

Diagram

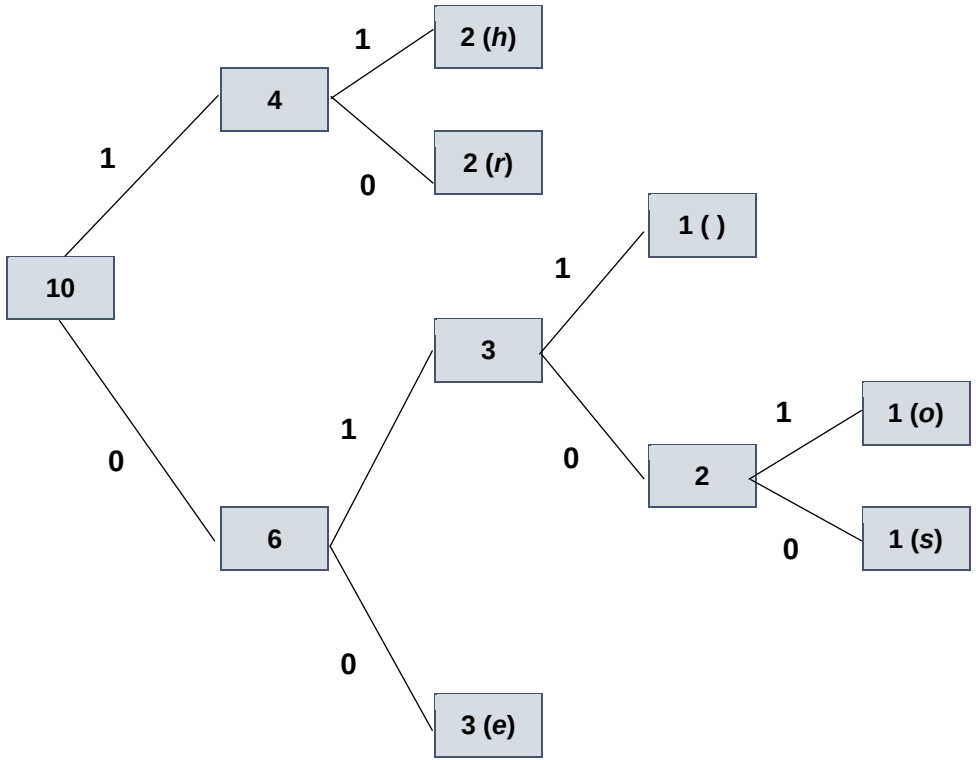


Table – for interest only

h	11
r	10
e	00
o	0101
s	0100
space	011

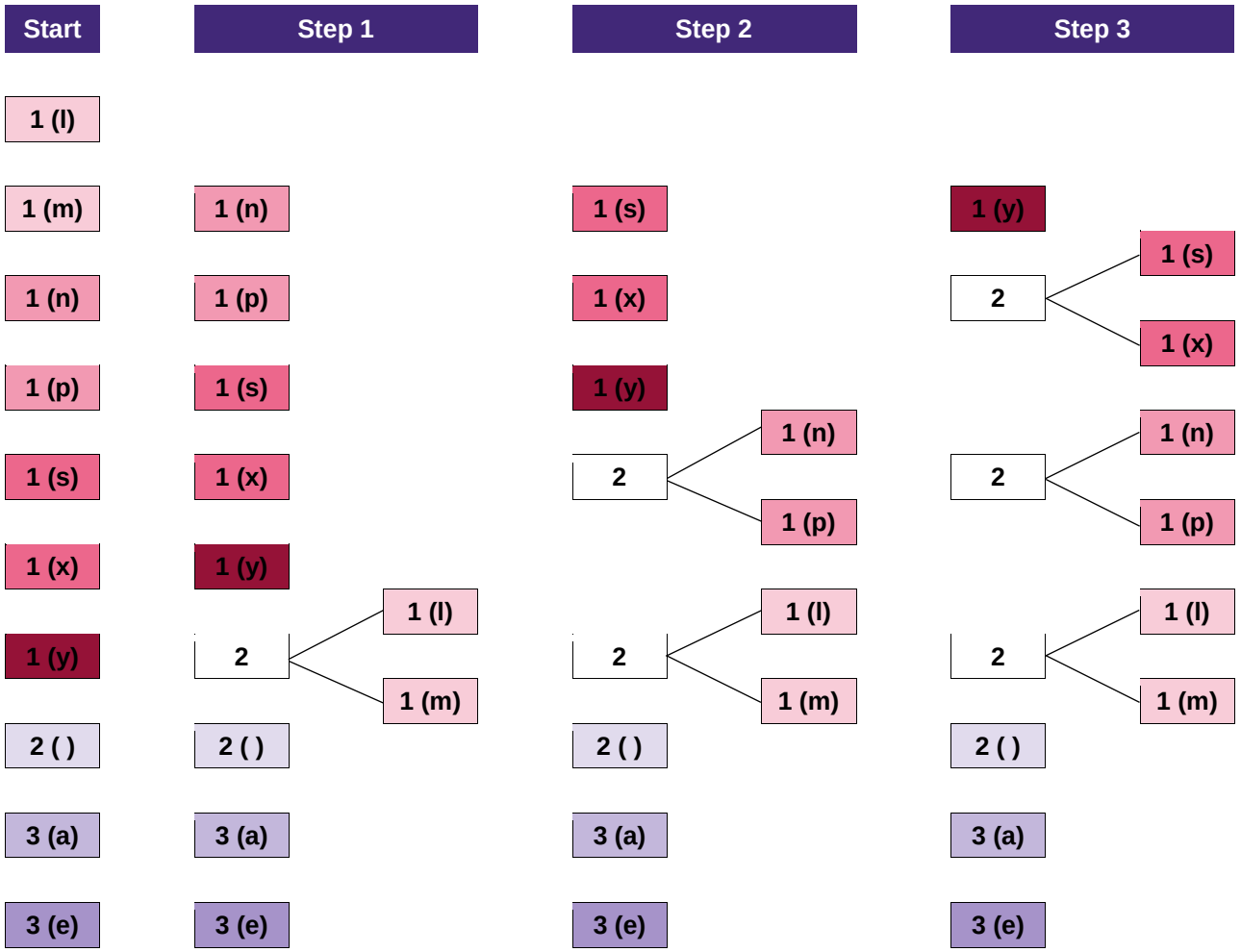
Quiz 3 – Some general considerations when using Huffman coding

Question
Look at the following two extracts of text: **1.** *Chemical formula $C_3H_4Br_2$* **2.** *The sun's diameter is 866,000 miles. Rotates every 606 hours. The length of time its current carries the sun over its orbit is unknown. The sun remains a melted mass; its vibration is maintained; has but little vapour and its theme reflected on the surface of its obsequious attendants which gives them heat and light.* Which of the two pieces of text might benefit from the use of Huffman coding to save space? Why?

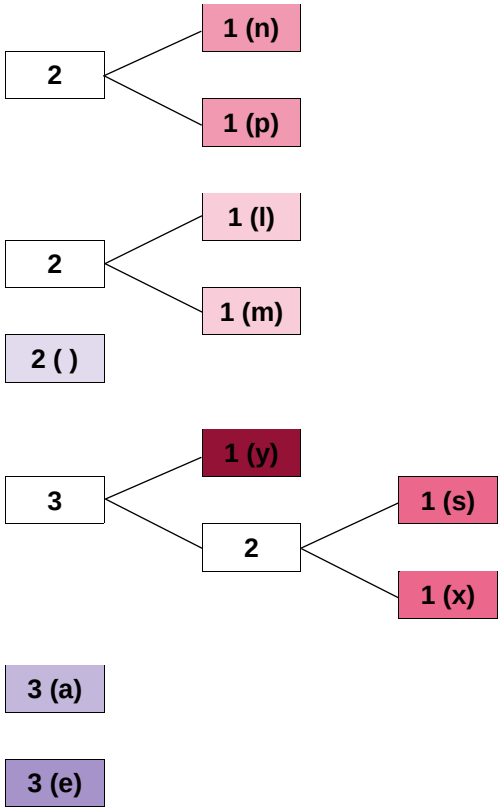
Quiz 3 – Some general considerations when using Huffman coding – answers

Question
Look at the following two extracts of text: 1. *Chemical formula $C_3H_4Br_2$* 2. *The sun's diameter is 866,000 miles. Rotates every 606 hours. The length of time its current carries the sun over its orbit is unknown. The sun remains a melted mass; its vibration is maintained; has but little vapour and its theme reflected on the surface of its obsequious attendants which gives them heat and light.* Source: gutenberg.org/cache/epub/1331/pg1331-images.html Which of the two extracts might benefit from the use of Huffman coding to save space? Why?
There is no set answer but you should look for the following thinking: • Extract 1 is a relatively **short** piece containing a large variety of characters all of which will need their own encoding sequence and lead to a disproportionately large Huffman tree being produced. • The shortness of the first piece makes it inefficient to encode as also need to include the associated Huffman tree to enable decoding. Give extra credit if students attempt a (basic) character count to back up their arguments.

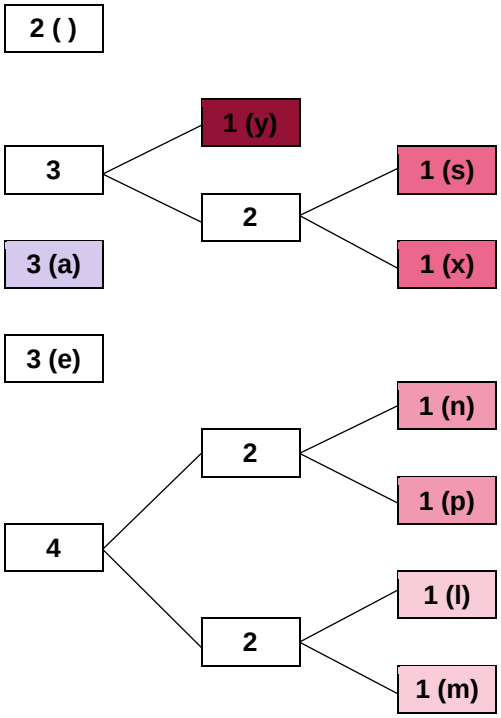
Tables from worked example



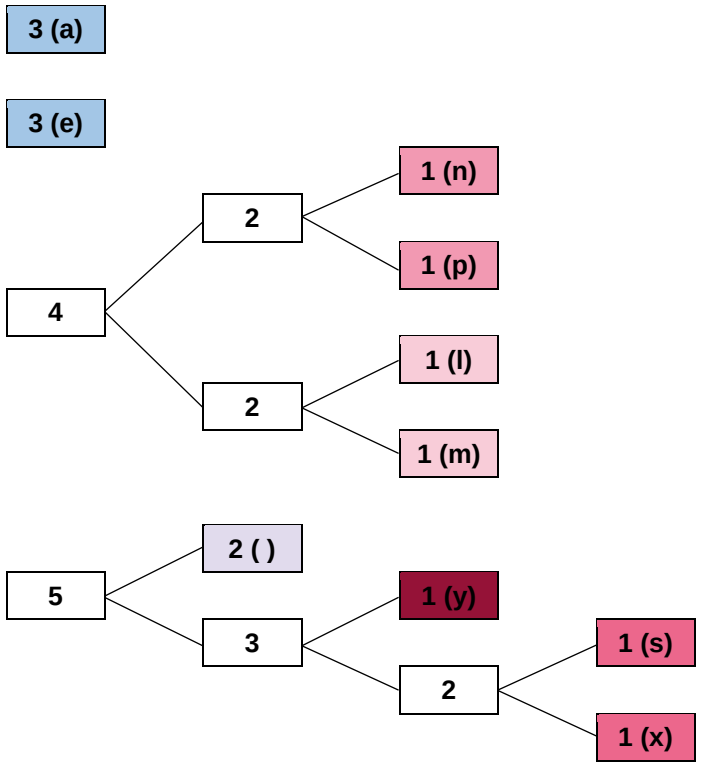
Step 4



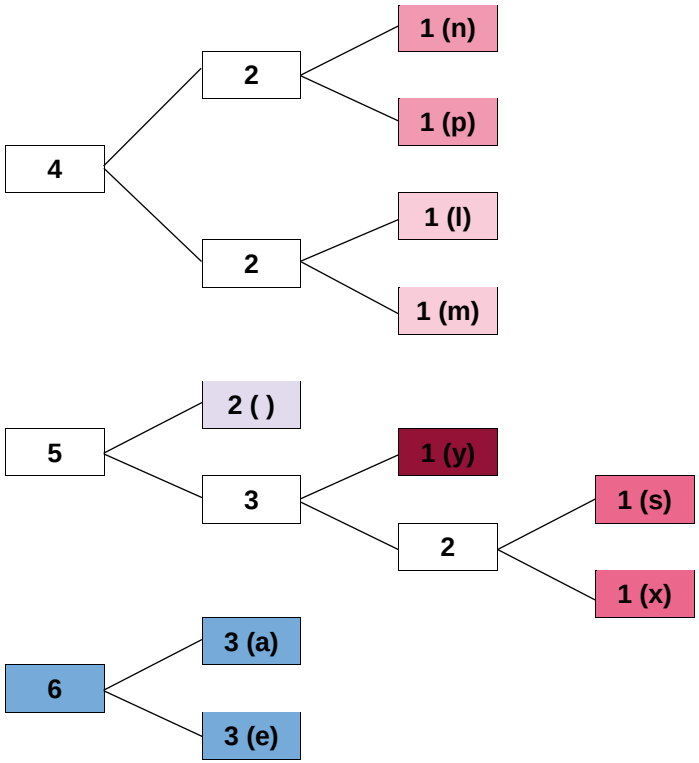
Step 5



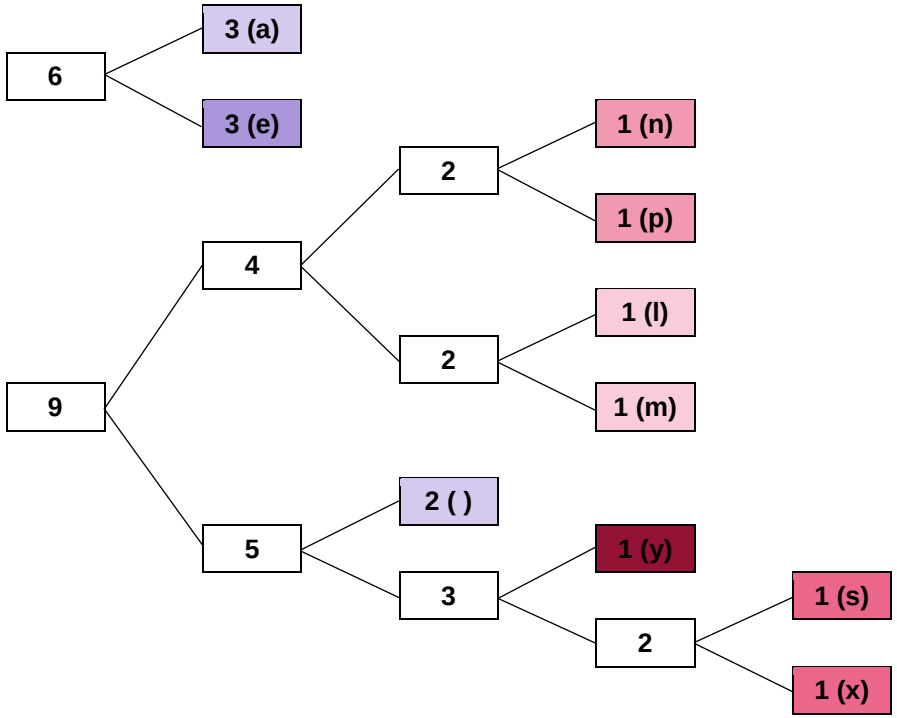
Step 6



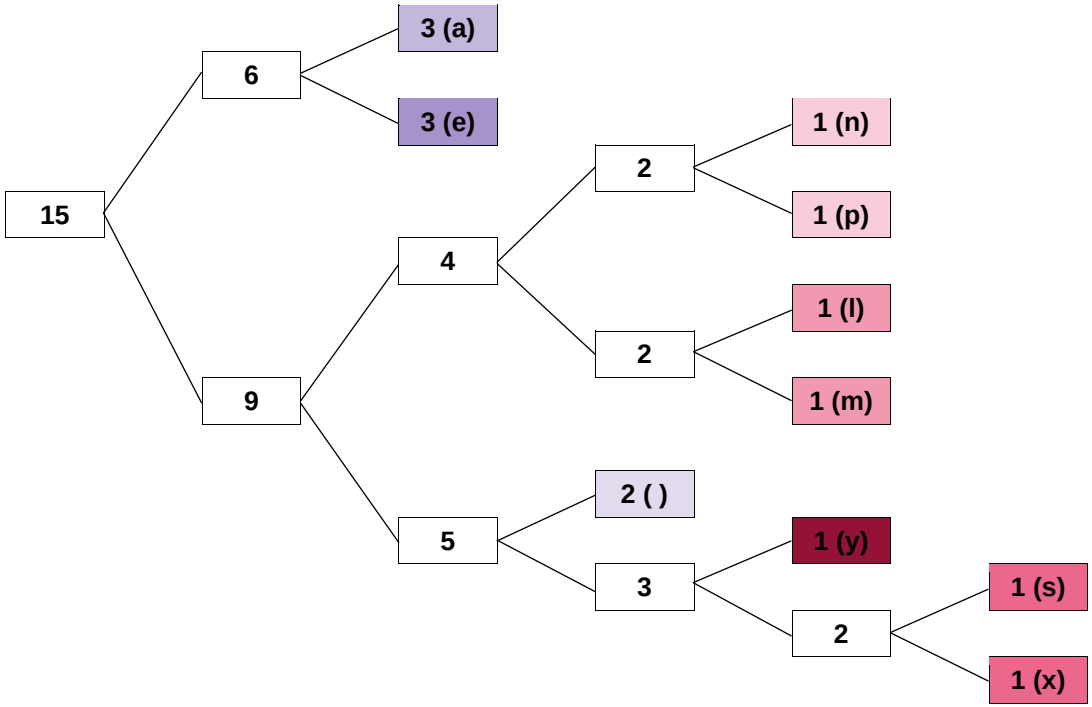
Step 7



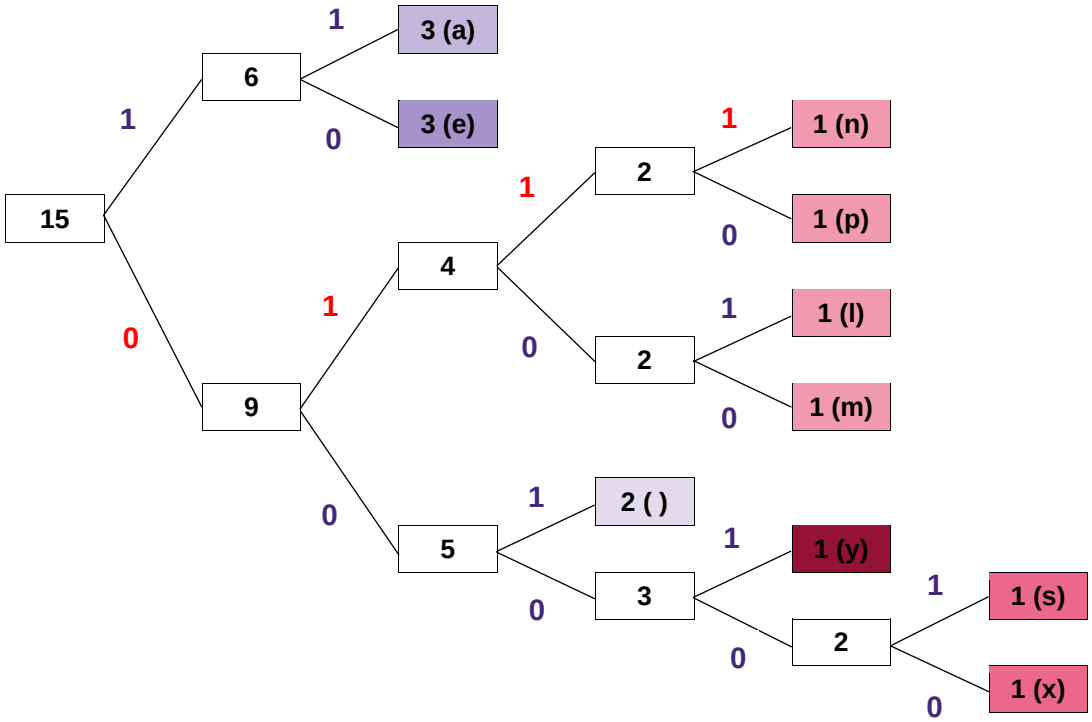
Step 8



Step 9



Step 10 – Add labelling



Coding produced	
l	0101
m	0100
n	0111
p	0110
s	00001
x	00000
y	0001
space	001
a	11
e	10