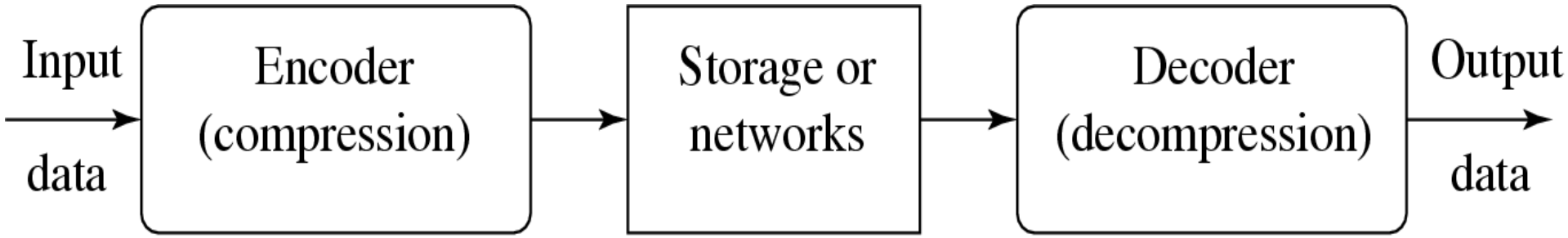
Lecturer : Dr. Amira Gamal Mohamed

Introduction

- **Compression:** the process of coding that will effectively reduce the total number of bits needed to represent certain information.



- **Why do we need Data Compression?**
  - Compressed data take up less storage space.
  - Compressed data can be transmitted more quickly.

- **Compression ratio:** is defined as the ratio between the uncompressed size and compressed size.

$$compression\,ratio = \frac{B_0}{B_1}$$

B_0 – number of bits before compression

B_1 – number of bits after compression

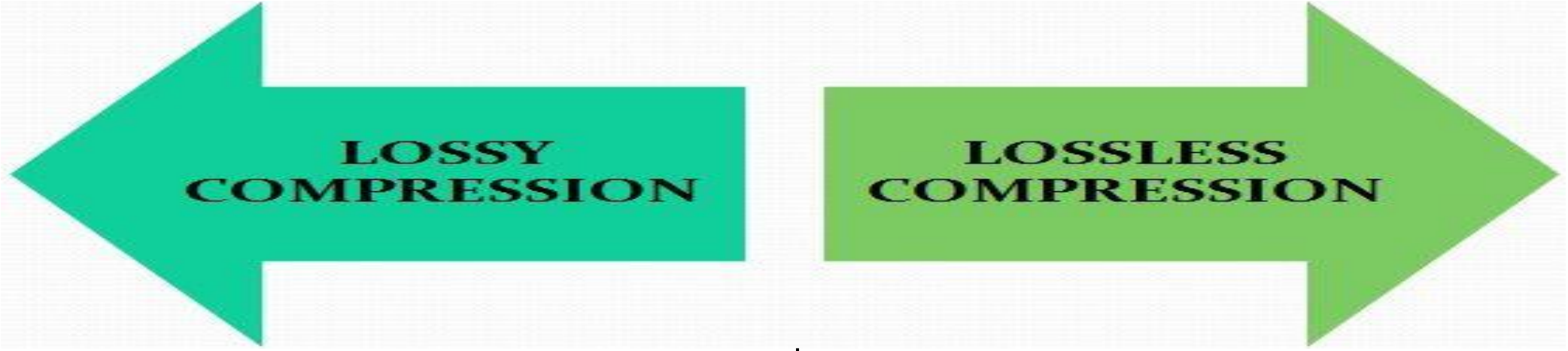
- **Example:** a representation that compresses a 10 MB file to 2MB has a compression ratio of 10/2 = 5, often notated as an exploit ratio , 5:1

- **Saving Percentage:** is defined as the reduction in size relative to the uncompressed size.

$$\text{Space Savings} = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}}$$

- **Example:** a representation that compresses a 10 MB file to 2 MB would yield a space saving of 1-(2/10) = 0.8 often notated as a percentage, 80 %

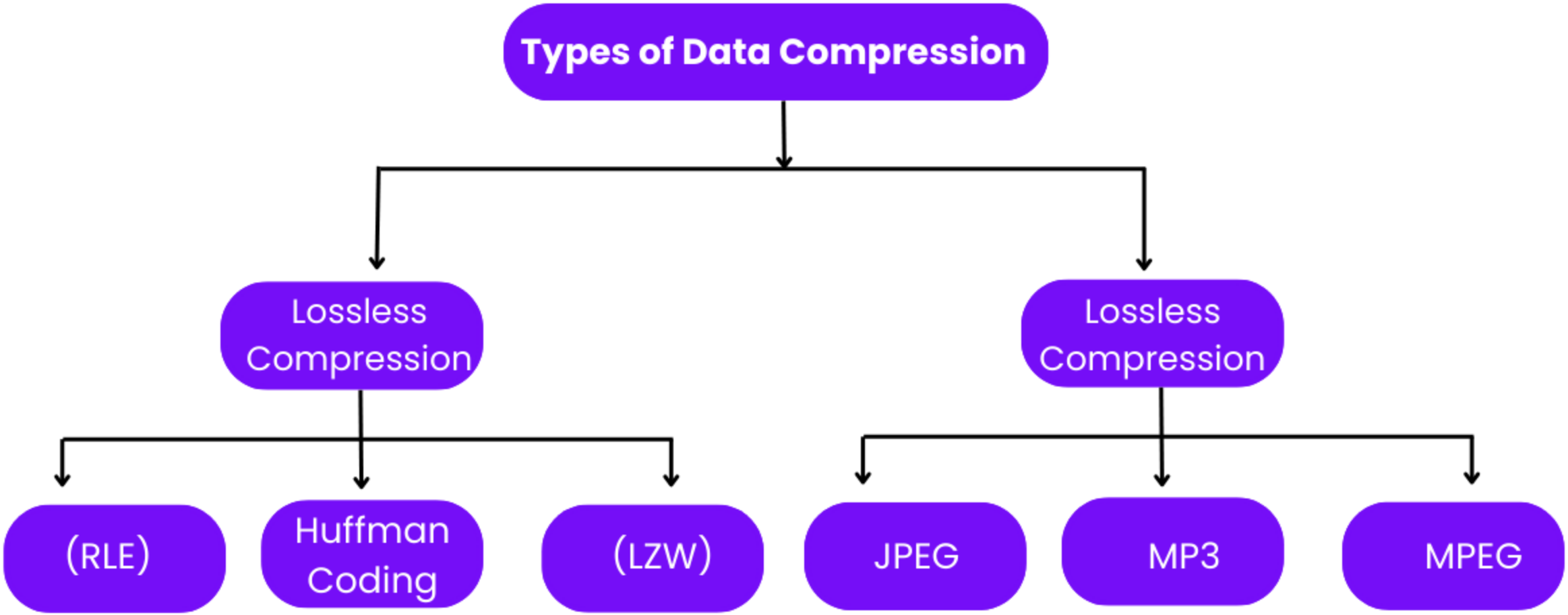
Types of Data Compression



- lossy compression, the data in a file is removed and not restored to its original form after decompression
- data is permanently removed, which is why this method is also known as irreversible compression.

- Lossless compression restores and rebuilds file data in its original form after the file is decompressed.
- The file can be decompressed to its original quality without any loss of data. This compression method is also known as *reversible compression*.

Types of Data Compression



Lossless Compression: Run Length Code (RLC)

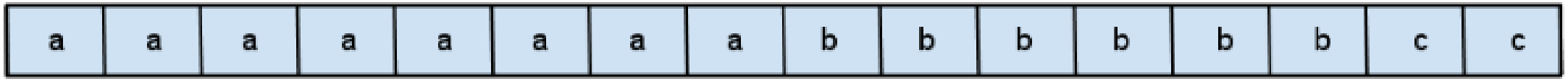
a	a	a	a	a	a	a	a	b	b	b	b	b	b	c	c

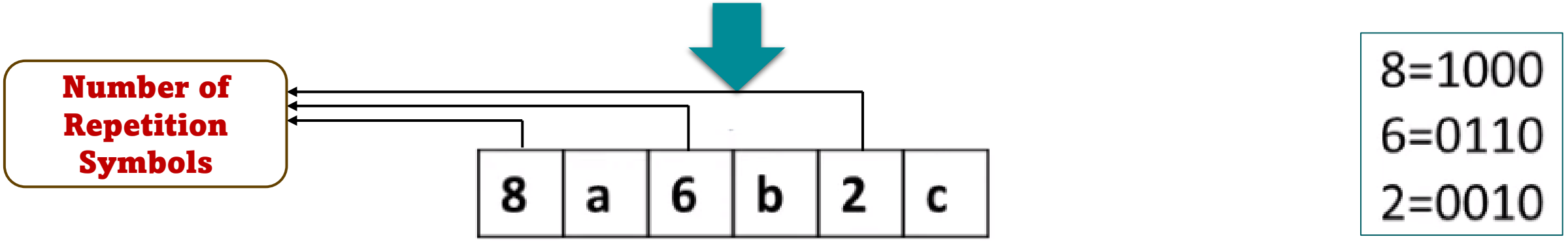
$a=(97)_{10}=01100001$
$b=(98)_{10}=01100010$
$c=(99)_{10}=01100011$

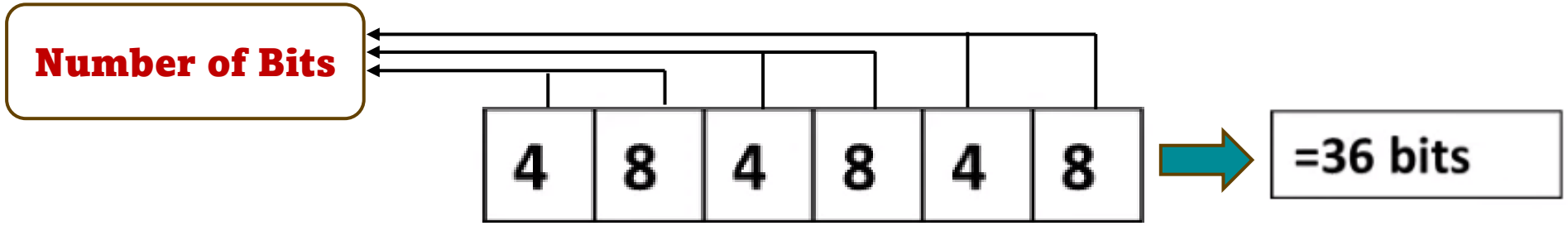
Size= 16 *8 bits=128 bits

Binary	Decimal	Character	Binary	Decimal	Character
01000001	65	A	01100001	97	a
01000010	66	B	01100010	98	b
01000011	67	C	01100011	99	c
01000100	68	D	01100100	100	d
01000101	69	E	01100101	101	e
01000110	70	F	01100110	102	f
01000111	71	G	01100111	103	g
01001000	72	H	01101000	104	h
01001001	73	I	01101001	105	i
01001010	74	J	01101010	106	j

Run Length Code (RLC)







120 Bits (original) → 36 Bits (Compressed)

Example:

Use the RLE method to compress the following sequence of symbols :

AAAACCBBBDDDDDDEFF

then calculate the compression ratio?

- Answer:

AAAACCBBBDDDDDDEFF

4A 2C 3B 5D 1E 2F

Original size= 17*8=136 bits
Compressed size=6*3+6*8=66 bits

Decoding: **4A 2C 3B 5D 1E 2F**

Example:

Use the RLE method to compress the following sequence of symbols :

000000111111111111110000000000000111111111

Answer:

000000111111111111110000000000000111111111

6 14 13 9 =42 bits

0	6	1	14	0	13	1	9

14=1110
13=1101
9=1001
6=0110

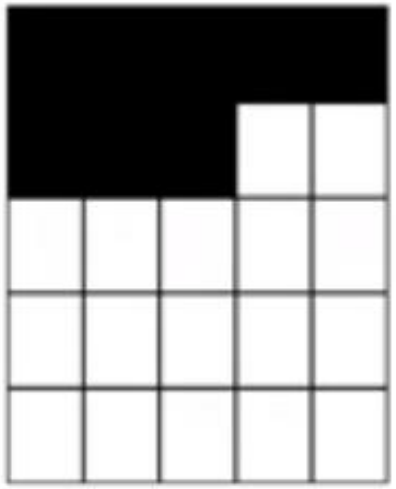
No of bits=	1	4	1	4	1	4	1	4	=20 bits

0	0110	1	1110	0	1101	1	1001

00110111100110111001

Example:

- Use the RLE method to compress the following binary image:



0	0	0	0	0
0	0	0	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Pixel=1 bit=0 or 1

Image size =5 *5= 25 bits

0	5			=4 bits
0	3	1	2	=8 bits
1	5			=4 bits
1	5			=4 bits
1	5			=4 bits

Image size=24 bits

Lossless Compression: Huffman Coding

What is Huffman Coding?

Huffman coding is a greedy algorithm frequently used for lossless data compression. The basic principle of Huffman coding is to compress and encode the text or the data depending on the frequency of the characters in the text.

Huffman Coding Example

- Let us understand how Huffman coding works with the example below: Consider the following input text:

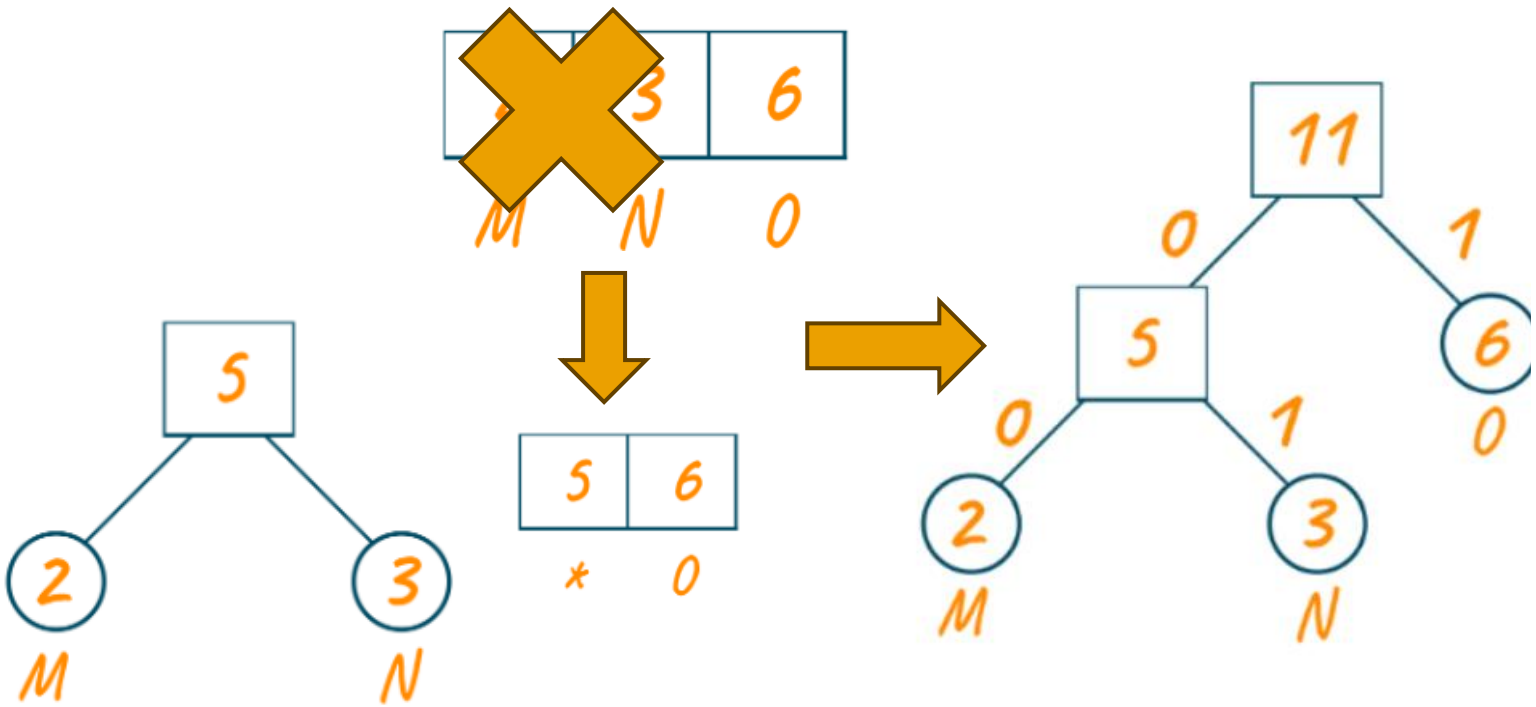
M	N	O	N	N	M	O	O	O	O

Step 1: Looking at the text, the frequencies of the characters will be as shown in the below image.

6	2	3
O	M	N

Step 2: we will sort the frequencies string of the characters in increasing order.

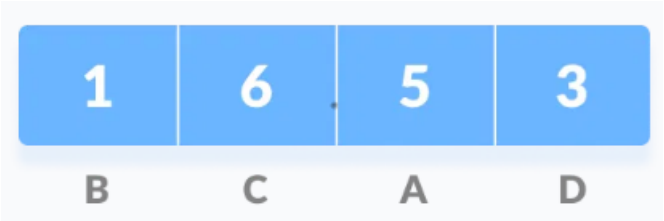
Step 3: we will create the Huffman tree using this priority queue.



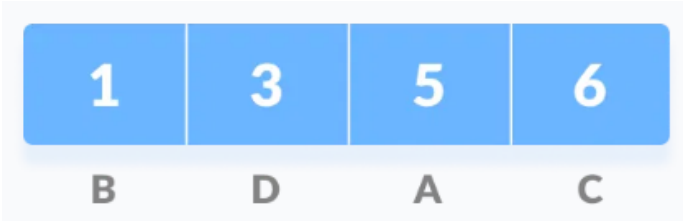
Character	Frequency	Code	Size
M	2	00	2*2 = 4
N	3	01	3*2 = 6
O	6	1	6*1 = 6
3*8 = 24 bits	11 bits		16 bits

Example:

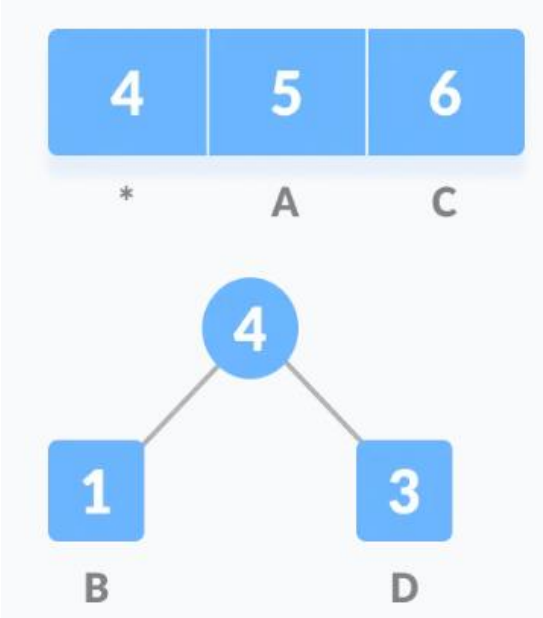
Step 1



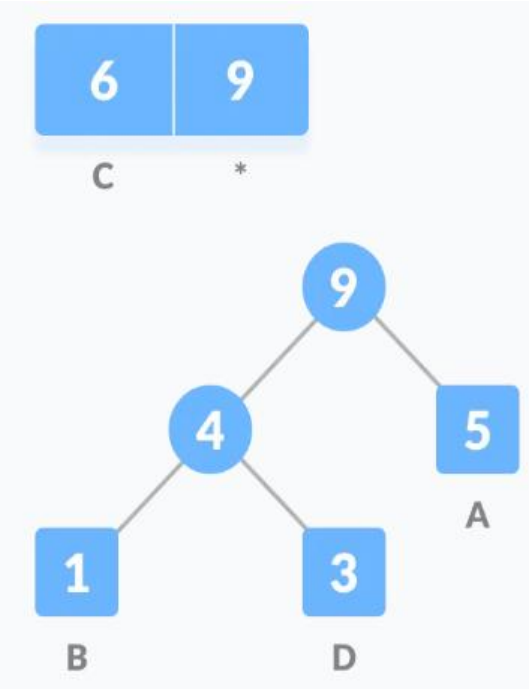
Step 2



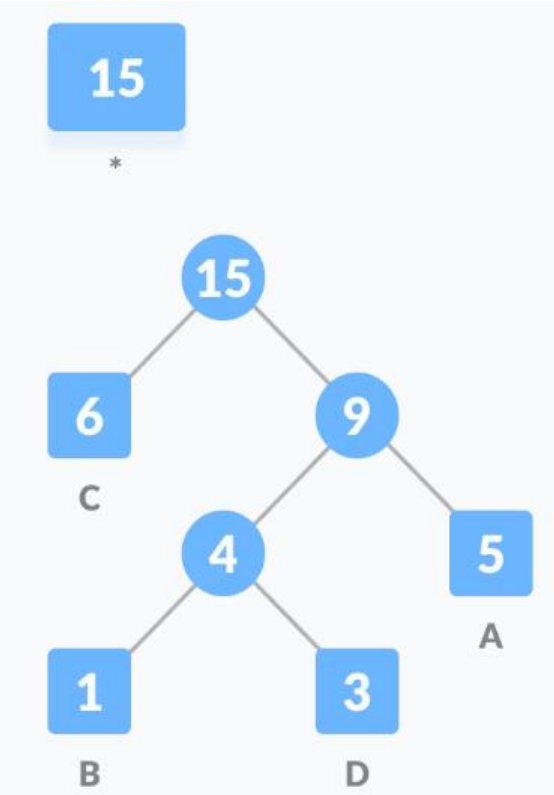
Step 3

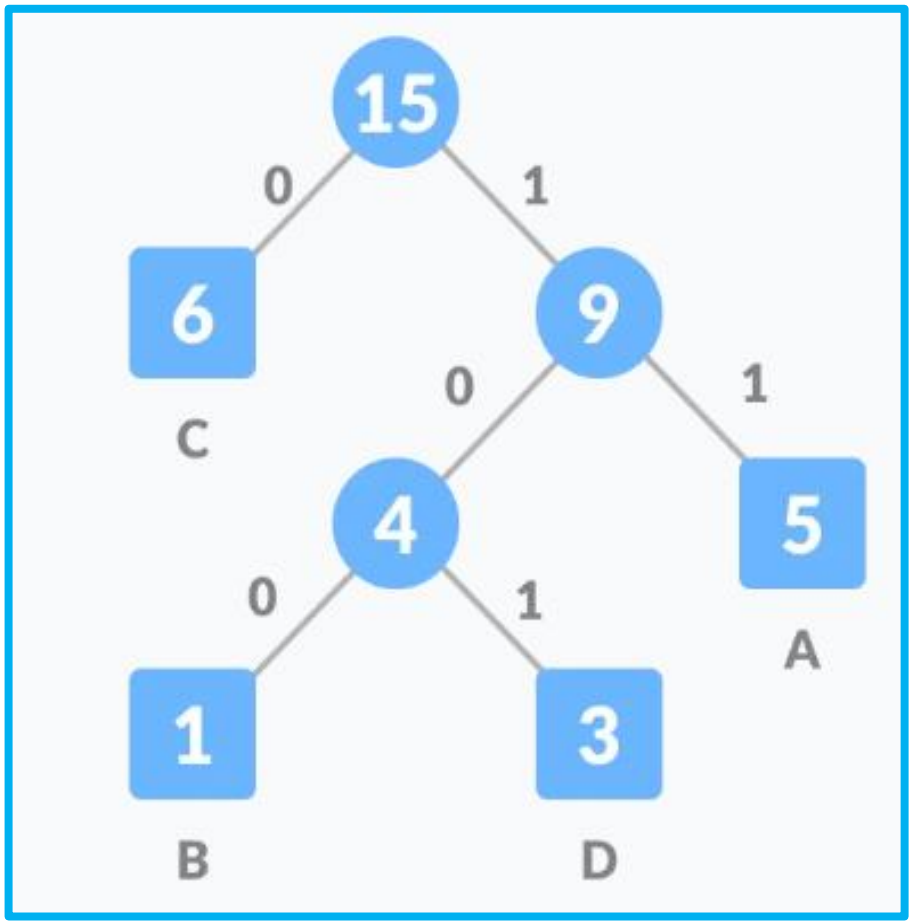


Step 4



Step 5





Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

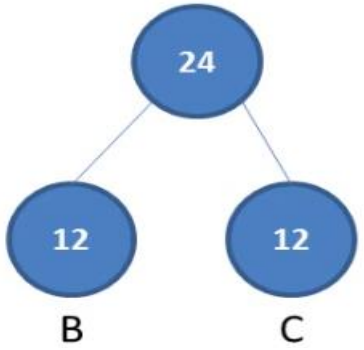
Example:

A	B	C	D	E
17	12	12	27	32

Step 1

B	C	A	D	E
12	12	17	27	32

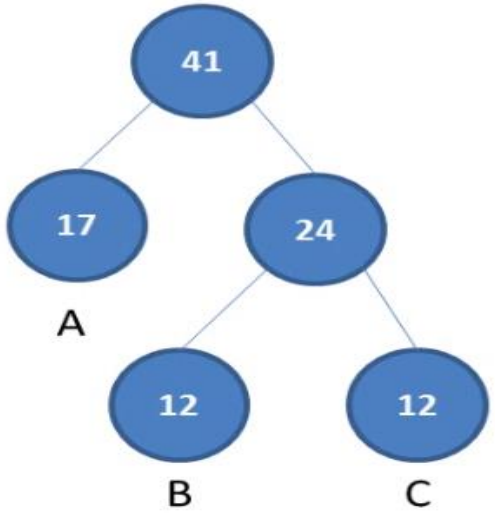
Step 2



.	A	D	E
24	17	27	32

Step 3

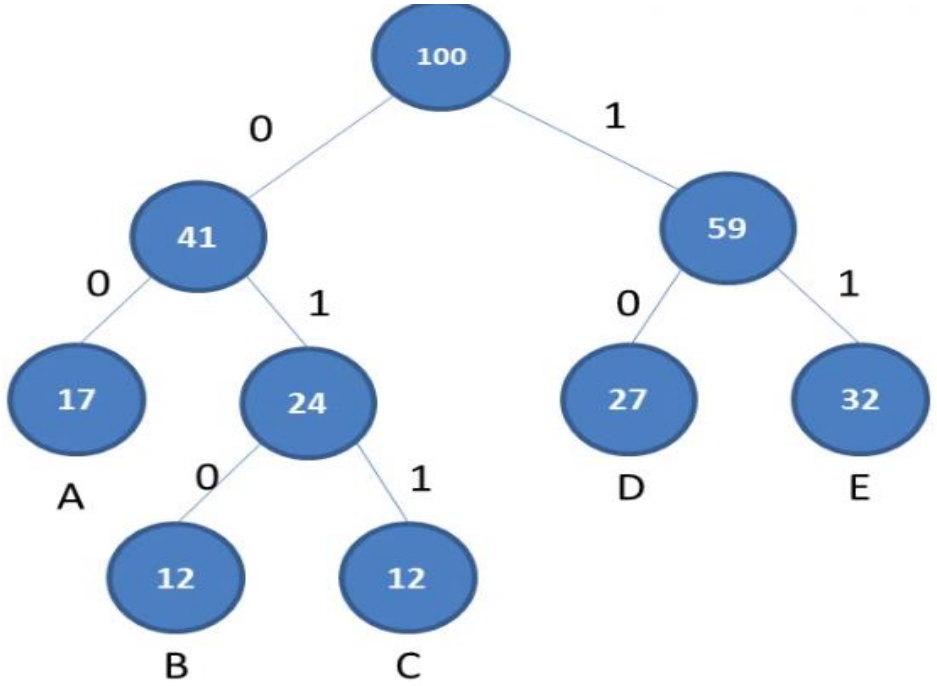
A	.	D	E
17	24	27	32

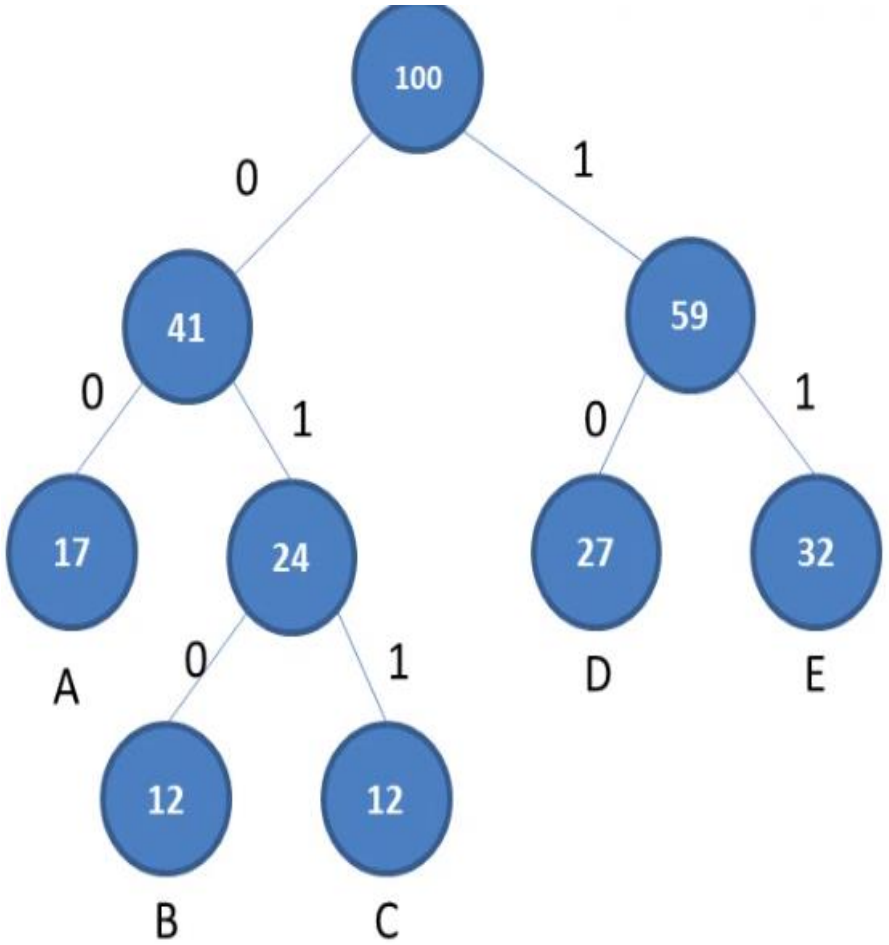


.	D	E
41	27	32

Step 4

D	E	.
27	32	41





A	B	C	D	E
00	010	011	10	11
3	1	1	1	4

	EAEBAECDEA
Size before compression	10*8bits= 80 bits
Size after compression	3*2+1*3+1*3+1*2+4*2= 22 bits