UNIT -II

Text compression

It is the process of reducing the amount of data needed for the storage or transmission of a given piece of information.

A technique known as compression is first applied to the source information prior to its transmission. This is done either to reduce the volume of information to be transmitted- text, fax and images or to reduce the bandwidth required for its transmission-speech, audio and video.

Compression principle

- source encoders and destination decoders.
- lossless and lossy compression.
- Entropy encoding
- Source encoding

The basic principles of text compression are set out to achieve a reduction in file size by encoding data more efficiently.

source encoders and destination decoders

Prior to transmitting the source information relating to a multimedia application , a compression algorithm is applied to it.

This implies that in order for destination to reproduce the original source information , a matching decompression algorithm must be applied to it.

The application of the compression algorithm is the main function carried out by the source encoder and the decompression algorithm is carried out by the destination decoder.

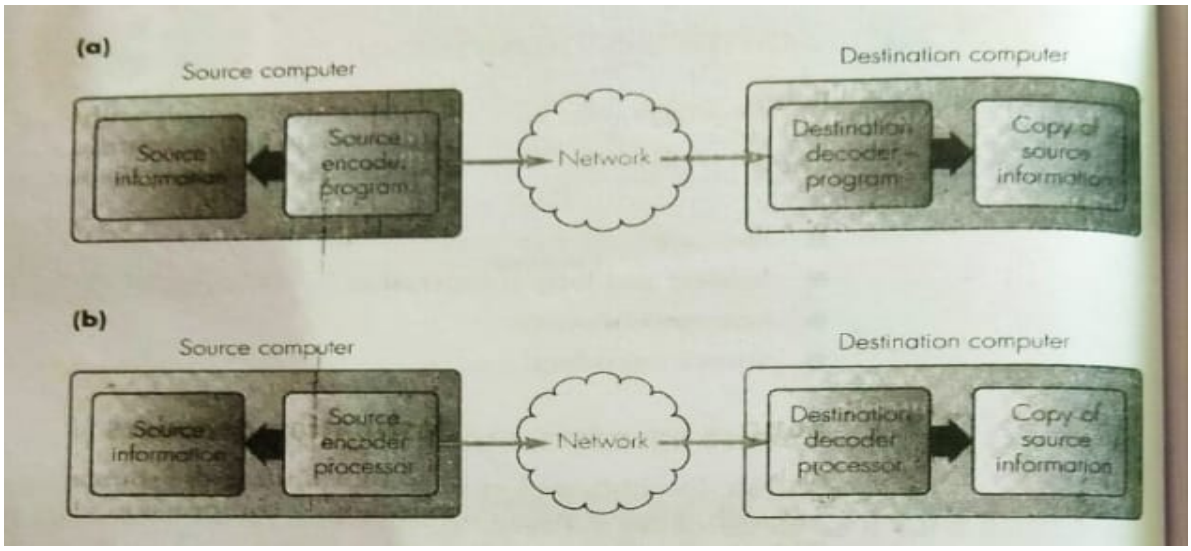


Fig. source encoder and destination decoder

(a) Software only (b) special processor/hardware

Lossless and Lossy compression

Lossless compression :-In the case of lossless compression algorithm the aim is to reduce the amount of source information to be transmitted in such a way that, when the compressed information is decompressed, there is no loss of information. Lossless compression is said to be reversible. An example of lossless compression is for the transfer over a network of a text file.

Lossless compression algorithms use statistic modelling technique to reduce the repetitive information in a file. Some of the methods may include removal of spacing character, representing a string of repeated character with a single character or replacing recurring characters with smaller bit sequences.

Lossy compression:- the aim of lossy compression algorithm is normally not to reproduce an exact copy of the source information after decompression but rather a version of it which is perceived by the recipient as a true copy. example of lossy compression are for the transfer of digitized images, audio, and video streams. Any fine details that may be missing from the original source signal after decompression are not detectable.

Entropy encoding

Entropy encoding is lossless and independent of the type of information that is being compressed. It creates and assigns a unique prefix code to each unique symbol that occur in the input.

Run length encoding

This type of encoding is used when the source information comprises long substrings of the same character or binary digit. In this the source string is transmitted as a different set of codewords which indicates not only the particular character or bit being transmitted but also the number of characters or bits in the substring. Then providing the destination knows the set of codewords being used, it simply interprets each codeword received and outputs the appropriate number of characters/bits

e.g. output from a scanner in a Fax Machine

000000011111111110000011

will be represented as

0,7 1,10 0,5 1,2

If we ensure the first substring always comprises **0s** , then the string will be represented as **7,10,5,2.**

Statistical encoding

A set of ASCII codewords are often used for the transmission of strings of characters.

However, the symbols and hence the codewords in the source information does not occur with the same frequency.

E.g. in string of text **A** may occur more frequently than **P** which may occur more frequently than **Z** and so on. Therefore the **statistical encoding** uses a set of variable length codewords with the shortest codeword used to represent the most frequently appearing symbol.

It is necessary to ensure that a shorter codeword in the set does not form the start of longer codeword otherwise the decoder will interpret the string on the wrong codeword boundaries. To avoid this a codeword must possess the prefix property. Example of this property is **Huffman encoding algorithm**.

Source encoding

Differential encoding

This type of coding is used where the amplitude of a value/signal covers a large range but difference in ampltitude between successive value/signals is relatively small.

Instead of using a set of large codewords to represent the amplitude of each value/signal, a set of smaller codeword can be used which indicate the difference in ampltitude between current value/signal being encoded and immediately preceding value/signal.

Eg if the digitization of analog signal requires **12 bits** but the maximum difference in amplitude between successive sample of the signal requires only **3 bits,** then by using only the difference values a saving of **75%** on transmission bandwidth can be obtained.

It can be either lossless or lossy.

If the number of bits is sufficient to cater for the maximum difference value then it is lossless otherwise it is lossy.

Transform encoding

It involves transforming the source information from one form to another, the other form lending itself more readily to the application of compression.

The transform is typically lossless.

This technique is used in a number of application including images and video.

It is used to convert spatial image pixel values to transform coefficient values.

The digitization of a continuous monochromatic image produces a two dimensional matrix. As we go from one position in the matrix to the next the magnitude of each pixel value may vary. The rate of change in magnitude as one transverse the matrix give rise to term known as **spatial frequency.**

For a particular image there will be mix of different spatial frequencies whose amplitude are determined by the related change in the magnitude of the pixel.

If we scan the matrix in either horizontal or vertical direction, it gives rise to the terms **horizontal** and **vertical frequency** components of the image.

Discrete cosine transform technique is used for the transformation of two-dimensional matrix of pixel values into an equivalent matrix of spatial frequency components

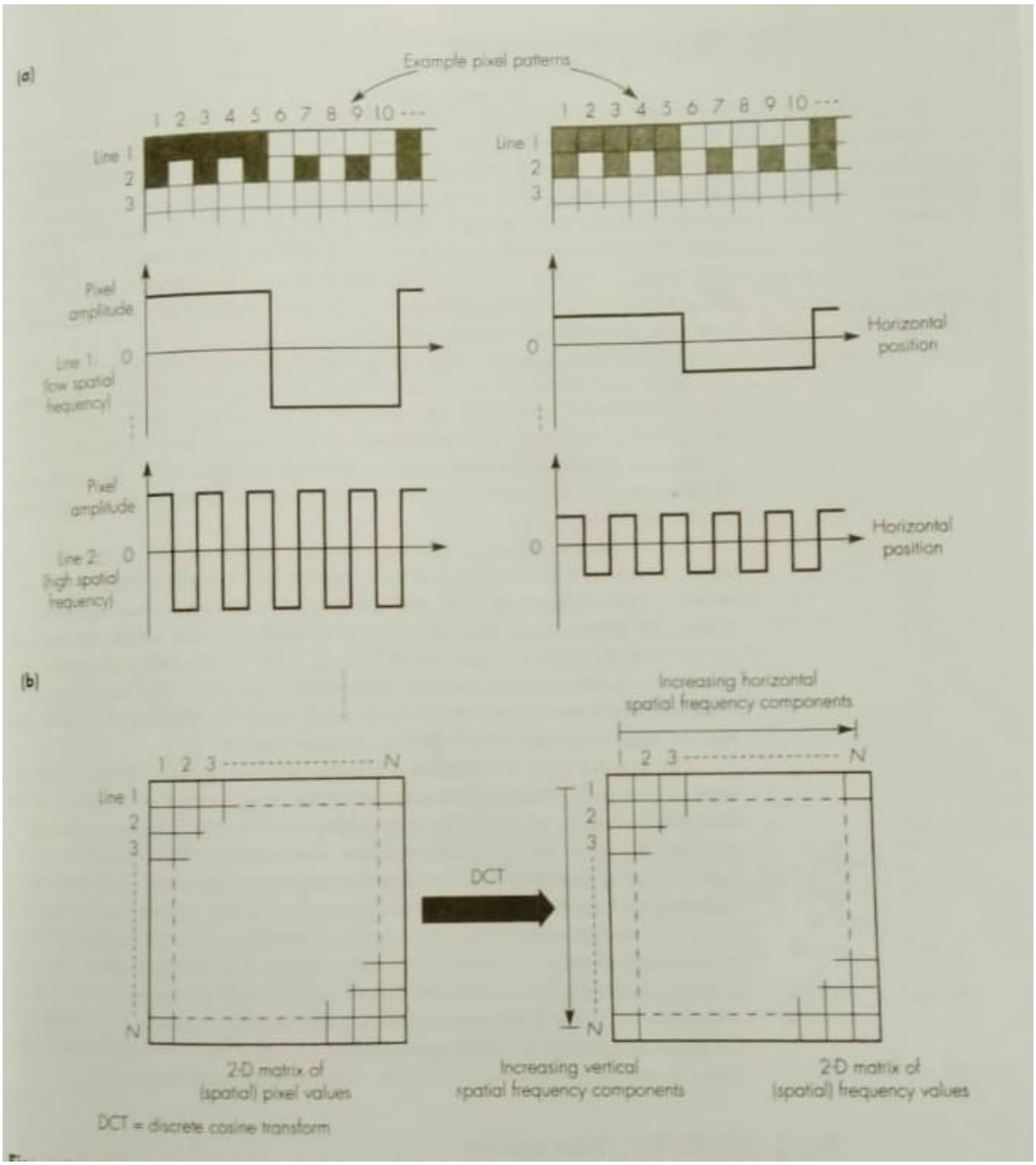
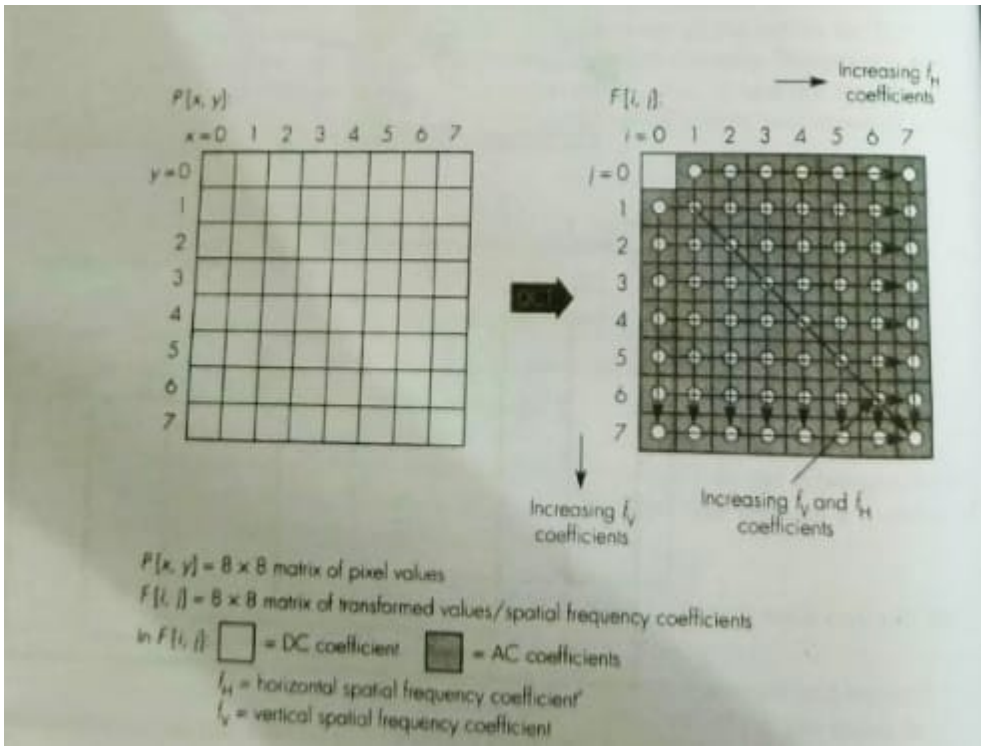


Fig. transform coding

(a) Pixal pattern (b) DCT transform priniciples



DCT transform principle

Text compression technique

Static Huffman coding

The character string to be transmitted is first analysed and the character types and their relative frequency is determined.

The coding operation involves creating an unbalanced tree with some branches shorter than the other. The degree of imbalance is function of relative frequency of occurrence of the characters.

The resulting tree is known as **Huffman code tree**. It is a binary tree with branches assigned the value 0 or 1, binary 0 for the left branch and binary 1 for the right branch.

The base of the tree is known as **root node**.

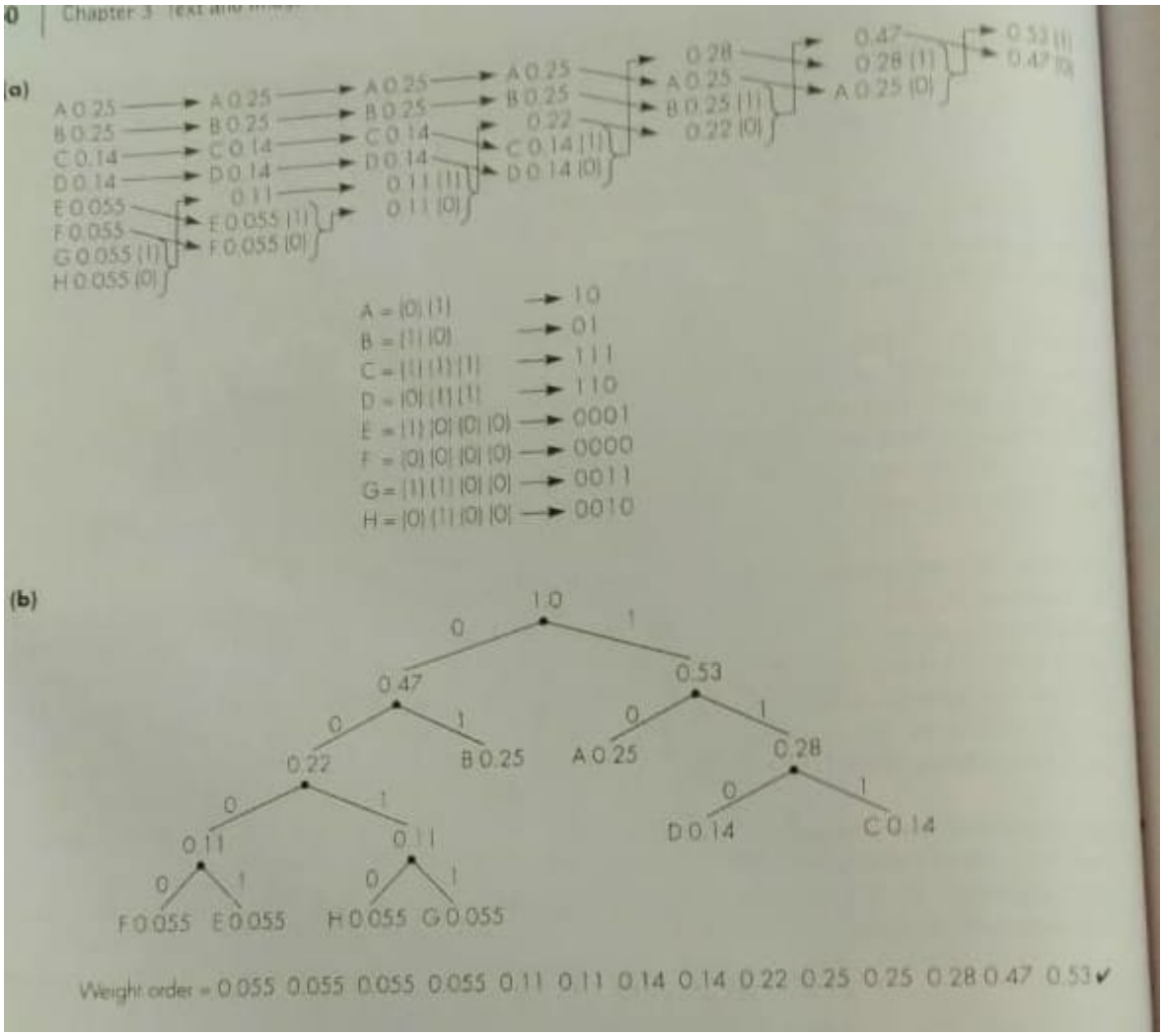
The point at which branch divides is known as **branch node**.

The termination point of a branch is known as **leaf node** to which the symbols being encoded are assigned.

The codewords used for each characters are determined by tracing the path from the root node out to each leaf and forming a string of binary

values associated with each branch traced.

Huffman encoding example



(a) Codeword generation (b) Huffman code tree

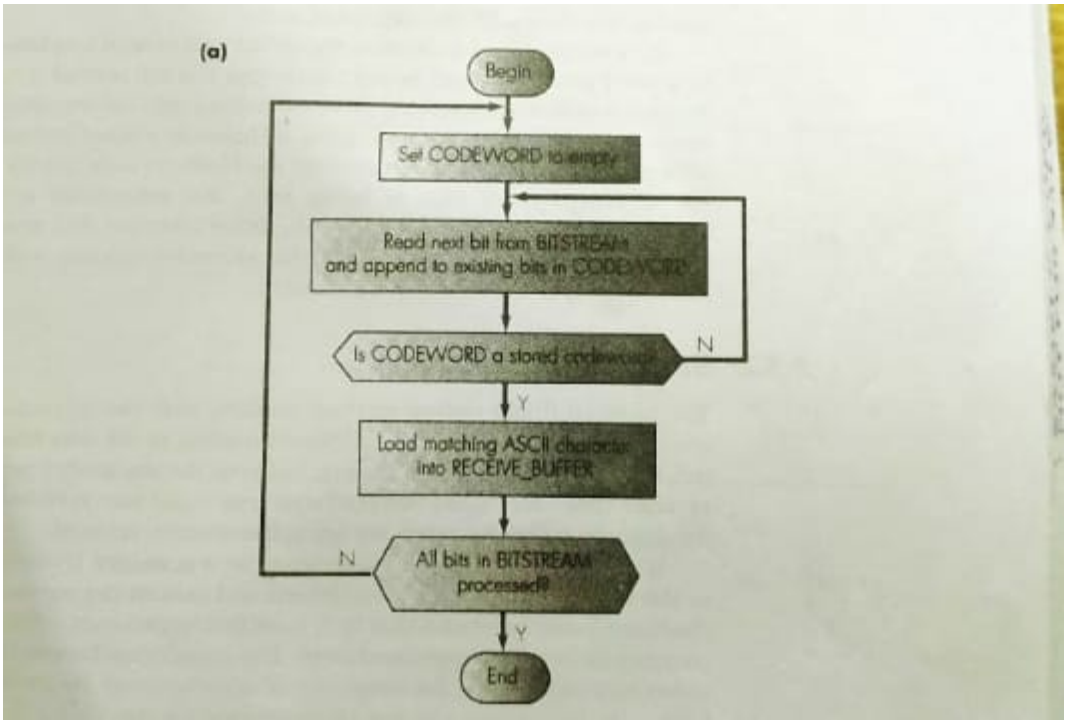
Huffman code tree varies for different set of characters being transmitted therefore receiver must known the codewords relating to the data being transmitted .

This is done in two ways, one is the codeword relating to the next data are sent before the data is transmitted and the second is the receiver knows in advance what codeword are being used.

A flowchart of a suitable decoding algorithm is shown in figure. The algorithm assumes a table of codewords is available at the receiver and this also holds the corresponding ASCII codeword.

The received bit stream is held in the variable BIT STREAM and the variable codeword is used to hold the bits in codeword while it is being

constructed. Once a codeword is identified the corresponding ASCII codeword is written into the variable RECEIVE BUFFER. The procedure repeats until all the bits in the received string have been processed.



Decoding algorithm

Dynamic Huffman coding

Static Huffman coding requires both the transmitter and the receiver to know the table of codewords relating to the data being transmitted. But with dynamic Huffman coding the transmitter and the receiver build the Huffman tree and hence the codeword table dynamically as the characters are being transmitted / received.

Encoder starts by reading the first character ,since tree is empty, it sends the first character in uncompressed form. On reception , since the decoder tree is also empty , it interprets the received bit string as an uncompressed character and proceeds to assign the character to its tree in the same way.

The encoder first checks whether the character is already present in the tree. If it is, then the encoder sends the current codeword for the character in normal way ,if it is not present then the encoder sends the

current codeword for the **empty leaf** .

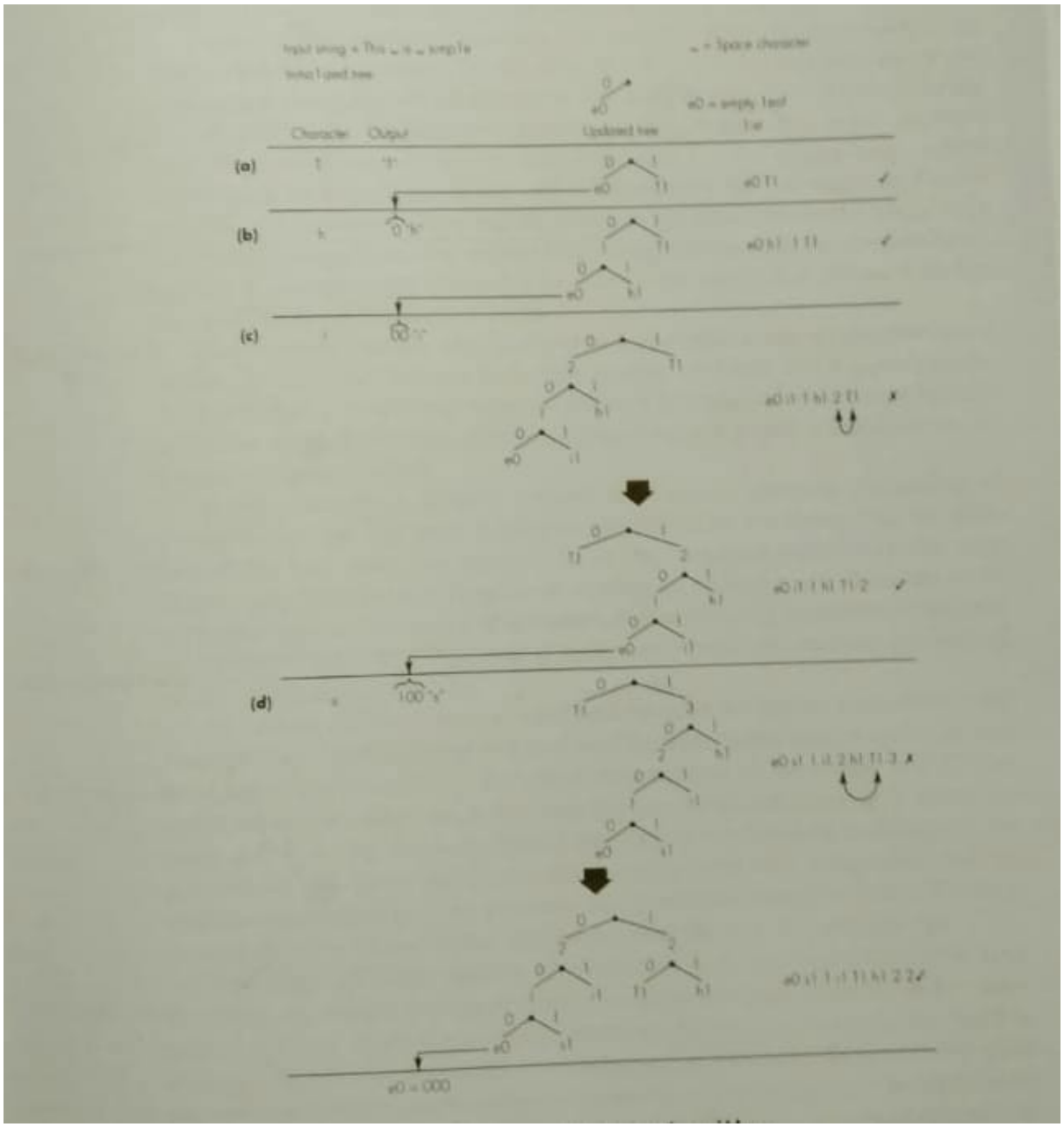
If the character is already present in the tree, then the frequency of occurrence of the leaf node is incremented by unity.

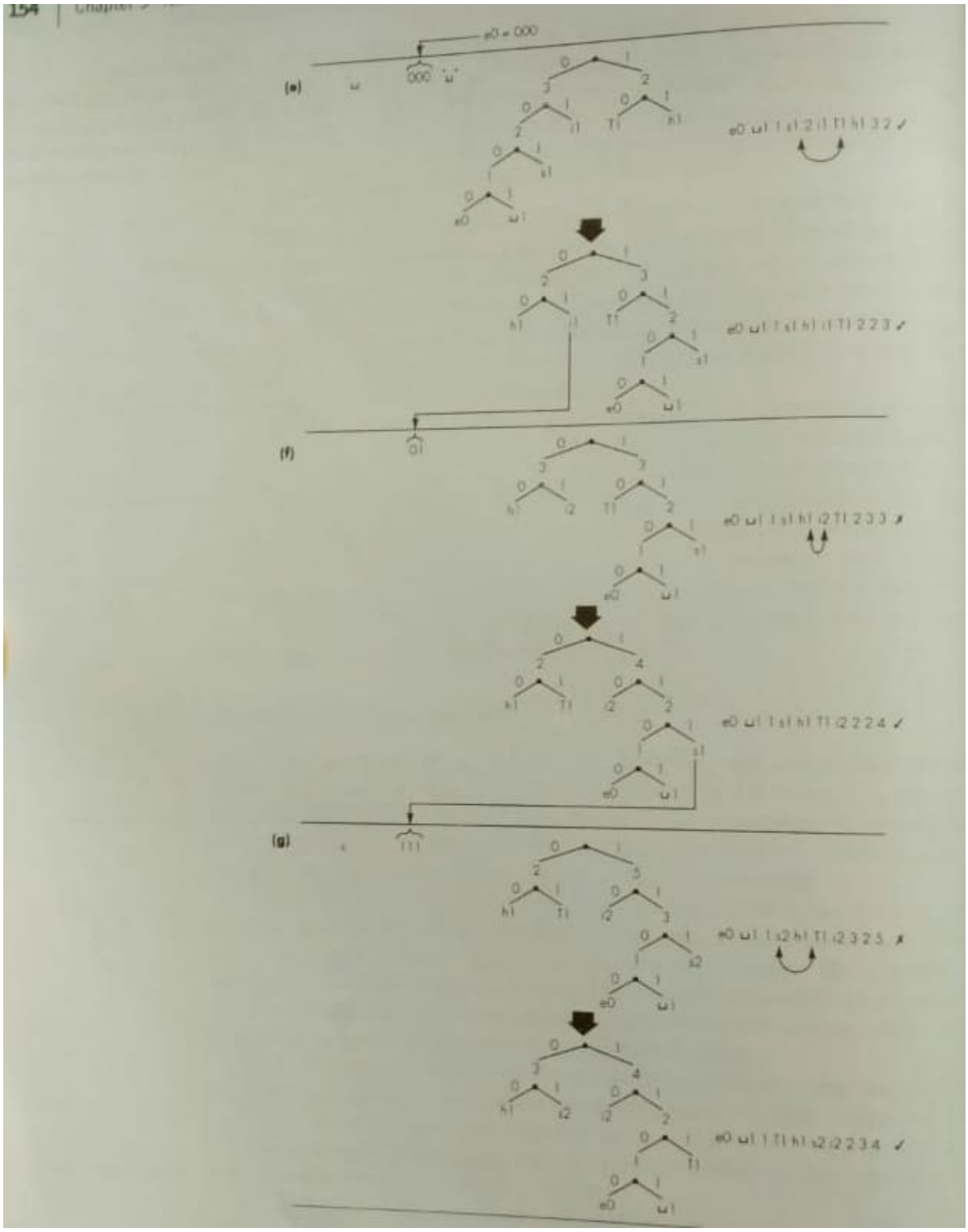
To ensure that both the encoder and decoder do in a consistent way , first list the weights of a leaf and branch in the updated tree from left to right and from bottom to top starting at the empty leaf. The tree is left unchanged , if they are all in weight order otherwise the structure of tree is modified by exchanging the position of the node with the other node in the tree.

Let us assume that the data to betransmitted starts with the following character string : **this is simple**

Both transmitter and receiver start with a tree that comprises the root node and a single empty leaf node (a leaf node with zero frequency of occurrence assigned to its zero(0) branch.

There is always one empty leaf node in the tree and its position and codeword varies as the tree is begin constructed.


Input string = This ␣ is ␣ simple
Initial coded tree
␣ = Space character
Character
Output
Updated tree
(a)
(b)
(c)
(d)
100 "s"
e0 = 000



Arithmetic coding

Huffman coding uses a separate codeword for each character whereas arithmetic coding yields a single codeword for each encoded string of charcaters. **Arithmetic coding** encodes data by creating a code string

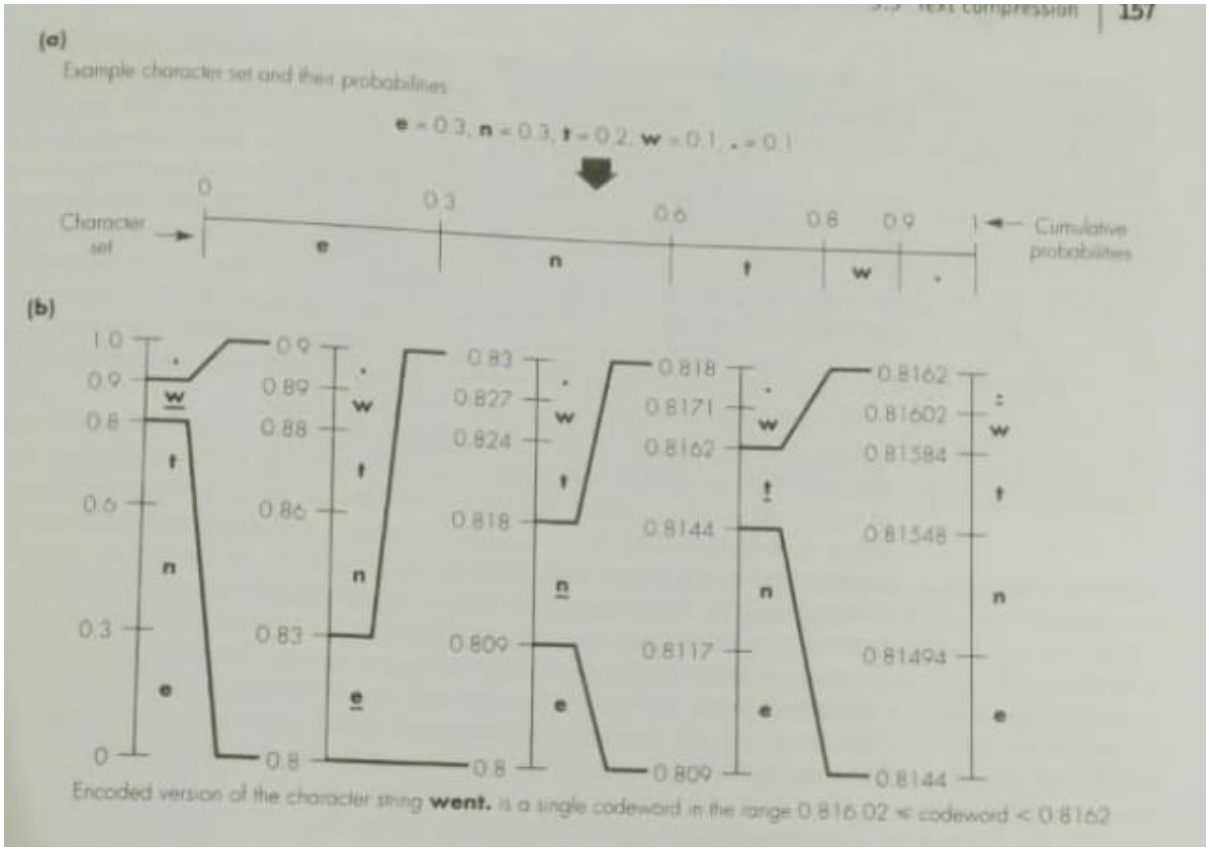
which represent a fractional value on the number line between 0 and 1.

The first step is to divide the numeric range from 0 to 1 into a number of different characters present in the message to be sent and the size of the each segment by the probability of the related character.

Eg. Consider the transmission of a message comprising a string of characters with probabilities of

e=0.3, n=0.3,t=0.2,w=0.1, .=0.1

we assume the message to be encoded is the single word **went.** .



Arithmetic coding principle

the character **e** has the probability of **0.3** and hence assigned the range from **0.0 to 0.3**, the character **n** also has a probability of 0.3 , the range from **0.3 to 0.6** , the character **t** has probability of **0.2** and hence the range from **0.6 to 0.8** and so on.

The first character to be encoded **w** is in the range from **0.8 to 0.9** hence the range **0.8 to 0.9** is itself subdivided into five further segments, the width of each segment again determined by the probability of the five

characters.

Range of symbol =lower limit :lower limit + d * probability of symbol

Where **d(difference) = upper limit – lower limit**

0.8 to 0.9 range the upper limit is 0.9 and lower limit is 0.8 so the difference d is 0.1

Next **range of symbol= 0.8 : 0.8+ 0.1(0.3)**

i.e from **0.8 to 0.83**

this procedure continue until the termination character . is encoded.

At this point, the segment range of . is from **0.81602 to 0.812**

Hence the codeword for the complete string is any number within the range

0.81602≤ codeword >0.8162

The decoder can follow the same procedure as that followed by the encoder to determine the character string relating to each received codeword.

If the received codeword is **0.8161**, then decoder can determine from this that te first character is **w (** because its range is **0.8 to 0.9**)

This procedure then repeats until it decodes the termination character.

Lempel-ziv (LZ) coding

It uses strings of characters, instead of using a single characters as the basis of coding operation. Eg. In text compression, a table containing all the possible character strings that occur in the text to be transferred held by both the encoder and decoder.

Instead of sending the word as a set of individual (**ASCII**) codeword the encoder sends only the index of where the word is stored in the table and on receipt of each index, the decoder uses this to access the corresponding word/string of characters from the table and proceeds to reconstruct text into its original form.

Thus the table is used as a dictionary and the algorithm is known as a **dictionary-based compression algorithm.**

The basic requirement with the **LZ algorithm** is that a copy of the dictionary is held by both the encoder and the decoder.

It can be relatively inefficient if the text to be transmitted comprises only a small subset of the words stored in the dictionary.

Hence a variation of the **LZ algorithm** has been developed which allow the dictionary to be built up dynamically by the encoder and the decoder as the compressed text is being transferred.

Most word –processing packages have a dictionary associated with them which is used for both spell checking and for the compression of text , they contain in the region of **25,000 words** and hence **15 bits (32768 combination**) are required to encode the index

Eg. to send the word '**multimedia'** dictionary require just **15 bits** instead of **70 bits** with **7-bit ASCII codeword**.

Shorter words will have a lower compression ratio as compare to longer words

Lempel –ziv-walsh (LZW) coding

The principle of the **Lempel –ziv-walsh(LZW)** coding algorithm is for the encoder and the decoder to build the contents of the dictionary dynamically as the text is being transferred.

Initially the dictionary held by both the encoder and the decoder contains only the character set (i.e **ASCII)** that has been used to create the text.

The remaining entries in the dictionary are then build up dynamically by both the encoder and decoder and contain the words that occur in the text.

Eg. If the character set comprises **128 characters** and the dictionary is limited to **4096 entries** , then the first **128 entries** would contains the single characters that make up the character set and the remaining **3968 entries** would each contain strings of two or more characters that make up the words in the text being transferred.

Eg. The text to be compressed starts with string:

This is simple as it is…..

Initially the dictionary held by both encoder and decoder contains only

the individual character from the character set being used (128 character in the ASCII character set)

Hence the first word in text is sent by the encoder using the index of each of the four characters T, h, i, s.

When encoder reads the next character from the string (first space character) ,it determines that this is an alphanumeric characters therefore it transmits the character using index as before, in addition to this it also interprets it as terminating the first word.

Similarly the decoder on receipt of the first codeword, read the character stored at each index and commence to reconstruct the text. When it determines that the fifth character is space character, it interprets this as a word delimiter and proceeds to store the word this in its dictionary.

The same procedure is followed by both the encoder and the decoder transferring the other words.

Figure LZW algorithm

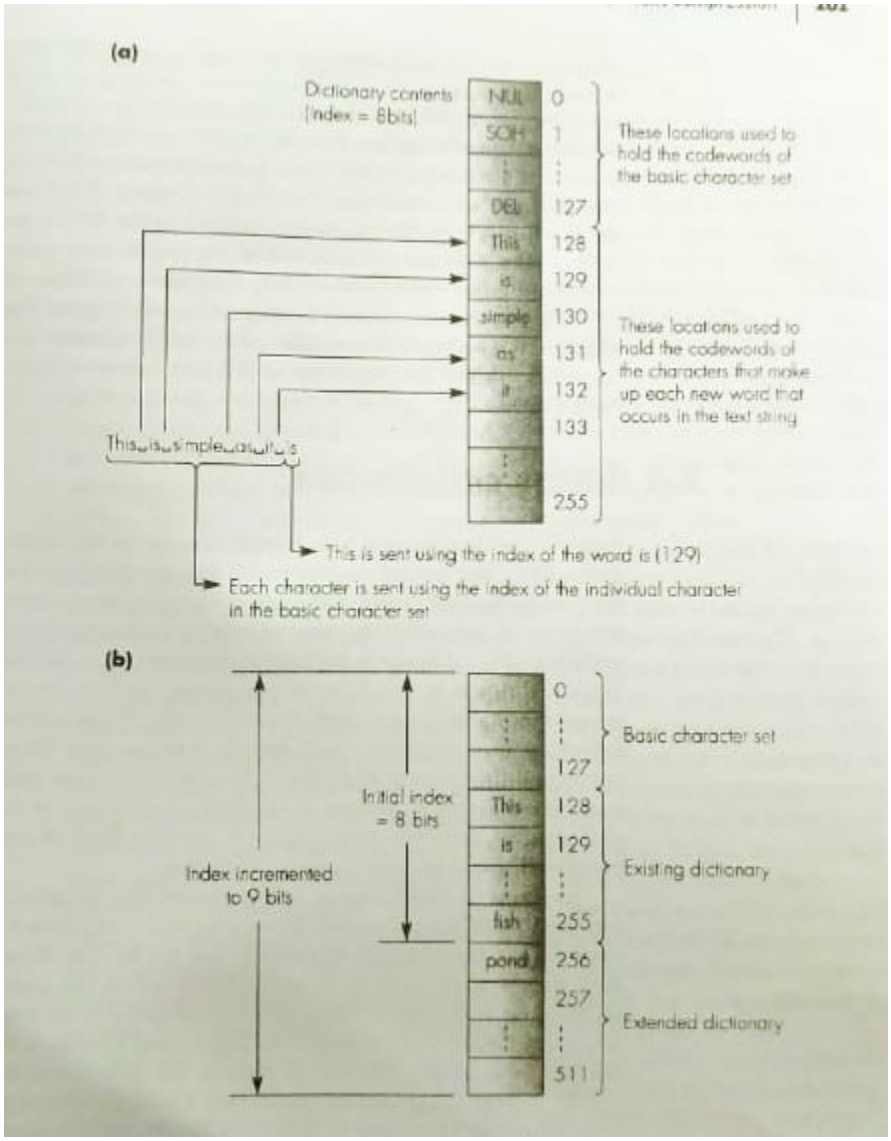
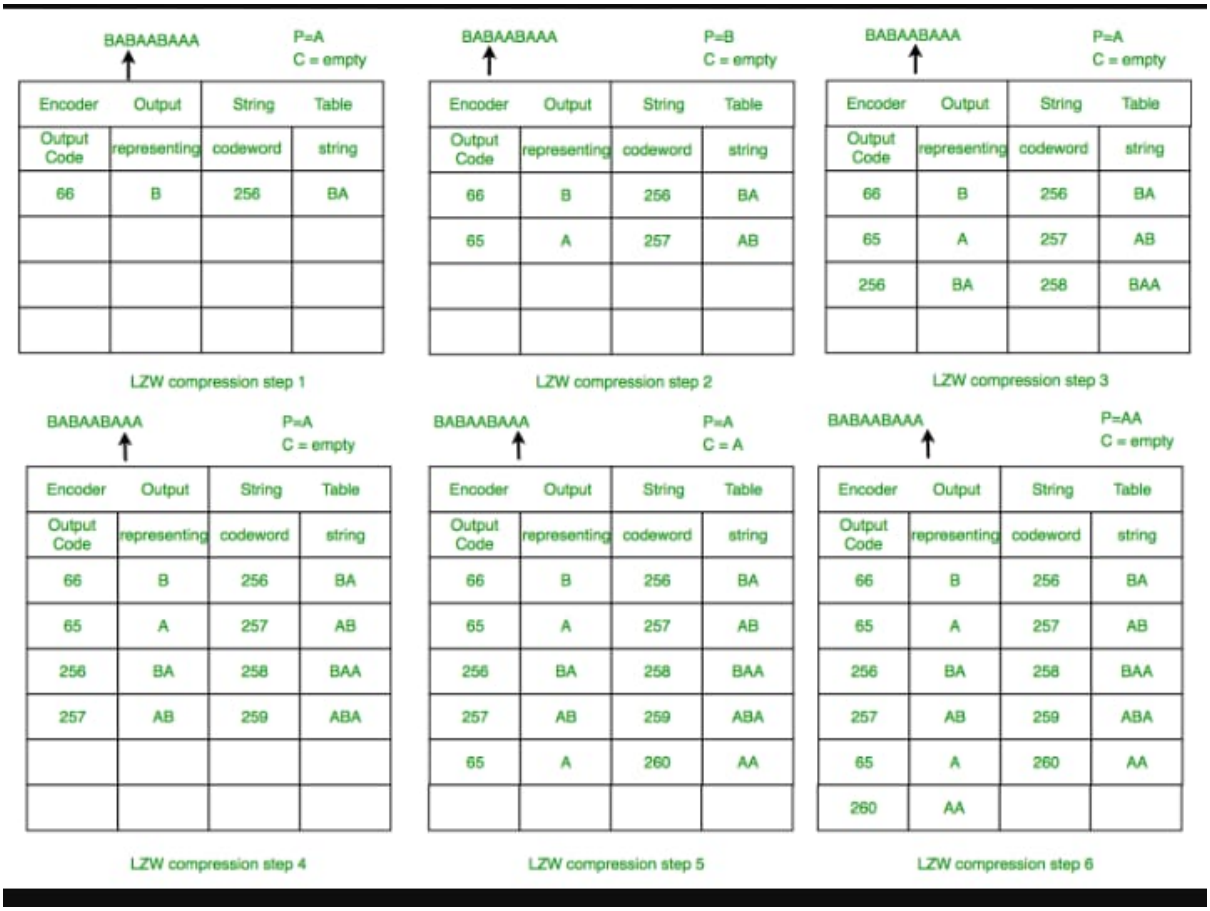


Fig. LZW compression algorithm (a) basic operation (b) dynamically extending the number of entries in the dictionary



With static dictionary the no. of entries is fixed . eg. The dictionary contains **25000 words** require **15 bits** to encode the index but in dynamically dictionary at the commencement of each transfer the no. of entries is set to a relatively low but when the available space become full, then the no. of entries is allowed to increase incrementally.