Reprint ISSN 0973-9424

INTERNATIONAL JOURNAL OF MATHEMATICAL SCIENCES AND ENGINEERING APPLICATIONS

(IJMSEA)

www.ascent-journals.com

International J. of Math. Sci. & Engg. Appls. (IJMSEA)
ISSN 0973-9424, Vol. 19 No. II (December 2025), pp. 15-26

Analysis of Lossless Text Compression using a Huffman Coding Technique

Mr. Juned Khatri
Research Scholar, Department of Mathematics,
SAGE University, Indore
Email id- junedkhatrijnt@gmail.com

Dr. Aarti Sharma
Associate Professor
Department of Mathematics,
SAGE University, Indore
Email id- sharma.aarti@sageuniversity.in

Abstract

Huffman coding is very popular in data compression area its coding is a great way to compress down and to able to shrink it down by bites and chunks. Huffman coding is a method of storing strings of data as binary code in an efficient manner and also Huffman coding uses "lossless data compression" which means no information is lost which you are coded. Compression is used about everywhere. Compression refers to reducing the size or quantity of data used to represent a file, image or video content without reducing the quality of the original data. It also reduces the number of bits required to store and/or transmit digital media. To compress something means that you have a piece of data, and you decrease its size. In this paper we proposed the lossless method of text data compression using a simple coding techniques called Huffman coding. This technique is simple in implementation and utilized less memory.

Keywords :- Huffman coding, Compression, Lossless compression.

Introduction :-

A text is a collection of characters or strings into a single unit. It contains many characters in it that always cause problems in limited storage device and speed of data transmission at the time [1]. Although storage can be replaced by another larger one, this is not a good solution if there is another solution. And this is making everyone try to think to find a way that can be used to compress text [2]. This analysis may be performed by comparing the measures of the compression and decompression.

Compression: -

Data compression is simply a means for efficient digital representation of a source of data such as image, sound and text. The aim of data compression is to represent a source in digital form with as few bits as possible while meeting the minimum demand of reconstruction. This thing is achieved by removing any redundancy present in the source. Compression techniques are two types: Lossy compression and Lossless compression.

Lossy compression: -

Lossy compression is based on the assumption that the current data files save more information than human being can perceive. In this compression some data is lost when it is decompressed [3].

Lossless compression: -

Lossless compression means that when the data is decompressed, the result is a bit-for-bit perfect match with the original one. The name lossless means "no data is lost", the data is only saved more efficiently in its compressed state, but nothing of it is removed [4].

Lossless compression is used in cases where it is important that the original and the decompressed data be identical, or where deviations from the original data would be unfavourable. Common examples are executable programs, text documents, and source code. Some image file formats, like PNG or GIF, use only lossless compression, while others like TIFF and MNG may use either lossless or lossy methods. Lossless audio formats are most often used for archiving or production purposes, while smaller lossy audio files are typically used on portable players and in other cases where storage space is limited or exact replication of the audio is unnecessary [5].

Methodology: -

Huffman coding :-

Huffman coding is a greedy approach based lossless data compression techniques invented by David Huffman. It used variable length encoding to compress the data. The main idea is Huffman coding is to assign each character with a variable length code[6].

To compress something means that you have a piece of data or object and you decrease its size. there different techniques and they all have their own advantages and disadvantages. Huffman coding is based on the frequently of occurrence of a data item i.e. pixel in images. The technique is to use a lower number of bits to encode the data into binary codes that occurs more frequently. these codes are of variable code length using integral number of bits. It is used in JPEG files[7].

Compression Ratio (CR): -

Compression ratio (CR) is defined as number of bits to represent the size of original image to the number of bits to represent the size of compressed image. Compression ratio shows that how much time the image has been compressed [8]. It is calculated using following formula-

$$\textbf{Compression ratio} = \frac{\textbf{original size}}{\textbf{compressed size}}$$

Compression gained: -

Compression gain tells us how much size was reduced during compression [9]. It is usually expressed as a percentage and calculated with the formula-

$$\textbf{Compression gain} = \frac{\textbf{original size} - \textbf{compressed size}}{\textbf{original size}} \times 100$$

Advantage of data compression:-

- Less memory required.
- Byte order independent.
- Easy for file transfer.

Disadvantage of data compression:

- Need to decompress all previous data.
- Added complication.

Current research:

- Cancer imaging:- For dignose and be aware of the tumor.
- Brain Imaging: – For focuses on the normal and abnormal development of brain.
- Computer aided detection in mammography.
- Magnetic resonance imaging in low back pain.

Huffman Coding Algorithm: -

The Huffman Coding algorithm is used to implement lossless compression. From this example we will investigate how this algorithm can be implemented to encode/compress textual information.

The principle of this algorithm is to replace each character (symbols) of a piece of text with a unique binary code. However, the codes generated may have different lengths. To optimize the compression process, the idea behind the Huffman Coding approach is to associate longer codes to the less frequently used symbols and shorter codes to the most frequently used symbols.[10].

So, let's consider the following message which consists of 22 characters: -

S	A	G	E	—	U	N	I	V	E	R	S	I	T	Y	—	I	N	D	O	R	E

Note that the Huffman coding algorithm is normally used to compress larger amount of data and would not really be used to compress such a small message.

Step 1: Frequency Analysis: -

The first step of the Huffman Coding algorithm is to complete a frequency analysis of this message by:

- Identifying all the symbols used in the message,
- Identifying their weights (or frequencies) by counting their occurrences (the number of times they appear) within the message,
- Sorting this list of symbols in ascending order of their weights.

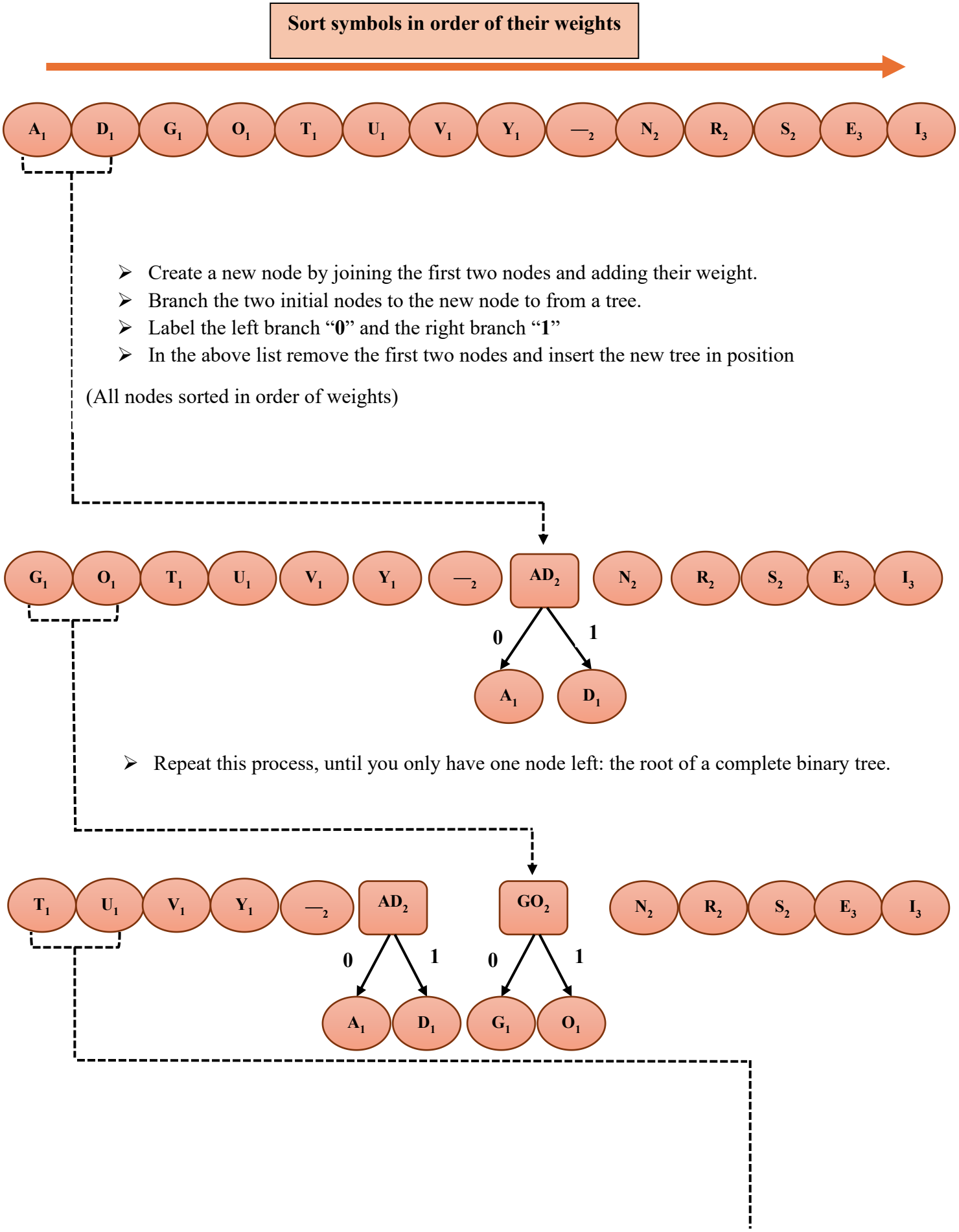
S	A	G	E	—	U	N	I	V	E	R	S	I	T	Y	—	I	N	D	O	R	E

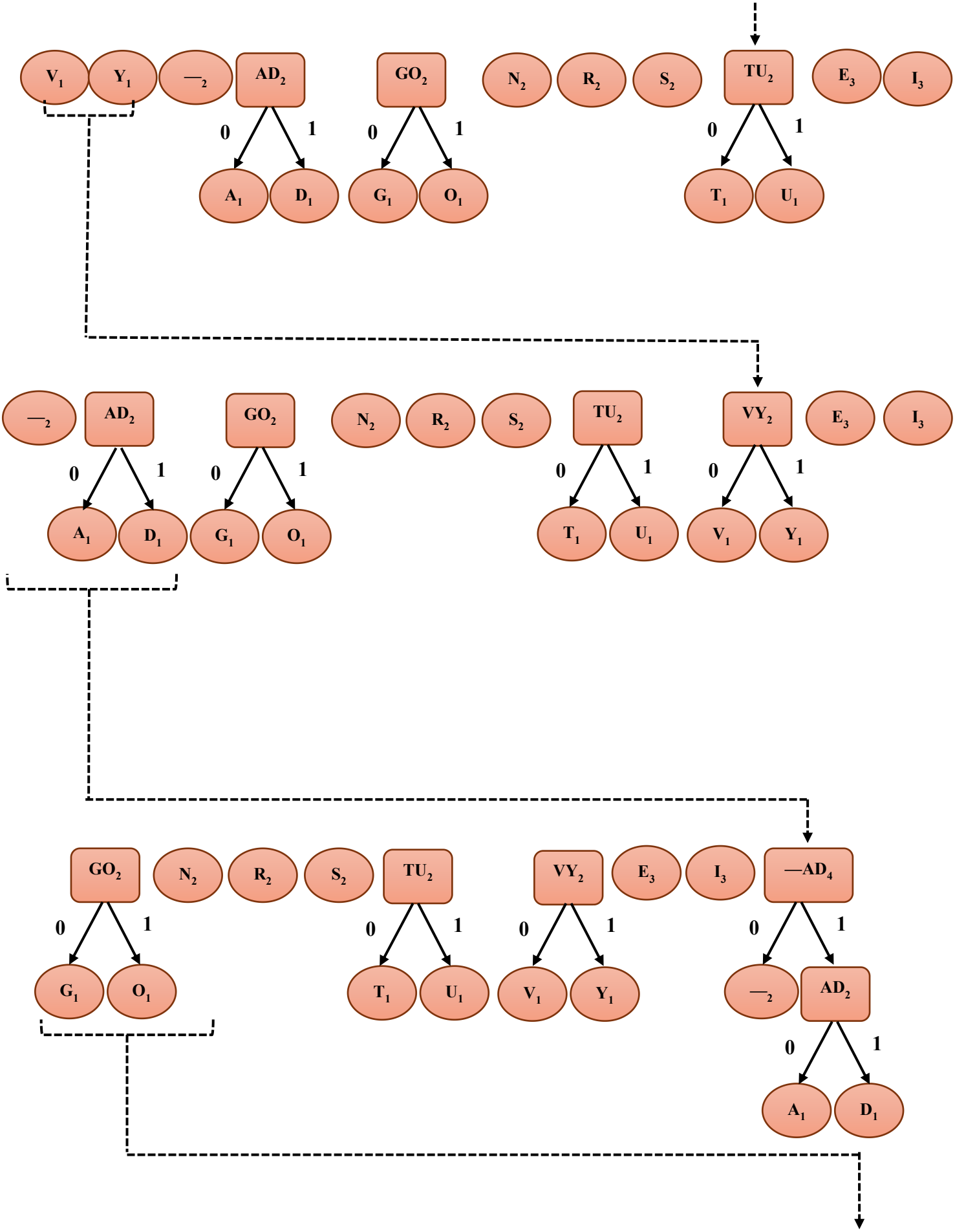
SYMBOL	NUMBER OF OCCURRENCES (WEIGHT) ⬇	FREQUENCY (%)
A	1	4.5454…
D	1	4.5454…
G	1	4.5454…

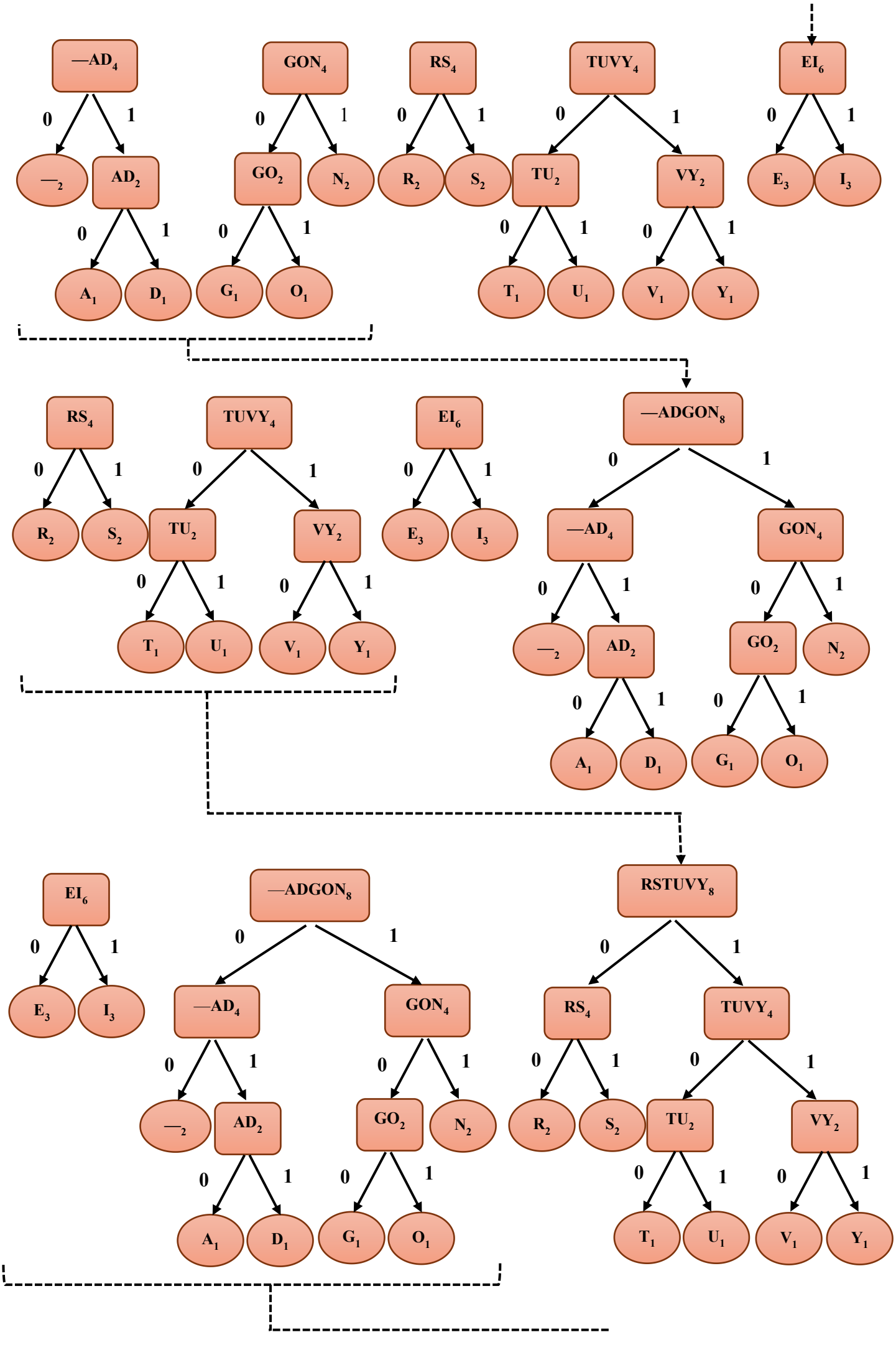
O	**1**	**4.5454…**
T	**1**	**4.5454…**
U	**1**	**4.5454…**
V	**1**	**4.5454…**
Y	**1**	**4.5454…**
—	**2**	**9.0909…**
N	**2**	**9.0909…**
R	**2**	**9.0909…**
S	**2**	**9.0909…**
E	**3**	**13.6363…**
I	**3**	**13.6363…**

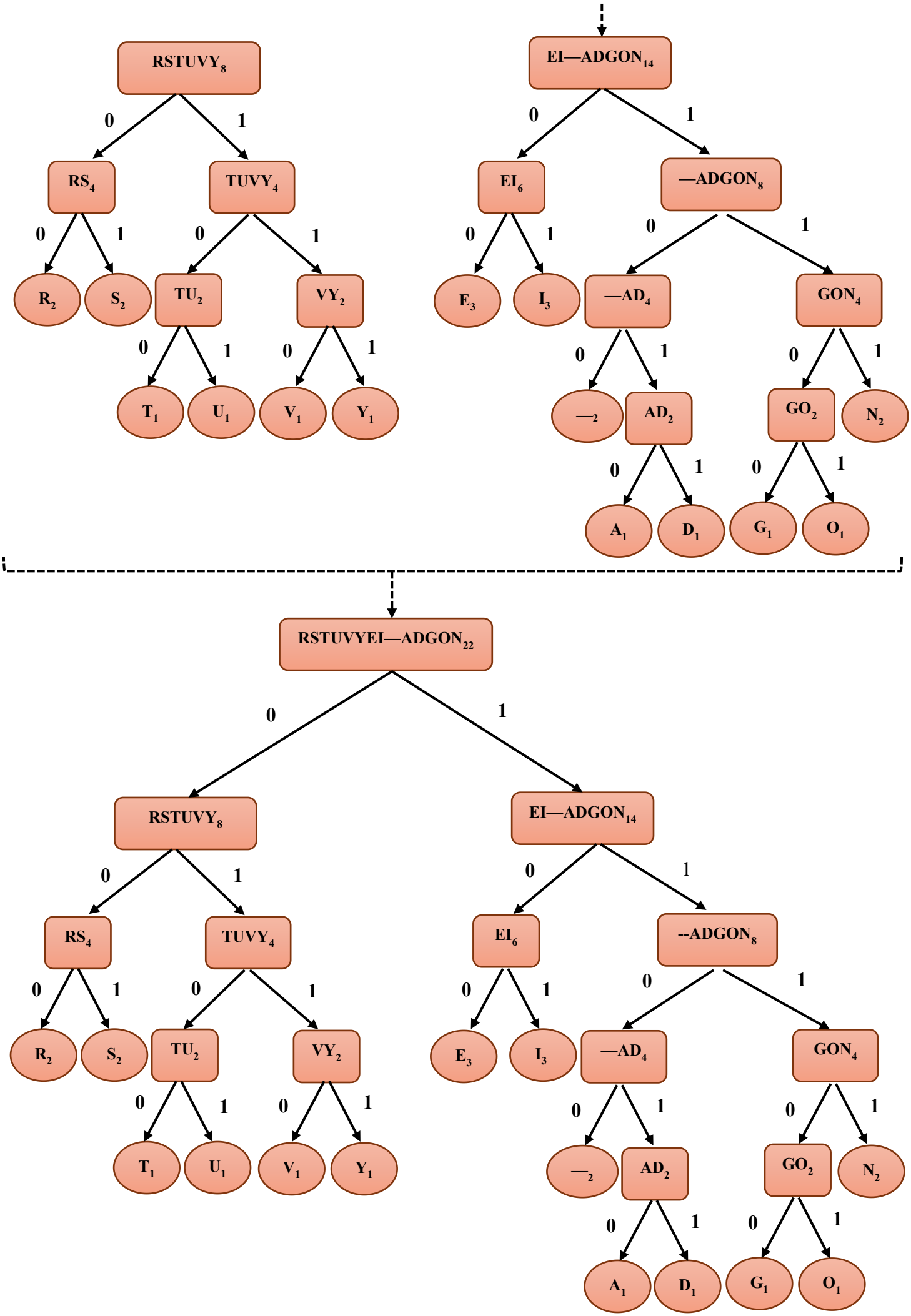
Step 2: Organising Symbols as a Binary Tree: -

To do so each symbol becomes a node storing the symbol itself and its weight.
Then the tree is constructed through the following iterative process: -

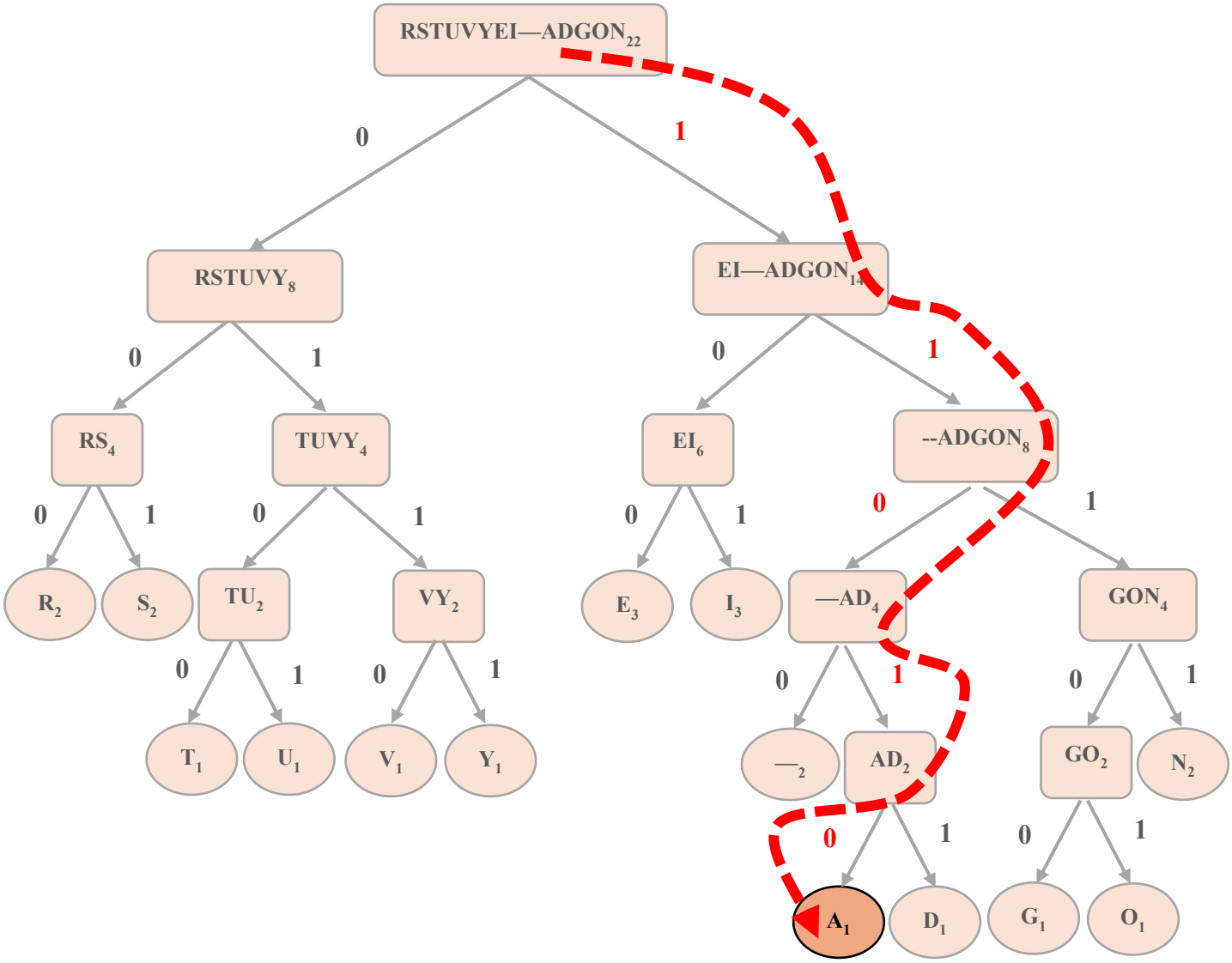

Sort symbols in order of their weights
A1 D1 G1 O1 T1 U1 V1 Y1 —2 N2 R2 S2 E3 I3
Create a new node by joining the first two nodes and adding their weight.
Branch the two initial nodes to the new node to from a tree.
Label the left branch "0" and the right branch "1"
In the above list remove the first two nodes and insert the new tree in position
(All nodes sorted in order of weights)
G1 O1 T1 U1 V1 Y1 —2 AD2 N2 R2 S2 E3 I3
0
1
A1
D1
Repeat this process, until you only have one node left: the root of a complete binary tree.
T1 U1 V1 Y1 —2 AD2 GO2 N2 R2 S2 E3 I3
0
1
0
1
A1
D1
G1
O1


V1 Y1 —2 AD2 GO2 N2 R2 S2 TU2 E3 I3
0 1 0 1 0 1
A1 D1 G1 O1 T1 U1
—2 AD2 GO2 N2 R2 S2 TU2 VY2 E3 I3
0 1 0 1 0 1 0 1
A1 D1 G1 O1 T1 U1 V1 Y1
GO2 N2 R2 S2 TU2 VY2 E3 I3 —AD4
0 1 0 1 0 1 0 1
G1 O1 T1 U1 V1 Y1 —2 AD2
0 1
A1 D1


—AD4
GON4
RS4
TUVY4
EI6
—2
AD2
GO2
N2
R2
S2
TU2
VY2
E3
I3
A1
D1
G1
O1
T1
U1
V1
Y1
—ADGON8
RSTUVY8
0
1


RSTUVY8
0
1
RS4
TUVY4
0
1
0
1
R2
S2
TU2
VY2
0
1
0
1
T1
U1
V1
Y1
EI—ADGON14
0
1
EI6
—ADGON8
0
1
0
1
E3
I3
—AD4
GON4
0
1
0
1
—2
AD2
GO2
N2
0
1
0
1
A1
D1
G1
O1
RSTUVYEI—ADGON22
0
1
RSTUVY8
EI—ADGON14
0
1
0
1
RS4
TUVY4
EI6
--ADGON8
0
1
0
1
0
1
R2
S2
TU2
VY2
E3
I3
—AD4
GON4
0
1
0
1
0
1
0
1
T1
U1
V1
Y1
—2
AD2
GO2
N2
0
1
0
1
A1
D1
G1
O1

Using the above tree, we can now identify the Huffman code for each symbol (leaves of the tree. This is done by highlighting the path from the Root node of the tree to the required leave/symbol. The labels of each branch are concatenated to form the Huffman code for the symbol.



As you can see, most frequent symbols are closer to the root node of the tree, resulting in shorter Huffman codes.

The resulting Huffman Codes for our message are: -

Symbol	Huffman code
R	000
S	001
E	100

I	101
T	0100
U	0101
V	0110
Y	0111
—	1100
N	1111
A	11010
D	11011
G	11100
O	11101

Encoding the message: -

We can now encode the message by replacing each symbol with its matching Huffman code.

S	A	G	E	—	U	N	I	V	E	R	S	I	T	Y
001	11010	11100	100	1100	0101	1111	101	0110	100	000	001	101	0100	0111

—	I	N	D	O	R	E
1100	101	1111	11011	11101	000	100

Encoded/compressed message: -

00111010111001001100010111111010110100000011010100011111001011111110111101000100

(Red color is used to differentiate the binary code of each character/symbol.)

Total bits used: -

Count the bits in each code:

Total bits:

3+5+5+3+4+4+4+3+4+3+3+3+3+4+4+4+3+4+5+5+3+3=82

Compression Comparison: -

Original data size (ASCII): -

22 characters × 8 bits = 176 bits

Compressed with Huffman Coding: -

82 bits

Compression Ratio: -

Compression Ratio = $\frac{176}{82} \approx 2.15$

Space savings (Compression gain): -

Compression gain = $\frac{176-82}{176} \times 100 \approx 53.41\%$

So, the compressed data is about 54% smaller than original.

Conclusion/Summary: -

Huffman coding, a lossless data compression technique, was applied to an original dataset (**SAGE UNIVERSITY INDORE**) of 176 bits. The compression reduced the size to 82 bits without any loss of information. This result in a compression gain of approximately 54%, indicating that the compressed data is significantly more efficient in terms of storage or transmit.

Feature	Original	Huffman compressed
Size in bit	176	82
Compression type	—	Lossless
Compression Gain	—	~54% smaller

References: -

[1] Mamta Sharma, "Compression Using Huffman Coding", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010, 133-141.

[2] Suherman and Andysah Putera Utama Siahaan, "Huffman Text Compression Technique",SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE) – volume 3, Issue 8–August 2016, 103-108.

[3] Guy E. Blelloch 'Introduction to Data Compression'. Carnegie Mellon University, 31 January , 2013.

[4] Sanjay Kumar Gupta "AN ALGORITHM FOR IMAGE COMPRESSION USING HUFFMAN CODING TECHNIQUES", International Journal of Advance Research in Science and Engineering. Volume 5, Issue 07, July 2016, 69-75.

[5] https://en.wikipedia.org/wiki/Lossless_compression.

[6] Sanjay Kumar Gupta "AN ALGORITHM FOR IMAGE COMPRESSION USING HUFFMAN CODING TECHNIQUES", International Journal of Advance Research in Science and Engineering. Volume 5, Issue 07, July 2016, 69-75.

[7] Dr. Pushpa R.Suri and Madhu Goel, "Ternary Tree & A new Huffman Decoding Technique", IJCSNS International Journal of Computer Science and Network Security, Vol.10 N0.3, March 2010.

[8] Mark Nelson and Jean-loup Gailly, "The Data Compression Book" (2nd edition). December 1995.

[9] https://en.wikipedia.org/wiki/Data_compression_ratio

[10] Julie Zelenski, Keith Schwarz," Huffman Encoding and Data Compression", Spring May 2012