Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

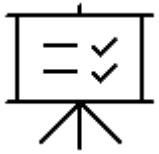
Topic:

Huffman Coding

AIM OF THE SESSION

To familiarize students with the concept of Huffman Coding

INSTRUCTIONAL OBJECTIVES



This Session is designed to:

1.Demonstrate :- Huffman Coding.

2.Describe :- Concept of Encoding and Compression of data for data transmission using Huffman coding

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Huffman Coding.
2. Describe :- Concept of Encoding and Compression of data for data transmission using Huffman coding.
3. Summarize:- Expected to know the importance of encoding and decoding using Huffman coding.

Encoding and Compression of Data

- **Compression:**
  - Data Compression, shrinks down a String so that it takes up less space. This is desirable for data storage and data communication.
  - Encoding means converting the String into Binary codes
- **Decompression:**

  In the decompression we convert the Binary codes into the Original string.

Fixed & Variable length code

- **Fixed length code:**

If every word in the code has the same length, the code is called a fixed-length code, or a block code.

Advantage	Disadvantage
Access is fast because the computer knows where each Word starts	Using Fixed length records, the records are usually larger and therefore need more storage space and are slower to transfer

- **Variable length code:**

Variable-length codes can perform significantly better as frequent characters are given short code words, while infrequent characters get longer code words.

Advantage	Disadvantages
Variable-length codes over fixed length is short codes that can be given to characters that occur frequently.	Where a character ends and another begins, difficult to identify.

Prefix Property

- A code has the prefix property if no character code is the prefix (start of the code) for another character
- Example:

Symbol	Code
P	000
Q	11
R	01
S	001
T	10

01001101100010

R S T Q P T

000 is not a prefix of 11, 01, 001, or 10

11 is not a prefix of 000, 01, 001, or 10 ….

Huffman Coding

- Developed by **David Huffman** in 1951
- Huffman Coding is a famous Greedy Algorithm.
- It is used for the lossless compression of data.
- It uses variable length encoding.
- It assigns variable length code to all the characters.
- The code length of a character depends on how frequently it occurs in the given text.
- The character which occurs most frequently gets the smallest code.
- The character which occurs least frequently gets the largest code.
- It is also known as **Huffman Encoding**.

Prefix Rule

1. Huffman Coding implements a rule known as a prefix rule.

2. This is to prevent the ambiguities while decoding.

3. It ensures that the code assigned to any character is not a prefix of the code assigned to any other character.

Major Steps in Huffman Coding-

There are two major steps in Huffman Coding-

1. Building a Huffman Tree from the input characters.
2. Assigning code to the characters by traversing the Huffman Tree.

Huffman Tree

The steps involved in the construction of Huffman Tree are as follows-

Step-01:

- Create a leaf node for each character of the text.
- Leaf node of a character contains the occurring frequency of that character

Step-02:

- Arrange all the nodes in increasing order of their frequency value.

Step-03:

- Considering the first two nodes having minimum frequency,
- Create a new internal node.
- The frequency of this new node is the sum of frequency of those two nodes.
- Make the first node as a left child and the other node as a right child of the newly created node.

Step-04:

- Keep repeating Step-02 and Step-03 until all the nodes form a single tree.
- The tree finally obtained is the desired Huffman Tree.

Important Formulas-

The following 2 formulas are important to solve the problems based on Huffman Coding-

Formula-01:

$$\text{Average code length per character} = \frac{\Sigma\,(\text{frequency}_i \times \text{code length}_i)}{\Sigma\,\text{frequency}_i}$$

$$= \Sigma\,(\text{probability}_i \times \text{code length}_i)$$

Formula-02:

- Total number of bits in Huffman encoded message
- = Total number of characters in the message **x** Average code length per character
- = $\sum$ (frequency_i x Code length_i)

PRACTICE PROBLEM BASED ON HUFFMAN CODING

Problem-

A file contains the following characters with the frequencies as shown. If Huffman Coding is used for data compression, determine-

1. Huffman Code for each character
2. Average code length
3. Length of Huffman encoded message (in bits)

Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

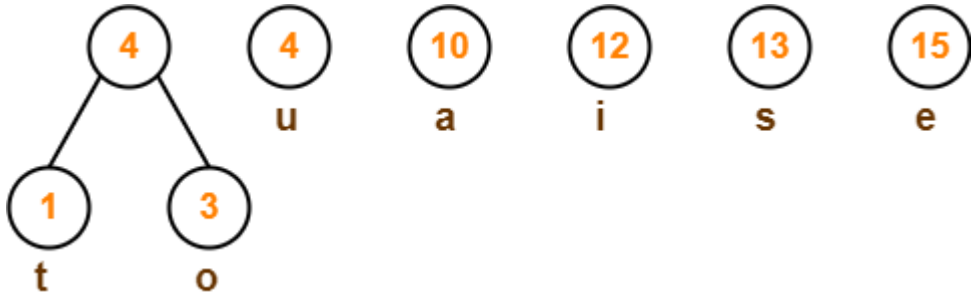
Solution

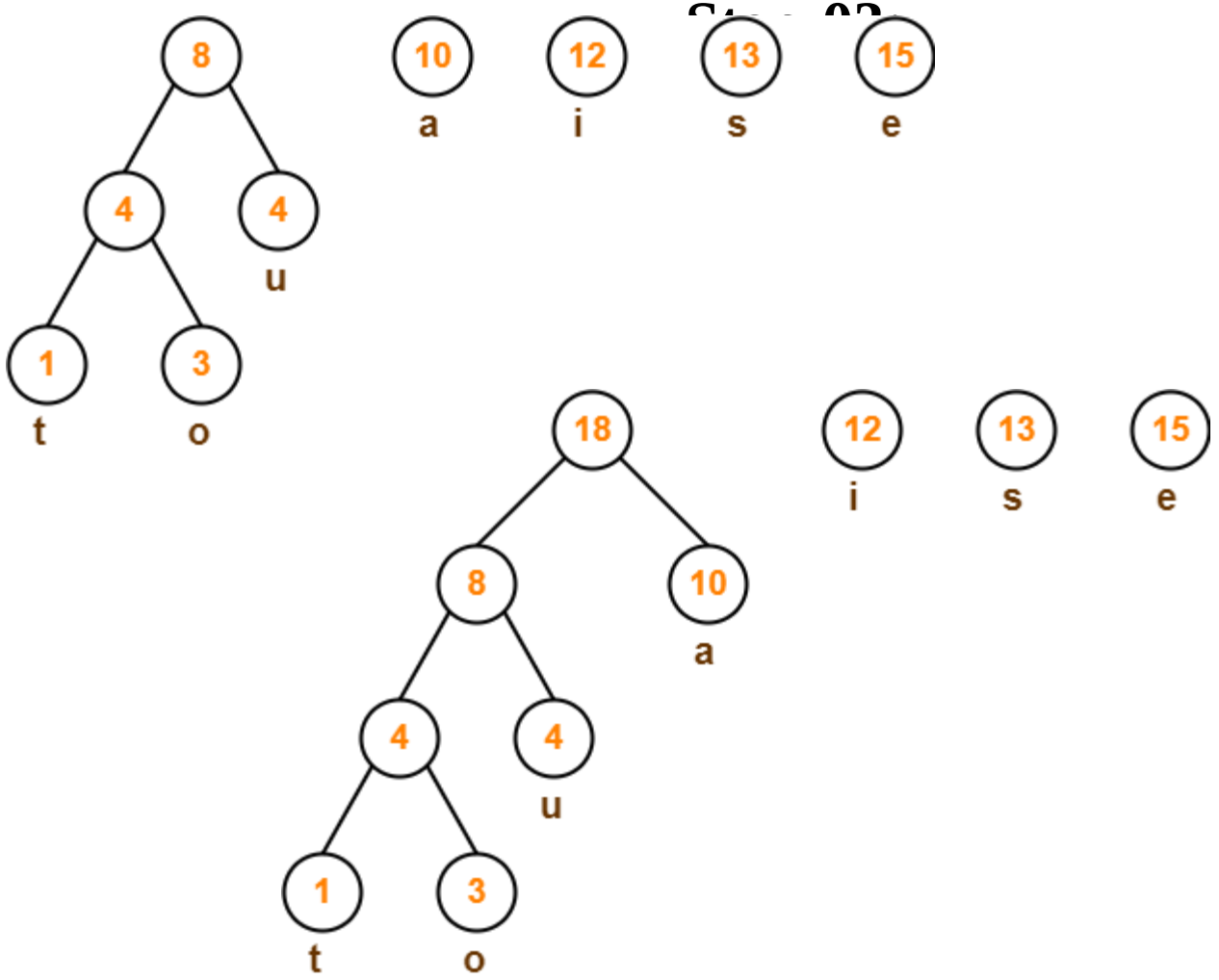
First let us construct the Huffman Tree.

- Huffman Tree is constructed in the following steps-

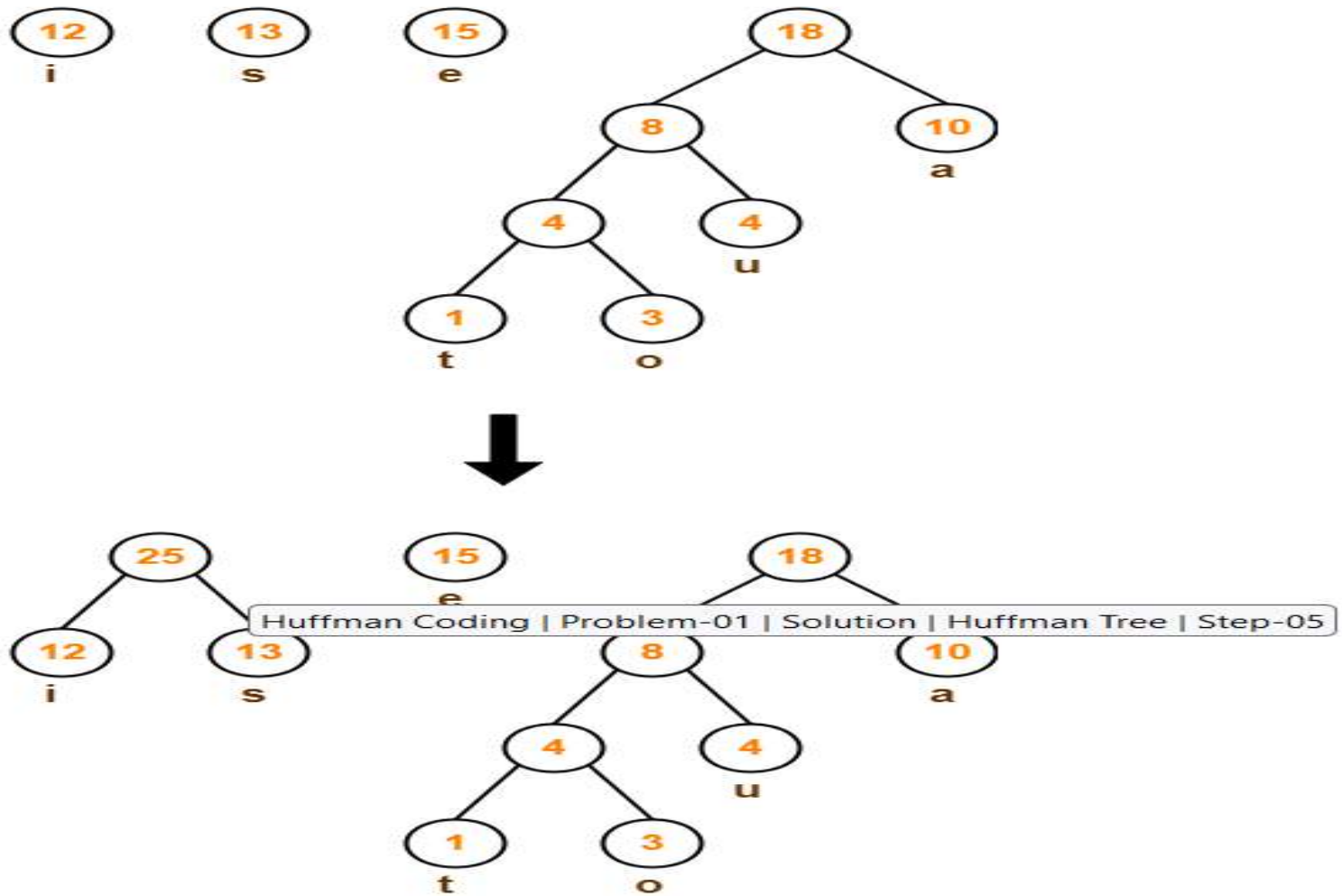
Step-01:

Step-02:

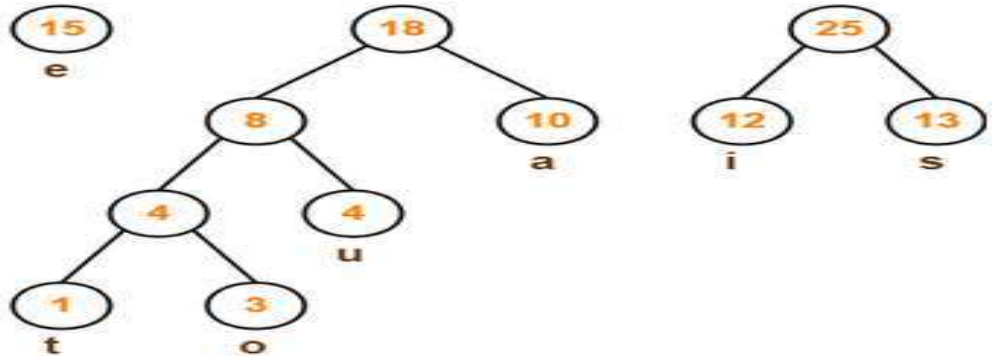


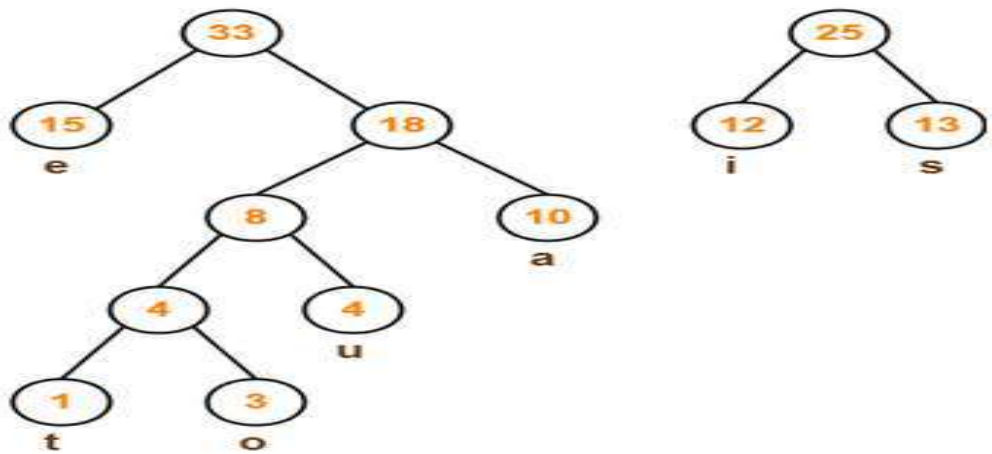


Step-05:

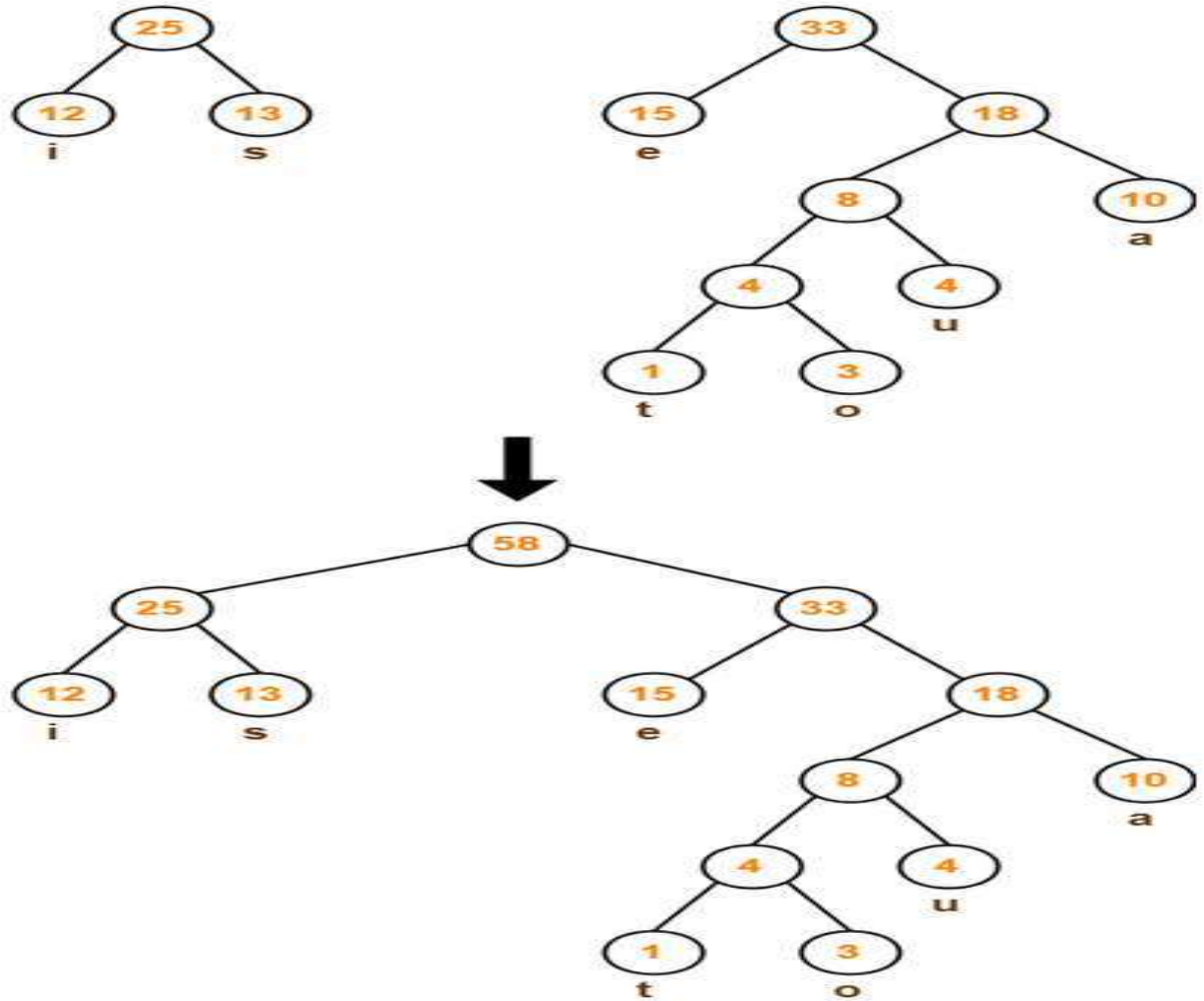


Step-06:


15
e
18
8
10
a
4
4
u
1
t
3
o
25
12
i
13
s


33
15
e
18
8
10
a
4
4
u
1
t
3
o
25
12
i
13
s

Step-07:


25
12
i
13
s
33
15
e
18
8
10
a
4
4
u
1
t
3
o
58
25
12
i
13
s
33
15
e
18
8
10
a
4
4
u
1
t
3
o

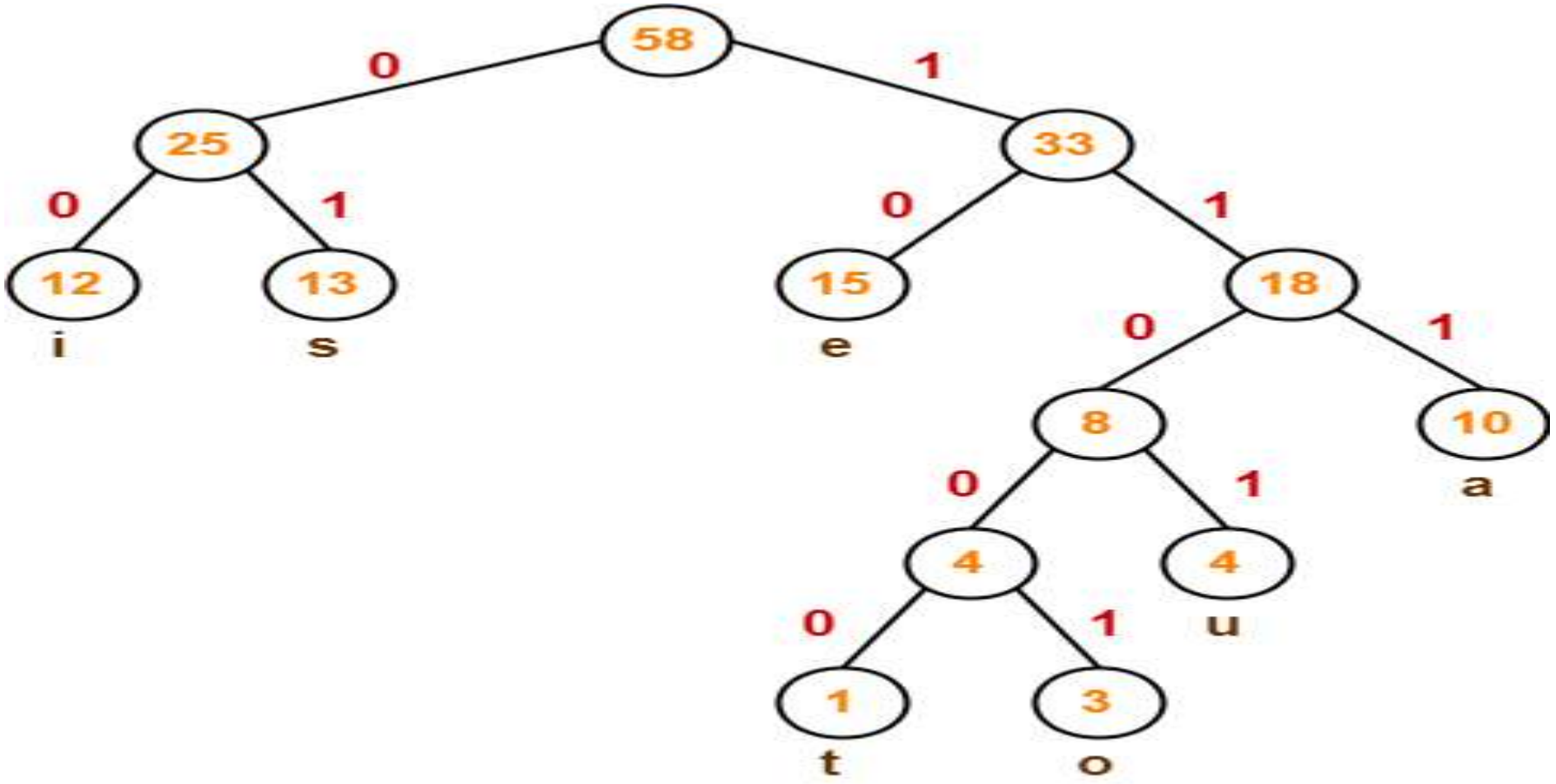
Now,

- We assign weight to all the edges of the constructed Huffman Tree.
- Let us assign weight ‘0’ to the left edges and weight ‘1’ to the right edges

Rule

- If you assign weight ‘0’ to the left edges, then assign weight ‘1’ to the right edges.
- If you assign weight ‘1’ to the left edges, then assign weight ‘0’ to the right edges.
- Any of the above two conventions may be followed.
- But follow the same convention at the time of decoding that is adopted at the time of encoding.

After assigning weight to all the edges, the modified Huffman Tree is-



Huffman Tree

Now, let us answer each part of the given problem one by one-

Huffman Code For Characters

To write Huffman Code for any character, traverse the Huffman Tree from root node to the leaf node of that character.

Following this rule, the Huffman Code for each character is-

- a = 111
- e = 10
- i = 00
- o = 11001
- u = 1101
- s = 01
- t = 11000

From here, we can observe-

- Characters occurring less frequently in the text are assigned the larger code.
- Characters occurring more frequently in the text are assigned the smaller code.

2. Average Code Length-

Using formula-01, we have-

Average code length

$= \sum (\text{frequency}_i \times \text{code length}_i) / \sum (\text{frequency}_i)$

$= \{ (10 \times 3) + (15 \times 2) + (12 \times 2) + (3 \times 5) + (4 \times 4) + (13 \times 2) + (1 \times 5) \} / (10 + 15 + 12 + 3 + 4 + 13 + 1)$

$= 2.52$

3. Length of Huffman Encoded Message

Using formula-02, we have-

Total number of bits in Huffman encoded message

= Total number of characters in the message x Average code length per character

= 58 x 2.52

= 146.16

≅ 147 bits

Applications

- Huffman coding is a technique used to compress files for transmission
- Uses statistical coding
  - more frequently used symbols have shorter code words
- Works well for text and fax transmissions
- An application that uses several data structures

SUMMARY

Huffman coding is a compression technique that assigns variable-length codes to symbols based on their frequencies, achieving efficient data storage by minimizing average code length and enabling straightforward decoding using a constructed binary tree.

SELF-ASSESSMENT QUESTIONS

What type of tree is used in Huffman coding?

(a) AVL Tree

(b) Red-Black Tree

(c) Binary Heap

(d) Binary Tree

Huffman coding is primarily used for

(a) Data Encryption

(b) Data Compression

(c) Data Decompression

(d) Data Transmission

TERMINAL QUESTIONS

1. Explain about coding and de-coding
2. What is compression

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.
2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.
3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.
4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=
2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU