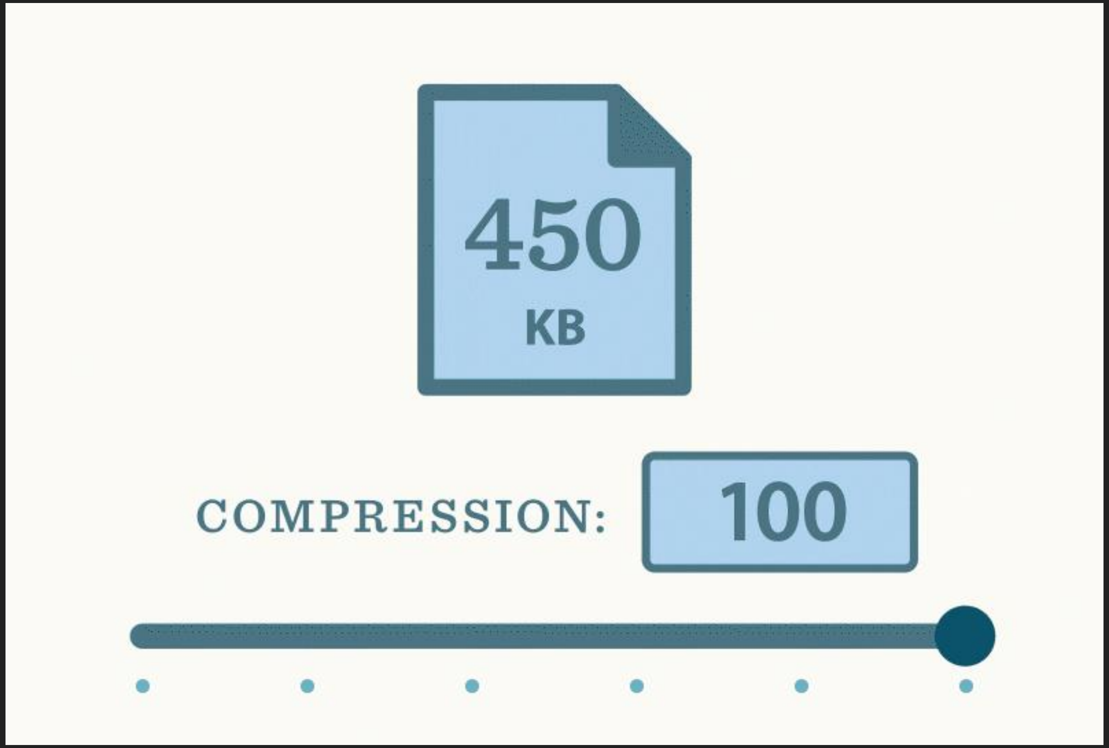
Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology, Pune

Department of AI & DS

A Mini Project of

Data Structure and Discrete Mathematics

Name	PRN No.	Roll No.
Rhishikesh Ravindra Sonawane	22010585	272053
Siddhesh Ramesh Songire	22010862	272054
Omsai Shankar Zadi	22011132	272065



DATA COMPRESSION USING HUFFMAN CODING

DATA COMPRESSION

- Data compression is a process of encoding, restructuring, or otherwise modifying data in order to reduce its size.
- Compression is done by a program that uses functions or an algorithm to effectively discover how to reduce the size of the data.
- Text compression is usually done by removing all unnecessary characters. By replacing a smaller bit string for a more common bit string.

Huffman Coding

- Huffman coding is a lossless data compression algorithm.
- The most frequent character gets the smallest code, and the least frequent character gets the largest code.
- The variable-length codes assigned to input characters are Prefix Codes.
- This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit stream.

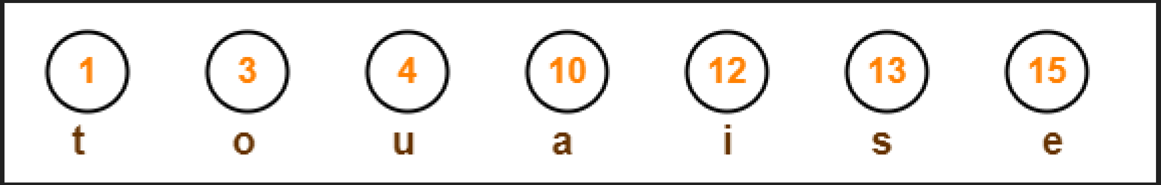
HOW DOES HUFFMAN CODING WORK?

- Each character occupies 8 bits. There are a total of 58 characters in the above string.
  Thus, a total of 8*58 = 464 bits are required to send the string.
- Using the Huffman coding technique, we can compress the string to a smaller size.

CHARACTERS	FREQUENCY
A	10
E	15
I	12
O	3
U	4
S	13
T	1
Total Characters:	58

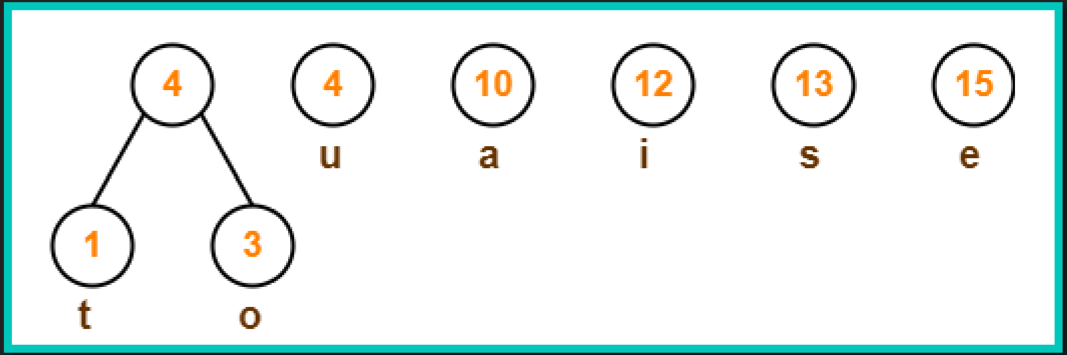
HOW IS HUFFMAN CODING DONE:

- STEP-1 Calculate the frequency.
- STEP-2 Sort the characters in increasing order of the frequency.

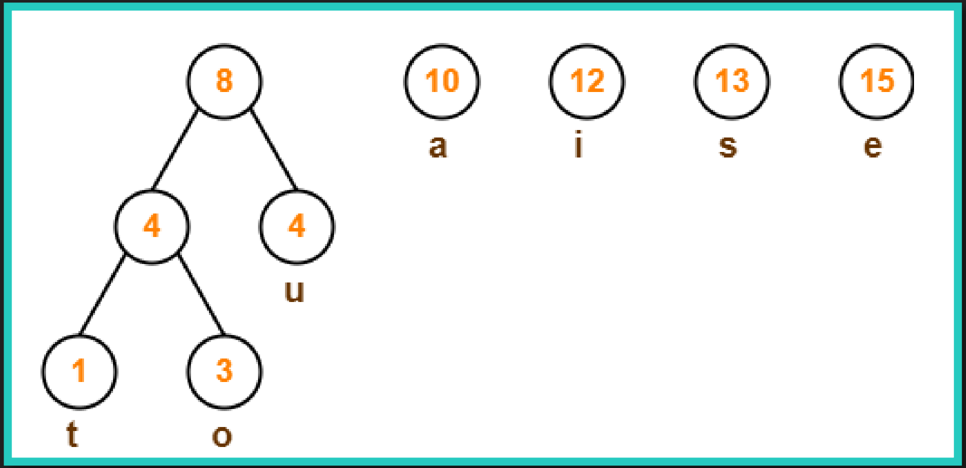


- STEP-3 Create an empty node and Assign 1st and 2nd minimum frequency to the left and right child respectively and set the value of the node as the sum of the above two minimum frequencies.

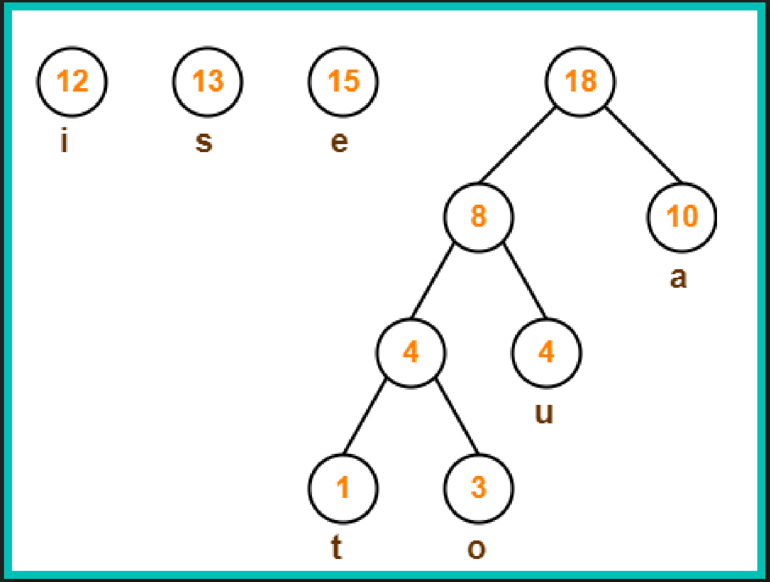
- **STEP-4** Remove these two minimum frequencies from Queue and add the sum into the list of frequencies.

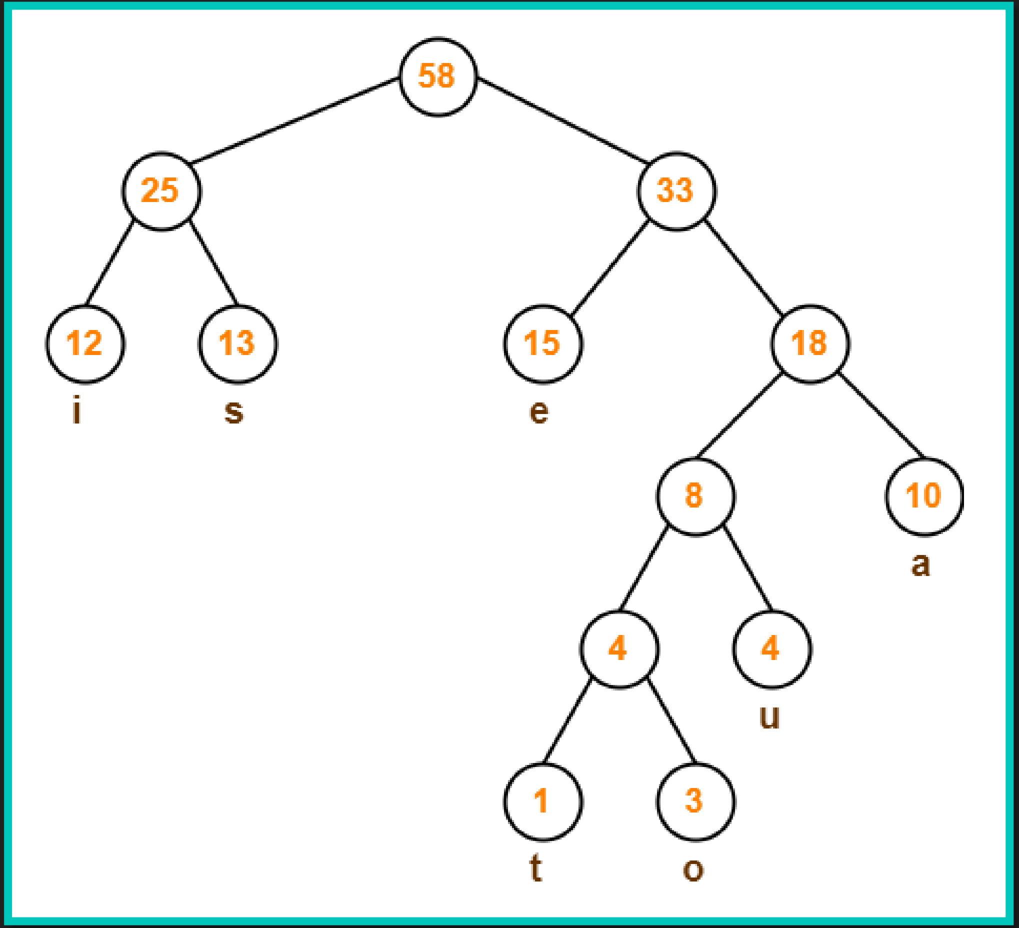


- **STEP-5** Insert this node into the tree.



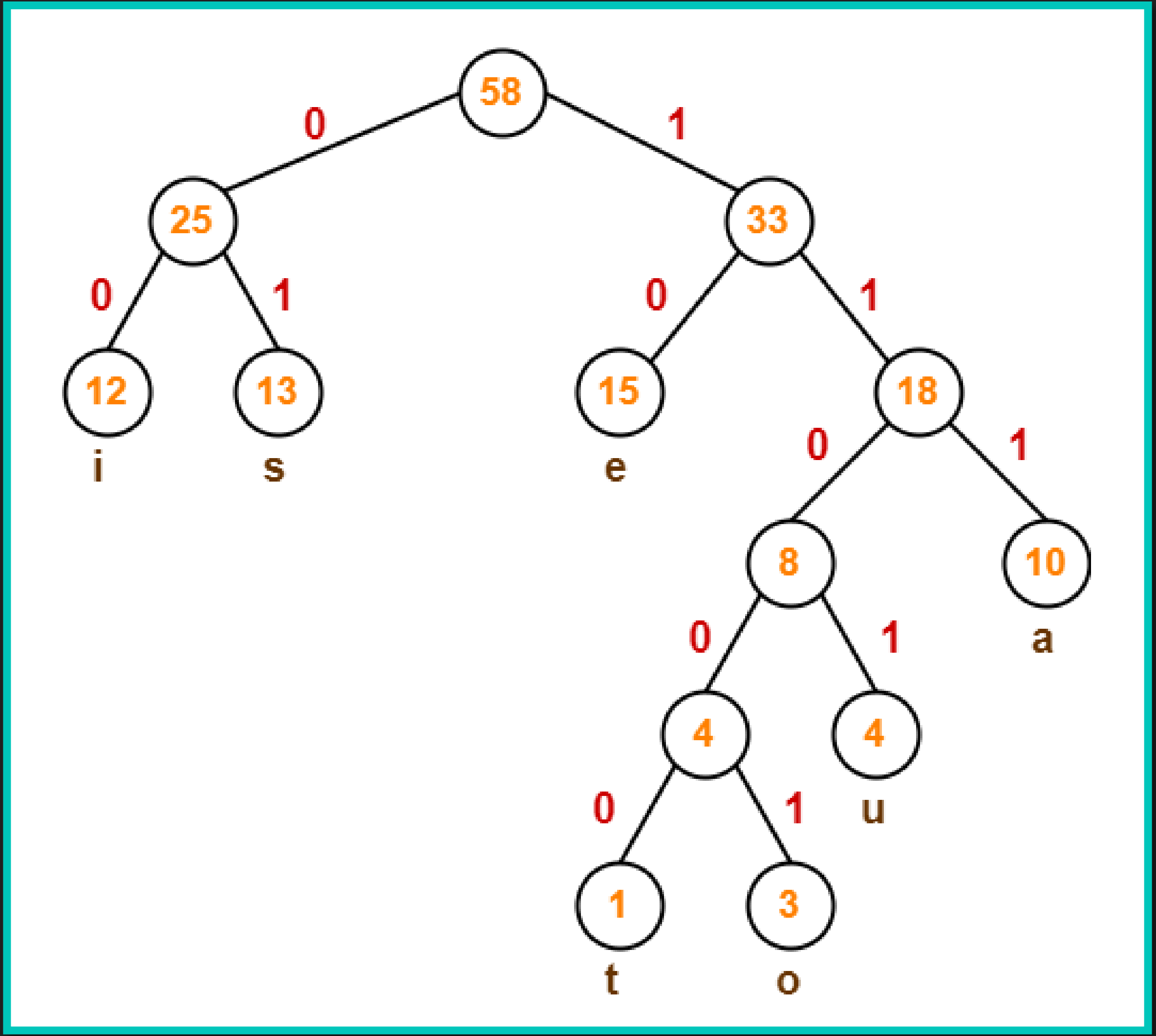
- **STEP-6** Repeat steps 3 and 4 for all the characters.





The Final Tree

- **STEP-7** Finally for each non-leaf, assign 0 to the left edge and 1 to the right edge.
- *After assigning weight to all the edges, the modified Huffman Tree is-*

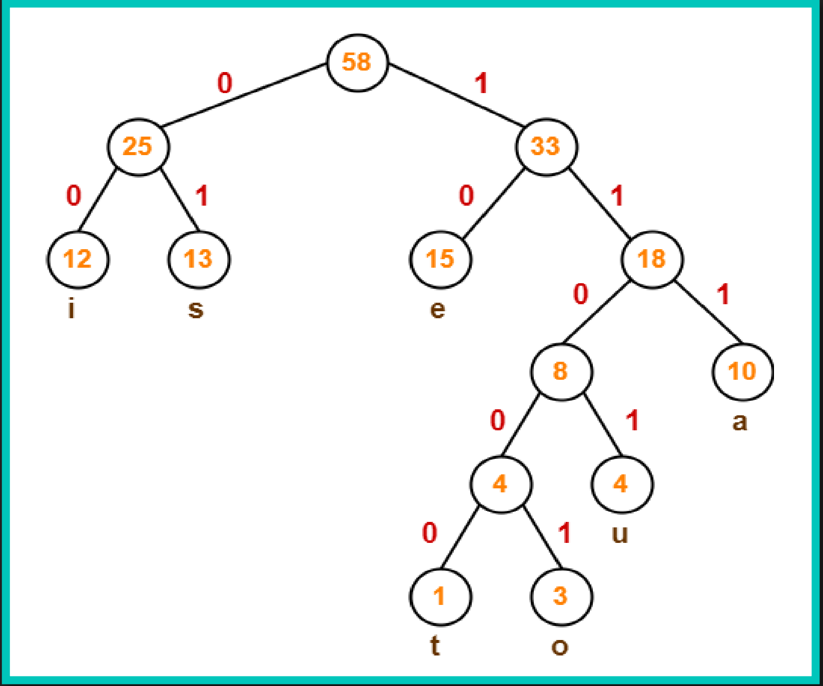


HUFFMAN CODE FOR CHARACTERS

- 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 and vice versa.*



CHARACTERS	FREQUENCY	CODE	SIZE
A	10	111	10*3=30
E	15	10	15*2=30
I	12	00	12*2=24
O	3	11001	3*5=15
U	4	1101	4*4=16
S	13	01	13*2=26
T	1	11000	1*5=5

In the above table we can see that all the characters are having prefix-free code words and their size.

Comparison (464 bits Vs 146 bits)

Calculating the total size,

before Huffman Encoding

- Each character $\longrightarrow$ 8 bits.
- Total $\longrightarrow$ 58 characters.
- Thus, a total of 8*58 $\longrightarrow$ 464 bits

Calculating the total size,

after Huffman Encoding

- 30 + 30 + 24 + 15 + 16 + 26 + 5 = 146 BITS
- Total size is reduced by 464 - 146= 318 Bits

Size is reduced by 68.53%

Advantages Vs Disadvantages

- Generates shorter binary codes for encoding symbols and characters.
- The binary codes generated are prefix-free.

- Lossless data encoding schemes, like Huffman encoding, achieve a lower compression ratio compared to lossy encoding techniques.
- Huffman encoding is a relatively slower process.

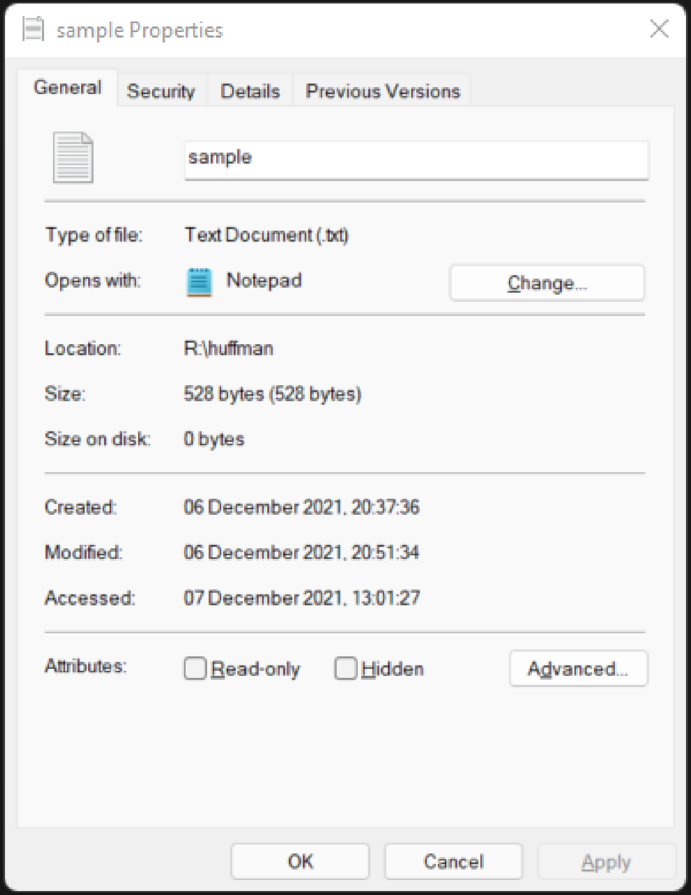
Real-life Applications Of Huffman Encoding

Huffman encoding is widely used in compression formats like GZIP, PKZIP (winzip) and BZIP2.

Multimedia codecs like JPEG, PNG and MP3 uses Huffman encoding (to be more precised the prefix codes)

Huffman encoding still dominates the compression industry since newer arithmetic and range coding schemes are avoided due to their patent issues.

Compression file details



sample Properties

General | Security | Details | Previous Versions

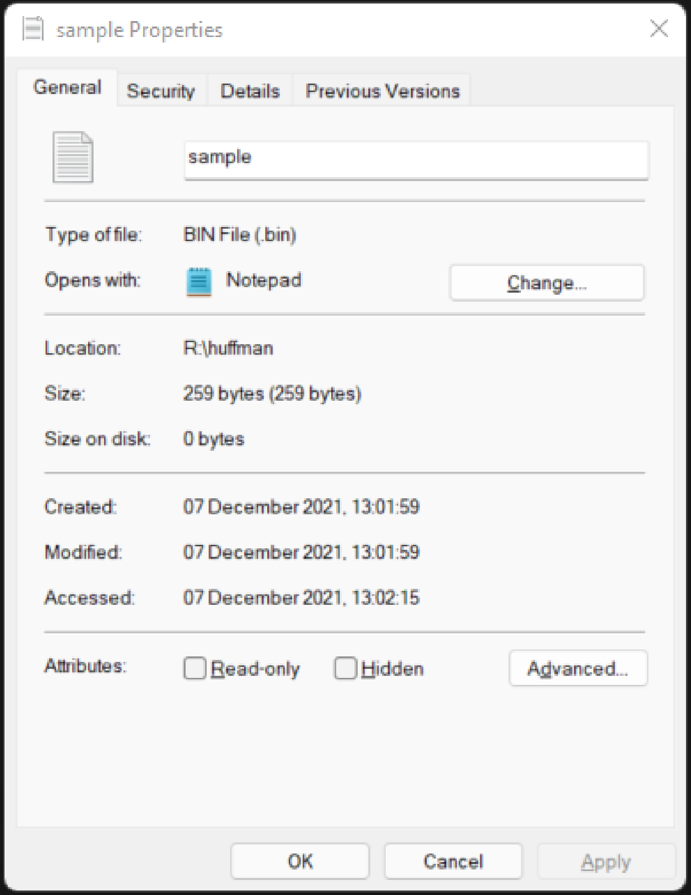
sample

Type of file: Text Document (.txt)
Opens with: Notepad

Location: R:\huffman
Size: 528 bytes (528 bytes)
Size on disk: 0 bytes

Created: 06 December 2021, 20:37:36
Modified: 06 December 2021, 20:51:34
Accessed: 07 December 2021, 13:01:27

Attributes: ☐ Read-only ☐ Hidden



sample Properties

General | Security | Details | Previous Versions

sample

Type of file: BIN File (.bin)
Opens with: Notepad

Location: R:\huffman
Size: 259 bytes (259 bytes)
Size on disk: 0 bytes

Created: 07 December 2021, 13:01:59
Modified: 07 December 2021, 13:01:59
Accessed: 07 December 2021, 13:02:15

Attributes: ☐ Read-only ☐ Hidden

File size reduced from

528 bytes
to
259 bytes

References

- https://www.cplusplus.com/reference/stl/
- https://www.geeksforgeeks.org/huffman-decoding/?ref=lbp
- https://www.programiz.com/dsa/huffman-coding
- https://cppsecrets.com/users/9999109111117110105971171035048484864103109971051084699111109/C00-Greedy-Approach-Huffman-Codes.php
- https://www.barracuda.com/glossary/data-compression

THANK YOU !