VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT**

**ANALYSIS AND DESIGN ALGORITHM**

*Lecturers* **Assoc.Prof. LÊ ANH CƯỜNG**

*Students*: **NGUYỄN TẤN LỘC – 518H0645**

**NGUYỄN KHẮC MINH LUÂN – 518H0398**

Class **: 18H50302**

Course  **: 22**

**HO CHI MINH CITY, YEAR 2020**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT**

**ANALYSIS AND DESIGN ALGORITHM**

*Lecturers* **Assoc.Prof. LÊ ANH CƯỜNG**

*Student*: **NGUYỄN TẤN LỘC**

**NGUYỄN KHẮC MINH LUÂN**

Class **: 18H50302**

Course  **: 22**

**HO CHI MINH CITY, YEAR 2020**

ACKNOWLEDGEMENT

I would like to express my very great appreciation to **Mr. Lê Anh Cường** for his valuable and constructive suggestions during the planning and development of this design and analysis algorithm project. Advices given by **Mr. Lê Anh Cường** has been a great help in doing this essay.

I would also like to thank my project external all the people who provided me with the facilities being required and conductive conditions for my project.

Sincerely thanks with all my love and respect.

TABLE OF CONTENTS

[ACKNOWLEDGEMENT 1](#_Toc56977256)

[TABLE OF CONTENTS 2](#_Toc56977257)

[HUFFMAN ENCODE AND DECODE 3](#_Toc56977258)

[1. HUFFMAN: 3](#_Toc56977259)

[2. HUFFMAN ENCODING 3](#_Toc56977260)

[3. HUFFMAN DECODING 6](#_Toc56977261)

[4. INTERFACE 7](#_Toc56977262)

[5. RESULT OF PROGRAM: 8](#_Toc56977263)

[6. Huffman Coding Complexity 9](#_Toc56977264)

[PLAGIARISM DETECTION PROBLEM, FINDING COPIED PARAGRAPHS OF TEXT A FROM DOCUMENTS IN A GIVEN TEXTS. 9](#_Toc56977265)

# HUFFMAN ENCODE AND DECODE

1. HUFFMAN:

* Huffman coding tree or Huffman tree is a full binary tree in which each leaf of the tree corresponds to a letter in the given alphabet.
* Define the weighted path length of a leaf to be its weight times its depth. The Huffman tree is the binary tree with minimum external path weight, i.e., the one with the minimum sum of weighted path lengths for the given set of leaves. So the goal is to build a tree with the minimum external path weight.
* Huffman codes are optimal prefix-free binary codes (The **greedy algorithm** builds the Huffman tree with the minimum external path weight for a given set of letters).

1. HUFFMAN ENCODING

The first step of Huffman encoding is building the Huffman tree. Given a set of characters and their associated frequencies, we can build an optimal Huffman tree as follows:

1. Construct leaf Huffman trees for each character/frequency pair
2. Repeatedly choose two minimum-frequency Huffman trees and join them together into a new Huffman tree whose frequency is the sum of their frequencies.
3. When only one Huffman tree remains, it represents an optimal encoding.

Then the code for each character can be obtained by following the path from the root of the tree to the leaf holding the given character, assigning and accumulating a '0' when following a left edge and a '1' when following a right edge. The accumulated zeros and ones at each leaf constitute a Huffman encoding for those symbols. The image below illustrates this process.

For example: String: abbccc

String Encoding

a 10

b 11

c 0

0 1

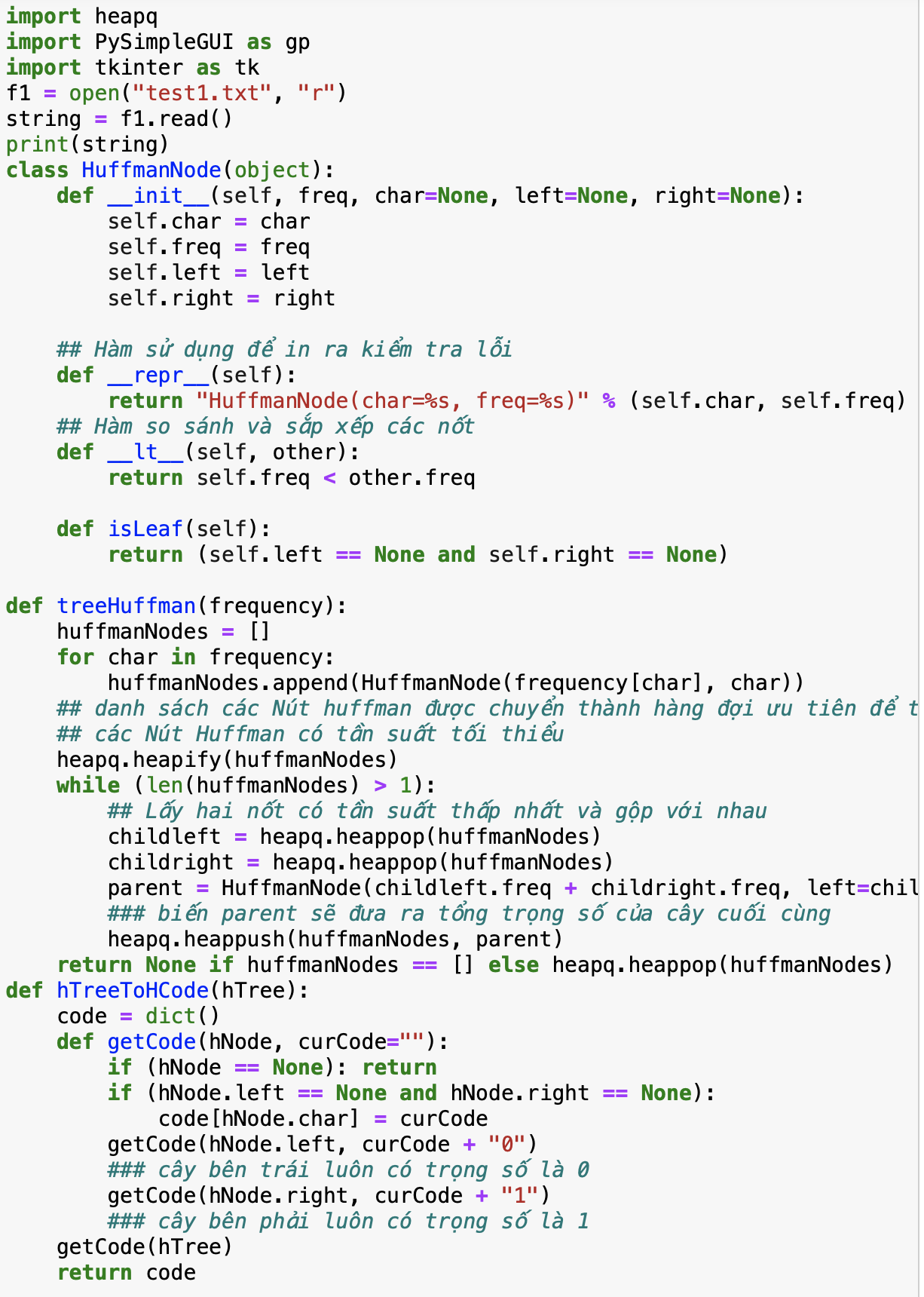
c

0 1

c

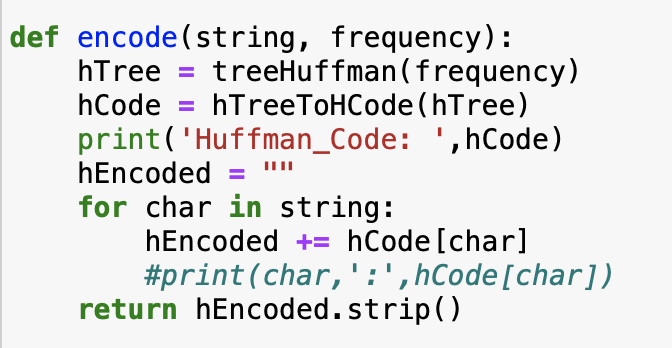
c

Code build Tree:

:

*Figure 2: Built tree implement*

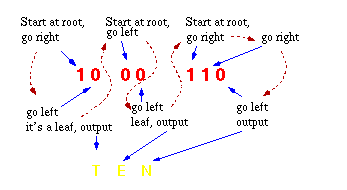
Code encoding:



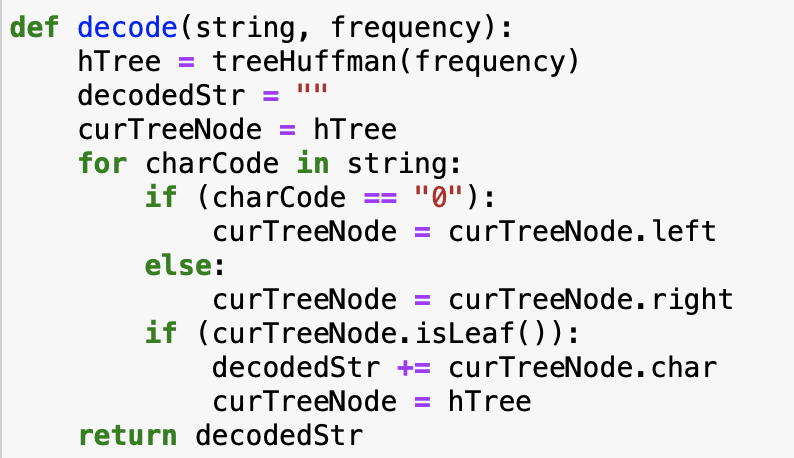
*Figure 3: Encode Implement*

1. HUFFMAN DECODING

The decoding procedure is deceptively simple. Starting with the first bit in the stream, one then uses successive bits from the stream to determine whether to go left or right in the decoding tree. When we reach a leaf of the tree, we've decoded a character, so we place that character onto the (uncompressed) output stream. The next bit in the input stream is the first bit of the next character.

****

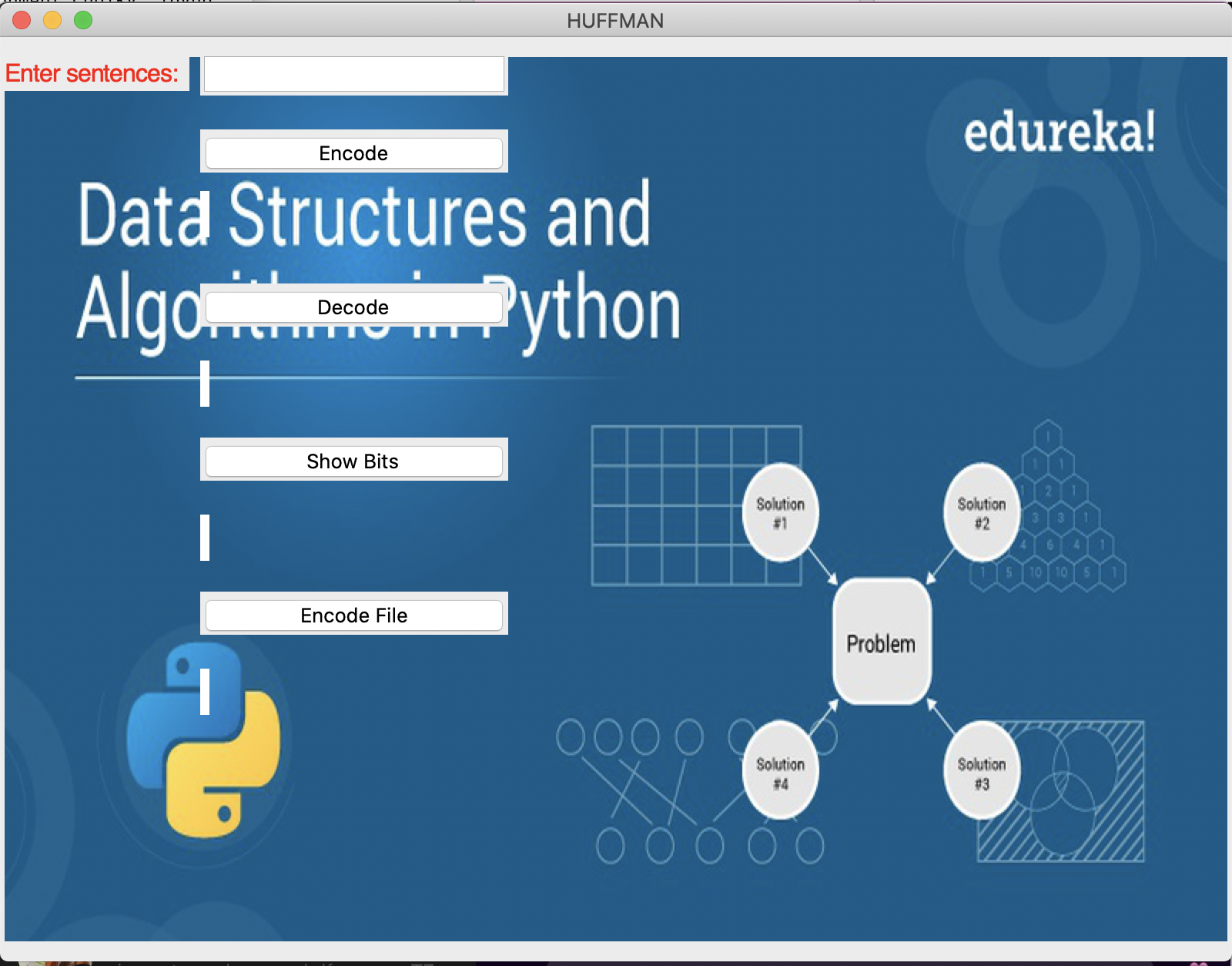
*Figure 4: Decode process*

Code Decoding: 

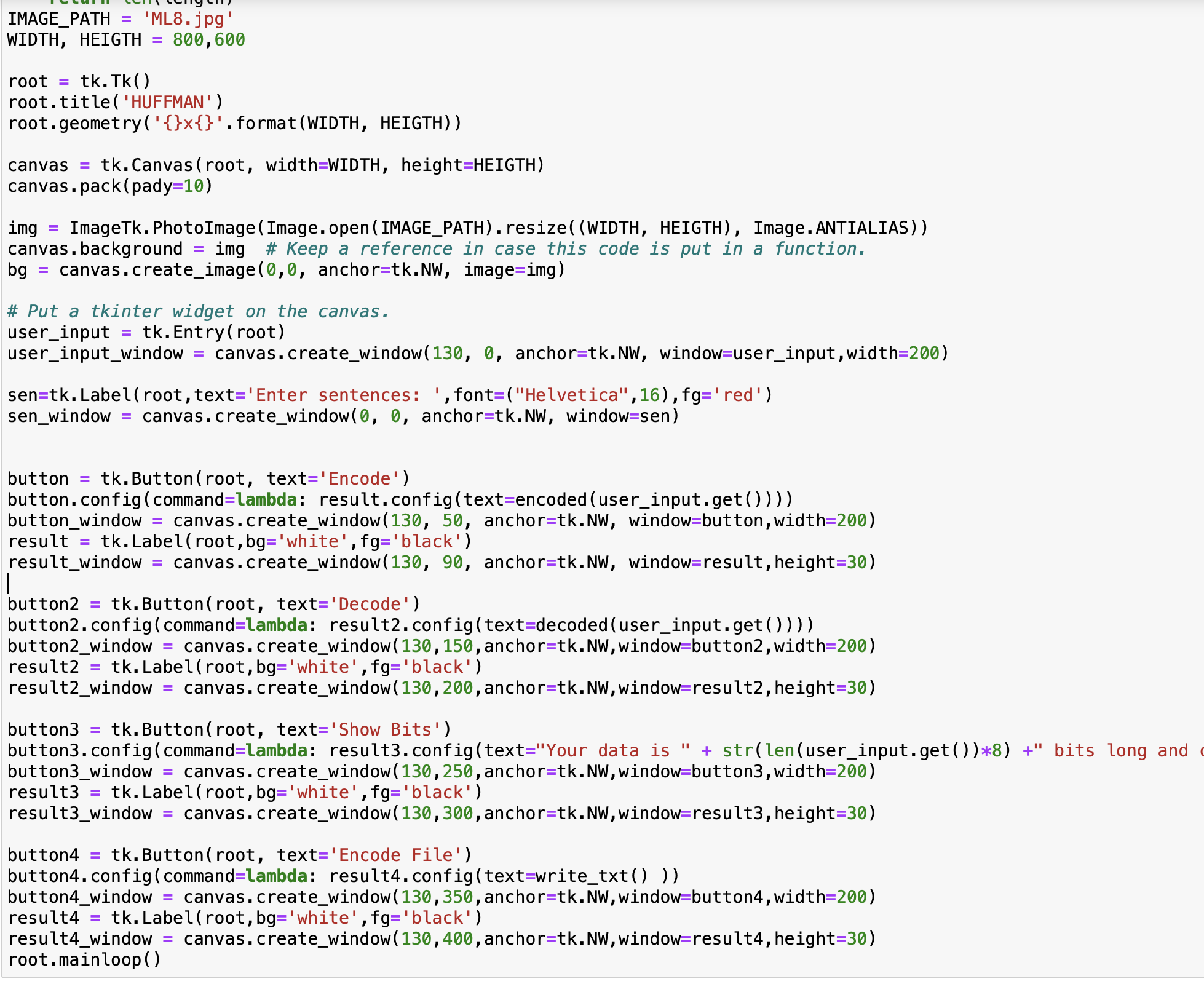
*Figure 5: Decode implement*

1. INTERFACE

Our project using tkinter library to create the interface for the encode – decode program



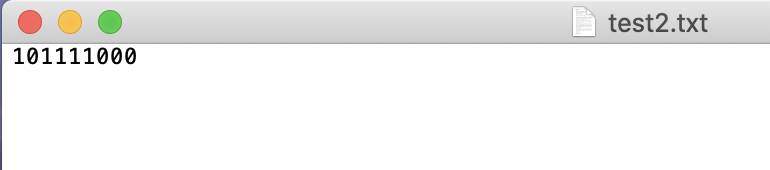
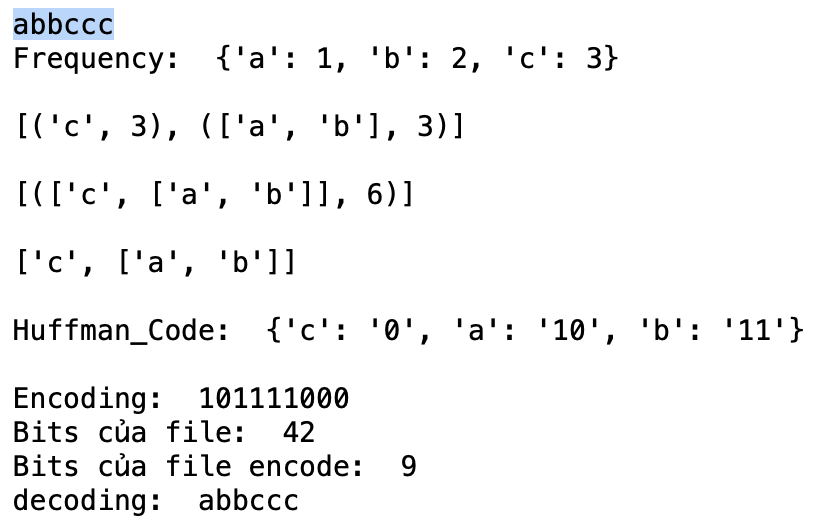
*Figure 6: Huffman encode – decode interface*



*Figure 7: Huffman interface implement*

1. RESULT OF PROGRAM:

Example : run with string is **abbccc** and print into file

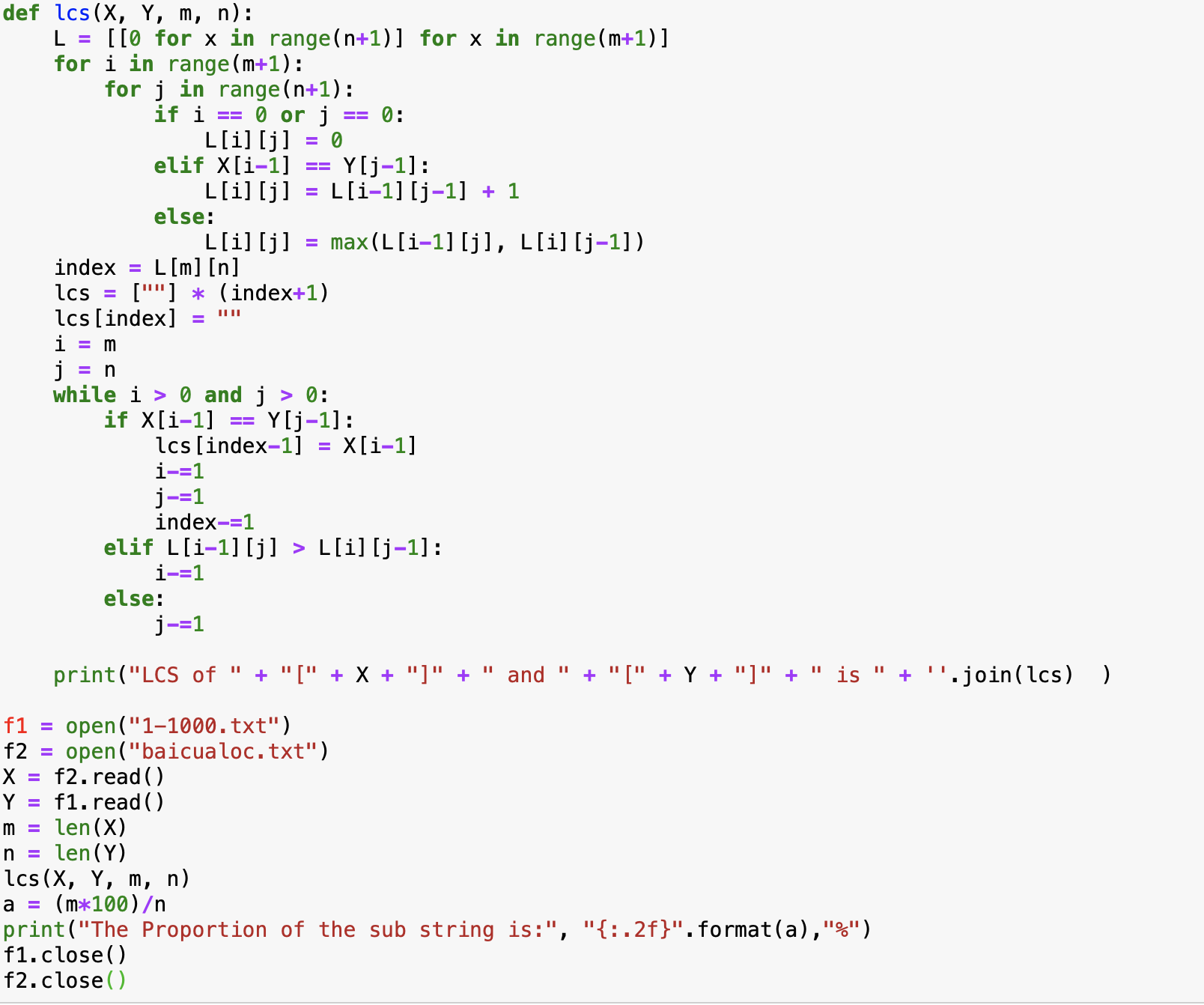


* Calculation size bits:
  + String: length of string \* 8
  + Encode: 1 \* 2 + 2 \* 2 + 3 \* 1

## Huffman Coding Complexity

The time complexity for encoding each unique character based on its frequency is O(nlog n)

# PLAGIARISM DETECTION PROBLEM, FINDING COPIED PARAGRAPHS OF TEXT A FROM DOCUMENTS IN A GIVEN TEXTS.



Result:

