TASK 2: You have to implement HUFFMAN ENCODING in both of files and show its performance in form of time complexity, size of file.

For this task, we designed a Huffman Algorithm using trees and priority queues. We used 3 different data structures: Trees, Queues and Vectors. All these three vectors were hand-written by us.

HUFFMAN Encoding Algorithm

The idea behind Huffman coding is based upon the frequency of a symbol in a sequence. The symbol that is the most frequent in that sequence gets a new code that is very small, the least frequent symbol will get a code that is very long, so that when we'll translate the input, we want to encode the most frequent symbols will take less space than they used to and the least frequent symbols will take more space but because they're less frequent it won't matter that much.

In our implementation, we are using 3 data structures:

1. Trees *(implemented in **Tree.h**)*
2. Queue *(implemented in **Queue.h**)*
3. Vectors *(implemented in **Vectors.h**)*

Moreover, we are using another file **"Encoding.h"**. This file has the implementation of file reading, the data encryption and the decryption. Moreover, some of the bonus tasks were also implemented which is, after encryption and decryption, you will be shown a menu having the following functionalities:

- Files are same or not?
- Size of original file?
- Size of compressed file
- Size of reconstructed file

We then performed tests using two of test files named ***"file1.txt"*** and ***"Challange.txt"***. Following is the results after running the tests. *(For most accurate results, we ran each file 3 times and calculated its average and store it)*

To run the file, follow the following steps:

- **g++ huffman.cpp**
- **./a.out**

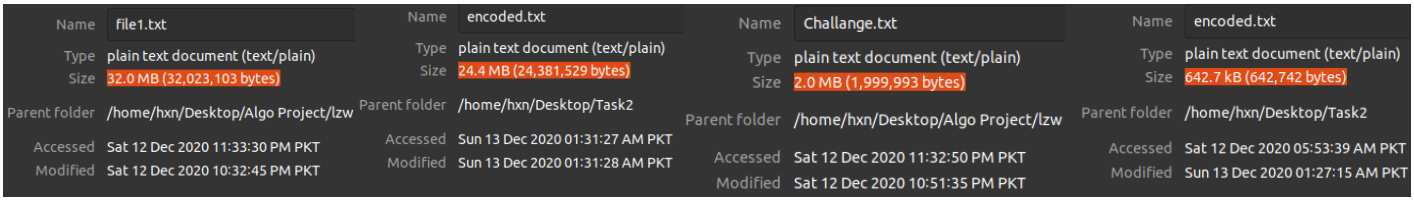
It will then ask you to enter the name of the file. Enter the name **without the extension (i.e., ".txt")**. Enter the name of the file you want to encrypt.

Results:

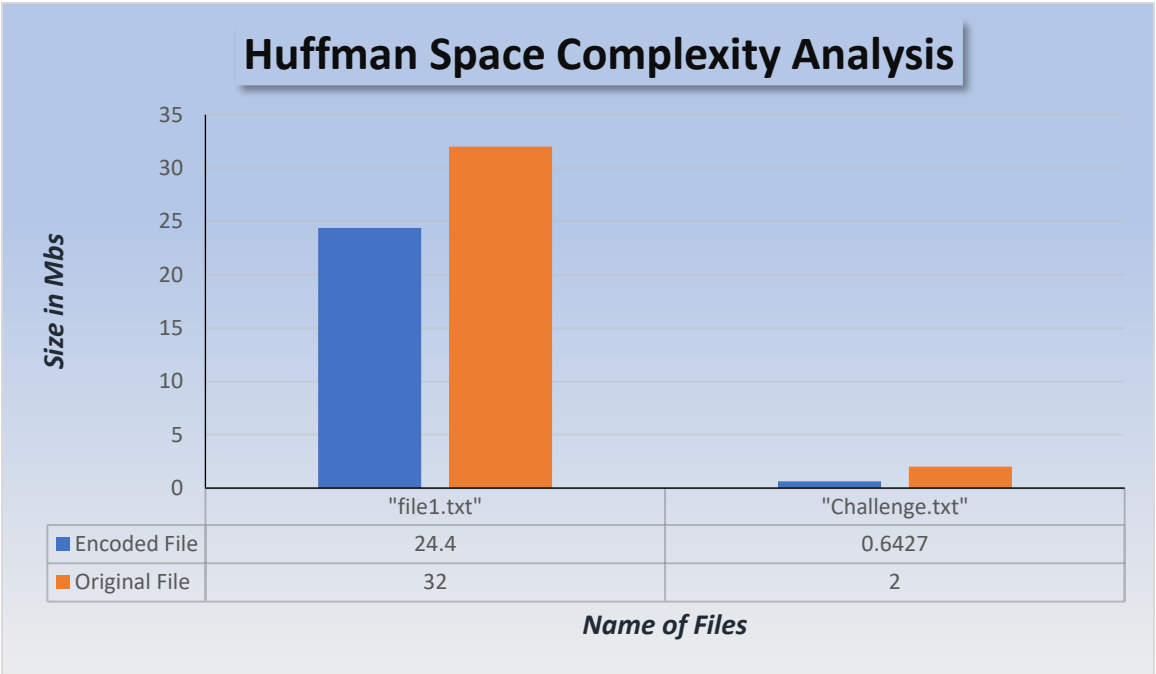
The results of the tests are as follows:

Name of the file	Original Size (MBs)	Size after Encryption (MBs)	% Compressed	Time for Encryption
"file1.txt"	32.0 (32,023,103B)	24.4 (24,381,529B)	23.9%	11.1s (0.19 mins)
"Challange.txt"	2.0 (1,999,993B)	0.6427 (642,742B)	67%	3.1s (0.05 mins)

Note: The "Time of Encryption" includes the time taken to read the test file.



Huffman Encoding Algorithm Benchmarks:



As can be seen from the above graph, Huffman data compression compressed both the files quite efficiently with respect to both time and space as compared to AES Algorithm as seen before. Huffman algorithm however was ***in-efficient to some extent*** while compressing "Challange.txt" due to consecutive repetition of characters.