Lab 3 Report

Compression/Huffman Trees

by Connor McLeod, Matthew Jesus

Mathematical Experiments in Computer Science

November 8, 2023

Section 1:

Data compression is a crucial field where mathematical principles and computer science techniques converge to achieve efficient storage and transmission of information. In this lab, we will explore the close relationship between mathematics and computer science, specifically in the realm of lossless data compression.

Lossless data compression is a technique used to reduce the size of files or data collections without any loss of information. This process involves encoding data using algorithms like Huffman Encoding, and Ziv-Lempel Compression. Both of these algorithms rely heavily on mathematical concepts and data structures, making mathematics a fundamental component of the lab.

One of the primary mathematical aspects in data compression is the concept of frequency analysis. When encoding data using Huffman Encoding, you must first build a frequency chart for all characters in your text. This frequency chart is essentially a mathematical representation of the occurrence of each character in the text. The more frequently a character appears, the shorter its encoded representation in the Huffman tree, resulting in efficient compression.

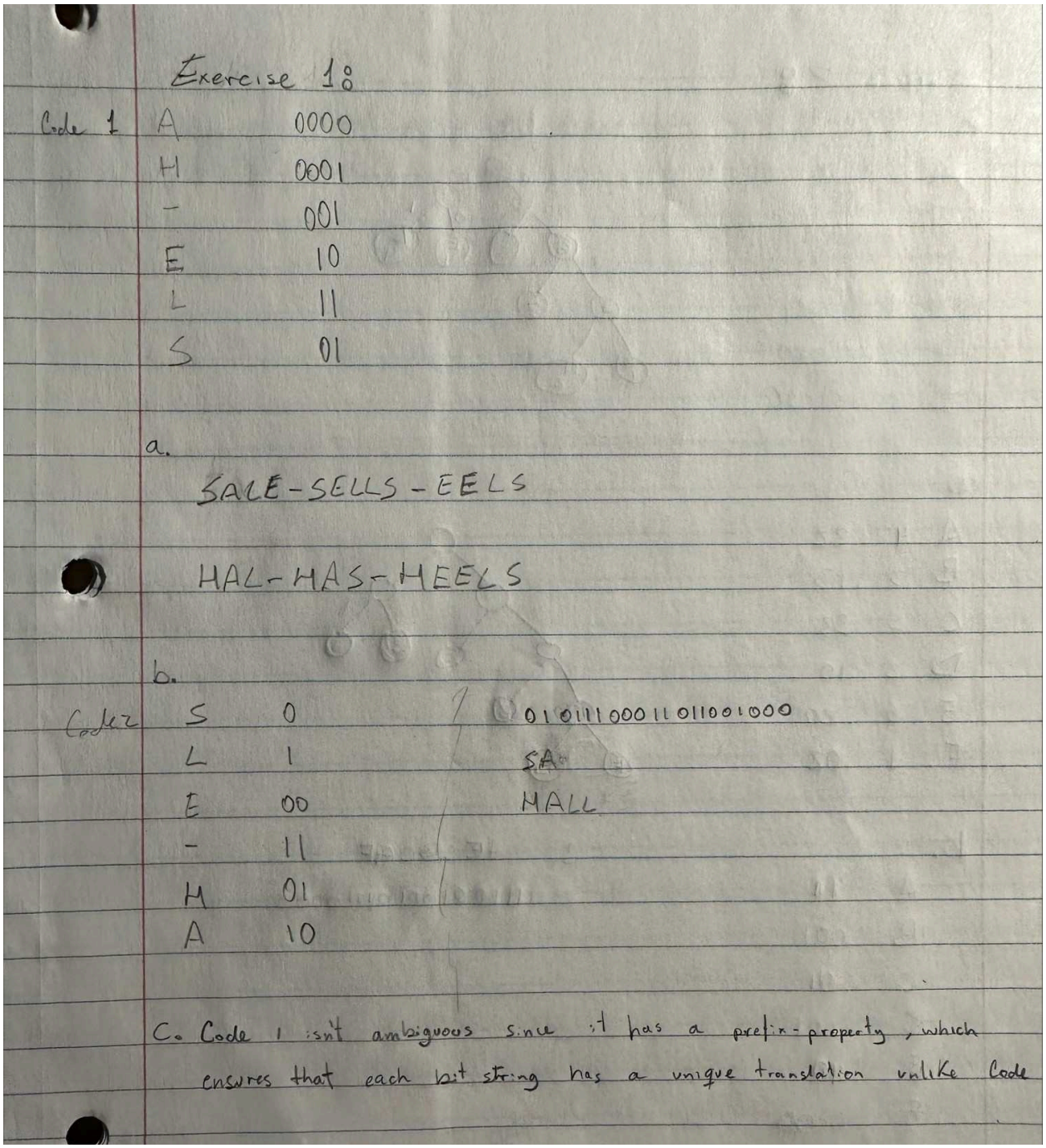
Furthermore, the encoding process itself relies on mathematical concepts of binary representation. The codewords generated during the Huffman encoding are binary sequences, which directly correspond to mathematical concepts of binary numbers. The codewords are assigned to each character in the text, and the efficiency of the compression depends on how these codes are constructed and used.

In "Real Life Compression," mathematics plays a significant role in converting bytes into bits for more efficient storage. This process involves converting eight '0' or '1' characters (bytes) into a single byte using binary arithmetic. The mathematical operations involved in this conversion ensure that the compression is achieved optimally.

Computer science, on the other hand, provides the tools and techniques to implement these mathematical concepts effectively. Students are required to write programs to encode and decode text using the Huffman algorithm. This process involves designing data structures and algorithms to manipulate data efficiently, which is at the core of computer science.

Section 2:

Exercise 1:



Exercise 1:

Code 1

A	0000
H	0001
-	001
E	10
L	11
S	01

a.

SALE-SELLS-EELS

HAL-HAS-HEELS

b.

Code 2

S	0
L	1
E	00
-	11
H	01
A	10

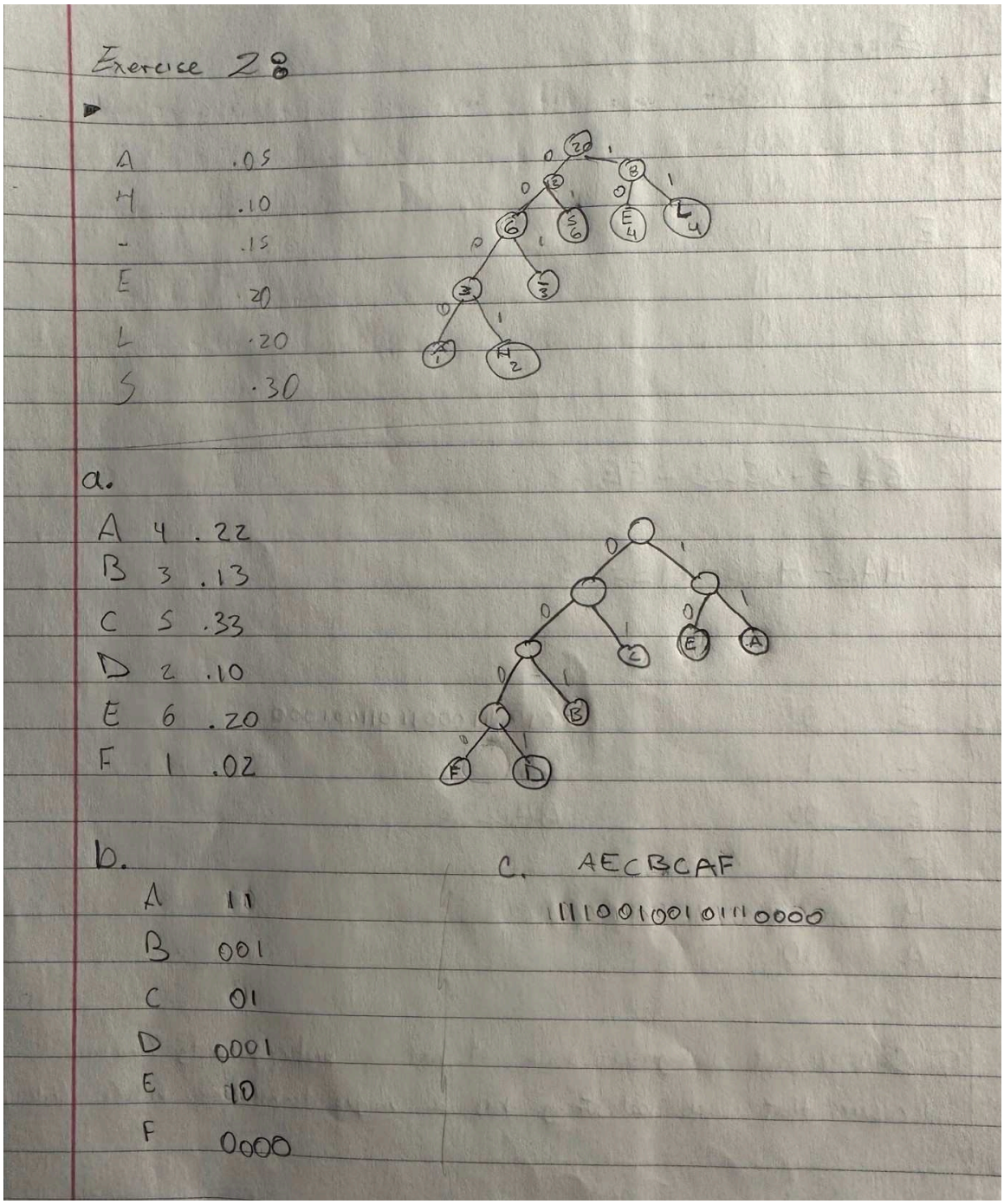
01011100011011001000

SA

HALL

c. Code 1 isn't ambiguous since it has a prefix-property, which ensures that each bit string has a unique translation unlike Code

Exercise 2:

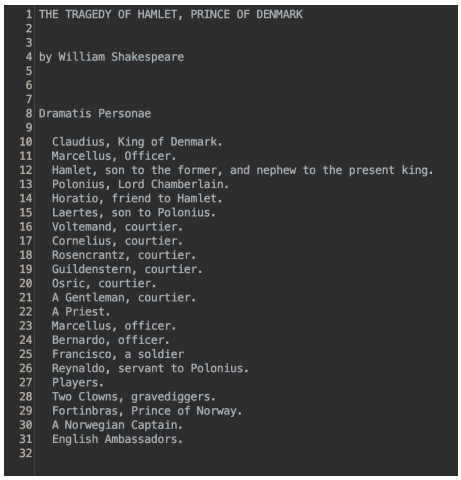


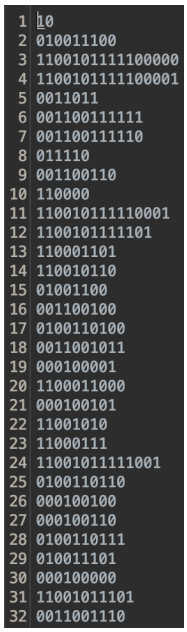
Program 1:

The purpose of program one, is to take in a normal text file and output a ‘compressed’ version of the file that they just provided. The logic behind this process is that the program will create a table that contains the number of times that each letter or symbol was used in the file that was provided. Using that frequency table, the program builds something that is called a

huffman tree. Using a Huffman tree allows the program to assign a code to each possible letter without any repetition. Once the Huffman tree has been built, the program creates an array of codewords for each possible character and then uses the codewords to encode the originally provided file. However, this is a program that lays down the foundation work to be able to compress files and this program actually makes the files larger because each character which is represented by one byte, actually turns into several different characters which each take up 1 byte of storage.

The photos showcase the file before compression (left), the codewords table (middle), and finally the compressed file (right)



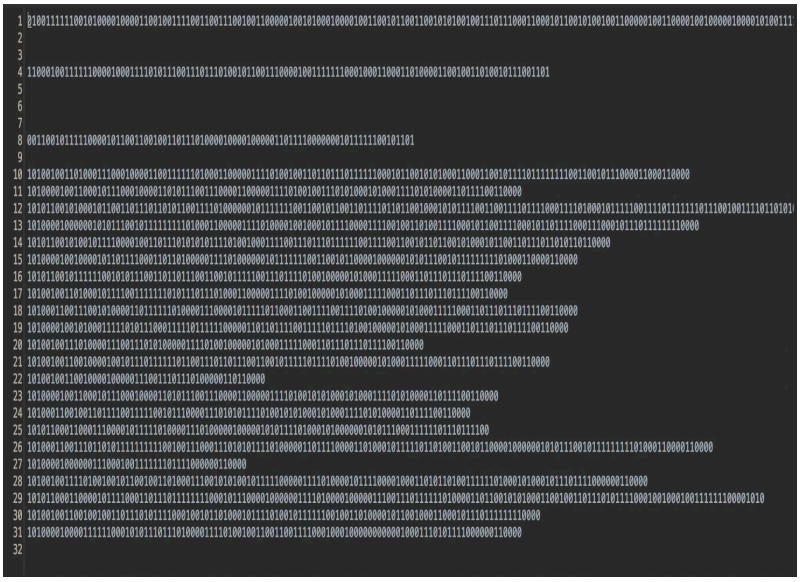


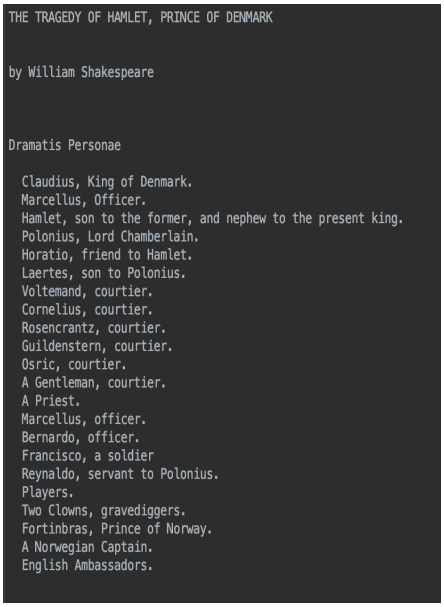
Program 2 & Exercise 3:

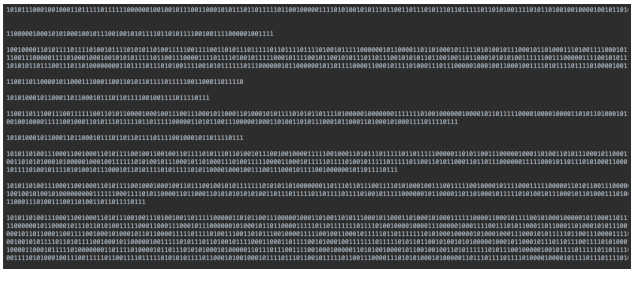
For program 2, the objective was to be able to take in an encoded file (as created in program 1) and decoding it so that you can read it as it was before. The program works by taking the frequency table that was made in the first program and turning it back into a Huffman tree, because as long as it's logic is the same as when it was encoded, you will end up with the

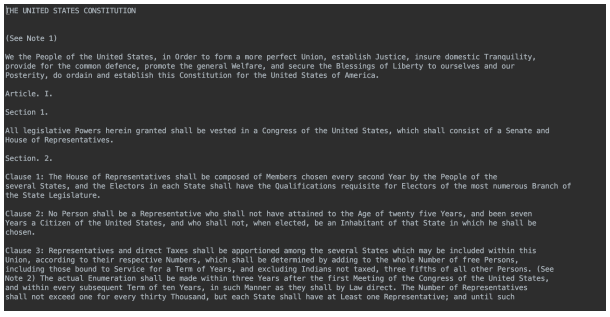
exact Huffman tree. Once the Huffman tree has been built, the code will then use the huffman tree in order to make a codeword table again which finally is used to decode the encoded file.

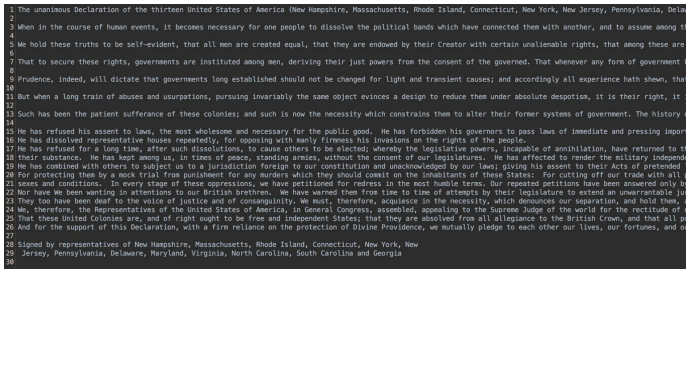
Below you will see a encoded file on the left that has been encoded by program 1, and on the right you will see the same file but decoded by program 2. The files shown will be Hamlet, The Constitution, and finally the Declaration of Independence.







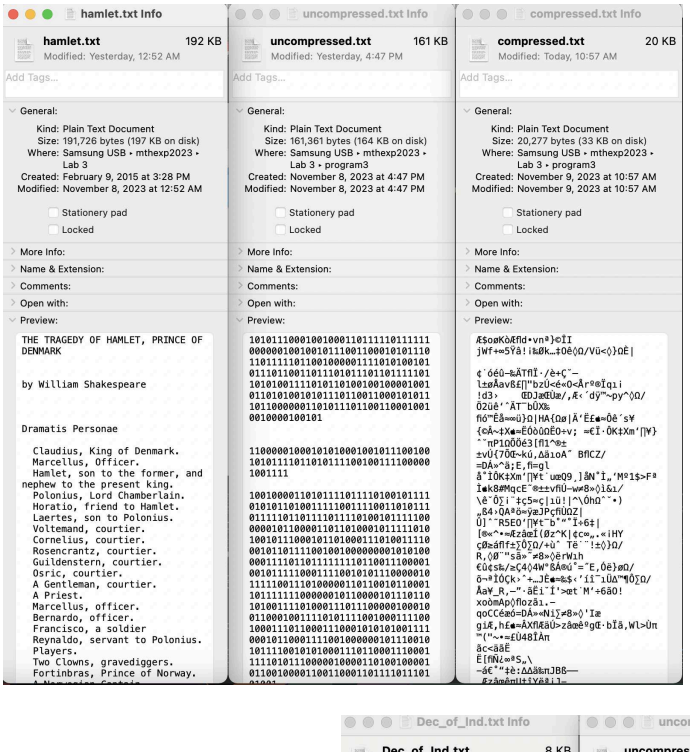


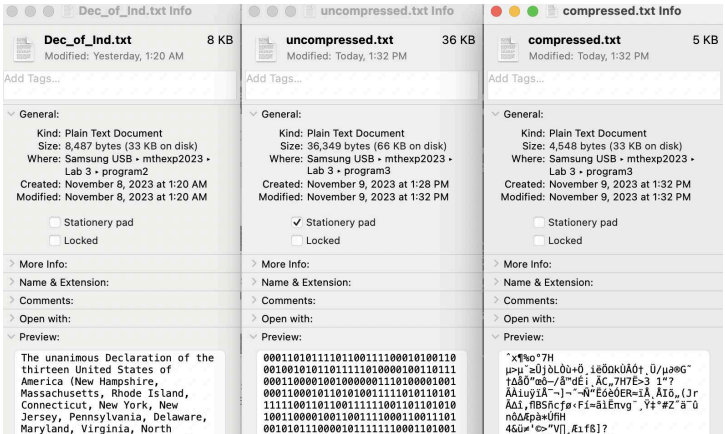


Program 3:

For program 3, the objective was to take the output of program 1 and actually compress it by taking all of the ones and zeroes out of the given file and then taking eight at a time and assigning them to one byte. This will decrease the size of the file because previously, each one and zero was automatically given a size of one byte so by turning eight of them into one, we see a drastic decrease in size. The final step of the program is it finally writes all of the bytes into a file for the user's use. Something that did however surprise me during this lab is that Hamlet and the Constitution compressed down to the same size despite starting with differing sizes. I did also double check this and it ran the same both times.

	Text file	Winzip	Huffman
Hamlet	191,726 bytes	69,800 bytes	20,277 bytes
Constitution	36,267 bytes	11,100 bytes	20,277 bytes
Declaration of Independence	8,487 bytes	3,380 bytes	4,548 bytes





Mathematical Questions:

Exercise 4:

Huffman’s Algorithm produces a code with the prefix property by constructing a binary tree in a way that shorter codes are assigned to shorter symbols. This makes sure that no code assigned to a symbol is a prefix of another symbol's code.

Exercise 5:

Prove: An optimal prefix code for a file is always represented by a full binary tree…*
Let's consider the properties of an optimal prefix code:

1. Shorter codes are assigned to more frequent symbols.
2. Longer codes are assigned to less frequent symbols.

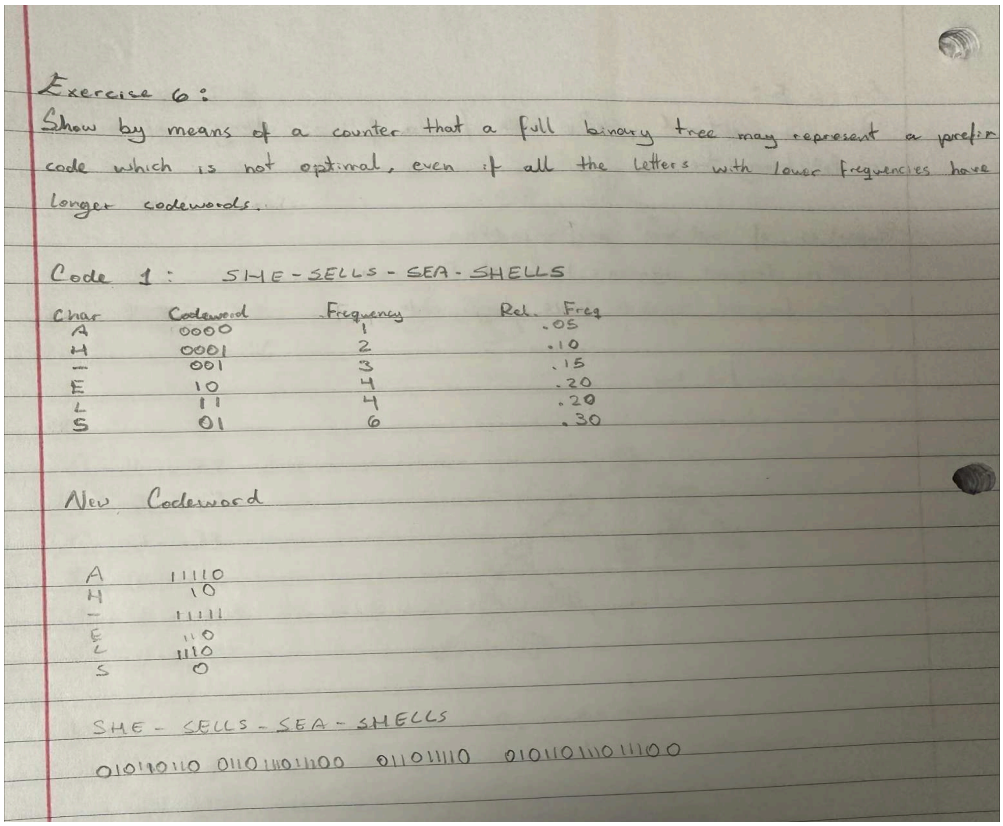
In a non-full tree, we have a non-leaf node "N" with a single child. This child must be a leaf node because all other non-leaf nodes have two children in a binary tree. Let's call this leaf node "L." "L" represents a symbol with a lower frequency because, by definition of an optimal prefix code, the more frequent symbols receive shorter codes.

Now, we can create a better code by merging node "N" and its child "L" into a new node, which has two children, like a full binary tree. The code for this new node would be the concatenation of the codes for "N" and "L."

This merging process reduces the height of the tree because we remove "N" and "L" from the tree and replace them with a single node. This reduction in height implies that the average code length for the entire tree is reduced because the code length for "N" and "L" is replaced by the code length of the merged node, which is shorter. This contradicts the optimality of the original code because we have found a way to achieve a shorter average code length, which should not be possible for an optimal code.

Therefore, we have reached a contradiction, demonstrating that an optimal prefix code for a file cannot be represented by a non-full binary tree. It must be represented by a full binary tree, where every non-leaf node has two children, to ensure that the code is as efficient as possible.

Exercise 6:



Exercise 6:

Show by means of a counter that a full binary tree may represent a prefix code which is not optimal, even if all the letters with lower frequencies have longer codewords.

Code 1: SHE-SELLS-SEA-SHELLS

Char	Codeword	Frequency	Rel. Freq
A	0000	1	.05
H	0001	2	.10
-	001	3	.15
E	10	4	.20
L	11	4	.20
S	01	6	.30

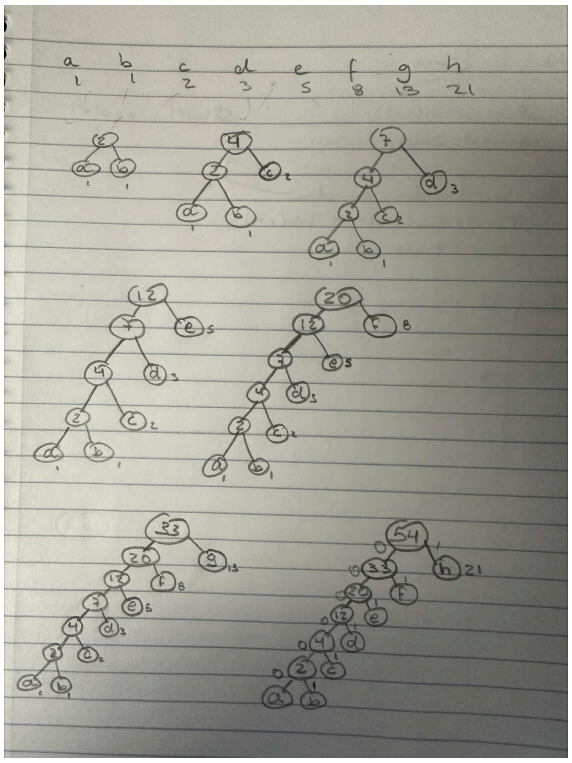
New Codeword

A	11110
H	10
-	11111
E	110
L	1110
S	0

SHE-SELLS-SEA-SHELLS

01011011011100 01101110100 01101110 0101101110111000

Exercise 9:



a.

n	Fib(n)	$\sum_{i=1}^{n} Fib(i) = Fib(1)+Fib(2)+\dots+Fib(n)$
1	1	1
2	1	2
3	2	4
4	3	7
5	5	12
6	8	20
7	13	33
8	21	54
9	34	88

Conjecture:

$\sum_{i=1}^{n} Fib(n) = Fib(n+2)-1$

Base case: $n=1$

$\sum_{i=1}^{1} Fib(1) = Fib(3)-1 = 2-1 = 1$

$k = n$

$\sum_{i=1}^{k} Fib(n) = Fib(n+2)-1$

$k = n+1$

$\sum_{i=1}^{k+1} Fib(n) = \sum_{i=1}^{k} Fib(n) + Fib(n+1)$

$= (Fib(n+2)-1) + Fib(n+1)$

$Fib(n+2) = Fib(n+1) + Fib(n) = Fib(n+3)-1$

Therefore, $\sum_{i=1}^{n} Fib(n) = Fib(n+2)-1$ holds for all positive ints n

b.

n	Fib(n)	C(n)
1	1	0 · 1 = 0
2	1	1 · 1 + 1 · 1 = 2
3	2	2 · 1 + ... = 6
4	3	3 · 1 + ... = 13
5	5	5 · 1 + ... = 30
6	8	8 · 1 + ... = 68
7	13	13 · 1 + ... = 140
8	21	21 · 1 + ... = 284
9	34	34 · 1 + ... = 576

Prove $C(n) = C(n-1) + \sum_{i=1}^{n} Fib(n)$, where $C(1) = 0$

For $n=1$

$C(1) = 0$

For each n, the cost is obtained by adding the cost of the previous tree ($C(n-1)$) with the sum of the first n Fib numbers ($\sum_{i=1}^{n} Fib(n)$).

c. Solve $C(n) = C(n-1) + \sum_{i=1}^{n} Fib(n)$, where $C(1) = 0$

$\sum_{i=1}^{n} Fib(n) = Fib(n+2)-1$

$C(n) = C(n-1) + \sum_{i=1}^{n} Fib(n)$

Substitute $\sum_{i=1}^{n} Fib(n)$ with $Fib(n+2)-1$

$C(n) = C(n-1) + (Fib(n+2)-1)$

$C(n) = (C(n-2) + \sum_{i=1}^{n-1} Fib(n-1)) + (Fib(n+2)-1)$

Simplify:

$C(n) = C(n-2) + \sum_{i=1}^{n-1} Fib(n-1) + Fib(n+2)-1$

$C(n) = C(n-3) + \sum_{i=1}^{n-2} Fib(n-2) + Fib(n+2)-1 + Fib(n+1)-1$

$\vdots$

$C(n) = 0 + \sum_{i=1}^{2} Fib(2) + Fib(n+2) + Fib(n+1) + Fib(n) - n$

$C(n) = Fib(4)-1 + Fib(n+2) + Fib(n+1) + Fib(n) - n$

Use $Fib(n+2) = Fib(n+1) + Fib(n)$

$C(n) = Fib(n+2) + Fib(n) - 1 - n$

$F(n) = Fib(n+2)$

$C(n) = F(n) + Fib(n) - 1 - n$

$C(n) = F(n) + Fib(n) - 1 - n$

Movie Quiz:

1. What was Fermat's "day job," i.e., how did he make a living? **Fermat's day job was working as a magistrate in the Parlement of Toulouse.**

2. Where did Andrew Wiles live when he worked on Fermat's Last Theorem? **He lived in Princeton University.**

3. The statement of Fermat's Last Theorem was written in the margin of Fermat's copy of Diophantus' book on Arithmetic. According to his notes, why didn't Fermat include his proof? **He didn't include his proof because it was too small to be contained.**

4. How many years elapsed (to the nearest hundred) between the statement of Fermat's Last Theorem and its proof? **About 350 years elapsed between both.**

5. What two mathematical subjects that "live on different planets" are bridged by the Taniyama-Shimura Conjecture? **Elliptic curves and modular forms.**

6. Goro Shimura died in 2019 at the age of 89. What happened to Taniyama? **Taniyama committed suicide in 1958.**

7. The song "One Way or Another" that plays in the movie is by what female 1980's punk vocal artist? (singer or group name is acceptable) **The group's name is Blondie.**

8. What did Andrew Wiles' wife say she wanted for her birthday? **She wanted him to prove the theorem.**

9. For how many years (within 3) did Wiles work on Fermat's Last Theorem? **Wiles worked on Fermat's Last Theorem for 3 years**

10. Which mathematician was not mentioned in the movie The Proof? John Coates*, Andrew Wiles*, Peter Sarnak*, Ken Ribet, Noam Elkies, Sophie Germain, John Conway, Pierre de Fermat*, Goro Shimura*, Barry Mazur, Nick Katz*. **Noam Elkies was not mentioned in "The Proof".**