Team 10 report

Compression Table:

The compression tables list 16, 5(short-bit)characters and 46, 8(long-bit)characters. The most common (red highlighted) characters have smaller binary codes do to them most likely being used more

Programs

The first function takes a string of text and finds the binary value for the first character or sequence of characters at the start of a string. It checks to see if the string given starts with any of the entries in the allocation table/compression table. Then maps that part of text into binary. After that, it removes the matched part from the string and returns both the binary code and leftover string.

This works because the REGEX matches the patterns at the start of the string. This allows for the matching of multi_character sequences like, “th” before shorter ones like “t”. This prevents incorrect encoding. By repeatedly applying this function, the entire text can be broken down into binary code represented in the compression table.

The latter part of the program reads the text from the input file and uses the previous function to convert the text into binary. Also keeps track of the total binary string and number of bits. The result is then written into the file “BinOutput.text” in the format D.B. D=number of bits B=binary string.The encoding function runs repeatedly until the entire input text is gone through.

Next, the **decoding program** performs the reverse operation. It reads the encoded binary file, removes the bit count, and reconstructs the original text by translating each binary code back into its matching character using the same compression table. The decoded text is then saved in another file, restoring the original message exactly as it was before compression.

Finally, the **comparison program** evaluates the accuracy and effectiveness of the entire process. It compares the original text file and the decoded file character by character to determine the **percentage of matching characters**. It also calculates the **percentage reduction in file size**, showing how much space was saved by compression. The program prints a summary of the results, confirming that the decoded text matches the original and displaying how efficient the compression was.

Resources and Citation:

Github project examples: https://github.com/topics/text-compression-algorithm

Python dictionary: https://docs.python.org/3/tutorial/datastructures.html, https://docs.python.org/3/howto/regex.html
AI usage: Chat GPT to write up a test based on the functions I created, Gauth to search for potential research materials like sites and other media
Teachers and and faculty: Asked CIS workshop facilitator to help organize ideas, also asked CIS professor for help