Assignment 6 Lempel-Ziv Compression

Writeup

Derrick Ko - Winter 2023

1 Summary

This writeup document must include everything you learned from this assignment. Make sure to mention everything in detail while being as precise as possible. How well you explain all the lessons you have learned in this assignment will be really important here.

1. Explain what you learned
2. Discuss the functionality of LZ78 compression
3. Demonstrate how the efficiency of your compression changes with entropy. Explain the relation.

2 What I learned

2.1 Static Variables

I didn't fully understand how these worked but now I learned it is a variable that is allocated once and persists throughout the lifetime of a program. When a variable is declared as static inside a function or a file scope, its lifetime is the entire duration of the program, but its scope is limited to the function or file in which it is declared. When it is declared as a global variable it means that the variable is visible only within the file in which it is declared.

2.2 Bit-Shifting

This is essentially moving bits left or right. When bits are shifted to the right, the value of the number is divided by 2 raised to the power of the number of bits shifted. If the original number is a signed integer, the sign bit (the most significant bit) is preserved by shifting in copies of the sign bit. This is known as arithmetic right shifting. When bits are shifted to the left, the value of the number is multiplied by 2 raised to the power of the number of bits shifted.

2.3 Trie

A trie, also called digital tree or prefix tree is a type of data structure used to efficiently store and retrieve strings or other data that can be viewed as a sequence of characters. It is similar to a tree, but instead of storing data at the nodes, it stores the data as paths from the root to a leaf node. Each node in the trie represents a single character, and the paths from the root to the leaf nodes represent complete strings. In regards for this assignment it is used as a dictionary to store all the previously seen phrases in the input data. Whenever a new phrase is encountered, the algorithm looks for the longest previously seen phrase in the Trie and outputs its index and the next symbol in the input. If the new phrase is not present in the Trie, it is added to the Trie and its index and next symbol are output.

3 Functionality of LZ78 Compression

LZ78 is a lossless compression algorithm that works by building a dictionary of sequences of previously encountered data, and then encoding new data as a reference to those sequences. Specifically, it uses a sliding window over the input data, and maintains a dictionary of sequences (called phrases) that have been encountered so far. The encoding process works by reading in a new input symbol and searching for the longest previously encountered sequence in the dictionary that matches a prefix of the new symbol. One of its key abilities is to handle repetitive data well. When the algorithm encounters a repetitive sequence in the input data, it can store that sequence as a single phrase in the dictionary and then refer to it multiple times in the encoded output. This can result in significant compression of the input data, particularly when dealing with large amounts of repetitive data. It also has very fast encoding times which makes it very good for thiings like streaming media.

4 Entropy on efficiency on compression

The efficiency of our compression algorithm depends on the redundancy present in the input data. If the input data has high entropy, which is that it has a lot of randomness and no predictable patterns, then the algorithm may not be able to compress the data efficiently. This is because the algorithm relies on finding repeated patterns in the data and encoding them using a dictionary. If there are no repeated patterns in the data, the dictionary will be less effective, and the compressed data may end up being larger than the original data. In contrast, if the input data has low entropy, meaning that it has a lot of redundancy and predictable patterns, then the LZ78 algorithm can be very effective in compressing the data. This is because the algorithm can find repeated patterns and encode them using a dictionary, resulting in a much smaller compressed data size.