Follow the (En)code to De(The)Code

Mylo Lynch

March, 11, 2023

1 Intro

So many files and so little space. How do we store an ever-growing amount of files in an ultimately finite amount of memory? Of course there's always data deletion, but there will always be things we want to archive and can't delete. This is where the magic of compression comes in.

Truly random files can't be compressed (though storing and compressing them would be useless in most scenarios), however, code and even the english language have grammar, syntax, prefixes, suffixes, and many more elements that make data able to be compressed using tricks like word tries and lz78 compression. The structure of words allows for new characters and words to be stored in a way that refers to an index representing a previously used succession of characters, only using the bare minimum amount of data to store everything that needs to be encoded. For example: the words she and shells. The word she is part of the word shells, so they'd share a branch of a trie. In lz78 compression, you'd append characters to previously stored characters to store the word shells. Or at least, this is what I feel I've learned about this concept.

2 Key Takeaways

This was my first time working with a buffer and it makes sense why it is important to this whole concept. Once setting the size of it to the efficient size of 4kb, it'll read through the file, encoding it (or decoding it) and writing once its filled up, or flushing and writing once the file is done (since it leaves an incomplete buffer). This way, the program doesn't have to do as many calls to write, but writes in an optimized size of 4kb bytes. This also implies that if a file is only one bit large its buffer will still take up 4kb of memory.

Tries are essential for encoding, as you go through each node and append new symbols to previously encoded symbols to create a new word that is encoded as a pair consisting of a number and a symbol. Word tables were necessary in order to decode everything that was encoded, and the process for decoding is basically just the reverse of encoding.

I also learned that depending on the endianness of a certain operating system, you either read bits from left to right, or from right to left. Therefore, one must check within their

code which way the bits are supposed to be read, and then flip the bits accordingly. Flipping/getting/setting bits can easily be done with bit shifts, bit masks, and logic operators. This interoperability is part of what makes the internet work in the first place.

3 Conclusion

So much space is saved when archiving unneeded files using a compression algorithm. This makes it an essential tool for the internet to be able to keep running. Though memory and storage keep growing larger and larger, so do the amount of things that actually exist as data and files, and tricks like these are like magic for clearing up space. Decoding, though ultimately being used less than encoding, is just as important though. If you can compress a file but can't bring it back, then you might as well be wasting space with useless data in the first place. P.S. The magic number being "bad back" was extremely relatable.