Sometimes Less is More: LZ78
Assignment 6 Writeup

Sam Leger

March 12, 2023

Professor Long
CSE13S Winter Quarter

1 LZ78

Compression is the act of reducing the number of bits required to represent data. This technique is invaluable, allowing for increased speed and capacity when transferring data. There are two types of data compression algorithms- lossy and lossless. The former compresses more data, with the downside of losing some of it. For this reason, it is used primarily when dealing with audio and video, where drops in quality are less important. Lossless compression is used when the data must remain intact, like when dealing with text files, binary programs, or code. This program contains the implementation of the Lempel-Ziv (LZ78), a lossless compression algorithm that was devised in the late seventies. It works by finding repeated patterns in input data and recording them in a dictionary.

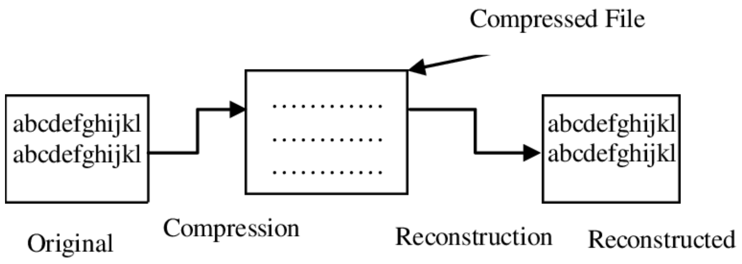


Figure 1: Compression Technique

2 Trie.c

The `trie.c` module was tricky at first but became clear very quickly. I spent time taking notes and visualizing prefix trees and their implementation became simple. I have had some work with trees in previous classes, like CSE30, so it was fairly easy to pick back up on. The main leap in difficulty was the use of dynamic memory allocation in C. When working with Python, dynamic memory allocation is handled automatically by the interpreter via garbage collection. Another aspect that was tough at first was the use of recursion since some of the trie functions lent themselves to the technique.

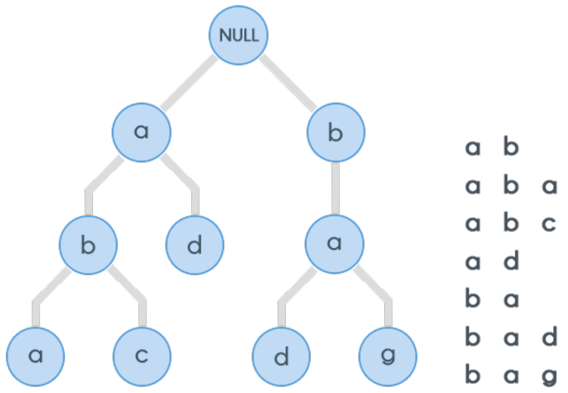


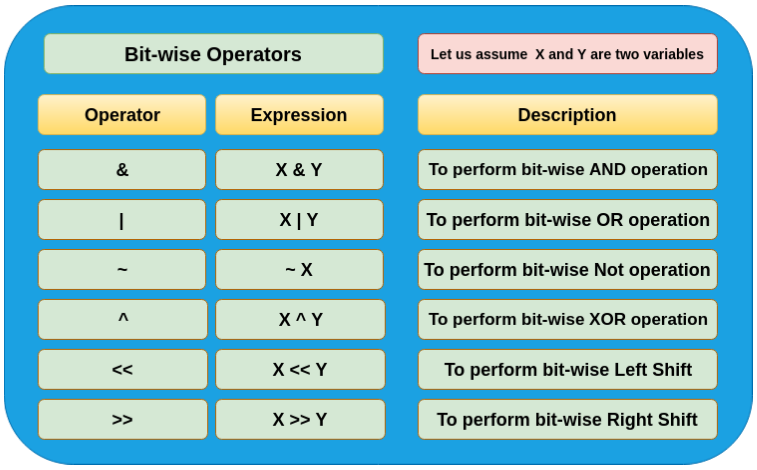
Figure 2: Prefix Tree

3 Word.c

word.c was somewhat tougher than *trie.c*. I had worked thoroughly with dictionaries in the past so the implementation of word tables was intuitive. I understood how to index and iterate through them. Actually, I experienced a minor hiccup in the middle of the assignment, when I realized that my word table reset and delete functions were not working properly. I was trying to call delete on words that didn't even exist, and it all came down to me not reading the instructions properly. Once I came back to the module after a short time, I was able to quickly spot and resolve my issue.

4 Io.c

This module was far and away the most difficult, but I'm sure that was a shared experience in the class. The read/write bytes functions and the header functions made sense to me, but when it came to the pair functions, I was lost. I definitely did not put enough time into understanding buffers, or the bit-wise operations that were fundamental to the module. I spent days on this one module and had bugs up to my eyes. Every time I thought I had resolved an issue, another one sprouted and took hours even to identify. Thankfully, I got help during office hours and tutoring sections, and after a while, I understood the problem (to some extent).



Bit-wise Operators

Let us assume X and Y are two variables

Operator	Expression	Description
&	X & Y	To perform bit-wise AND operation
|	X | Y	To perform bit-wise OR operation
~	~ X	To perform bit-wise Not operation
^	X ^ Y	To perform bit-wise XOR operation
<<	X << Y	To perform bit-wise Left Shift
>>	X >> Y	To perform bit-wise Right Shift

Figure 3: Bitwise Operators

5 Encode and Decode

The two main program files were the easiest to implement. Since I had experience with parsing command line options in the last four assignments, that part was relatively straightforward. This assignment was different in that it used `open()` to get the file descriptors for the respective input and output. After learning how to open, read, write, and close files with the new technique, I was able to quickly implement the rest of the compression and decompression programs thanks to the pseudocode provided by Prof. Long in the assignment document.

6 Closing Thoughts

Honestly, I didn't enjoy this assignment as much as the previous one, because it was less math-intensive and it was less intuitive for me. I did, however, find it really interesting. I had heard of the concept of entropy but I had never really learned about it in depth. One find from this assignment that I thought was really cool was that compression does not always make a file smaller, sometimes actually making it larger due to the overhead that comes with the compression algorithm. It was also great getting hands-on experience with compression and its fundamentals due to how relevant it is in today's world. With how widely used data-compression techniques are today, everything I was working on felt practical, and that was satisfying.