WRITEUP

Andrew Kuznetsov

March 2023

1 Introduction

In this assignment, I was tasked with implementing Lempel-Ziv compression. I utilized ADTs such as Tries and Word/WordTables, and most importantly I utilized buffers to lower memory consumption

2 Trie ADT

In this assignment, we used a Trie ADT to compress the data. A trie ADT is a tree that holds prefixes of words, each vertex of the tree being a symbol in ASCII as well as possessing a unique code and an array of pointers to other ASCII characters. For example the sentence "abbab" we first set up the Trie we can usual the following table to visualize it.

word	code

Then we index the letter "a" since our table doesn't have any inputs we add the letter "a" with code 2 (the starting code)

word	code
"a"	2

Next we index the letter "b" since we don't have a "b" in the trie we add "b" with code "3"

word	code
"a"	2
"b"	3

Now the next letter is the letter "b" as well since we already have a "b" in our trie we don't add b to the trie but we add it to a variable called "seen_word".

Next we have the letter a, we add the letter "a" to our last seen word giving us "ba" we have a "b" in the trie at code 3 so we add "a" with code 4 s a child to "b" as seen below.

word	code
"a"	2
"b"	3
"ba"	4

3 Word/WordTable ADT

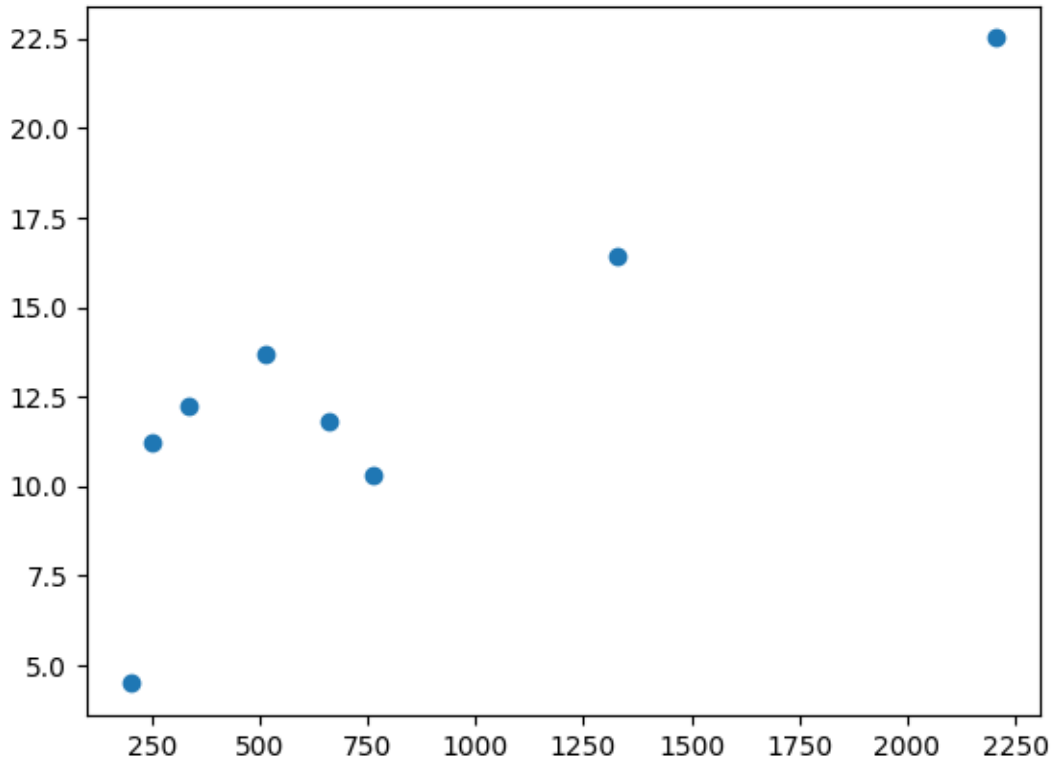
The Word and Word Table ADT is used for the other part of compression and decompression compared to Trie. The Word and WordTable ADT are used specifically for decompressing files, it's also a lot simpler to understand than the Trie ADT. Words have two components a list of syms and a len, the list of syms contains bytes of symbols we read from the compressed files, and len is simply the length of that list. WordTables on the other hand, is a list of Words, in other words, Words are an array while WordTables are a 2d array.

4 Space Saving

Space Saving is a value gotten by using the following formula:

$$Space\_Size = 100 * (1 - \frac{compressed\_size}{uncompressed\_size})$$

Space saving is a value that we calculate to see how much space we save by using the blocks we do in the io.c file. the value compressed_size is the total number of bits we process and the uncompressed_size is the total number of syms we process. Its the true showcase of how crucial compression and decompression are and you can really see this in the graph below.



In this graph, the y values are the Space saving percentages and the bottom values are the number of bits in the file. So as you can see there is a clear upwards trend in the graph, proving the use of compression. Even with such a small amount as 2000 bytes, we save 20% of memory.

Compression is used in everyday life, we might just not notice it. Whenever we download something or upload something to Google Drive that file is compressed to save space, and whenever we download something from Google Drive it's decompressed back to its original size. Compression is truly a magnificent and crucial creation as you can see in the above graph it truly works.