Rodrigo Hernandez #48035611
Rajiv Koya #21027060

CS 165 Project 3 – Analysis

Compile and Run

This program consists of two main files: LZ.cpp and EXPAND.cpp. Also included are header files: ParamChecker.h and DoublyLinkedList.h. To run this program, run the makefile to compile and build the program. From there, run the files using the linux commands as in the project specifications. Expand worked on all test files provided.

	BOOK1	**KENNEDY.XLS**
ENCODE TIME		
DECODE TIME		

Algorithm(s)

Encode:

To encode data, file input is taken as a string and passed to the encode function. The encode function utilizes a brute force algorithm in which the window is represented by a string of max size W-F. Searching the string requires the use of the "find" function. In the case of searching a string, even with a naïve algorithm, linear time will be needed. In this case, it would be O(n*m) where m is the length of the substring being matched in the search. When inserting into the window, linear time will also be needed. This is because the string makes a copy of itself with double the size every time a character is appended to the string. Worst case time taken for encoding will be O(n). Average case will also be O(n) because of the search.

Decode:

To decode, a doubly linked list was used. The list acts as a window of size W that runs in constant time. The doubly linked list allows us to remove and insert items efficiently in and out of the window. The first loop runs based on the size of the file, linear time. The second loop will run in linear time. Expand also uses two helper functions: readBits and bitToInt. Each one utilizes inner loops to translate the bits as necessary. Each loop will run in linear time. This means that in the worst case and average, expand will be take linear O(n) time due to loop constraints.

Worst Case and Average Case (Theoretical Analysis):

Structures - String searching: linear time; doubly linked list: constant time

Loops used:

The "while" loops (iterates until i < counter)

$$T(n) \leq n + (n+1) + (n+2) + (n+3) \ldots + (n+i)$$
$$= n + n + 1 + n + 2 + n + 3 \ldots + n + i$$
$$= n(11 + i)$$
$$= n$$

We do not care about constants; thus, T(n) = n time taken

The "for" loops (n = index_count)

$$\sum_{i=0}^{n} \mathrm{i} = (0) + (1) + (2) \ldots (n)$$
$$= 3 + n$$

$= n \rightarrow$ loop will run as far as n; thus, n queries will be executed