Dov Kassai (dk778) and Noah Green (ncg44)

Asst2: Readme

File Compression

In our program there are a few major components for the case where we are given many of files or directories that we are supposed to compress. The first step in the process, when we are given the files, is that we iterate through all the files and/or directories and collect the tokens. The tokens are stored in a doubly linked list (DLL); at this point in the program we include duplicates of each word. The next step in the process is then compressing the DLL, so that there is only one of each unique word. The way that this is done is we create a hash table, which includes the amount of times that specific word is used. This is useful so whenever we hash a specific word we can then immediately get the times that unique word is used in the files. With the hash table we now have a new list with all the unique words and the frequency of each word. Then we go into the Huffman coding algorithm. This is adding everything to a heap, then taking the two smallest things and merging them together for a binary tree and then adding the binary tree back to the heap. We do this until there is only one item left in the heap. We realized that an AVL would be more memory efficient since it grows dynamically, but the reasons that we used a hash table is because it is easier to code and the runtime is the same (N log(N)). Once everything has been made into a binary tree and added back to the heap, we then go through every word and encode them and put them in a hash table. The reason we used a hash table is because when you want to look up a single word it is going to be a O(1) runtime per word and since there are N word it would be O(N). Now that the hash table is acting as our codebook all we have to do is write the words to the file. After all the steps the total runtime of our program will be (N log(N)).

For the case where we are to compress the files, we go through all the files individually and then do the compression process for that. This is building the list of tokens from that file and then get the encoded version and then write to the file. The way we get the encoded version is that we look at the hash table and that is O(1) time.

For the case where we decompress we go through all the files and instead of tokenizing it we go through it until we get the encoded word. Then we just write to the file.