Matthew Bolda

ECE 368

10/10/19

# Project 2: Milestone 1

For this project we will be using our knowledge about queues and trees to compress and decompress information. The foundation of our code will be Huffman coding. Huffman coding is basically making the most used data easily to access and only using the bits you need. This process can be done using trees in a way that left equal 0 and right equals 1, then every leaf has data.

Beginning with our implementation, we can either count the occurrence of each character individually or could the non-white spaces. It would seem that counting the non-white spaces would be easier however counting individual characters would be more helpful, so I will have to decide which one to do. Once we have that we must create a single tree using greedy Huffman algorithm. Once the tree is complete, we must create a table of bit sequences for each leaf.

Our header file must store the initial information that needs to be compressed or decompressed. This can be done by either storing the character could for each character or store the tree at the beginning and run through it using pre-order traversal. I found it easier to store the count for each character but am not sure I will continue to use this method.

Huff.c is our program that will compress an uncompressed file. Upon completion of this program the file returned should be the same name with ".huff" appended to it. Unhuff.c is basically the opposite of huff.c and requires us to uncompress a file that has already been compressed, this returns a file that has the same name as the last with ".unhuff" appended to it.

My approach to compressing will be to begin with counting each character in the file and how many times that character is used. From there I hope to use a linked list that is ordered from least used to most used. Ideally that linked list will be easy to convert into a tree by taking the smallest items and connecting them to a root until I have a full tree. Once the tree has been completed it is a process of finding the binary path to every character. Now beginning at the start of the file again I parse through the tree to find the needed binary path for each character. Something I struggle to understand is the pseudo EOF and hopefully I can find a way to implement that.

Decompressing is a little trickier and I have not found a great way to implement this. What I would like to do is find a way to label parent nodes and leaf nodes. Once I know difference between parent nodes and leaf nodes it is of finding the end of the tree. You can parse the given tree and convert the binary path into an ASCII character and unhuff the file.

I do not believe that creating the files will cause any difficult, furthermore I understand how to use Huffman code to compress a file. It will take much more time and reading to figure out how to unhuff a file.