Chun-Wei Chen

Assignment #6 Written Report

2012/03/12

I planned my program by reading all the specification completely. I also browsed through the Wikipedia entry for Huffman Coding provided in the specification in order to fully understand the main idea of the Huffman Coding. Because I started the assignment a little bit late, so there are already several questions and the answers associated with the assignment posted in the discussion forum; therefore, I saved a lot of the time before I started coding.

I organized my program by following the specification of the assignment. When the specification says about creating a map, building a tree, or encoding/decoding, I wrote a method to do so. I think the easier part of the assignment is to create maps and using the PriorityQueue since we’ve covered most of the important idea of those things in class. The most difficult part for me is using private recursive methods with an extra parameter except HNode (root). I spent a lot of time to figure out how to write the general cases. For example, I have a private recursive method called fillLookupTable with two parameters, HNode and String. Although I knew I need another parameter to do this at first, but I didn’t know how I should change the String in order to make the subtrees work at first. It really took me a lot of time and drove me crazy. JFileChooser is not difficult for me as well. I looked at the Java tutorial (<http://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html>) so many times and tried really hard to understand how to use it.

I didn’t test my methods immediately after I finished them; instead, I test them after I finished most of them, just before I started the JFileChooser. I initially used a Scanner in the main method to test my program. I created a simple file, which contained “seattle seahawks season,” to test my program. The first problem I encountered was I ran out of my memory while I called my buildHuffmanTree method. I was very surprised. And then I found that although I used the PriorityQueue correctly, I forgot to update the variable p1, p2 I used to hold the two elements I took out from the queue, so the method kept using the same elements and the recursive method never ended. After I updated the variable, it worked fine. Another method I encountered is about my decode method. I used String to hold the character in my assignment, so I just used substring in my recursive decode method to do the general case. However, I just copied and pasted the statement s.substring(1, s.lengt()) for every statement where I needed to use substring and forgot that I should pass in the original string to the method call as an argument. Consequently, when I first decoded the encoded message, it didn’t match the original file’s content. After I fixed that bug, the method worked fine.

Furthermore, I used one of my program’s outputs to determine the other output. I wrote the frequency table and the PriorityQueue on the paper and then used them to draw the HuffmanTree to check if the tree my program produced was correct. The result confirmed. After that, I used the tree I drew to make the lookup table and then find the encoded message. Both results matched with my program’s outputs. Finally, I used the encoded message I got on the paper to find the decoded message, and the result appeared to be the same as original text and program’s output.

This assignment helps me improve my skill of using private recursive method along with public non-recursive method. I’ve feel more comfortable with using extra parameter, such as String object and integer value. Moreover, I’ve realized that Huffman Coding used fewer bits than ANSI and ASCII. Before this assignment, I thought they were all the same and didn’t really care about them. Nevertheless, I find this encoding algorithm is very cool. It’s really amazing that it can store the same information with less memory cost. Even if I spent a lot of time designing this program, I really learn a lot from it.