Tony Diep
4-15-17
PRTL 1120
Design Document

Assignment 12 - Huffman

The purpose of this assignment is to explore the idea of Huffman encoding, which is essentially the mechanism to compress text files by utilizing the “bit” pattern to represent symbols such as the alphabet. However, the assignment will extend to just more than letters in the alphabet. Further, we assign short bit lengths to frequent characters and long bit lengths to infrequent characters to optimize our reduction of the text file’s length. We will take this idea to gain a further understanding of building the Huffman Tree algorithm and the Huffman encoding in terms of the capability of compressing onto a file.

The idea of the Huffman Tree is very similar to a Binary Search Tree (BST) as well as a heap. To simplify and optimize the process of finding the appropriate symbol node in a big Huffman Tree, we will utilize the HashTable data structure and update every node to have a reference to its parent node. This strategy will allow us to, according to the assignment specifications, “not only proceed from root to leaf in the tree, but to follow a path from leaf to root.”

Because I am proficient with JUnit Testing, we will apply the JUnit Test First step on this assignment. This way, we are able to provide ourselves a proficient guideline as to what our program should behave in terms of methods. However, me and my partner, Alex Cervantes, will need to solidify our understandings of the Huffman Tree before starting on the assignment. Thus, we suggest to watch Sedgewick’s videos and the read the provided online text describing the Huffman Encoding process and the Huffman Tree. Additionally, we are to also recall the mechanisms and operations behind the HashTable and Heap data structures since we will incorporate those once again in this assignment. To simplify the process, we will be using Java’s HashTable and Heap data structures.

Once we understand the fundamental concepts behind the Huffman Encoding and Huffman Tree, we are to focus on the static methods that help build the Huffman algorithm. The idea is to understand how to translate provided pseudocode (especially in lecture) into Java implementation. We need to also prepare ourselves to proficient utilize the debugging system because we will expect incorrect values passed onto incorrect references (e.g. incorrect node references). It would also be wise to visualize these operations by either whiteboard or pencil and paper.

Before meeting in class on Tuesday, we are hoping to familiarize ourselves with the whole assignment structure as well as complete some JUnit Tests. If possible, we also plan to finish a couple of static methods to give us a good start to the assignment. Because we are allowed more time to turn in the assignment a couple days after the intended due date, we may take advantage of the extra given amount of time. Otherwise, we are planning on working and completing the analysis of this assignment on Thursday. Nonetheless, we are hoping to complete the implementation between now and the end of Wednesday night.

Again, this may be our last assignment for the semester. If that is the case, then we are hoping that we will proficiently complete this assignment.