Huffman Tree Documentation

Step 1

The first step runs in O(n) time where n is the number of characters in the file. This is because the do..while loop runs once for every character. The time it takes to run is therefore dependent on the number of characters in the file, n.

Step 2

This step runs in O(k logk) time where k is the number of unique characters in the file. The for loop runs k times. Everything inside the loop is independent of k except for the insert function. This function is part of a heap based priority queue, where at the worst runs in O(logk) time. This is when the inserted node goes from the leaf node to the root node, in which case there are log k iterations. Therefore, this step runs in O(k logk) time.

Step 3

The third step runs in O(k logk) as well. As before, k represents the number of unique characters in the file. During each while loop (which runs until there is only one item left in the queue i.e. k times), 2 nodes are dequeued, and 1 node is inserted. This is a net decrease of decrease of one node. As discussed before, the insert takes O(logk) time and for a similar reason, dequeue takes O(logk) as well. Therefore, this step runs in O(k logk) time.

Step 4

This step runs in O(1) time. This is because the last node is dequeued, once, when there is only one item in the priority queue. Therefore, it is independent of the number of characters (unique or otherwise).

Step 5

The fifth step runs in O(k) time, where k represents the number of unique characters in the file. This is because of the for loop runs k times, and the instructions within are independent of k.

Step 6

Step six runs in O(n) time. The instructions within are independent of n. The file is read again and for reasons similar to Step 1, runs in O(n) time.