CS 583 - Project 2 Write Up

Text Used

https://www.gutenberg.org/cache/epub/1661/pg1661.txt

Description and Design Choices

For this project, I used Java to implement the Huffman code algorithm. I was able to create a separate class for each node that will be holding information for the ascii character and its frequency in the text. First, I counted each character in the text file and put it in a frequency table in an ArrayList. Then to create the algorithm, I used the built-in min-heap priority queue and created a custom comparator for the queue to sort each node appropriately. After, I ran the algorithm and returned the root node. For the root node, I used depth first search prefix to run through each leaf node and map it to a hashmap that held the ASCII character and its Huffman encoded string into a key, value pair respectively. Finally, I counted each bit given a fixed length of each character of seven, and also counted each bit using the frequencies and the length of the encoded string to find how many bits are saved using Huffman encoding.

Comparisons

After running the program and finding how many bits were used given both methods, I found that with the fixed encoding length of 7 bits for each character, the total amount of bits were to be utilized is 3,936,576 bits. Now compared to using Huffman code to perform lossless compression, the total amount of bits to be used based on each frequency of the ASCII character is 2,484,605 bits. This saves us 1,451,971 total bits and a 36.88% bit reduction when utilizing this algorithm!