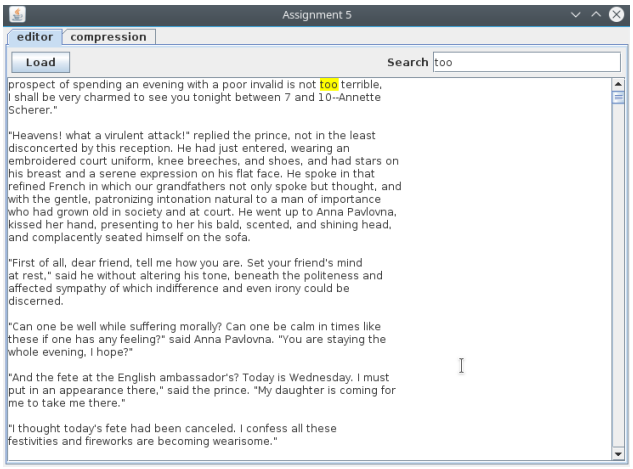
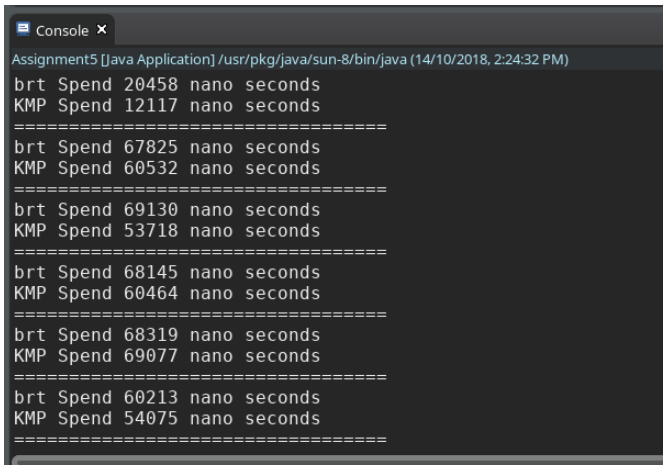
COMP261 Assignment5 Report

Question 1: Write a short summary of the performance you observed using the two search algorithms.





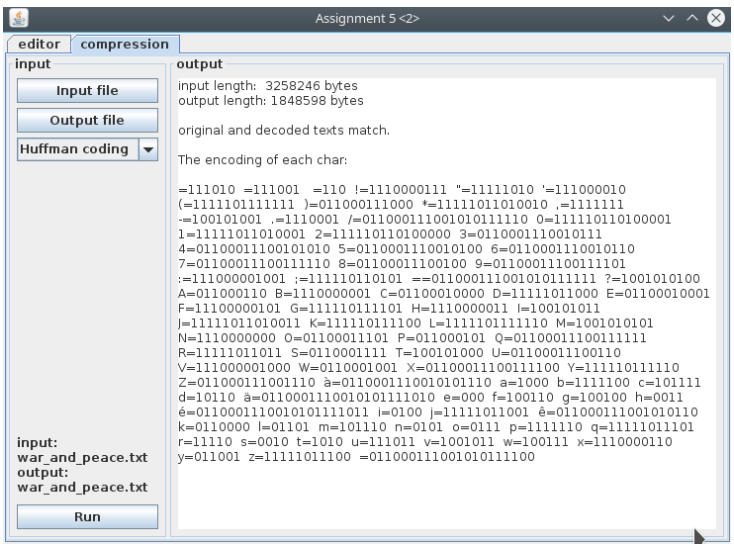
I searched the word “too” six times and print out how long it takes of each time of both ways. The result, as shown above, take the average number of the result:

Bruteforce search used 59,015 nanoseconds in average

KMP search used 51,664 nanoseconds in average

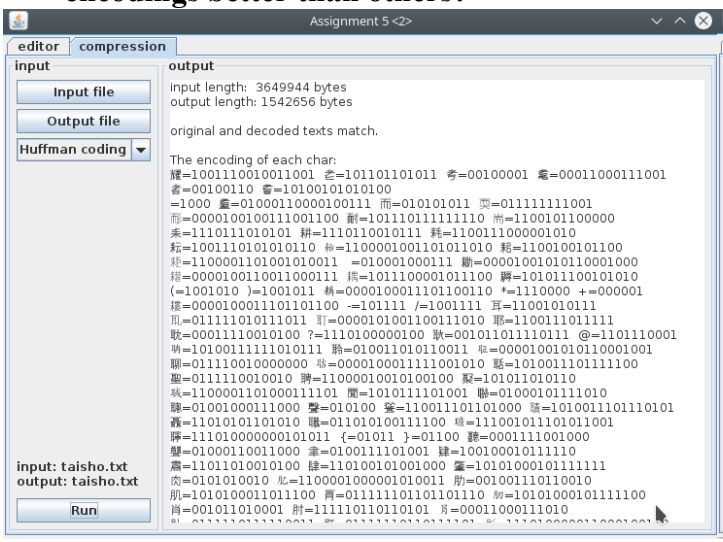
Therefore, sometimes it may take much time to get the result by KMP search, but on average, KMP search is more efficient than BruteForce. Due to the test files are so small, the advantage of KMP search has not shown, even slower than BruteForce sometimes. However, if it is a big file like 1GB or more, KMP search will be much efficient.

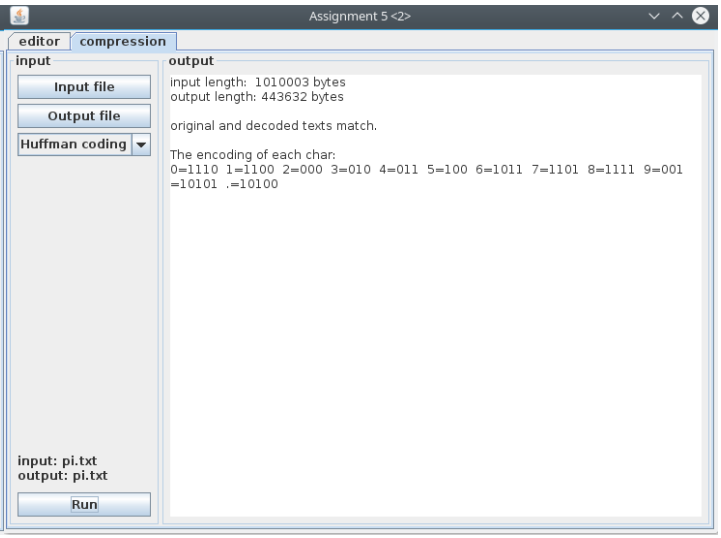
Question 2: Report the binary tree of codes your algorithm generates, and the final size of War and Peace after Huffman coding.



As above, the final size of War and Peace after Huffman coding is 1848598 bytes. And I have also shown the encoding of each char. (0 for left and 1 for right)

Question 3: Consider the Huffman coding of war_and_peace.txt, taisho.txt, and pi.txt. Which of these achieves the best compression, i.e. the best reduction in size? What makes some of the encodings better than others?





As shown on above and Q2, the compression of the three txts are:

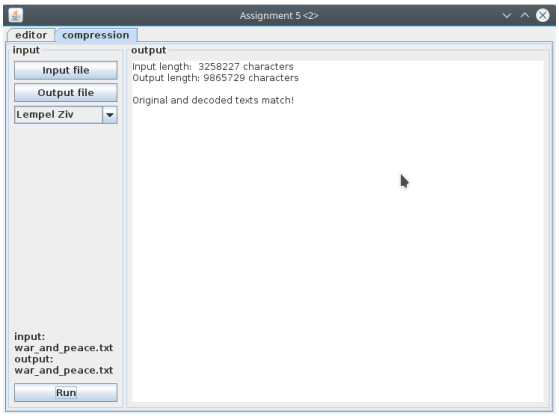
War and Peace.txt: 1848598/3258246 = 56.7%

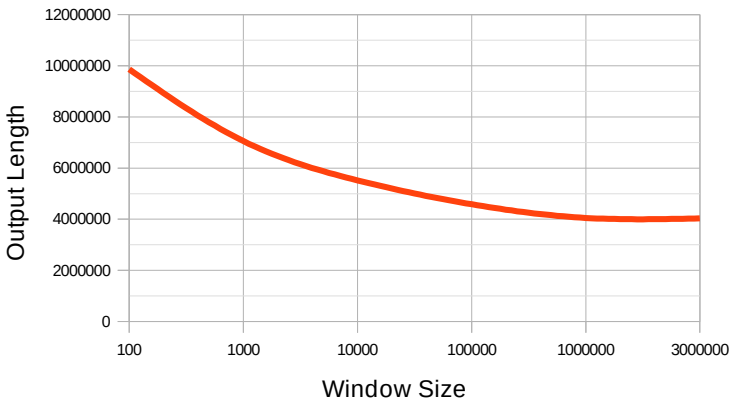
taisho.txt: 1542656/3649944 = 42.3%

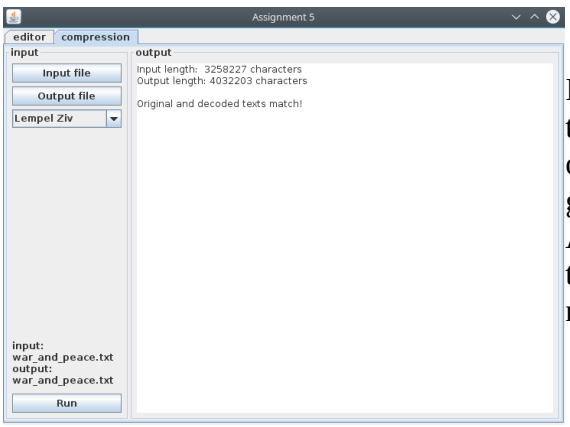
pi.txt: 443632/1010003 = 43.9%

Obviously, Huffman coding achieves the best compression in taisho.txt, which from Chinese Buddhist Canon. Theoretically speaking, Huffman coding should do well with an article which has lots of repeated words like pi.txt. It did well but not as good as taisho, I think the reason is the language, maybe there is some difference between compressing Chinese and English. But compare pi and War and Peace, pi is much better because it has more repeated words and less type of words.

Question 4: The Lempel-Ziv algorithm has a parameter: the size of the sliding window. On a text of your choice, how does changing the window size affect the quality of the compression?

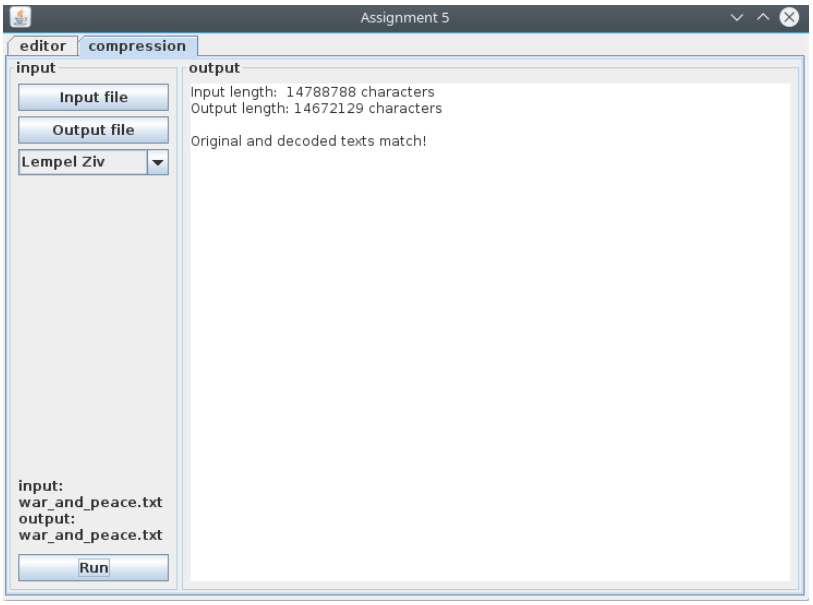






It is easy to see, at first, with the increase in window size, the output length decreases a lot which means the quality of the compression getting higher. Yet when window size greater than 10000, the affection becomes not obvious. Another point is, with the increase of window size, the time it takes become much longer, when window size = 3 million, it spends 30 minutes to compress.

Question 5: What happens if you Huffman encode War and Peace before applying LempelZiv compression to it? Do you get a smaller file size (in characters) overall?



The input length is 14788788 characters and the output is 14672129 characters which are smaller than the input. Though the change is only a little, compare to the result in Q4, it indeed compressed. If it is a huge file, I think the result will be more obvious.