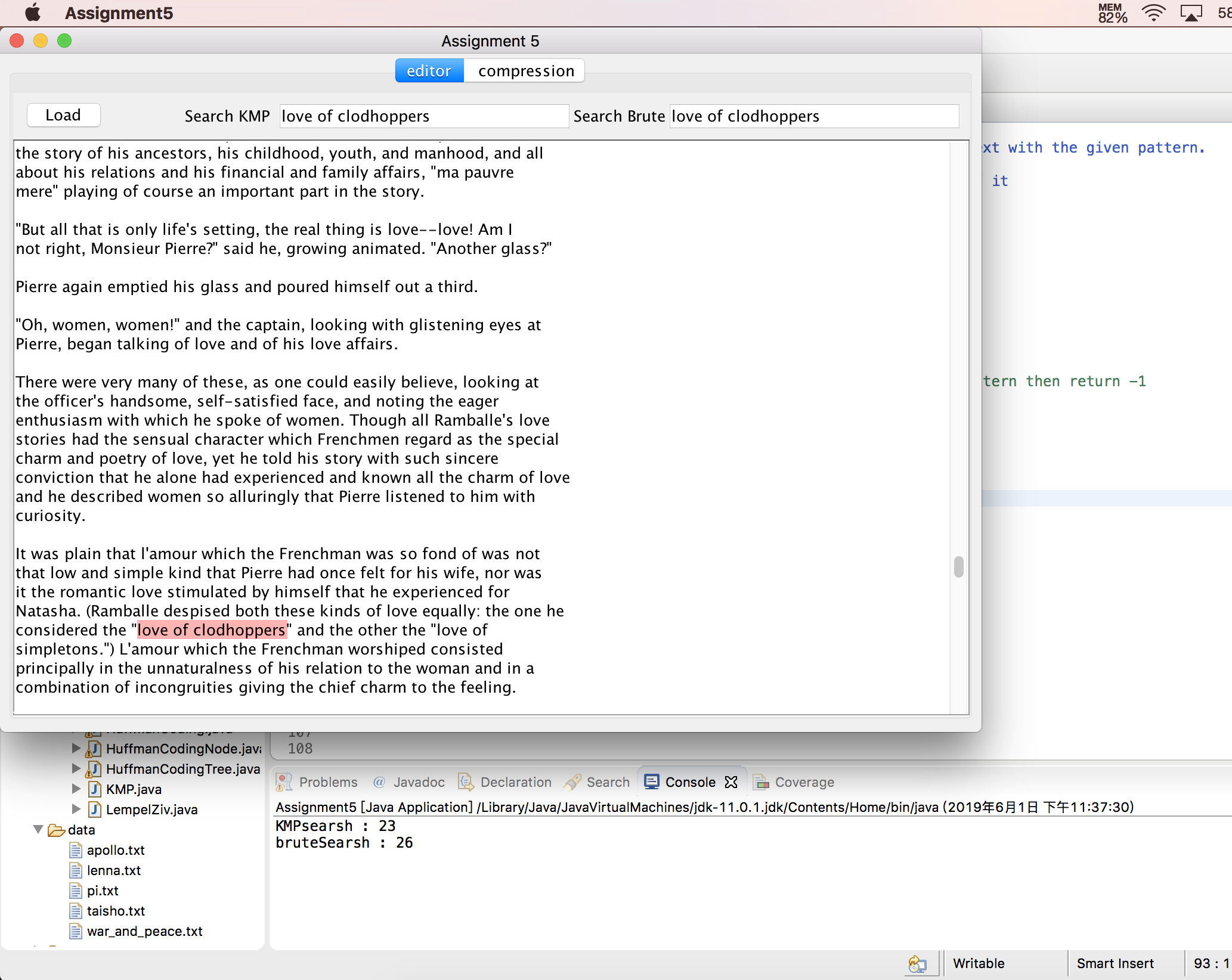
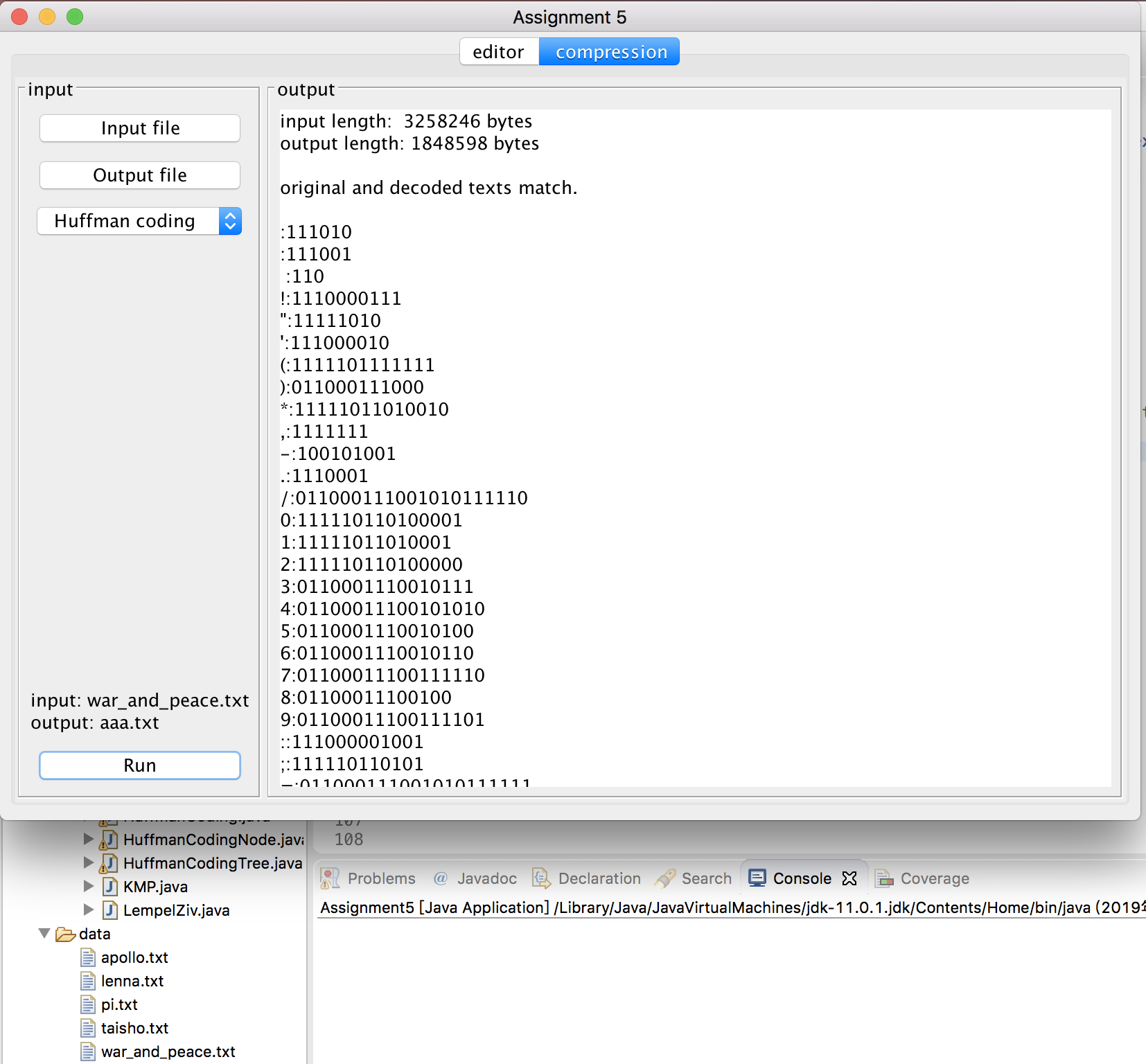
**Question 1**: Write a short summary of the performance you observed using the two search algorithms.

**Answer :**



The time used by KMP is less than Brute Search.

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

**Answer :** 

input length: 3258246 bytes

output length: 1848598 bytes

original and decoded texts match.

:111010

:111001

:110

!:1110000111

":11111010

':111000010

(:1111101111111

):011000111000

\*:11111011010010

,:1111111

-:100101001

.:1110001

/:011000111001010111110

0:111110110100001

1:11111011010001

2:111110110100000

3:0110001110010111

4:01100011100101010

5:0110001110010100

6:0110001110010110

7:01100011100111110

8:01100011100100

9:01100011100111101

::111000001001

;:111110110101

=:011000111001010111111

?:1001010100

A:011000110

B:1110000001

C:01100010000

D:11111011000

E:01100010001

F:11100000101

G:111110111101

H:1110000011

I:100101011

J:11111011010011

K:111110111100

L:1111101111110

M:1001010101

N:1110000000

O:01100011101

P:011000101

Q:01100011100111111

R:11111011011

S:0110001111

T:100101000

U:01100011100110

V:111000001000

W:0110001001

X:01100011100111100

Y:111110111110

Z:011000111001110

à:0110001110010101110

a:1000

b:1111100

c:101111

d:10110

ä:0110001110010101111010

e:000

f:100110

g:100100

h:0011

é:0110001110010101111011

i:0100

j:11111011001

ê:011000111001010110

k:0110000

l:01101

m:101110

n:0101

o:0111

p:1111110

q:11111011101

r:11110

s:0010

t:1010

u:111011

v:1001011

w:100111

x:1110000110

y:011001

z:11111011100

﻿:011000111001010111100

**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?

**Answer :**

war and peace : input length: 3258246 bytes

output length: 1848598 bytes

taisho : input length: 3649944 bytes

output length: 1542656 bytes

pi : input length: 1010003 bytes

output length: 443632 bytes

Then use input/output , we get :

war and peace : 1.7625

taisho : 2.3660

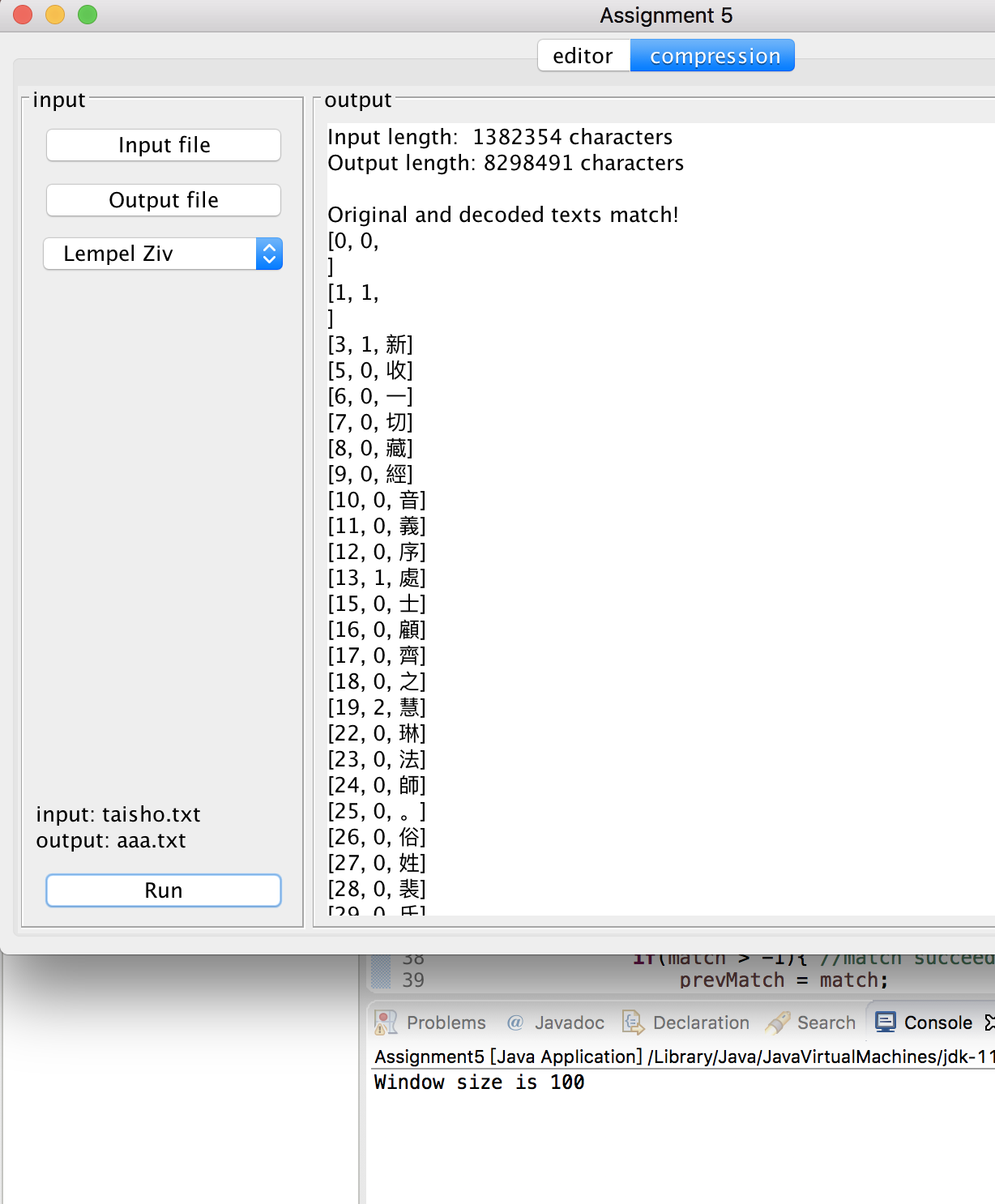
pi : 2.2767

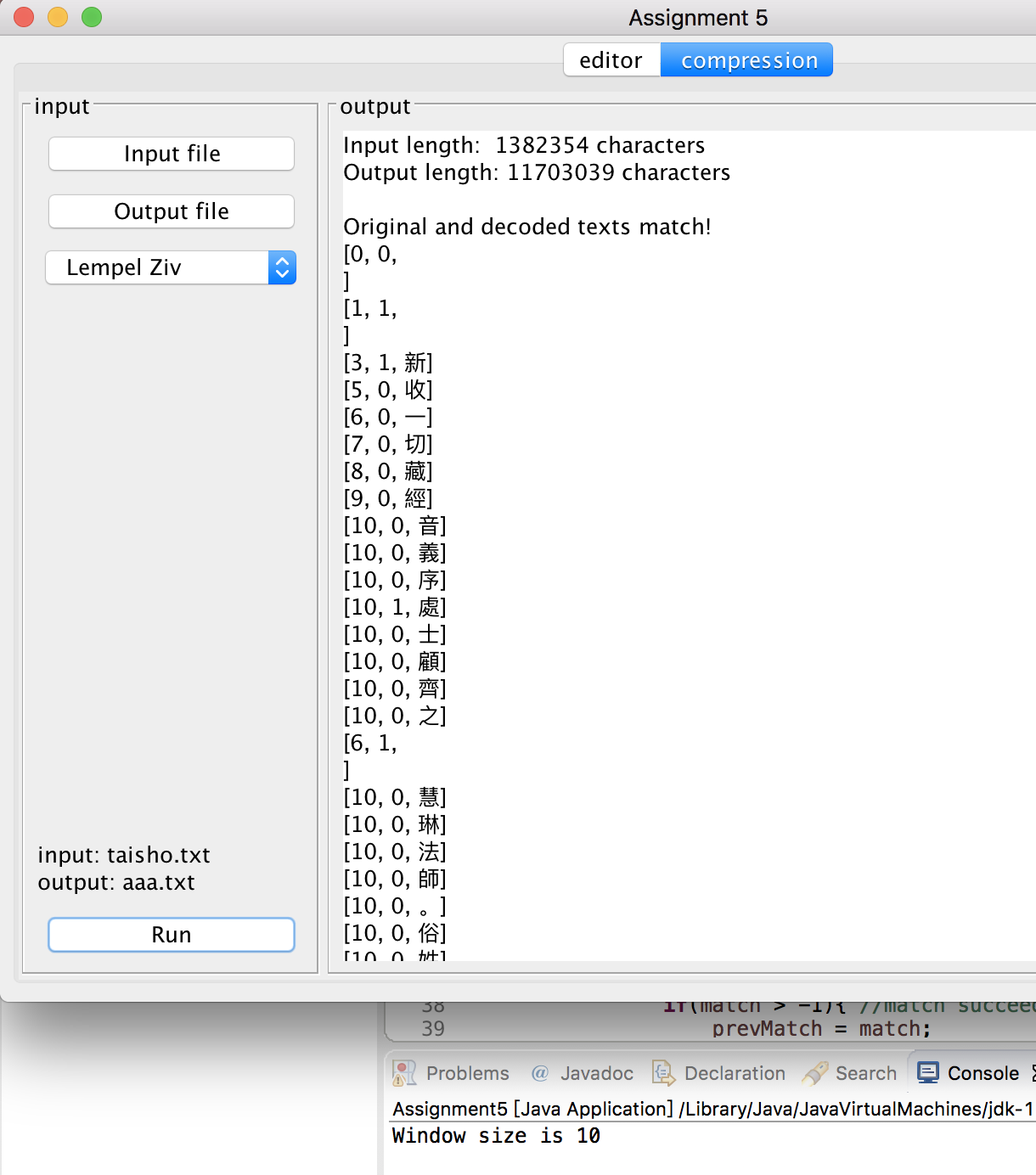
According to this conclusion we can see taisho have the highest value , however in my opinion the highest should be pi , after all pi only have 0123456789 ten numbers.

So I think the reason why taisho is biggest is that chinese character occupy more memory.

**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?

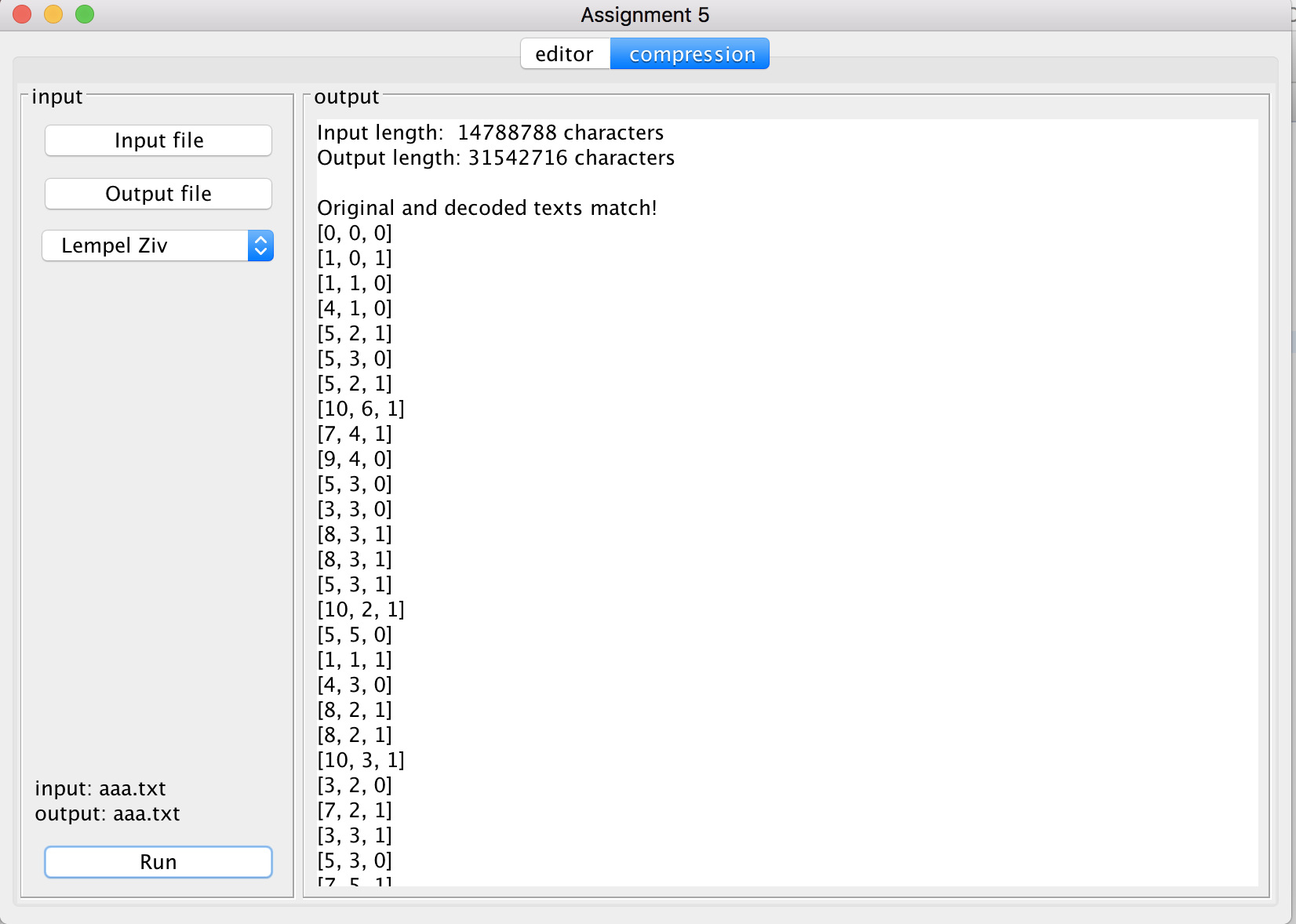
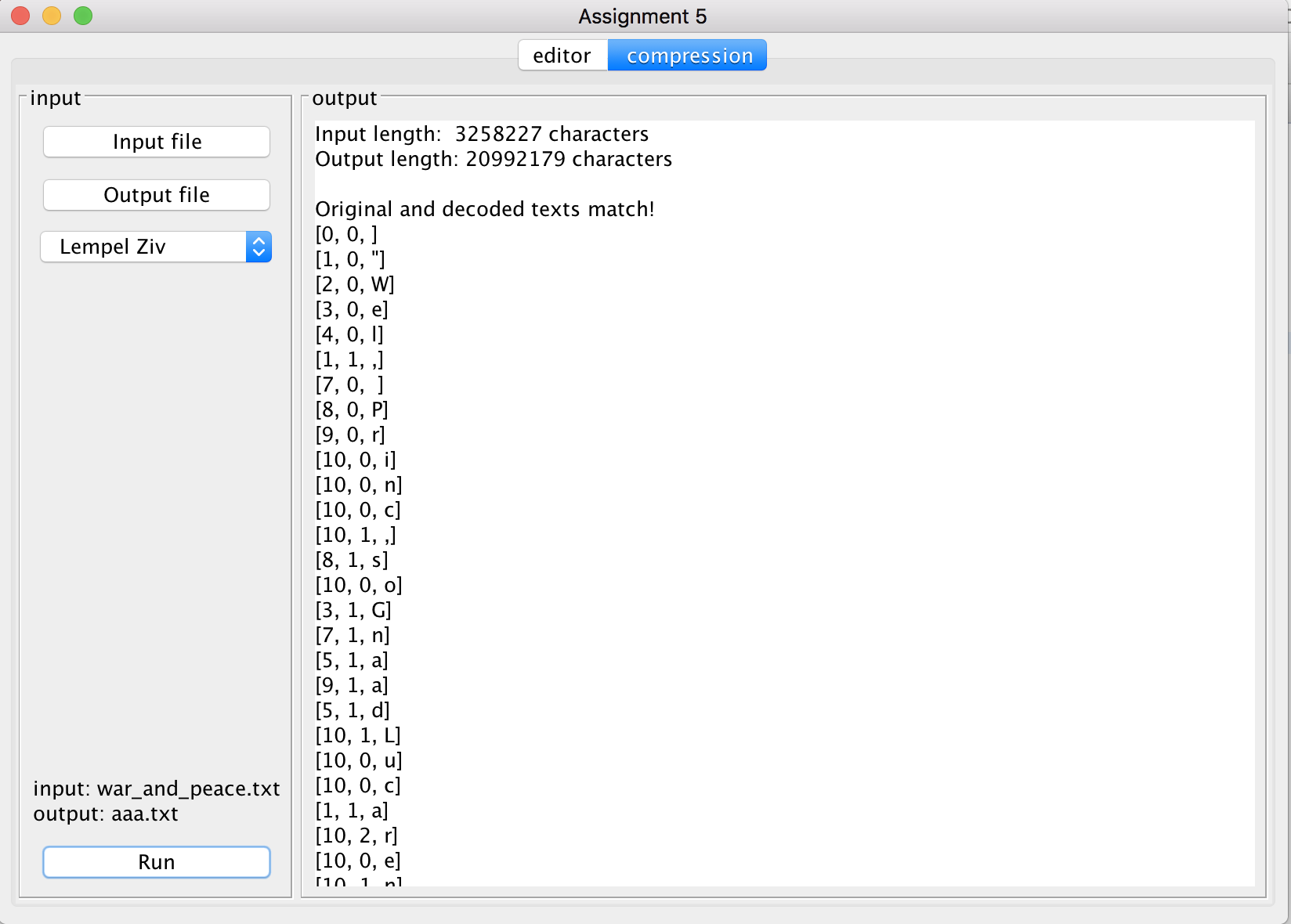
**Answer :**





When we use big value of window size it can apparently reduce the size of tuples and output.

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

**Answer : **

The first picture is use LZ directly without encoding , which size of file is bigger than using huffman code

Yan zichu

300476924