# Coursework for CSC3621 Cryptography

## Part 1

### Exercise 1

Letter Frequencies  
Cipher Text Frequency - { i, x, e, w, m, r, v, s, l, p, h, y, g, q, a, t, k, c, j, f, z, o, u, b, n }

Book Text Frequency - { e, t, a, o, i, n, h, s, r, d, l, u, m, w, c, y, f, g, p, b, v, k, x, j, q, z }

English Letter Frequency - { e, t, a, o, i, n, s, h, r, d, l, u, c, m, f, w, y, p, v, b, g, k, j, q, x, z }

#### Findings

By completing the first coursework exercise I have deduced that the frequency distribution of letters in the English alphabet is not uniform and therefore every encoded text can be exploited by this. Some of the exploitations that can be used are:

* Most English texts have roughly the same frequency analysis with ‘e’ and ‘t’ the first and second most common letter respectively.
* Certain texts if you know the topic or title may lead to decryption – Sherlock Holmes book using the word Watson a lot would increase the frequency of a less used letter like ‘w’.
* If there are spaces within the cipher text you can tell which letters are ‘a’ or ‘I’ as they can stand by themselves.

#### Program

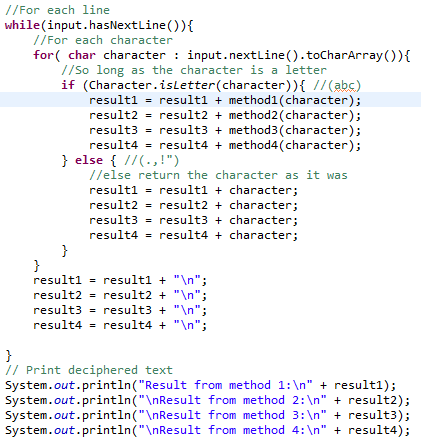
5 Java Classes were used for the first exercise.

* FrequencyAnalysis.java which contains the main method.
* CharacterCount.java which reads in a given text and counts each individual character and stores the information within a TreeMap and an Array. The tree map which maps the character count to the count whilst the array just returns the ordered list of characters from most frequent to least.
* TreeValueComparator.java which is used to overwrite the compare method so the TreeMap is sorted by value instead of key.
* Decryptor.java which has 4 methods which were my attempts at decrypting the cipher. These methods are all called by the constructor.
* StaticHelpers which is a class of static methods to be used throughout the coursework.

#### Steps to cryptanalyze the given cipher text

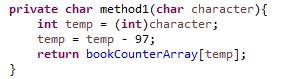
The 4 methods within my Decryptor.java class show how I approached the problem.

Screenshot to show the Decryptor.java constructor. Each character is processed individually by a private method.



Method 1:

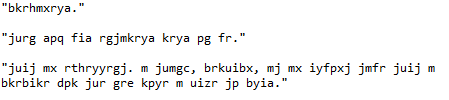
\*Called from Decryptor.java constructor



Method 1 takes each character from the cipher file and converts it to an integer value from 0-25 ( a = 0, b = 1, etc…) . It then uses this value to access the book text frequency array and swaps the values. So where ever there is ‘a’ in the text, it is replaced by the most frequent letter in the book (e). ‘b’ is then replaced with the second most common. This method ended up as a dead-end with the result unreadable as shown below.

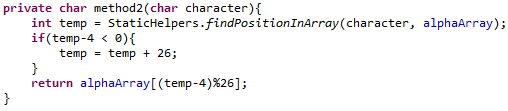
This would be a substation cipher with the key being the frequency analysis of the book.

Output of first 3 Paragraphs



Method 2:

Method 2 involved me looking at the pattern of the text. There are only two letters that commonly form a word by themselves in the English language. These are ‘e’ and ‘a’.



Within the cipher text the two letters that make a single letter word. These are e and m.

I therefore can deduce that the text will need to be shifted so that all e’s are equal to a’s or i’s and all m’s are equal to a’s or i’s. This is likely to involve a Ceaser Cipher like position shift.

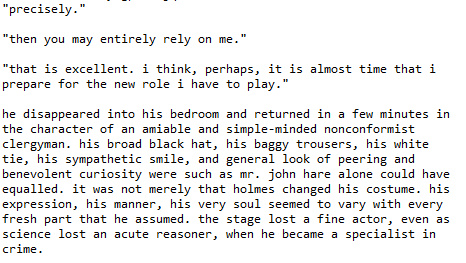
If you take the position of ‘e’ in the alphabet and move it back 4 places you will get an ‘a’. If you take the position of ‘m’ in the alphabet and move it back 4 places you will get an ‘i’ .

Method 2 processes each character, takes 4 away from its position and returns character in that position.

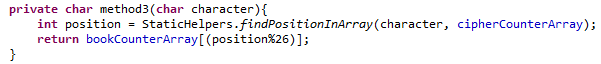
The result of this process is a fully decrypted text. The text was encrypted via a Ceaser Cipher where each characters position is shifted 4 places.

Although I has solved the cipher it did not use the character count of the book or the cipher text. Therefore I continued to look into the links between the two.

Output of the first 3 paragraphs:



Method 3:

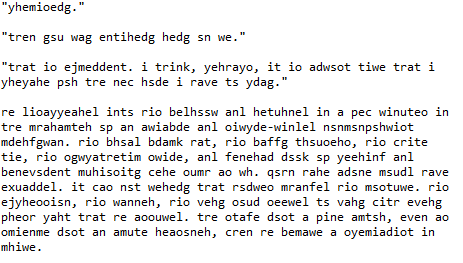


Method 3 introduces the character count of the cipher text and simply does a straight swap between the two. This is another type of substitution cipher.

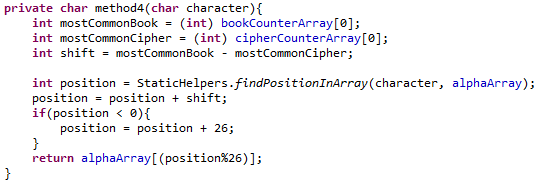
Although this did not solve the cipher text it did come out with an output where the third paragraph could be used to deduce some of the words. I quite easily make out the sentence ‘that is excellent. I think, perhaps, it is almost time that I prepare for the new role I have to play.’

If I had not already got an answer for the cipher I would have filled in the rest of the alphabet using a pen and paper from what was shown by method 3.

First 3 paragraphs from output:



Method 4:



Method 4 uses the two frequency counts and assumes that the most common letter in the book is also the most common letter in the cipher. It then works out how many positions the letter has shifted. It then applies this position shift to each character.

Note: This shift value is calculated for each character despite being the same each time but I wanted to separate each deciphering attempts into their own methods rather and am not too worried about performance.

This method is very similar to that of the second method and fully deciphers the message into plaintext.

Most common letter in the book is an ‘e’, the most common letter in the cipher text is an ‘i’. To get from ‘i’ to ‘e’ you need a shift value of -4. This is what is calculated by the method and the full decoded output is below.

