Cryptography Exercise 1

For this exercise I was asked to produce a Java program to run a frequency analysis on English text and understand how frequency analysis can be used in cryptanalysis.

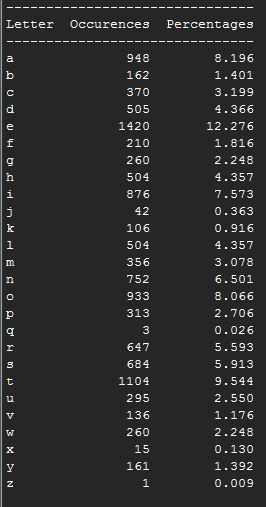
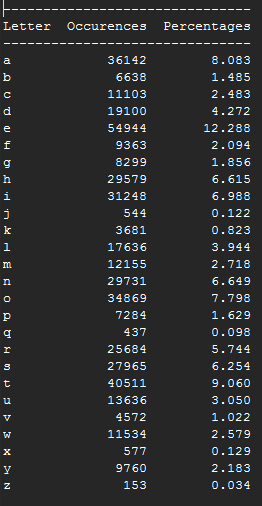
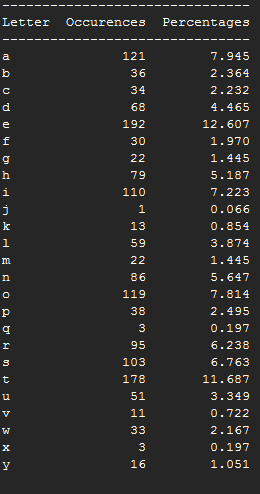
# Frequency Analysis

My first task was to run a frequency analysis on text from the following source:

<http://homepages.cs.ncl.ac.uk/feng.hao/teaching/pg1661.txt>

To do this I created a program with a method “analyseFrequency” that iterated over the all the characters within the text file and counted how many times each individual letter occurred within the text, my function created a hash map of characters to integers (letter to number of occurrences), I decided on a hash map for its O(1) lookup time which would help with analysis of large inputs. I could then use this to calculate the percentage of occurrences each letter took up. To view the results of this process easily I created a method to print a table for these values converting the hash map to a tree map so I can order by the key (letter), I then ran the program using the text files including the one above as input, the results of which are below.

Test File Results from left to right respectively:

* pg1661.txt (test.txt) : 447145 Letters
* test2.txt : 11567 Letters
* test3.txt : 1523 Letters

I then compared the letter frequencies my program produced against the commonly known results found at <https://en.wikipedia.org/wiki/Letter_frequency>.

As shown by the above graph my program outputted values that follow the patterns in the known frequency value on all tests. It would also appear that the longest of the text files (test.txt) produced results most similar to the known values whereas the shorter text files tended to follow the pattern slightly less and produce more extreme values (t on test 3 being a good example of this). However this is to be expected as shorter inputs will have fewer letters which will in turn cause more extreme values.

# Cryptanalysis using frequency analysis

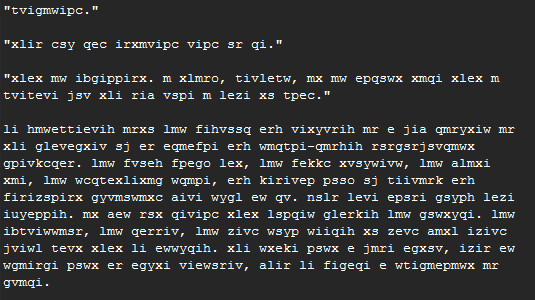
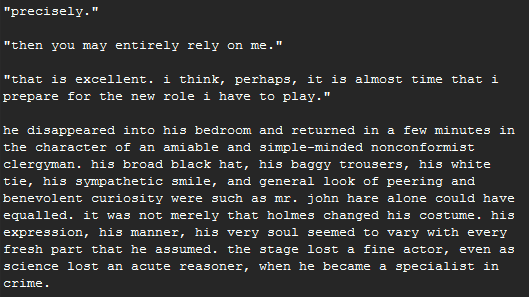
I was then presented with some cipher text and was asked to recover the plain text.

First I decided to run a frequency analysis on the cipher text, what I found was that the frequencies of all the letters did not match the known frequencies for normal English text. But it was clear that they still followed a pattern similar to the known frequencies, only the values had been shifted.

I looked for the most common letter, usually ‘e’ it was now ‘i’. There was also a spike at ‘e’ of 7.774%, followed by f : 1.395%, g : 2.924%, h : 3.654%. I noticed this pattern is very similar to the values produced from a – e in normal English text.

This suggested that the cipher text had been encrypted using a shift cipher as the letter distribution had remained the same. Using this and the fact the most common letter was now ‘i’ I determined that the letters must have been shifted 4 places down the alphabet (e -> i).

I created a method to perform a monoalphabetic cipher on some text and shifted the cipher text -4 places back into its original position. Which produced the following result.



As shown above my program correctly deciphered the cipher text into normal English.

# Conclusion

This exercise has shown how simple monoalphabetic ciphers can be easily broken. Through the use of letter frequency analysis and by taking advantage of the non-uniform distribution of letters within the English language I was able to easily decipher the text.