The Affine Cipher

Reyansh Tellis

1 Section A: General Information and Mathematics

1.1 Overview

The Affine Cipher is a type of monoalphebetic substitution cipher where each letter of the cipher text is mapped to another letter using a simple mathematical function. Each letter is assigned its equivalent numerical value and the encryption function is applied to this number to give a new number: the numerical equivalent of another letter in the alphabet. The encryption key for an affine cipher is on ordered pair of integers, commonly referred to with the letters a and b, where a is a multiplier and b is a constant as expressed in the general encryption function for a single letter using modular arithmetic:

$$E(x) = (ax + b) \bmod m \tag{1}$$

where modulus m is the size of the alphabet, 26 in this case. It is crucial that the values of a and m are coprime in order for the decryption algorithm to work. Therefore, the only valid values of a are:

$$a = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25 \tag{2}$$

The decryption function uses the Extended Euclidean Algorithm in order to calculate modular multiplicative inverses of certain values, where each a value and its inverse a^{-1} satisfy the following equation:

$$aa^{-1} \bmod m = 1 \tag{3}$$

Hence, the general decryption algorithm for a single letter is:

$$D(x) = a^{-1}(x - b) \bmod m \tag{4}$$

Proof that the decryption function is the inverse of the encryption function is shown in subsection **1.2**

1.2 Proof of Inverses

$$\begin{aligned}
D(E(x)) &= a^{-1}(E(x) - b) \bmod m & (1)\\
&= a^{-1}(((ax + b) \bmod m) - b) \bmod m & (2)\\
&= a^{-1}(ax + b - b) \bmod m & (3)\\
&= a^{-1}ax \bmod m & (4)\\
&= x \bmod m & (5)
\end{aligned}$$

2 Section B: My Code

2.1 Overview and Runtime

As a brief overview, my code has 4 core functions, which I go into a more detail in subsection **2.2**. Calculating modular inverses, implementing the affine cipher converter, calculating the most probable outcome and a *main* to bring everything together and let the user decide whether to continue decryption or not. Since there are 12 possible a values and 26 possible b values, the total number of affine ciphers is: $(12 \cdot 26) - 1$, yielding a total of 311 as the mapping: $x \rightarrow 1x + 0$ is unusable as it does not disguise the letters. Therefore, the code is solely based on a brute force method due to the limited *keyspace* that the affine cipher has. Note that the code works best with larger ciphertexts, due to the usage of monogram analysis.

The average runtime of the program is approximately 0.7 seconds. However, the runtime can have slight variations depending on the size of the ciphertext provided.

2.2 Exploring Individual Functions

i. **Modular Inverse:** The modular inverse function uses the the Extended Euclidean Algorithm in order to iteratively search for a valid solution which satisfies the equation. The iteration only exists in the range from $1 \rightarrow 25$ as the results are always taken within the range determined by the modulus. Once a value satisfies the equation, the iteration stops and the modular inverse to the input number to the function is returned.

ii. **Affine Converter:** The affine converter function is the function responsible for taking in 3 inputs: the a value, the b value and the *ciphertext* string and outputting the resultant string after the general encryption formula has been applied. It iterates through all the letters in the *ciphertext* string, calculates the numerical values, applies the encryption formula, and appends the outcome letter to a *final* list which is then joined and returned as output.

iii. **Calculating Most Probable:** This function's main purpose is to calculate the affine cipher shifts of all 311 possibilities and determine, using monogram analysis, which shift most closely matches the standard letter distributions for letters in the English alphabet. The function also finds the a and b values used for the encryption algorithm and returns these values along with the *plaintext* in the form of a tuple.

iv. **Main:** Finally, the main function handles user input, displaying of results and also gives the user an option of trying new a and b values for further manual decryption.