REPORT DI CHIARA DE NARDI E REBECCA GILI

EXERCISE 1

Describe the strategy used for breaking the monoalphabetic cipher. Which were the first guesses? How did you manage to guess the other letters?

We observed the three most common trigrams in the ciphertext, comparing them with both the graph of the most common English letters and the most frequently used trigrams in the English language, looking for correspondences. For example, the most common trigram in the ciphertext was “*FXW*”. We supposed that “*W*”, the most frequent letter in the ciphertext, could correspond to the most used letter in English, “*e*”. This was supported by the fact that the most used trigram in English is "*the*", reinforcing the intuition that “*W*” in the ciphertext could correspond to “*e*”. We could confirm this hypothesis by analyzing the second most frequent trigram in the ciphertext, which is “*ZRN*”. Since “*Z*” is the third most used letter in the cipher and “*a*” is also the third most used letter in the English alphabet, this suggested that the trigram “*ZRN*” corresponds to the word "*and*." While a perfect correspondence between the most frequent letters is not guaranteed, it is highly probable they follow a similar distribution. The same procedure was applied to the third most frequent trigram, “*JRB*”: assuming that “*R*” corresponded to “*n*” as previously supposed, it was likely that “*JRB*” corresponded to the English trigram "*ing*." We substituted these 8 letters in the ciphertext using the replace function. By observing the partially substituted text and keeping in mind the letters that had already been replaced, we made further guesses based on common sounds and digraphs. We continued this process, simplifying our attempts as more letters were revealed, until all letters in the ciphertext were substituted. We noticed that one letter, “*O*”, is not present at all in the ciphertext. Since it was the last letter to be assigned, we deduced it could correspond to “*j*”.

Considering the above strategy, how could you write a function to automate this? Would the provided frequency tables be sufficient for such a function? If not, what other prior information could be useful? (think of the way you made your guesses to answer this)

To automate this process, one could associate a score with each trigram found in the ciphertext, using an iterative algorithm that progressively substitutes the ciphertext letters with different permutations until the score is considered optimal. A score is considered optimal when it is associated with a "translation" that resembles the English language. For this purpose, the provided frequency tables are not sufficient. Indeed, tables of the most frequent English quadgrams would also be necessary, and a dictionary of common words would be useful. However, this method could fail with very short texts, where different letter permutations might still produce readable English text. Moreover, a limitation is that the algorithm doesn't truly "understand" English; it could get very close to the correct decryption without identifying the right permutation as the best one. Finally, the efficiency of this automation depends on the nature of the text, as proper names or invented words could alter the results.

EXERCISE 2

Estimate the complexity of the algorithm used to break the Vigenere cipher as a function of the key length. Is this polynomial in the key length? (You can make assumptions on the maximum key length that you are expecting)

The complexity is determined by the key length and the length of the text to be decrypted. The algorithm can be broken down into two main phases:

- Estimating the key length, which is the more complex phase.
- Identifying the key itself.

To find the key lenght (*keylen*), the algorithm has a polynomial complexity, specifically quadratic, of O($maxkeylen^2$), where *maxkeylen* is the maximum length that the key can assume. This can be explained by considering the operations in the algorithm. The outer loop iterates *Maxkeylen* times (i.e., for every possible key length from 1 to *Maxkeylen*). For each of these iterations, the inner loop iterates a number of times proportional to the key length itself. Consequently, the total number of operations is approximately the sum (1+2+3+...+*Maxkeylen*), which is directly proportional to $MaxKeylen^2$.

(Mathematical note: sum(x) from 1 to n = n(n+1)/2 => order of magnitude n^2)

If you should break Vigenere by hand, using only tools that estimate the frequencies of ciphertext letters as in the case of monoalphabetic cipher, would this be more or less complex than the monoalphabetic cipher? Explain your point in the answer.

A brute-force attack would have a much higher complexity. In fact, if the alphabet has A characters (e.g., 26 for English) and the key has length k, the number of possible keys is A^k. The complexity, in this case, would be O(A^k). This is an exponential function of the key length k, which is not polynomial. The runtime grows explosively with each increase in key length, quickly becoming computationally impossible.

BONUS QUESTIONS

Bonus question: how has cryptogram03.txt been generated? Describe how did you find out.

Cryptogram03 was generated using the Vigenère cipher. We tried to decrypt it with the Vigenère algorithm, rather than the monoalphabetic one, because it was faster and already automated. The result showed that the key is "mylongpassphrase" and the text is in English.

Bonus question: how would you use the “Vigenere cracking algorithm” for attacking a stream cipher where the same key has been re-used for multiple encryptions? (You will be able to answer this after the lecture on stream ciphers)

The goal is to cancel out the keys in the different ciphertexts using the XOR operation. For example, with two ciphertexts, the result will be c1 XOR c2 = p1 XOR k XOR p2 XOR k = p1 XOR p2. At this point, the problem can be solved using the Caesar cipher and frequency tables.