Arcana imperii

Carthage cryptographers intercepted messages sent to Tarraco, and could decrypt many of them since not all of them used a secure encryption system. In addition, they have devised an improved version of the classic Cèsar encryption, which incorporates a word as a password.

We have intercepted messages that have been sent to some scholars who contain old texts, somehow of prophecy that can be important. Fortunately, our spies have given us information that will help us to be able to decrypt the messages: We know a word that appears in the clear text.

Encryption system

First, we need to know the new encryption system. A word is used as a password, where each letter represents a number according to its position in the Latin alphabet, being 0 for the A, and 22 for the Z. Starting with the first letter of the text, this one is numbered with a Cèsar encryption, using the position number of the first letter of the key as the offset. For the second letter of the text, the second letter of the key is used, etc. Once all the letters of the password have been used, it is started again with the first.

The system only encrypts the letters of the Latin alphabet, the rest of the symbols are copied to the encrypted text and do not allow us to pass to the next letter of the key. In addition, it keeps if the letters are lower or upper case. For example, given the " Ab, initio. " Text, encrypted with the CAT key, the resulting encryption would be " Cb, cpinmo. ". To decrypt, the same process is followed, only when calling the Cèsar function, it will be with the decryption mode (negative number).

For more information, you can see a similar system, called Vigenère encryption.

Requirements

- You must implement a function to encrypt / decrypt a text with the explained system (Vigenère adapted, using the Latin alphabet), which has as parameters the text to encrypt / decrypt and the key to use (a string) . You can perform a function that can do both operations (with another parameter to differentiate, for example), or a function for each.
  Tip: you can take the code you have in `cesar.py`, which you will find in Sakai in practice 1, which encrypts / decrypts a message based on

a key (number), depending on whether it is positive or negative. This will be an auxiliary function for the main program.

- Remember that the alphabet is Latin, which contains the following 23 letters: " emph abcdefghilmnopqrstuvxyz".
- The main program must be in texttt python, whose parameter is passed the name of the file with the encrypted text, a known word of the clear text, the minimum length of the key and the maximum length of the key. Your program must find all the possible combinations.

  Example: if the alphabet is only AB, the minimum is 2 and the maximum is 3, then the keys of length 2 are AA, AB, BA, BB, and those of 3 would be AAA, AAB, ABA, ABB, BAA, BAB, BBA, BBB.
- To find the correct key, the function will call the decryption function with the new system, with the generated key, and check if the known word appears in the decrypted text: if so, the program will end and return the emph on key encounter. Optionally, you can display on screen or save a deciphered text file. If the function ends without finding the correct key, it will mean that its length is not between the minimum and the maximum, and will return None.
- You must make a report containing, on the most important algorithm: pseudo-code, theoretical and experimental cost. Keep in mind, for costs, how they vary depending on the length of the text and the length of the key. Additionally, you have to put in the report the key found for each of the texts in Sakai with practice.

Arguments and parameters

The implementation of the application must follow the following syntax:

```
$ python keyfinder.py file_code.txt word_configured min_len max_len
```

Where the `known_word` is a string (without spaces), and `min_len`, texttt max_len are the minimum and maximum lengths of the keys to be tested.

Avaluació

En l'informe de la pràctica, els algorismes principals, les taules i les gràfiques han d'estar comentats.

Els programes han de compilar sense errors ni warnings, i si el text ha estat xifrat utilitzant una clau entre les longituds donades, i la paraula coneguda apareix en el text en clar, ha de trobar sempre una clau (es possible si feu proves, que tingueu falsos positius, no passa res).

Evaluation

In the practice report, the main algorithms, the tables and the Graphics must be commented.

The programs must be compiled without errors or warnings, and if the text has been encrypted using a key between the lengths given, and the word known appears in the clear text, you must always find a key (it is possible if you do tests, which have false positives, nothing happens).

The baremation of the practice (over 10) will be the following:

Costs 3 points, cost analysis of theoretical (recursive and iterative) and empirical algorithms;

Design 3 points, recursive and iterative (the cost of the algorithm will be taken into account);

Implementation 4 points

Delivery

The assignment is in pairs, representing a 20 % of the final note Present the practice at the Virtual Campus of the UdL with two files: a pdf for the report, and a compressed file, `tgz` or `zip` (other compression formats are not supported), for the source code.

The programs that have to be compiled have to do with:

```
$ make
```

The programs are tested by:

```
$ make test
```

No practice will be evaluated that does not have all the files to be able to do the compilation with texttt make and test tests with texttt make test.

The practice could be verified individually on the first day of the laboratory after delivery.