MAT 331-Fall 20: Project 1

In this project, you can use any code from the previous homeworks.

You are **NOT allowed** to use the functions pow, **sqrt**, or any external library except the string library.

You are allowed to use any previous homework code, as long as it does not break the above rule.

For each exercice, each student has to decrypt a unique file. Take your student id, the caesar encrypted files are "**caesar_encrypted_(your id).txt**" where "(your id)" is replaced by your student id number.

For example if my student id is "123456", then the associated filenames are:
caesar_encrypted_123456.txt,
vigenere_encrypted_123456.txt
rsa_encrypted_123456.txt

Exercice 1. *(Caesar cryptography) The file* ***"caesar_encrypted_(your id).txt"*** *has been encrypted using a Caesar cypher (Hint: the first line of the real message contains the word "the").*

1. *Write a program that decrypts it (using a python code) and creates a file with the decrypted message inside "caesar_decrypted_(your id).txt".*

2. *(Math part) Explain your algorithm and describe its complexity.*

Solution for (2). **How we find the right key.**

The decryption key is a letter of the alphabet, the padding number p_0 which is an integer between 0 and 25. If the encryption key is e (an integer between 1 and 26), the decryption padding number $p = -e$ modulo 26. We first write a function **caesar_encrypt(text,key)** which takes a string text, a padding key and returns the decrypted chain using the key. There are exactly 25 possibilities for p. To find the right padding p, we test all the possibilities. The hint shows us that the first sentence contains the word **the**. So we write a function **determine_caesar_key(filename)** which opens the encrypted file filename, reads the first line m, takes for each integer p, decrypts the first sentence using the padding $p = 1..25$ and saves it into the chain m_p and then determines whether the word "the" belongs in the decrypted sentence m_p (using the function str.find()). If it does, then we print the integer p and the string m_p.

Let us determine the complexity of the above function. There are 25 different keys p, and the first line has 54 characters, so the each decryption has complexity 54 (we decrypt 54 characters and change them one by one). The complexity of the function str.find is linear, we apply this function 25 times, overall the complexity is $2 \cdot 25 \cdot 54 = 2700$.

Once we find the right key p_0, then we proceed by decrypting the whole file using the function caesar_encrypt_file(inputname,outputname,key) using the key p_0, which takes the encrypted file inputname, and decrypts it and stores the decrypted message into the file outputname. □

Exercice 2. *(Vigenère cryptography) Alice has a message written in English, with only lowercase letters, breaklines and spaces.*

She encrypts her message in the file ***"vigenere_ encrypted_ (your id).txt"*** *by replacing the letters using a Vigenère cypher with* 2 *characters and transfers it to Bob. You intercept her file and want to decypher her message.*

For example, if her message was :
*"**abcd\nefghi**" and her key is "ab", then the encrypted file would contain "acdd\n ffhhj"*

1. *Write a program that decrypts it (using a python code) and creates a file with the decrypted message inside "vigenere_ decrypted_ (your id).txt".*

2. *(Math part) Explain your algorithm and describe its complexity.*

Solution for (2). We do the same as in the Caesar decryption method.

A key for the vigenere cypher is a pair of numbers (p_0, p_1) between 0 and 25. The total number of keys is $26 \cdot 26 = 676$. We write a function determine_vigenere_key(filename) which reads the first line l of the encrypted text filename. Then for each possible key $k = (i, j)$ where $i, j = 0..25$, we decrypt the line l using the key k and stores it in a string m_k. Then we test whether the string m_k contains the word "the" using the function str.find(). If it does, then we print the key k and the string m_k. This way we don't print the 676 possibles messages m_k but only a few.

Testing on the encrypted file, we find that only one key (p_0, p_1) gives an English message. We then decrypt the whole file using the function vigenere_encrypt_file(inname,outname,key) using the key (p_0, p_1) we just found.

Let us now describe the complexity of the above method. The first line has 43 characters, and for each of the 676 possible keys, we encrypt 43 characters, so the number of character conversion we do is: $676 \cdot 43 = 29068$. Also for each of the decrypted messages m_k, we search whether it contains the word the, so the complexity is proportional to 43. Overall, the total complexity for the function determine_vigenere_key is of the order:

$$2 \cdot 676 \cdot 43 = 58136.$$

□

In the following exercice, you can use the functions *str.strip*(), the function *chr*(), and the function *ord*().

Exercice 3. *(RSA) Alice and Bob communicate using a RSA cypher. We denote by (e, n) the public key, (d, n) the private key only known to Bob. Message has an original message containing 5830 characters. To send the encrypted message to Bob, she proceeds as follows:*

Step 1 She creates a file called ***"rsa_ encrypted_ (your id).txt"****.*

Step 2 She writes in this file "exponent = " and puts her encryption exponent and breaks a line.

Step 3 She then writes "integer n = " and writes the value of n and breaks a line.

Step 4 She converts her message into a sequence of numbers with three digits using the ASCII convention, for example the text with 15 characters "This is a test" would be converted to the list of numbers

$$[084, 104, 105, 115, 032, 105, 115, 032, 097, 032, 116, 101, 115, 116, 046]. \quad (1)$$

Step 5 *She then regroups all the numbers 5 by 5. For example, for the above sentence "This is a test" she would then get the list:*

$$[84104105115032, 105115032097032, 116101115116046] \tag{2}$$

Step 6 *She encrypts each of these numbers using the RSA method by raising each of these integers at the power e modulo n. She obtains a list of numbers and writes them in the file separating each one by a comma.*

You intercept her message. Your goal is to decrypt her message.

1) Write a program that decrypts her message and write the decrypted message inside a file called "rsa_ decrypted_ (your id).txt".

2) (Math part) Explain your algorithm and describe its complexity.

Solution for (2). We know $n = 3387981625011481$ can be written as $p \cdot q = n$ where p, q are two prime numbers. We first write a function find_factor(n) that takes n and determines the two factors p and q. Note that n is of the order $4 \cdot 10^16$, and we test whether each number between 2 and $\sqrt{n}$ divides n, so in our case approximately 2^33 numbers. So the number of euclidean divisions we do is approximately 2^33.

We thus find that the prime numbers p and q are 32452843 and 104397067, and we find these numbers in less than 1 minute.

We store the integers p and q in our program and continue with the decryption and set $N = (p-1) \cdot (q-1)$.

We write a function compute_decryption_key(e,N) which takes the exponent e and N and find the decryption exponent d using the euclidean division algorithm (done in the function bezout()). The complexity of this algorithm is proportional to $log_2(N) \simeq 51.5$. We then store the decryption exponent d into a variable of the same name.

We then continue and write a function fast_expand(m,d,n) that takes a number m, an exponent d and n and computes m^d mod n. The way we do this is by a fast exponentiation method, so the complexity is $O(\log_2(d))$.

Finally, our program rsa_decrypt_file_output(filename,outname) reads each number m in the third line of the encrypted file filename, decrypts m by taking the exponent $m' = m^d$ mod n and then converts m' into a string using the function reconvert. We then print everything into an output file called outname. □