

Universidad Nacional Autónoma de México
Facultad de Ciencias

Criptografía y Seguridad

Semestre: 2025-1

Tarea 1

García Ponce José Camilo 319210536

Equipo Pingüicoders
Arrieta Mancera Luis Sebastian
Cruz Cruz Alan Josué
García Ponce José Camilo
Matute Cantón Sara Lorena

1. Investiga y reporta qué son las Máximas de Kerchoffs. ¿Siguen vigentes?

Son buenas prácticas para evaluar métodos criptográficos publicadas en 1883 en un ensayo (El diario de las ciencias militares y La criptografía militar) publicado por Auguste Kerckhoffs (un lingüista y criptógrafo).

Son las siguientes:

- Si el sistema no es teóricamente irrompible, al menos debe serlo en la práctica.
- La efectividad del sistema no debe depender de que su diseño permanezca en secreto.
- La clave debe ser fácilmente memorizable de manera que no haya que recurrir a notas escritas.
- Los criptogramas deberán dar resultados alfanuméricos.
- El sistema debe ser operable por una única persona.
- El sistema debe ser fácil de utilizar.

Dicen que un criptosistema tiene que ser seguro aun cuando todo lo relacionado al sistema, menos la llave, es de conocimiento público.

Siguen vigentes, aunque no todos los puntos.

El primer punto es muy importante y se sigue en los métodos actuales de cifrado, debido a que existen formas de poder romper los cifrados pero toman muchos recursos en forma de tiempo y espacios para nuestros dispositivos y medios de cómputos actuales, lo cual genera que en la práctica sean muy difíciles o casi imposibles de romper.

El segundo punto también se mantiene relevante, debido a que en la actualidad con el uso del internet y la gran cantidad de comunicación los diseños de estos sistemas son más difíciles de mantener en secreto o que no sean tan conocidos y aun así los métodos actuales de cifrado siguen siendo eficiente sin importar toda la información que uno pueda encontrar en internet o diferentes fuentes.

Los demás puntos ya no son tan relevantes en la actualidad, esto debido a que con el gran avance de la tecnología y por lo tanto de la criptografía, los sistemas se vuelven más complejos en varios aspectos, como los alfabetos, algoritmos y claves que se usan, ya que con el uso de computadoras podemos hacer que realicen operaciones más complejas.

Fuentes usadas para este ejercicio:

curis2ria. (2020, January 30). Los principios de Kerckhoffs y la criptografía moderna.

Curistoria. <https://www.curistoria.com/2020/01/los-principios-de-kerckhoffs.html>

Wikipedia Contributors. (2019, December 17). Kerckhoffs's principle. Wikipedia;

Wikimedia Foundation. https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

2. ¿Cuántas transformaciones afines existen para un alfabeto de 26 letras? ¿Y para uno de 27?

Sabemos que una transformación afín es de la forma $\alpha * x + \beta \text{ mod } n$, donde α y β pueden tomar valores entre 0 y $n-1$, entonces como usamos un alfabeto de 26 letras entonces usamos mod 26 y los valores son entre 0 y 25, entonces α y β tienen 26

posibles valores, de esta manera tenemos $26^2 = 676$ posibles transformaciones afines, pero hay que notar que en Z_{26} no todos los números de 0 a 25 tienen inversos multiplicativos (los cuales son muy importantes para poder generar una transformación de regreso), por lo tanto necesitamos saber cuántos números tienen inverso multiplicado en Z_{26} , sabemos que para que tengan inverso un número n en Z_m n y m deben ser coprimos, por lo tanto tenemos que encontrar cuántos números entre 0 y 25 son coprimos de 26, para esto existe una fórmula muy útil, la Función ϕ de Euler (la cual $\phi(n)$ nos da el número de enteros positivos menores de n que son coprimos con n) y para calcular ϕ se puede usando estas propiedades.

$\phi(p) = p-1$, si p es un primo

$\phi(p^k) = (p-1)p^{k-1}$ si p es un primo y k un natural

$\phi(mn) = \phi(m)\phi(n)$ si m y n son coprimos

Por lo tanto veamos cuánto es $\phi(26)$, sabemos que $26=2^1*13^1$, además 2 y 13 son coprimos (todos los numeros primos son coprimos), por lo tanto $\phi(26) = \phi(2)*\phi(13) = (2-1) * (13-1) = 1 * 12 = 12$.

Por lo tanto hay 12 números entre 0 y 25 que tienen inverso multiplicativo en Z_{26} , por lo tanto α sólo puede tener 12 posibles valores para que la transformación afín tenga inversa, entonces hay $12*26 = 312$ posibles transformaciones afines que sí tienen inversa.

Ahora calculemos lo mismo pero para el alfabeto de 27 letras, primero α y β tienen 27 posibles valores, de esta manera tenemos $27^2 = 729$ posibles transformaciones afines, después calculemos $\phi(27)$, notemos que $27 = 3^3$, entonces $\phi(27) = \phi(3^3) = (3-1) * 3^{3-1} = 2 * 3^2 = 2 * 9 = 18$ (hay 18 números entre 0 y 26 que tienen inverso multiplicativo en Z_{27}) por lo tanto α sólo puede tener 18 posibles valores para que la transformación afín tenga inversa, entonces hay $18*27 = 486$ posibles transformaciones afines que sí tienen inversa.

Fuentes usadas para este ejercicio:

Can. (2021, May 15). Can all affine cyphers be expressed with this formula.

Cryptography Stack Exchange.

<https://crypto.stackexchange.com/questions/89991/can-all-affine-cyphers-be-expressed-with-this-formula>

Wikipedia Contributors. (2020, February 10). Euler's totient function. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Euler%27s_totient_function

Clases y ayudantías

3. ¿Qué pasa si ciframos un texto dos veces, la primera usando un cifrado monoalfabético afín y la segunda usando un alfabeto aleatorio? ¿y si los dos cifrados son afines? ¿Aumenta el espacio de búsqueda si quisieramos romper el alfabeto por fuerza bruta? ¿Se modifican la distribución de las letras?

Caso 1) Ciframos primero usando un cifrado monoalfabético afín y luego usando un alfabeto aleatorio.

Notemos primero que cifrar con un cifrado monoalfabético afín es aplicar una función afín de la forma $\alpha * x + \beta \text{ mod } n$ (donde n es el tamaño del alfabeto, α y β son constantes) y cifrar usando un alfabeto aleatorio es usar una permutación del alfabeto y es un cifrado monoalfabético, por lo tanto su combinación resultará en otro cifrado monoalfabético de sustitución.

También observamos que tanto usar un alfabeto aleatorio y una transformación afín va a preservar la distribución de las letras del alfabeto original, por lo tanto la combinación de ambos también preservará la distribución de las letras ya que es un cifrado monoalfabético de sustitución y estos preservan la distribución de las letras, lo cual nos dice que cifrar primero con transformación affinity luego con un alfabeto aleatorio seguirá siendo muy débil contra análisis de frecuencias, que son unas de las formas que vimos para poder romper los criptogramas cifrados con transformaciones afines y alfabetos aleatorios, por lo tanto no aumentará la complejidad para romper el cifrado, entonces el espacio de búsqueda si lo queremos romper a fuerza bruta tampoco aumentará, ya que notamos que es otro cifrado monoalfabético y usar análisis de frecuencia puede ser muy útil.

Caso 2) Ciframos primero usando un cifrado monoalfabético afín y luego usando un cifrado monoalfabético afín.

Observemos que usar un cifrado monoalfabético afín es aplicar una transformación lineal de la forma $\alpha * x + \beta \text{ mod } n$ (como visto en el caso de arriba), por lo tanto veamos cuando se aplican dos a un texto, digamos que la transformación del primer cifrado es $f(x) = \alpha_1 * x + \beta_1 \text{ mod } n$ y la transformación del segundo es $g(x) = \alpha_2 * x + \beta_2 \text{ mod } n$, por lo tanto aplicar estas dos transformaciones sería como la composición de dos funciones $g(f(x)) = \alpha_2 (\alpha_1 x + \beta_1) + \beta_2 \text{ mod } n$, usando algo de álgebra tenemos que $g(f(x)) = \alpha_2 \alpha_1 x + \alpha_2 \beta_1 + \beta_2 \text{ mod } n$ y acomodando un poco obtenemos esto $g(f(x)) = (\alpha_2 \alpha_1) x + (\alpha_2 \beta_1 + \beta_2) \text{ mod } n$ lo cual es otra transformación afín, por lo tanto usar dos transformaciones afines resulta en otra transformación afín.

Entonces como resulta en otro cifrado afín significa que la distribución de letras no se modifica (las de las letras individuales) ya que los cífrados afines son cífrados monoalfabéticos de sustitución.

Por último veamos el espacio de búsqueda, al resultar otro cifrado afín entonces las mismas formas de romperlo se puedan aplicar aquí por lo tanto en el caso general no debería aumentar tanto el espacio de búsqueda, pero veamos alguno interesante, si tenemos que las funciones afines usadas fueron $f(x) = \alpha_1 * x + \beta_1 \text{ mod } n$ y $g(x) = \alpha_2 * x + \beta_2 \text{ mod } n$ entonces la nueva función afín es $g(f(x)) = (\alpha_2 \alpha_1) x + (\alpha_2 \beta_1 + \beta_2) \text{ mod } n$, pero hay que notar que si $\alpha_1 \alpha_2$ no tiene inverso multiplicativo en Z_n entonces la transformación afín no va a tener inversa y por lo tanto poder descifrar el texto no va a ser posible o sería demasiado difícil, entonces en este caso la complejidad para intentar descifrar el cifrado sería mucha pero en los casos donde $\alpha_1 \alpha_2$ sí tiene inverso multiplicativo en Z_n la transformación $g(f(x))$ si va a tener inverso y por lo tanto las mismas estrategias usadas para un descifrar un cifrado afín van a servir y no se aumentará tanto el espacio de búsqueda.

Fuentes usadas para este ejercicio:

Udofia, E. (2024, August 9). Do affine ciphers preserve frequency - EITCA Academy. EITCA Academy.

<https://eitca.org/cybersecurity/eitc-is-ccf-classical-cryptography-fundamentals/history-of-cryptography/modular-arithmetic-and-historical-ciphers/do-affine-ciphers-preserve-frequency/>

How do I decrypt an affine cipher without a key? (2019). Quora. <https://www.quora.com/How-do-I-decrypt-an-affine-cipher-without-a-key>

Does. (2016, April 4). Does composing multiple substitution ciphers improve security? Cryptography Stack Exchange.

<https://crypto.stackexchange.com/questions/34236/does-composing-multiple-substitution-ciphers-improve-security>

Affine, an. (2016, January 13). Affine Cipher over an Affine Cipher. Cryptography Stack Exchange.

<https://crypto.stackexchange.com/questions/31908/affine-cipher-over-an-affine-cipher>

CS 513 System Security -- Introduction to Cryptography. (2024). Cornell.edu. <https://courses.cs.cornell.edu/cs513/2000SP/L23.html>

Clases y ayudantías

4. Siguiendo las ideas de la pregunta anterior, ¿qué pasa si componemos un cifrado monoalfabético con un cifrado de Vigenère o viceversa?

Al combinar un cifrado Vigenere y uno monoalfabético se aumenta la dificultad para romperlo y la seguridad del cifrado, esto debido a que al usar Vigenere se usan diferentes alfabetos para cifrar el texto y luego aplicar otro alfabeto de cifrado hace que la clave usada para Vigenere sea más complicada de encontrar y por lo tanto intentar descifrar el texto, de manera similar primero usando un alfabeto para cifrar y luego usar Vigenere aumenta la seguridad del cifrado (ya que pasa algo similar)

Veamos algo relacionado con el espacio de búsqueda, este va a ser mayor debido a que para intentar resolver este nuevo cifrado vamos a tener que intentar encontrar la clave usada para el cifrado Vigenere y también el alfabeto usada para el monoalfabético, lo cual puede ser muy difícil como se nota en el artículo Modified Vigenere Cipher to enhance data security using monoalfabético cipher, donde se realizó un criptoanálisis y no logró el texto, en resumen hace más difícil romperlo y lo vuelve más seguro.

Y las distribuciones de las letras va a cambiar algo, sabemos que Vigenere cambia la distribución de las letras ya que usa varios alfabetos para encriptar y los monoalfabéticos no afectan la distribución, por lo tanto al combinarlos la distribución si va a cambiar (debido a aplicar Vigenere), por lo cual hacer análisis de frecuencias no va a resultar tan útil debido a que se usan varios alfabetos, lo cual hace aún más segura a esta combinación de cifrados.

Fuentes usadas para este ejercicio:

Can. (2012, August 23). Can I make a cipher (ex: Vigenère) harder to break? Cryptography Stack Exchange. <https://crypto.stackexchange.com/questions/3621/can-i-make-a-cipher-ex-vigen%C3%A8re-harder-to-break>

Agustini, S., Rahmawati, W. M., & Kurniawan, M. (2019). Modified Vegenere Cipher to enhance data security using monoalphabetic cipher. International Journal of Artificial Intelligence & Robotics (IJAIR), 1(1), 26–32. <https://doi.org/10.25139/ijair.v1i1.2029>

<https://drive.google.com/file/d/1OxHvtV4U14leT9DUMPbDuwsBnY7dzv4b/view>

Clases y ayudantías

5. Para un alfabeto de 26 letras, ¿Cuántas matrices de 2x2 hay que nos permitan cifrar a la Hill? Justifica tu respuesta.

Para resolver este ejercicio usamos el siguiente código (está en ejercicio5.py).

```
# Metodo para obtener el determinante de una matriz 2x2
def determinante(matriz):
    a, b, c, d = matriz[0][0], matriz[0][1], matriz[1][0], matriz[1][1]
    return (a * d - b * c)

# Metodo para encontrar las matrices invertibles en Z_26
# Una matriz es invertible en Z_n si y solo si el determinante no es
# congruente a 0 modulo p, para cada divisor de n
# Dos numeros x, y son congruentes modulo n si tienen el mismo residuo
# al ser divididos por n
# Al dividir 0 entre cualquier numero (excepto 0) el residuo es 0
def matrices_invertibles():
    matrices = []
    divisores = [2, 13]
    for a in range(26):
        for b in range(26):
            for c in range(26):
                for d in range(26):
                    matriz = [[a, b], [c, d]]
                    det = determinante(matriz)
                    if all(det % divisor != 0 for divisor in divisores):
                        matrices.append(matriz)
    return matrices

print(len(matrices_invertibles()))
```

Y obtuvimos que existen 157248 posibles matrices 2x2 para cifrar a la Hill, esto debido para que una matriz 2x2 pueda ser usada a la Hill primero necesitamos que los valores de cada entrada de la matriz sea un valor entre 0 y 25, por lo tanto cada una de las cuatro posiciones tiene 26 posibles valores, lo cual nos dice que hay $26^4 = 456976$ matrices 2x2 en Z_{26} , pero no podemos usar todas las matrices para cifrar, necesitamos que la matriz tenga una matriz inversa en Z_{26} para así poder descifrar el texto, entonces tenemos que eliminar las matrices que no tienen inversa, para esto sabemos que una matriz es invertible en Z_n si y sólo si el determinante de la matriz no es congruente a 0 módulo p, para cada divisor de n (lo vimos en la ayudantía del 12/9/24), además sabemos que dos números x y y son congruentes

módulo n si tienen el mismo residuo al ser divididos por n y por ultimo los divisores de 26 son 1, 2, 13 y 26, para el código no usamos a 1 ya que todos los números x y y son congruentes modulo 1, ya que todos los números son divisibles por 1.

Entonces con estos conocimientos creamos el método para calcular las matrices 2x2 con inverso en Z_{26} y así obtuvimos la cantidad de 15724.

También encontramos un escrito donde calculan lo mismo pero usando el Teorema Chino del Residuo (está en las fuentes), pero como no logramos encontrar fuentes de libros para respaldar esto (y se nos complicó algo entender) entonces usamos el código para calcularlo.

Fuentes usadas para este ejercicio:

Clases y ayudantías

Clases de Algebra Lineal

<https://uregina.ca/~kozdrone/Teaching/Regina/124Winter09/Handouts/hill.pdf> (escrito donde calculan el número de matrices invertibles módulo 26)

6. Tablita

Escoge un texto de al menos 1500 letras, en español, y cifralo usando Vigenère.

Calcula los índices de coincidencia de acuerdo con la siguiente tabla:

I/r	2	3	4	5	6	I
7	0.054155045 09001938	0.051533577 899111525	0.047894088 02330402	0.045090803 07804919	0.044449738 75453271	0.042346295 42407032
8	0.055868207 20640939	0.050322765 698826724	0.046956002 45453968	0.044057106 44880929	0.044532026 96231905	0.041498335 03526246
9	0.054630749 300746156	0.049366655 094071094	0.047061801 57883642	0.045368231 89287172	0.044585318 37307592	0.041340028 19742587
10	0.056224789 44015022	0.050799253 60677052	0.046662115 99815988	0.044859612 39903041	0.042654288 43035635	0.041220122 52322291
11	0.055379180 142993395	0.051459126 66349531	0.046618228 95400716	0.046148010 623799474	0.042929366 153527854	0.040292225 01827974
12	0.054851751 91594377	0.049244398 328217094	0.046373715 422299164	0.043794567 88111	0.044553970 48439541	0.039809467 53259985

Para esto se usó este texto (ya limpiado de símbolos, acentos y mayúsculas)
“enlagranjahabiaungranalborotolospolluelosdemamapataestabanrompiendoelcascar
onunoaunocomenzaronasalirmamapataestabatanemocionadaconsusadorablespatito
squeñonotoqueunodesushuevoselmasgrandedetodospermanciaintactoalaspocasho
raselultimohuevocomenzaroarompersemapatatodoslospolluelosylosanimalesdelagr
anjaseencontrabanalaexpectativadeconocerapequenoquetardabaennacerderependente
delcascaronsalounpatitonuyalegrecuandotodoslovieronsequedaronsorprendidosest
epatitonoerapequeniamarilloytampocoestabacubiertodesuavesplumasestepatitoer

agrande gris y en vez de esperar a que graznido cada vez que hablaban a mamá una corneta vieja aun que nadie dijo que los polluelos estaban demasiado feos. Pasaron los días y todos los animales de la granja se burlaban de él. El patito feo se sintió muy triste y una noche escapó de la granja para buscar un nuevo hogar.

El texto limpio tiene 1598 letras.

El texto original es este

"En la granja había un gran alboroto: los polluelos de Mamá Pata estaban rompiendo el cascarón.

Uno a uno, comenzaron a salir. Mamá Pata estaba tan emocionada con sus adorables patitos que no notó que uno de sus huevos, el más grande de todos, permanecía intacto.

A las pocas horas, el último huevo comenzó a romperse. Mamá Pata, todos los polluelos y los animales de la granja, se encontraban a la expectativa de conocer al pequeño que tardaba en nacer. De repente, del cascarón salió un patito muy alegre. Cuando todos lo vieron se quedaron sorprendidos, este patito no era pequeño ni amarillo y tampoco estaba cubierto de suaves plumas. Este patito era grande, gris y en vez del esperado graznido, cada vez que hablaba sonaba como una corneta vieja.

Aunque nadie dijo nada, todos pensaron lo mismo: "Este patito es demasiado feo". Pasaron los días y todos los animales de la granja se burlaban de él. El patito feo se sintió muy triste y una noche escapó de la granja para buscar un nuevo hogar.

El patito feo recorrió la profundidad del bosque y cuando estaba a punto de darse por vencido, encontró el hogar de una humilde anciana que vivía con una gata y una gallina. El patito se quedó con ellos durante un tiempo, pero como no estaba contento, pronto se fue.

Al llegar el invierno, el pobre patito feo casi se congela. Afortunadamente, un campesino lo llevó a su casa a vivir con su esposa e hijos. Pero el patito estaba aterrado de los niños, quienes gritaban y brincaban todo el tiempo y nuevamente escapó, pasando el invierno en un estanque pantanoso.

Finalmente, llegó la primavera. El patito feo vio a una familia de cisnes nadando en el estanque y quiso acercárseles. Pero recordó cómo todos se burlaban de él y agachó la cabeza avergonzado. Cuando miró su reflejo en el agua se quedó asombrado. Él no era un patito feo, sino un apuesto y joven cisne. Ahora sabía por

qué se veía tan diferente a sus hermanos y hermanas. ¡Ellos eran patitos, pero él era un cisne! Feliz, nadó hacia su familia."

sacado de la pagina <https://arbolabc.com/cuentos-clasicos-infantiles/el-patito-feo>

Para cifrar el texto usamos este método (en vigenere.py) donde la tablita es la tablita de Vigenere.

```
# Función para cifrar un texto a vigenere

def cifrar_vigenere([texto, clave, tablita, |nuevo]):
    salto = False
    with open(texto, 'r') as archivo:
        texto = archivo.read()
    texto_cifrado = ""
    print(len(clave))
    for i in range(len(texto)):
        letra = texto[i]
        fila = ord(clave[i % len(clave)]) - 97
        columna = ord(letra) - 97
        texto_cifrado += tablita[fila][columna]
    with open(nuevo, "w") as archivo:
        archivo.write(texto_cifrado)

    return texto_cifrado
```

Y para obtener el índice de coincidencia usamos este método (se encuentra en indice.py).

```
# Calcular el indice de coincidencia de un texto dado un alfabeto
def calcular_indice_coincidencia(texto, alfabeto):
    # Contar la frecuencia de cada letra
    with open(texto, 'r') as archivo:
        texto = archivo.read()
    frecuencias = {letra: 0 for letra in alfabeto}
    for letra in texto:
        letra = letra.lower()
        if letra in frecuencias:
            frecuencias[letra] += 1
    # Calcular el total de letras
    total_letras = sum(frecuencias.values())
    print("Total de letras:", total_letras)
    # Calcular el indice de coincidencia con la formula
    # IC = (f1 * (f1 - 1) + f2 * (f2 - 1) + ... + fn * (fn - 1)) / (n * (n - 1))
    indice_coincidencia = 0
    for frecuencia in frecuencias.values():
        indice_coincidencia += frecuencia * (frecuencia - 1)
    if total_letras > 1:
        indice_coincidencia /= total_letras * (total_letras - 1)
    else:
        indice_coincidencia = 0

    return indice_coincidencia

abecedario= 'abcdefghijklmnopqrstuvwxyz'
```

Entonces usamos las siguientes claves

Para longitud 7 y 2 alfabetos usamos las cadenas “ebebebe”
Para longitud 7 y 3 alfabetos usamos las cadenas “sbsmssb”
Para longitud 7 y 4 alfabetos usamos las cadenas “izipopi”
Para longitud 7 y 5 alfabetos usamos las cadenas “hyyowwi”
Para longitud 7 y 6 alfabetos usamos las cadenas “aeiouza”
Para longitud 7 y 7 alfabetos usamos las cadenas “rapidos”
Para longitud 8 y 2 alfabetos usamos las cadenas “ebebebee”
Para longitud 8 y 3 alfabetos usamos las cadenas “sbsmssbm”
Para longitud 8 y 4 alfabetos usamos las cadenas “izipopiz”
Para longitud 8 y 5 alfabetos usamos las cadenas “hyyowwio”
Para longitud 8 y 6 alfabetos usamos las cadenas “aeiouzaz”
Para longitud 8 y 8 alfabetos usamos las cadenas “rapidosz”
Para longitud 9 y 2 alfabetos usamos las cadenas “ebebebeeb”
Para longitud 9 y 3 alfabetos usamos las cadenas “sbsmssbmb”
Para longitud 9 y 4 alfabetos usamos las cadenas “izipopizp”
Para longitud 9 y 5 alfabetos usamos las cadenas “hyyowwiow”
Para longitud 9 y 6 alfabetos usamos las cadenas “aeiouzazo”
Para longitud 9 y 9 alfabetos usamos las cadenas “rapidoszu”
Para longitud 10 y 2 alfabetos usamos las cadenas “ebebebeebe”
Para longitud 10 y 3 alfabetos usamos las cadenas “sbsmssbmb”
Para longitud 10 y 4 alfabetos usamos las cadenas “izipopizpo”
Para longitud 10 y 5 alfabetos usamos las cadenas “hyyowwiowi”
Para longitud 10 y 6 alfabetos usamos las cadenas “aeiouzazou”
Para longitud 10 y 10 alfabetos usamos las cadenas “rapidoszum”
Para longitud 11 y 2 alfabetos usamos las cadenas “ebebebeebeb”
Para longitud 11 y 3 alfabetos usamos las cadenas “sbsmssbmbsm”
Para longitud 11 y 4 alfabetos usamos las cadenas “izipopizpoz”
Para longitud 11 y 5 alfabetos usamos las cadenas “hyyowwiowih”
Para longitud 11 y 6 alfabetos usamos las cadenas “aeiouzazoue”
Para longitud 11 y 11 alfabetos usamos las cadenas “rapidoszumx”
Para longitud 12 y 2 alfabetos usamos las cadenas “ebebebeebebb”
Para longitud 12 y 3 alfabetos usamos las cadenas “sbsmssbmbsm”
Para longitud 12 y 4 alfabetos usamos las cadenas “izipopizpozi”
Para longitud 12 y 5 alfabetos usamos las cadenas “hyyowwiowihy”
Para longitud 12 y 6 alfabetos usamos las cadenas “aeiouzazouea”
Para longitud 12 y 12 alfabetos usamos las cadenas “rapidoszumxf”

En la tablita podemos notar que entre más grandes sean la longitud y número de alfabetos el índice de coincidencia va siendo más pequeño, por lo tanto las claves largas y que pocas letras repetidas generan una mejor seguridad para el texto cifrado y lo hace un poco más difícil de romper.

Fuentes usadas para este ejercicio:

Clases y ayudantías

7. Cifrar Texto 1

Lo primero que hicimos fue limpiar al texto de espacios, saltos de líneas y acentos, usando estos métodos (todo el código del ejercicio 7 está en `cifrados.py`).

```
def Limpiar(original, nuevo):
    """Función que limpia un texto de espacios, saltos de linea y caracteres especiales"""
    with open(original, 'r') as archivo:
        texto = archivo.read()
        texto = texto.replace(' ', '') #Elimina espacios
        texto = texto.replace('\n', '') #Elimina saltos de linea
        texto = texto.lower() #Convierte todo a minusculas
        texto = ''.join([i for i in texto if i.isalpha()]) #Elimina caracteres especiales
        texto = normalizar(texto)
    with open(nuevo, 'w') as archivo:
        archivo.write(texto)
    return texto

def normalizar(texto):
    """Función que elimina dieresis y tildes pero mantiene la ñ"""
    a,b = 'áéíóú', 'aeiou'
    trans = str.maketrans(a,b)
    return texto.translate(trans)

##Alfabeto de 27 caracteres con la ñ
alfabeto = 'abcdefghijklmnñopqrstuvwxyz'
```

- Cifrado Afín con alfabeto de 27 letras

El texto cifrado obtenido es (archivo `Texto_Cifrado_Afin_2.txt`)

“ipxmbdjqbcmophbizismiqixyjziptjutoxdijdibuorizicyjzposmiichoujcianmi
chjcdbojzdbopoymopzjcdbcñixcopoothizybzbzhixñxihozbjzdipjcuizcog
icyomcozdjocbmzonophodixihizybzdibznjxuoybjzichoñxicizhicjtxihjdjizi
poutbhjdipoñmtpbybdodfopzjciyxjzcyclizhicdicmñxicizybocixouocdbnby
bpñoxopjcdcbcnieodjxicfñxnicbjzopicsmiichozbzqjpmyxodjcizpoxyoibjzdi
yjzhizbdjdbrbhopyxioxuizcogicsmiciozdbchbzbqjcfsmiyoñhizpoohizyb
zdipjcmcmoxbjc”

Para cifrar usamos la transformación afín $5x + 15 \bmod 27$

Y el código usado para cifrar es

```
def transformacion_afin_vallida(a,n):
    """Función que verifica si una transformacion afín es valida"""
    if mcd(a,n) == 1:
        return True
    return False

def cifrado_afin(original,a,b,alfabeto,nuevo):
    """Función que cifra un texto con el cifrado afín"""
    with open(original, 'r') as archivo:
        texto = archivo.read()
        cifrado = ''
        for letra in texto:
            print (alfabeto.index(letra))
            print (a*alfabeto.index(letra))
            print ((a*alfabeto.index(letra) + b))
            cifrado += alfabeto[(a*alfabeto.index(letra) + b)%len(alfabeto)]
    print(cifrado)
    with open(nuevo, 'w') as archivo:
        archivo.write(cifrado)
    return cifrado
```

- Matriz de Hill con alfabeto de 27 letras

El texto cifrado obtenido es (archivo Texto_Cifrado_Hill_2.txt)

"xcodeaqhxsojlñmnnsererekzmdnsukrmlmgmcalhvjuxnbhyhdocbfthhyex
joykñkthrctvpfpzzypfhipzpjettvpfnezmtmnxyqnbftzyaqglzmuolpzsufgxc
tvchyyrjhvxnwhroqenriubglñxmzmolnbftzylhdvnmxupzsunsrctsñyykgl
hyleñypdrongbxcouuipdlhnxtgrtwñeaxmfdjetvzmhdyyftnbjvqxhyibñyykxth
oykjaxjqxtzwñwñdjaydqxxsuñxmmjhyvvuggedizyhihyervsrcwhdvlasae
oxmtvnbnxeoquftzylhhdglnbeaadbmfphieoquxunbtmwyuqthykwhpfrcdvñ
tlaenergtsjvocajvxtsufgxctvfnoswtv"

Para cifrar usamos la matriz [[11,8],[12,13]]

Y el código usado para cifrar es

```
def dividir_texto_bloques(texto, k):
    """Funcion que divide un texto en bloques"""
    bloques = []
    for i in range(0, len(texto), k):
        if (len(texto) - i < k):
            bloques.append(texto[i:] + 'x' * (k - (len(texto) - i)))
        else:
            bloques.append(texto[i:i+k])
    return bloques

def numero_a_letra(secciones, alfabeto):
    """Funcion que convierte un numero a una letra"""
    resultado = []
    for i in range(len(secciones)):
        numero = []
        for j in range(len(secciones[i])):
            numero.append(alfabeto.index(secciones[i][j]))
        resultado.append(numero)
    return resultado
```

```
def bloque_por_matriz(bloque, matriz):
    """Funcion que multiplica un bloque por una matriz"""
    resultado = []
    for i in range(len(matriz)):
        suma = 0
        for j in range(len(matriz)):
            suma += matriz[i][j] * bloque[j]
        resultado.append(suma)
        print(resultado)
    return resultado

def bloques_a_texto(bloques, alfabeto):
    """Funcion que convierte bloques a texto"""
    resultado = ''
    for bloque in bloques:
        for i in range(len(bloque)):
            resultado += alfabeto[bloque[i]%len(alfabeto)]
    return resultado
```

```

def cifrado_hill(original, matriz, alfabeto, nuevo):
    """ metodo que cifra un texto con el cifrado hill"""
    with open(original, 'r') as archivo:
        texto = archivo.read()
    secciones = dividir_texto_bloques(texto, len(matriz))
    secciones = numero_a_letra(secciones, alfabeto)
    seccionesCifradas = []
    for bloque in secciones:
        seccionesCifradas.append(bloque_por_matriz(bloque, matriz))
    textoCifrado = bloques_a_texto(seccionesCifradas, alfabeto)
    print(textoCifrado)
    with open(nuevo, 'w') as archivo:
        archivo.write(textoCifrado)
    return textoCifrado

```

- Arreglo de Playfair con alfabeto de 27 letras

El texto cifrado obtenido es (archivo Texto_Playfair_2.txt)

“luumdkpzpdnohoodubrmszduetubgepugbykleesprbnbusbetugtnzlsbitu
 pbsvtzlcpekdtbaukdogtbngoupekdflsdubpgotibudtaudoqaduimcotbopr
 blupeusmbtosbbtmebweipborwbgphoibducobudtauesarlpunktbpocpta
 udfsqasbaeudipebulupndaipesgoomegtddkibwigupeduetmbdtbuocbes
 bmoudbsqbotbsninpbepktokatniuebepdbuibiusbviilsdpauogsbrmczcp
 bwarzpuzdqibpebugodqbodtauesetqabudkieakoiogdqounbubpodcmzl
 bsbwkdcpariozpdvrmssdtaocugtwocqboprblupemenoydpe”

Para cifrar usamos la clave “patitos”

Y el código usado para cifrar es

```

def generar_matriz_clave(clave, alfabeto):
    """Funcion que genera una matriz clave para el cifrado playfair"""
    clave = clave.replace('j', 'i') #<- Puedes remplazar la letra que quieras
    clave = clave.replace('ñ', 'n') # <- solo ten en cuenta que deben de ser 25 letras
    matriz = []
    for letra in clave:
        if letra not in matriz:
            matriz.append(letra)
    for letra in alfabeto:
        if letra not in matriz and letra != 'j' and letra != 'ñ':
            matriz.append(letra)
    return matriz_cuadrada(matriz, 5)

def matriz_cuadrada (matriz,n):
    """Funcion que convierte una matriz en una matriz cuadrada"""
    matriz_cuadrada = []
    for i in range(0, len(matriz), n):
        matriz_cuadrada.append(matriz[i:i+n])
    return matriz_cuadrada

```

```

def dividir_en_tuplas(texto):
    """Funcion que divide un texto en tuplas"""
    texto = texto.replace('j', 'i')
    texto = texto.replace('ñ', 'n')
    tuplas = []
    for i in range(0, len(texto), 2):
        evaluar = texto[i:i+2]
        if len(evaluar) == 1:
            evaluar += 'x' #<- Puede ser la letra que quieras siempre cuando este en la tabla
        if evaluar[0] == evaluar[1]:
            evaluar = evaluar[0] + 'x'
        tuplas.append(evaluar)
    return tuplas

def obtener_posicion(letra, matriz):
    """Funcion que obtiene la posicion de una letra en una matriz"""
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            if matriz[i][j] == letra:
                return (i,j)
    return None

```

```

def casos(tupla, matriz):
    """Funcion que maneja los casos del cifrado playfair"""
    pos1 = obtener_posicion(tupla[0], matriz)
    pos2 = obtener_posicion(tupla[1], matriz)
    if pos1[0] == pos2[0]:
        return (matriz[pos1[0]][(pos1[1] + 1)%5] + matriz[pos2[0]][(pos2[1] + 1)%5])
    elif pos1[1] == pos2[1]:
        return (matriz[(pos1[0] + 1)%5][pos1[1]] + matriz[(pos2[0] + 1)%5][pos2[1]])
    else:
        return (matriz[pos1[0]][pos2[1]] + matriz[pos2[0]][pos1[1]])

def cifrado_playfair(original, clave, alfabeto, nuevo):
    """Funcion que cifra un texto con el cifrado playfair"""
    with open(original, 'r') as archivo:
        texto = archivo.read()
    matriz = generar_matriz_clave(clave, alfabeto)
    tuplas = dividir_en_tuplas(texto)
    cifrado = ''
    for tupla in tuplas:
        cifrado += casos(tupla, matriz)
    with open(nuevo, 'w') as archivo:
        archivo.write(cifrado)
    return cifrado

```

Para el código de cifrado Playfair decidimos juntar las letras “n” con “ñ” y “í” con “j” (ya que las letras “ñ” y “j” no son tan usadas) para poder tener nuestro cuadrito de 5x5.

Fuentes usadas para este ejercicio:

Clases y ayudantías

cYpHeR. (2020, August 21). Hill Cipher Encryption and Decryption in Python (Part-I).

YouTube. https://www.youtube.com/watch?v=Y_UFbwClcEc

Hill Cipher (Encryption). (n.d.). Www.youtube.com. Retrieved June 18, 2022, from <https://www.youtube.com/watch?v=-EQ8UomTrAQ>

8. Criptograma 3

El texto descifrado (con espacios) es

“casi sobra decir que en este proceso de sublimacion de una esencia indigena la participacion de esta mujer fue mas bien pasiva como representacion y no como representante a pesar de sus multiples colaboraciones con los distintos investigadores que visitaban milpa alta el papel otorgado a luz jimenez consistio en informar a dichos investigadores representacion del ideal indigena luz nunca fue la autora de ninguna de estas obras ni en nahuatl ni en artes plasticas de tal forma que su nunca logro vivir de sus conocimientos tanto su voz como su imagen necesitaron de la representacion hecha por los intelectuales nacionales para ser vista o ser oida ser un icono de la indigeneidad no se tradujo en una mayor independencia economica para ella ni en un mas alto nivel de estudios deseо que dona luz manifestara en varias ocasiones”

Pensamos que el texto fue sacado de

[https://es.wikipedia.org/wiki/Luz_Jim%C3%A9nez_\(traductora\)#Legado](https://es.wikipedia.org/wiki/Luz_Jim%C3%A9nez_(traductora)#Legado)

Se usó el siguiente alfabeto

Alfabeto del texto cifrado

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
i	j	x	l	g	m	n	o	e	d	p	a	f	h	q	r	s	t	u	v	b	k	w	y	z	c

Alfabeto del texto original

Los pasos seguidos para descifrarlo fueron los siguientes

Primero quitamos los espacios de Criptograma_3.txt, generando un archivo llamado criptograma_3_s.txt para guardar el texto sin espacios.

Después revisamos las frecuencias de las letras del criptograma_3_s.txt, para eso usamos este método (todo el código esta en ejercicio8.py).

```
abecedario = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'

# Método para calcular las frecuencias de los caracteres del texto de un archivo
def calcular_frecuencias(archivo, abecedario):
    with open(archivo, 'r') as file:
        texto = file.read()
    # Usar un diccionario para contar las ocurrencias de cada letra
    frecuencias = {letra: 0 for letra in abecedario}
    total_caracteres = 0
    # Contar las ocurrencias de cada letra
    for caracter in texto:
        if caracter in abecedario:
            frecuencias[caracter] += 1
            total_caracteres += 1
    # Calcular las frecuencias de las letras
    for letra in abecedario:
        frecuencias[letra] /= total_caracteres
    # Ordenar de mayor a menor las frecuencias
    frecuencias = {letra: frecuencias[letra] for letra in sorted(frecuencias.keys(), key=lambda letra: frecuencias[letra], reverse=True)}
    return frecuencias
```

Y las frecuencias obtenidas fueron estas.

```
{'I': 0.12842712842712842, 'L': 0.11688311688311688, 'G': 0.09956709956709957, 'A': 0.08802308802308802, 'H': 0.08513708513708514, 'Q': 0.0836940836940837, 'P': 0.05339105339105339, 'R': 0.049062049062049064, 'Z': 0.046176046176046176, 'J': 0.044733044733044736, 'D': 0.04329004329004329, 'S': 0.04184704184704185, 'K': 0.025974025974025976, 'F': 0.024531024531024532, 'T': 0.01443001443001443, 'E': 0.012987012987012988, 'U': 0.008658008658008658, 'M': 0.007215007215007215, 'Y': 0.007215007215007215, 'N': 0.005772005772005772, 'O': 0.005772005772005772, 'B': 0.004329004329004329, 'X': 0.002886002886002886, 'C': 0.0, 'V': 0.0, 'W': 0.0}
```

Notemos que las letras C, V y W no aparecen en el criptograma, además las letras que más aparecen son I y L.

Posteriormente intentamos generar diferentes mapeos para las letras, usando ese mapeo sustituir el texto cifrado y luego intentar encontrar palabras usando un diccionario de palabras en Español (el diccionario lo sacamos de esta pagina https://github.com/xavier-hernandez/spanish-wordlist/blob/main/text/spanish_words.txt, pero lo modificamos para no tener acentos), para estos pasos usamos los siguientes métodos:

Método para generar un mapeo aleatorio usando un mapeo inicial y frecuencias.

```

# Metodo para crear un mapeo aleatorio basado en las frecuencias de las letras cifradas y reales y usando un mapeo inicial
def crear_mapeo_aleatorio(frecuencias_cifradas, frecuencias_reales):
    # Ordenar de mayor a menor las frecuencias de las letras cifradas y reales
    letras_cifradas = sorted(frecuencias_cifradas.keys(), key=lambda letra: frecuencias_cifradas[letra], reverse=True)
    letras_reales = sorted(frecuencias_reales.keys(), key=lambda letra: frecuencias_reales[letra], reverse=True)
    # Copiar el mapeo inicial
    mapeo = mapeo_inicial.copy()
    # Filtrar letras cifradas y reales ya mapeadas, y obtener las probabilidades de las letras reales disponibles
    letras_cifradas_disponibles = [letra for letra in letras_cifradas if letra not in mapeo_inicial]
    letras_reales_disponibles = [letra for letra in letras_reales if letra not in mapeo_inicial.values()]
    probabilidades_reales_disponibles = [frecuencias_reales[letra] for letra in letras_reales_disponibles]
    # Crear un mapeo aleatorio
    for letra_cifrada in letras_cifradas_disponibles:
        if letras_reales_disponibles:
            # Seleccionar una letra real disponible aleatoriamente, asignarla al mapeo y eliminarla de las disponibles
            letra_real = random.choices(letras_reales_disponibles, weights=probabilidades_reales_disponibles, k=1)[0]
            mapeo[letra_cifrada] = letra_real
            letras_reales_disponibles.remove(letra_real)
            # Actualizar las probabilidades de las letras reales disponibles
            probabilidades_reales_disponibles = [frecuencias_reales[letra] for letra in letras_reales_disponibles]
        else:
            break
    return mapeo

```

Método para sustituir texto usando un mapeo.

```

# Metodo para sustituir los caracteres de un texto usando un mapeo
def sustituir_texto(mapeo, archivo):
    with open(archivo, 'r') as file:
        texto = file.read()
    texto_descifrado = []
    # Sustituir cada caracter del texto usando el mapeo
    for caracter in texto:
        if caracter in mapeo:
            texto_descifrado.append(mapeo[caracter])
        else:
            # En caso de que el caracter no esté en el mapeo, se mantiene igual
            texto_descifrado.append(caracter)
    return ''.join(texto_descifrado)

```

Método para buscar palabras en un texto, usando un diccionario.

```

# Metodo para verificar cuantas palabras se encuentran en un texto descifrado usando un diccionario
def verificar_palabras(texto_descifrado, diccionario):
    palabras_encontradas = []
    # Longitud minima de las palabras a buscar
    longitud_minima_palabra = 8
    # Buscar palabras en el texto descifrado
    for i in range(len(texto_descifrado)):
        for j in range(i + longitud_minima_palabra, len(texto_descifrado) + 1):
            secuencia = texto_descifrado[i:j].lower()
            # Si la secuencia es una palabra en el diccionario y no ha sido encontrada previamente, se agrega a la lista
            if secuencia in diccionario and secuencia not in palabras_encontradas:
                palabras_encontradas.append(secuencia)
    # Regresamos cuantas palabras se encontraron y la lista de palabras encontradas
    return len(palabras_encontradas), palabras_encontradas

```

Método para generar el diccionario.

```

# Metodo para generar un diccionario de palabras a partir de un archivo de texto
def generar_diccionario(filepath):
    with open(filepath, 'r', encoding='utf-8') as file:
        palabras = set(file.read().splitlines())
    return palabras

```

Y por último el método para generar varios mapeos aleatorios hasta encontrar uno que genera x cantidad de palabras de longitud n.

```

# Generar mapeos aleatorios y descifrar el texto
listo = False
while not listo:
    # Crear un mapeo aleatorio y sustituir el texto cifrado
    mapeo_aleatorio = crear_mapeo_aleatorio(frecuencias_cifradas, frecuencias_espanol)
    texto_descifrado = sustituir_texto(mapeo_aleatorio, 'criptograma_3_s.txt')
    # Revisar cuantas palabras se encontraron en el texto descifrado
    palabras_correctas, palabras_encontradas = verificar_palabras(texto_descifrado, diccionario_espanol)
    # Si se encontraron al menos n palabras, mostrar el texto descifrado y el mapeo
    if palabras_correctas >= 5:
        print("Texto sustituido")
        print(texto_descifrado)
        print(f"\n{palabras_correctas} palabras encontradas en el diccionario")
        print("Palabras encontradas:", palabras_encontradas)
        print("\nAbecedario usado:")
        print(mapeo_aleatorio)
    listo = True

```

Entonces veamos como fue el progreso para descifrar el texto

El primer mapeo que usamos fue {I:e, L:a}, no usamos el vacío, ya que notamos que las letras que más aparecen en el texto cifrado son I y L, entonces las pusimos con las letras más frecuentes en el Español, usando este mapeo como inicial ejecutamos el código buscando al menos dos palabras de longitud seis, después de varios intentos encontramos las palabras "indigena" e "irisar", ademas notamos que "indigena" aparece dos veces por lo cual decidimos usar esta palabra para modificar el mapeo inicial, "irisar" no la usamos ya que no es una palabra tan común.

El segundo mapeo que usamos fue {A:i, E:g, G:n, I:e, J:d, L:a} (es como el primero que usamos pero agregamos las letras que usaba "indigena"), usando este mapeo ejecutamos el código buscando al menos tres palabras de longitud siete, después de varias ejecuciones encontramos las palabras "esencia", "dependencia" y "pendencia", usamos las palabras "dependencia" y "esencia" para modificar el mapeo, ya que son palabras largas y algo similares, "pendencia" no la usamos ya que no es una palabra normal.

El tercer mapeo que usamos fue {A:i, E:g, G:n, I:e, J:d, K:p, L:a, Q:s, Z:c}, usando este mapeo ejecutamos el código buscando al menos dos palabras de longitud ocho, tomo algunas ejecuciones pero encontramos las palabras "representante" y "distinto", usamos ambas palabras para modificar el mapeo.

El cuarto mapeo que usamos fue {A:i, E:g, G:n, H:o, I:e, J:d, K:p, L:a, P:r, Q:s, R:t, S:u, Z:c}, usando este mapeo ejecutamos el código pero ahora no buscamos una cantidad de palabras, sino para sustituir en el texto e intentar reconocer palabras en el texto sustituido, de este proceso identificamos las palabras "sobra" que se encontró al ver "casi so_ra decir", "que" que se encontró al ver "decir_ue en este" y "la" que se encontró al ver "esencia indigena _a participacion", usando estas palabras identificadas modificamos el mapeo.

El quinto mapeo que usamos fue {A:i, D:l, E:g, G:n, H:o, I:e, J:d, K:p, L:a, O:q, P:r, Q:s, R:t, S:u, U:b, Z:c}, usando este mapeo ejecutamos el código de igual manera que en el paso anterior, para intentar reconocer palabras en el texto sustituido, de este proceso identificamos solo pudimos identificar la palabra "sublimacion" que se encontro al ver "este proceso de subli_acion de una", modificamos el mapeo usando este palabra.

El sexto mapeo que usamos fue {A:i, D:l, E:g, F:m, G:n, H:o, I:e, J:d, K:p, L:a, O:q, P:r, Q:s, R:t, S:u, U:b, Z:c}, usando este mapeo ejecutamos el código otra vez para reconocer palabras en el texto sustituido, de este proceso identificamos las palabras "mujer", "fue" y "pasiva" que se encontraron al ver "de esta mu_er_ue mas bien pasi_a" y también la palabra "y" que se encontró "como representacion_no como representante", esta vez logramos identificar más palabras y por lo tanto avanzar más en el mapeo.

El séptimo mapeo que usamos fue {A:i, B:j, D:l, E:g, F:m, G:n, H:o, I:e, J:d, K:p, L:a, M:f, O:q, P:r, Q:s, R:t, S:u, T:v, U:b, X:y, Z:c}, usando este mapeo ejecutamos por última vez el código para intentar reconocer las letras que nos faltan N y Y (C, V y W también pero no aparecen en el texto cifrado), de este proceso encontramos las palabras "voz" que se encontró al ver "tanto su vo_como su imagen" y "hecha" se que encontró al ver "representacion_ec_a por los", por lo tanto hicimos la última modificación al mapeo.

Obteniendo el mapeo final {A:i, B:j, D:l, E:g, F:m, G:n, H:o, I:e, J:d, K:p, L:a, M:f, N:h, O:q, P:r, Q:s, R:t, S:u, T:v, U:b, X:y, Y:z, Z:c} o {A:i, B:j, C:x, D:l, E:g, F:m, G:n, H:o, I:e, J:d, K:p, L:a, M:f, N:h, O:q, P:r, Q:s, R:t, S:u, T:v, U:b, V:k, W:w, X:y, Y:z, Z:c} (poniendo valores aleatorios para C, V y W).

Por último solo sustituimos el texto, lo revisamos para que tenga sentido y le agregamos espacios.

Fuentes usadas para este ejercicio:

Clases y ayudantías

9. Criptograma 4

Se quitó de la tarea.

10. Criptograma 5

El texto descifrado (con espacios) es

"RIQUEZA FAMA PODER GOLD ROGER EL REY DE LOS PIRATAS CONQUISTO TODO LO QUE EL MUNDO PUEDE OFRECER SUS ULTIMAS PALABRAS ANTES DE MORIR HICIERON QUE LA GENTE CORRIERA HACIA EL OCANO QUIEREN MI TESORO SI LO BUSCAN PUEDEN QUEDARSE CON EL TODO LO QUE OBTUVE DE ESTE MUNDO LO ENCONTRARAN AH LOS HOMBRES SE DIRIGIERON A LA GRAN RUTA MARTIMA A CUMPLIR SUS SUENOS Y ASI COMENZO LA GRAN ERA DE LOS PIRATAS Z"

Pensamos que el texto fue sacado de

https://onepiece.fandom.com/es/wiki/Narraci%C3%B3n_de_los_openings

La matriz usada para descifrar el texto es [[5,18],[10,13]].

El código usado se encuentra en los cifrar_hill.py y ejercicio10.ipynb

Los pasos seguidos para descifrarlo fueron los siguientes.

Primero notamos las pistas dadas en clase y ayudantías para resolver este ejercicio, primero se mencionó que la matriz es 2x2, la primera letra es “r” la quinta letra es “e”, la penúltima letra es “s”, la última letra es “z” y el texto está relacionado con One Piece.

Lo primero fue quitar los espacios de Criptograma_5.txt

```
criptograma = None
with open('./archivos/Criptograma_5.txt', 'r') as file:
    # Eliminamos espacios y lo guardamos en una variable
    criptograma = file.read().replace(' ', '')
print(criptograma)
```

Después revisamos las últimas 4 letras del criptogramas las cuales son “N”, “Y”, “L” y “S”, luego revisamos cuales son sus valores numéricos (posición en el alfabeto de 26 letras, iniciando con “A” = 0) obteniendo “N” = 13, “Y” = 24, “L” = 18 y “S” = 11.

```
print(ALFABETO['N'])
print(ALFABETO['Y'])
print(ALFABETO['S'])
print(ALFABETO['L'])
```

Posteriormente, usando los valores obtenidos consideramos la matriz [[13,24],[18,11]], usando las pistas sabemos que la última letra es “Z” y la penúltima es “S” por lo tanto obtenemos sus valores numéricos, “S” = 18 y “Z” = 25.

```
print(ALFABETO['S'])
print(ALFABETO['Z'])
```

Luego, intentamos calcular la matriz inversa de la matriz que cifró el texto original (como se vio en las ayudantías), por lo tanto buscamos todas las matrices inversas de la matriz [[i,j],[18,25]] con i y j números entre 0 y 25, encontrando 312 posibles matrices para descifrar el criptograma, para esto usamos este código.

```
# Regresa el determinante de una matriz
def determinante(matriz):
    return determinante_aux(matriz, len(matriz))

def determinante_aux(matriz,n):
# Determinante
    det = 0
    if n == 1:
        det = matriz[0][0]
    else:
        for j in range(n):
            if matriz[0][j] != 0:
                det += pow(-1, j) * matriz[0][j] * determinante_aux(matriz_menor(matriz, n, 0, j), n-1)
    return det

# Regresa la submatriz
def matriz_menor(matriz, n, r, c):
    mm = []
    for i in range(n):
        if i != r:
            fila = []
            for j in range(n):
                if j != c:
                    fila.append(matriz[i][j])
            mm.append(fila)
    return mm
```

```

# Regresa la matriz transpuesta de una matriz cuadrada
def matriz_transpuesta(matriz):
    transpuesta = []
    n = len(matriz)
    for i in range(n):
        nueva_fila = [matriz[j][i] for j in range(n)]
        transpuesta.append(nueva_fila)
    return transpuesta

# Regresa la matriz adjunta de una matriz
def matriz_adjunta(matriz):
    return matriz_transpuesta(matriz_cofactores(matriz))

# Regresa la matriz de cofactores de una matriz
def matriz_cofactores(matriz):
    cofactores = []
    n = len(matriz)
    for i in range(n):
        fila = []
        for j in range(n):
            cofactor = pow(-1, i+j) * determinante(matriz_menor(matriz, n, i, j))
            fila.append(cofactor)
        cofactores.append(fila)
    return cofactores

```

```

# Regresa la matriz inversa de una matriz
def matriz_inversa(matriz):
    d = determinante(matriz)
    if mcd(d,M) != 1:
        return None
    d_inverso = inverso(d,M)
    matriz_inversa = []
    for fila in matriz_adjunta(matriz):
        nueva_fila = [modulo(d_inverso * x,M) for x in fila]
        matriz_inversa.append(nueva_fila)
    return matriz_inversa

```

```

matrices_con_inversa = []
for i in range(26):
    for j in range(26):
        matriz = [
            [i,j],
            [18,25]
        ]
        inversa = matriz_inversa(matriz)
        if inversa:
            matrices_con_inversa.append(matriz)
print(len(matrices_con_inversa))

```

Después, multiplicamos cada matriz obtenida en el paso anterior por la matriz obtenida de las últimas 4 letras ([[13,24],[18,11]]) para obtener posibles matrices usadas para cifrar el texto original, para eso usamos este código.

```

# Multiplica dos matrices
def multiplica_matriz_x_matriz(matriz1, matriz2):
    matriz_resultante = []
    n = len(matriz1)
    for fila in matriz1:
        columnas = [[matriz2[j][i] for j in range(n)] for i in range(n)]
        nueva_fila = [multiplica_vector_x_columna(fila, columna) for columna in columnas]
        matriz_resultante.append(nueva_fila)
    return matriz_resultante

# Reduce una matriz modulo m
def matriz_modulo(matriz,m):
    matriz_resultante = []
    for fila in matriz:
        nueva_fila = [modulo(entrada,m) for entrada in fila]
        matriz_resultante.append(nueva_fila)
    return matriz_resultante

```

```

claves = []
for matriz in matrices_con_inversa:
    inversa = matriz_inversa(matriz)
    clave = matriz_modulo(multiplica_matriz_x_matriz(inversa,matriz_cifrado),26)
    claves.append(clave)

```

Posteriormente, usamos las matrices obtenidas para calcular su inverso y descifrar el texto, guardando todos los textos descifrados, para luego revisar los textos y ver si alguno tiene sentido, para esto se usó este código.

```

# Divide una cadena en bloques
def dividir_en_bloques(cadena, k):
    bloques = []
    for i in range(0, len(cadena), k):
        if (len(cadena) - i < k):
            bloques.append(cadena[i:] + 'X' * (k - (len(cadena) - i)))
        else:
            bloques.append(cadena[i:i+k])
    return bloques

# Convierte una cadena a un vector
def vector_fila(cadena):
    return [ALFABETO[str(c)] for c in cadena]

# Cifra un vector, regresa una cadena
def cifrar_vector(vector):
    vector_cifrado = ""
    for v in vector:
        vector_cifrado += ALFABETO_INVERTIDO[modulo(v,M)]
    return vector_cifrado

```

```

# Multiplica un vector por una columna
def multiplica_vector_x_columna(fila, columna):
    resultado = 0
    for i in range(len(fila)):
        f = fila[i]
        c = columna[i]
        resultado += f * c
    return resultado

# Multiplica un vector por una matriz
def multiplica_vector_x_matriz(vector, matriz):
    n = len(matriz)
    nuevo_vector = []
    for i in range(n):
        columna = [matriz[j][i] for j in range(n)]
        nuevo_vector.append(modulo(multiplica_vector_x_columna(vector, columna), M))
    return nuevo_vector

```

```

# Cifra una cadena con una matriz (Cifrado Hill)
def cifrar_hill(cadena, matriz):
    k = len(matriz)
    bloques = dividir_en_bloques(cadena, k)
    vectores = [vector_fila(bloque) for bloque in bloques]
    texto_cifrado = ""
    for vector in vectores:
        nuevo_vector = multiplica_vector_x_matriz(vector, matriz)
        texto_cifrado += cifrar_vector(nuevo_vector)
    return texto_cifrado

```

Después, se revisaron los textos obtenidos y se encontró uno que tenía sentido, el texto con sentido fue la línea 217 del archivo, por lo tanto fue obtenido usando la matriz en la posición 216, por lo tanto se busco la matriz inversa de la matriz en la posición 216 para conocer con qué matriz se descifró el texto obteniendo la matriz $[[5,18],[10,13]]$, para esto se usó el siguiente código.

```

inversa = matriz_inversa(claves[216])
print(inversa)
print(cifrar_hill(criptograma,inversa))

```

Y por último se le agregaron los espacios adecuados al texto descifrado y revisamos que tuviera sentido.

Fuentes usadas para este ejercicio:

Ives, T. (n.d.). Find the Determinant of a Matrix with Pure Python without Numpy or Scipy – Integrated Machine Learning and Artificial Intelligence. <https://integratedmlai.com/find-the-determinant-of-a-matrix-with-pure-python-without-numpy-or-scipy/>

Las clases del profe vvmejia. (2022, September 28). Programación de la Matriz Inversa por cofactores. Explicación Paso a Paso En Lenguaje C. YouTube. <https://www.youtube.com/watch?v=GhWpLf7-mK4>

in. (2015, October 26). Inverting a matrix in $\mathbb{Z}/n\mathbb{Z}$. Mathematics Stack Exchange.

<https://math.stackexchange.com/questions/1498737/inverting-a-matrix-in-mathbbz-n-mathbbz>

Clases y ayudantías

11. Criptograma 6

El texto descifrado (con espacios, saltos de línea y sin "x"s de relleno) es
"capítulo vi

de los principados nuevos que se adquieren con las armas propias y el talento personal nadie se asombe de que al hablar de los principados de nueva creacion y de aquellos en los que solo es nuevo el principe traiga yo a colacion ejemplos ilustres los hombres siguen casi siempre el camino abierto por otros y se empenan en imitar las acciones de los demas y aunque no es posible seguir exactamente el mismo camino ni alcanzar la perfeccion del modelo todo hombre prudente debe entrar en el camino seguido por los grandes e imitar a los que han sido excelsos para que si no los iguala en virtud por lo menos se les acerque y hacer como los arqueros experimentados que cuando tienen que dar en blanco muy lejano y dado que conocen el alcance de su arma apuntan por sobre el no para llegar a tanta altura sino para acertar donde se lo proponian con la ayuda de mira tan elevada los principados de nueva creacion donde hay un principe nuevo son mas o menos dificiles de conservar segun que sea mas o menos habil el principe que los adquiere y dado que el hecho de que un hombre se convierta de la nada en principe presupone necesariamente talento o suerte es de creer que una u otra de estas dos cosas allana en parte muchas dificultades sin embargo el que menos ha confiado en el azar es siempre el que mas tiempo se ha conservado en su conquista tambien facilita enormemente las cosas el que un principe al no poseer otros estados se vea obligado a establecerse en el que ha adquirido pero quiero referirme a aquellos que no se convirtieron en príncipes por el azar si no por sus virtudes y digo entonces que entre ellos los mas ilustres han sido moises ciro romulo teseo y otros no menos grandes y aunque moises solo fue un simple agente de la voluntad de dios merece sin embargo nuestra admiracion si quiera sea por la gracia que lo hacia digno de hablar con dios pero tambien son admirables ciro y todos los demas que han adquirido o fundado reinos y si juzgamos sus hechos y su gobierno hallaremos que no deslucen ante los de moises que tuvo tan gran preceptor y si nos detenemos a estudiar su vida y sus obras descubriremos que no deben a la fortuna si no el haberles proporcionado la ocasion propicia que fue el material al que ellos dieron la forma conveniente verdades que sin esa ocasion sus meritos de nada hubieran valido pero tambien es cierto que sin sus meritos era inutil que la ocasion se presentara pues necesario que moises hallara al pueblo de israel esclavo y oprimido por los egipcios para que ese pueblo ansioso de salir de su sojuzgamiento se dispusiera a seguirlo se hizo menester que romulo no pudiese vivir en alba y estuviera expuesto desde su nacimiento para que llegase a ser rey de rome y fundador de su patria ciro tuvo que ver a los persas descontentos de la dominacion de los medas y a los medas flojos e indolentes como consecuencia de una larga paz no habria podido teseo poner de manifiesto sus virtudes si no hubiese sido testigo de la dispersion de los atenienses por lo tanto estas ocasiones permitieron que estos hombres realizaran felizmente sus designios y por otro lado sus meritos

permitieron que las ocasiones rindieran provecho con lo cual llenaron de gloria y de dicha a sus patrias

los que por caminos semejantes a los de aquellos se convierten en principes adquieren el principado con dificultades pero lo conservan sin sobresaltos las dificultades nacen en parte de las nuevas leyes y costumbres que se ven obligados a implantar para fundar el estado y proveer a su seguridad pues debe considerarse que no hay nada mas dificil de emprender ni mas dudosos de hacer triunfar ni mas peligroso de manejar que el introducir nuevas leyes se explica el innovador se transforma en enemigo de todos los que se beneficiaban con las leyes

antiguas y no se granjea sino la amistad tibia de los que se beneficiaran con las nuevas tibieza en estos cuyo origen es por un lado el temor a los que tienen de su parte a la legislacion antigua y por otro la incredulidad de los hombres que nunca fian en las cosas nuevas hasta que ven sus frutos de donde resulta que cada vez que los que son enemigos tienen oportunidad para atacar lo hacen energicamente y aquellos otros asumen la defensa con tibieza de modo que se expone uno a caer con ellos por consiguiente si se quiere analizar bien esta parte es preciso ver si esos innovadores lo son por si mismos o si dependen de otros es decir si necesitan recurrir a la suplica para realizar su obra o si pueden imponerla por la fuerza en el primer caso fracasan siempre y nada queda de sus intenciones pero cuando solo dependen de si mismos y pueden actuar con la ayuda de la fuerza entonces rara vez dejan de conseguir sus propositos de donde se explica que todos los profetas armados hayan triunfado y fracasado todos los que no tenian armas hay que agregar ademas que los pueblos son tornadizos y que si es facil convencerlos de algo es dificil mantenerlos fieles a esa conviccion por lo cual conviene estar preparados de tal manera que cuando ya no crean se les pueda hacer creer por la fuerza moises ciro teseo y romulo no habrian podido hacer respetar sus estatutos durante mucho tiempo si hubiesen estado desarmados como sucedio en nuestros a fray jeronimo savonarola que fracaso en sus innovaciones en cuanto la gente empezo a no creer en ellas pues se encontro con que carecia de medios tanto para mantener fieles en su creencia a los que habian creido como para hacer creer a los incredulos hay que reconocer que estos revolucionarios tropiezan con serias dificultades que todos los peligros surgen en su camino y que solo con gran valor pueden superarlos pero vencidos los obstaculos y una vez que han hecho desaparecer a los que tenian envidia de sus virtudes viven poderosos seguros honrados y felices

a tan excelsos ejemplos hay que agregar otro de menor jerarquia pero que guarda cierta proporcion con aquellos y que servira para todos los de igual clase es el de hieron de siracusa que de simple ciudadano llego a ser principe sin tener otra deuda con el azar que la ocasion pues los siracusanos oprimidos lo nombraron su capitán y fue entonces cuando hizo meritos suficientes para que lo eligieran principe y a pesar de no ser noble dio pruebas de tantas virtudes que quien ha escrito de el ha dicho quod nihil illi deerat ad regnandum praeter regnum licencio el antiguo ejercito y creo uno nuevo dejó las amistades viejas y se hizo de otras y asi rodeado por soldados y amigos adictos pudo construir sobre tales cimientos cuento edificio quiso

y lo que tanto le habia costado adquirir poco le costo conservar el principe nicolas maquiavelo”

Pensamos que el texto fue sacado de

<https://albalearning.com/audiolibros/maquiavelo/elprincipe06.html>

Se usó la clave “**nicolas maquiavelo**” para cifrar

Y por lo tanto la tablita se ve así

n	i	c	o	l
a	s	m	q	v
e	b	d	f	g
h	j	k	p	r
t	u	x	y	z

El codigo usado se encuentra en ejercicio11.py

Los pasos seguidos para descifrarlo fueron los siguientes

Primero notamos las pistas dadas en clase y ayudantías para resolver este ejercicio, primero se mencionó que la primera palabra del texto original es “capítulo” y luego se mencionó la cadena “hombres siguen” y que se cifraba como “pnsdhgmubsbzha”. Lo primero fue tomar las pistas obtenidas y intentar reconstruir la tablita de igual manera a la vista en la ayudantía 13/9/24, este proceso se realizó en hojas de papel (las cuales se encuentran al final de este pdf), luego de usar todas las pistas dadas se encontró la posible tablita usada para cifrar y de ahí concluimos la posible clave.

Después usamos estos métodos para descifrar el texto usando la tablita.

Métodos para descifrar un texto cifrado con Playfair usando la tablita que lo cifro.

```

# Metodo para descifrar un par de letras con Playfair usando una matriz 5x5
def descifrar_par_playfair(letra1, letra2, matriz):
    # Encontrar las posiciones de las letras en la matriz
    pos1 = None
    pos2 = None
    for i in range(5):
        for j in range(5):
            if matriz[i][j] == letra1:
                pos1 = (i, j)
            if matriz[i][j] == letra2:
                pos2 = (i, j)
    # Si las letras estan en la misma fila
    # Se usa la letra a la izquierda de cada letra
    if pos1[0] == pos2[0]:
        letra1_descifrada = matriz[pos1[0]][(pos1[1] - 1) % 5]
        letra2_descifrada = matriz[pos2[0]][(pos2[1] - 1) % 5]
    # Si las letras estan en la misma columna
    # Se usa la letra arriba de cada letra
    elif pos1[1] == pos2[1]:
        letra1_descifrada = matriz[(pos1[0] - 1) % 5][pos1[1]]
        letra2_descifrada = matriz[(pos2[0] - 1) % 5][pos2[1]]
    # Si las letras forman un rectangulo
    # Se usa la letra en la misma fila pero en la columna de la otra letra
    else:
        letra1_descifrada = matriz[pos1[0]][pos2[1]]
        letra2_descifrada = matriz[pos2[0]][pos1[1]]
    return letra1_descifrada + letra2_descifrada

# Metodo para descifrar un texto cifrado con Playfair usando una matriz 5x5
def descifrar_playfair(texto, matriz):
    # Separar el texto en pares de letras
    pares = []
    for i in range(0, len(texto), 2):
        pares.append(texto[i:i+2])
    # Descifrar los pares de letras
    texto_descifrado = ""
    for par in pares:
        texto_descifrado += descifrar_par_playfair(par[0], par[1], matriz)
    return texto_descifrado

```

Método para descifrar el texto de un archivo usando la tablita que lo cifró y guardar el resultado

```

# Metodo para descifrar el texto de un archivo cifrado con Playfair
# Recibe el archivo, la matriz 5x5 con la que se crio el texto y un archivo para guardar el texto descifrado
def descifrar_bonito(archivo_origen, matriz, archivo_destino):
    # Leer el archivo
    texto = ""
    with open(archivo_origen, "r") as archivo:
        texto = archivo.read()
    # Descifrar el texto
    texto_descifrado = descifrar_playfair(texto, matriz)
    # Guardar el texto descifrado en un archivo
    with open(archivo_destino, "w") as archivo:
        archivo.write(texto_descifrado)
    return texto_descifrado

# Clave : "nicolas maquiavelo"
tablita = [
    ['n', 'i', 'c', 'o', 'l'],
    ['a', 's', 'm', 'q', 'v'],
    ['e', 'b', 'd', 'f', 'g'],
    ['h', 'j', 'k', 'p', 'r'],
    ['t', 'u', 'x', 'y', 'z']
]

#descifrar_bonito("Criptograma_6.txt", tablita, "descifrado_6.txt")
#descifrar_bonito("Criptograma_6.txt", tablita, "texto_bonito.txt")

```

Y por último se le agregaron los espacios adecuados al texto descifrado y revisamos que tuviera sentido.

Fuentes usadas para este ejercicio:
Clases y ayudantías

- ① CA → NM
 ⑥ PI → JO
 ⑩ TU → UX
 ⑨ LO → NL
 ③ HO → PN
 ⑧ MB → SD
 ⑤ RE → HG
 ⑦ SI → BS
 ⑪ GU → BZ
 ② EN → HA
 ⑨ SX → MU

pista 1

① CA → NM

rectangle

C	N	N	C
M	A	A	M
<hr/>			
M	A	A	M
C	N	N	C

vertical

horizontal

CNAM

②

pista 2 EN → HA

rectangle

vertical

horizontal

E	A	H	E
A	N	N	A
<hr/>			
A	N	N	A
E	H	H	E

E

H

N

A

EHNA

combine ① & ②

vertical ① , vertical ②

vertical ① y horizontal ②

vertical ① & rectangle ②

no se pide

X C/H
X N
X A

no se pide

X E H X A
X M

C

N

A

M

no se pide

horizontal (1) y vertical (2)

E
H
~~C N A M~~
A
no se pide

horizontal (1) y horizontal (2)

E
H
~~C N A M~~
A
no se pide

horizontal (1) y rectangular (2)

H E
~~C N A M~~
C N A M
H E
Opción 1
Opción 2

rectangular (1) y vertical (2)

Opción 3
E
H
C N
M A
Opción 4
E
H
N C
A M

rectangular (1) y horizontal (2)

E H N A
no se pide

rectangular (1) y rectangular (2)

N
A
N A
no se pide

pista 3

(3)

HO → PN

rectangular

vertical

horizontal

H P	P H
N O	O N
—	
N O	O N
H P	P H

H
P
O
N

APON

Opción 1 y vertical (3)

Opción 5
HE
P -
O. M
C N A M

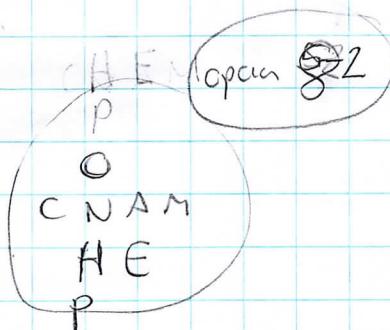
Opción 1 y horizontal (3)

HE
~~C N A M~~
no se pide

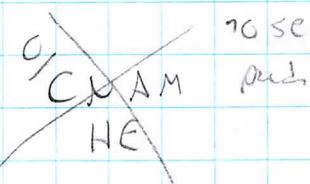
Opción 1 y rectangular (2)

Opción 6
P H E
O C N A M
HE P
C N A M O
Opción 7

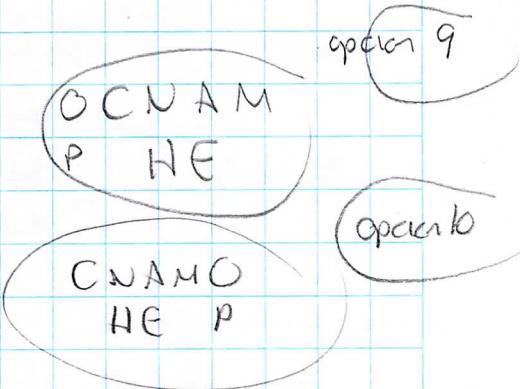
opcion ② y vertical ③



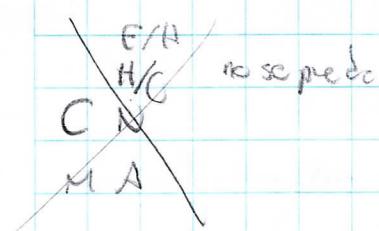
opcion ② y horizontal ③



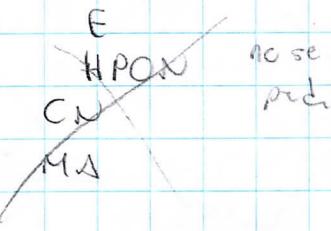
opcion ② y rectangular ③



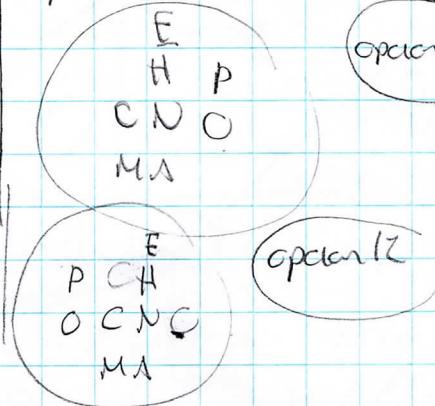
opcion ③ y vertical ③



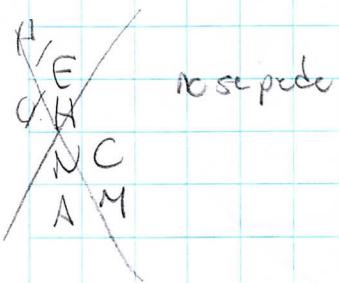
opcion ③ y horizontal ③



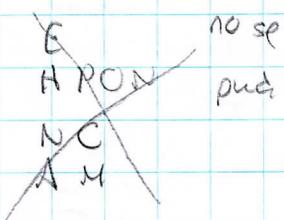
opcion ③ y rectangular ③



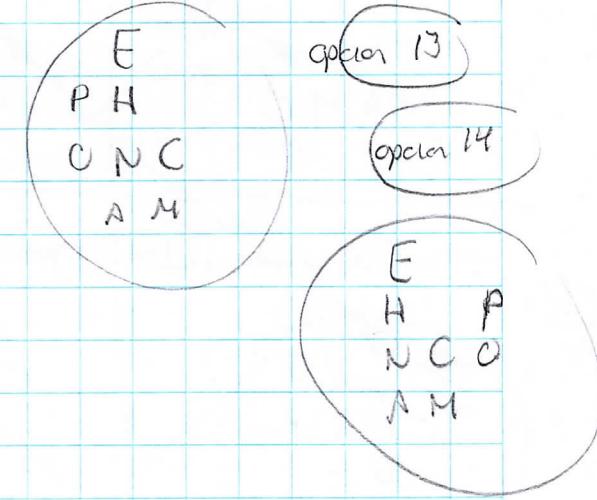
opcion ④ y vertical ③



opcion ④ y horizontal ③



opcion ④ y rectangular ③



pista (4) LO → NL

rectangle	vertical	horizontal
no sepe	N	OLN
option (5) y vertical (4)	option (3) y hori (4)	option (6) y vertical (4)
P H E C N A M no sepe	H E P C N A M no sepe	P H E C N A M no sepe
option (6) y hori (4)	option (7) y vertical (4)	option (7) y hori (4)
P H E C N A M no sepe	H E P C N A M no sepe	H E P C N A M no sepe
option (6) y vertical (4)	option (8) y hori (4)	option (9) y vertical (4)
C N A M H E P no sepe	C N A M H E P no sepe	C N A M P H E no sepe
option (9) y hori (4)	option (10) y vert (4)	option (10) y horizontal (4)
C N A M P H E no sepe	C N A M H E P no sepe	C N A M H E P no sepe

opcion ⑪ y vert ④

~~E H P~~ no se pide
~~C N O~~
~~M A~~

opcion ⑪ y hor ④

(E)
P H
C O L N
M A

opcion ⑫ y vert ④

~~E H~~ no se pide
~~C N~~
~~M A~~

opcion ⑫ y hor ④

(E)
P H
C O L N C
A M

opcion ⑬ y vert ④

~~E H~~ no se pide
O N C
A M

opcion ⑬ y hor ④

(E)
P H
C O L N C
A M

opcion ⑭ y vert ④

~~E H P~~ no se pide
~~N G O~~
~~A M~~

opcion ⑭ y hor ④

(E)
P H
C O L N
M A

opcion 15.

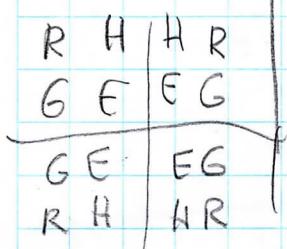
pista (S)

RE → HG

rectangular

vertical

horizontal



R
E
G

R H E G

opcion (S) y vertical (S)

~~E~~
~~R~~
~~H~~
~~C O L N~~
~~M A~~
no se pide

opcion (S) y hori (S)

~~E~~
~~P~~
~~H~~
~~C O L N~~
~~M A~~
no se pide

opcion (S) y rectangular (S)

~~E~~
~~B~~
~~P~~
~~H~~
~~R~~
~~C O L N~~
~~M A~~

opcion 17

opcion 18

opcion (S) y vertical (S)

~~E~~
~~P~~
~~H~~
~~C O L N C~~
~~A M~~
no se pide

opcion (S) y vertical (S)

~~E~~
~~P~~
~~H~~
~~C O L N C~~
~~A M~~
no se pide

opcion (S) y rectangular (S)

~~E~~
~~G~~
~~P~~
~~H~~
~~R~~
~~O L N C~~
~~A M~~

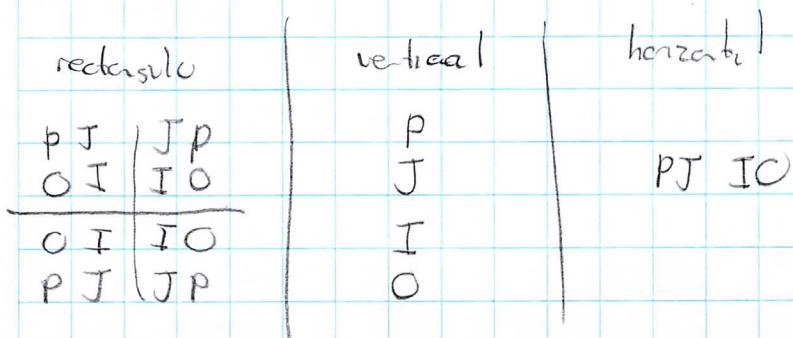
opcion 19

opcion 20

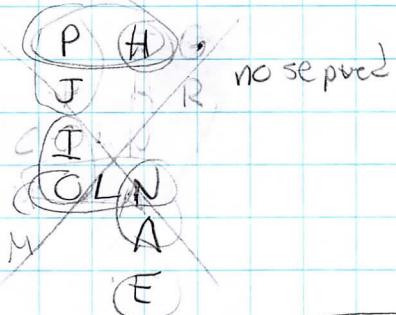
~~G~~
~~E~~
~~R P~~
~~H~~
~~O L N C~~
~~A M~~

- pista 1
- pista 2
- pista 3
- pista 4
- pista 5
- pista 6

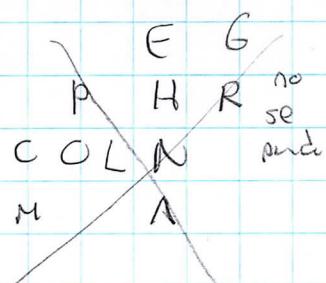
pista ⑥ PI → JO



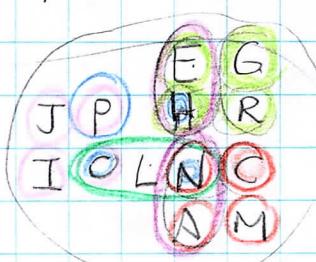
opcion ⑯ y vertical ⑥



opcion ⑯ y hor ⑥



opcion ⑯ y rect ⑥



opcion ⑰

rectangular ①
vertical ②
rectangular ③
horizontal ④
rectangular ⑤
rectangular ⑥

opcion ⑰ y vertical ⑥

isual qe

opcion ⑰ y horizontal ⑥

isual qe

opcion ⑰ y rect ⑥

isual qe

opcion ⑲ y ⑳ son similares

pista ⑦

SI → BS

rectangular	vertical	horizontal
no se pide	I	ISB
	S	
	B	

caso ⑩

rectangular ⑤
rectangular ⑥

vertical ②

rectangular ③

horizontal ④

IOLNC = longitud S horizontal ⑧ no se pide

vertical ⑦

E G
J P H R
I O L N C
S A M
B

caso ⑪

rectangular ① rectangular ⑤
vertical ② rectangular ⑥
rectangular ③ vertical ⑦
horizontal ④

pista ⑧ MB → SD

rectangle	vertical	horizontal	
M S S M	M		
D B B D	S		
D B B D	B		
M S S M	D		MS BD

opcion ⑨ y vertical ① no separam

~~M I
S B~~

opcion ⑩ y horizontal ⑧ no separam

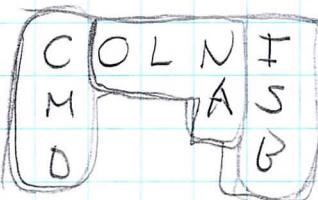
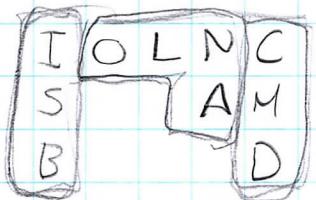
~~I S
B~~ MS BD

opcion ⑪ y rectangle ⑧

E G
J P H R

opcion 22

G E
R P H J



pista ⑨

$SX \rightarrow MU$

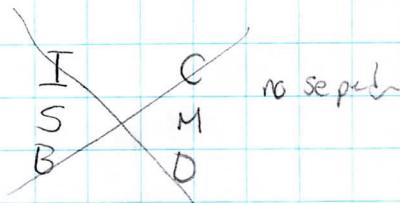
rectangulo	vertical	horizontal
S M	M S	
U X	X U	
U X	X U	
S M	MS	

opcion

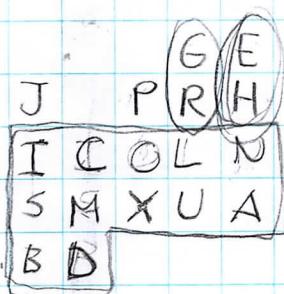
②2

- rectangulo ①
- vertical ②
- rectangulo ③
- horizontal ④
- rectangulo ⑤
- rectangulo ⑥
- vertical ⑦
- rectangulo ⑧

opcion ②2 y vertical ⑨



opcion ②2 y horizontal ⑨



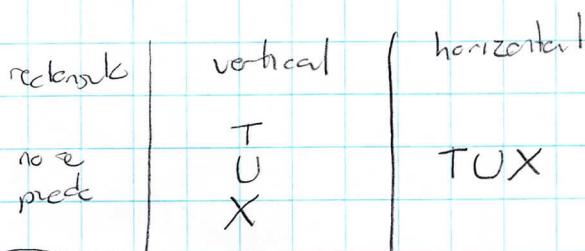
opcion 24



opcion 25

pista ⑩

TU → UX



opcion ⑯

- { rectangle ① rectangle ⑤ horizontal ⑨
- vertical ② rectangle ⑥
- rectangle ③ vertical ⑦
- horizontal ④ rectangle ⑧

opcion ⑯ → vertical ⑩

no se pide

SMXU ~~TUX~~

opcion ⑯ → horizontal ⑩

no se pide

SMXU ~~TUX~~

Opcion ⑯

- { rectangle ① rectangle ⑤ rectangle ⑨
- vertical ② rectangle ⑥
- rectangle ③ vertical ⑦
- horizontal ④ rectangle ⑧

opcion ⑯ → vertical ⑩

no se pide

SM ~~UX~~ T

opcion ⑯ → horizontal ⑩



pista (11) GU → BZ

rectangulo	vertical!	horizontal
GB	BG	G
ZU	UZ	B
—	—	GB UZ
ZU	UZ	U
GB	BG	Z

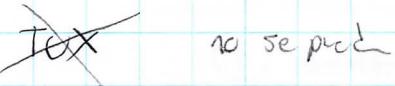
opcion (26) { rectangulo (1) rectangulo (5) rectangulo (9)
 vertical (2) rectangulo (6) horizontal (10)
 rectangulo (3) vertical (7) —
 horizontal (4) rectangulo (8) —

opcion (26) y vertical (11)



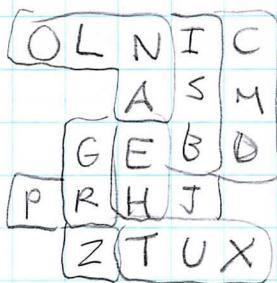
, no se pide

opcion (26) y horizontal (11)



, no se pide

opcion (26) y rectangulo (11)



opcion (27)

{ rectangulo (1) vertical (7)
 vertical (2) rectangulo (8)
 rectangulo (3) rectangulo (9)
 horizontal (4) horizontal (10)
 rectangulo (5) rectangulo (11)
 rectangulo (6) —

w y x mtas

opcion 27

N I C O L

A S M Q V

E B D F G

H J K P R

T U X Y Z

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

A B C D E F G H I J K

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

L M N O P Q R S T U V

✓ ✓

X Y Z

✓ ✓ ✓

letras
restantes
F, K, Q, V, Y
? ?

clave " Nicolas Maquavelo " ?