UNIVERSIDAD CENTRAL DEL ECUADOR

INTEGRANTES:
ANDRADE DENNISSE
LUGMAÑA ANDRES
NARVÁEZ ANTHONY
OCAPANA JOSUE
RIVAS DIEGO

CARRERA: COMPUTACIÓN

SEMESTRE: OCTAVO SEMESTRE

PARALELO C001

FECHA: 30/04/2024

TEMA: DESAROLLO DE ALGORITMOS DE CRIPTOGRAFIA

1. Algoritmo que escriba **todas las permutaciones posibles de una palabra de longitud n SIN espacios (Anagrama)**. La palabra se ingresa al iniciar el algoritmo. El algoritmo debe mostrar el número total de permutaciones y las 10 primeras ordenadas alfabéticamente.

   El proceso para el desarrollo de este algoritmo se divide en dos funciones principales:

   Función generar_anagramas(palabra)

   Esta función toma una palabra como entrada y genera todas las permutaciones posibles de esa palabra independientemente de la extensión de la palabra. Utiliza la función permutations de la biblioteca itertools para generar todas las permutaciones, y luego las convierte en una lista de cadenas lo cual permite ordenar esta lista alfabéticamente y la devuelve dicha cadena.

   Función main()

   es la función principal del programa. Solicita al usuario que ingrese una palabra, llama a la función generar_anagramas(palabra) para obtener todas las permutaciones de esa palabra y luego imprime el número total de permutaciones y las primeras 10 permutaciones ordenadas alfabéticamente.

   Formula: $_nP_r = n!$

   El bloque if __name__ == "__main__": garantiza que la función main() se ejecute solo cuando el script de Python se ejecute directamente ademas aprovecha la capacidad de la biblioteca estándar de Python para generar permutaciones y ordenar listas, lo que hace que el código sea simple y eficiente.

2. Algoritmo que realice el cifrado de un mensaje por **permutación de filas, teniendo como clave n filas**. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que n x n. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

Este código en Python realiza el cifrado de un mensaje por permutación de filas, utilizando una matriz de tamaño n x n como clave

El programa solicita al usuario que ingrese el número de filas (n) y el mensaje que desea cifrar. Esto se hace utilizando la función input() para capturar la entrada del usuario.

```
Ingrese el número de filas: 3
Ingrese el mensaje: hola mundo
```

Se elimina cualquier espacio en blanco del mensaje ingresado para garantizar que solo se consideren los caracteres relevantes. Esto se logra utilizando el método replace() para reemplazar los espacios en blanco por una cadena vacía, adicional a ello si el número de caracteres del mensaje es menor o igual a n x n. Si el mensaje excede esta longitud, se imprime un mensaje de error y el programa se detiene.

Se crea una matriz de cifrado de tamaño n x n utilizando la biblioteca NumPy. Inicialmente, todos los elementos de la matriz se llenan con el carácter *. Esto se logra utilizando la función full() de NumPy para crear una matriz llena de un valor específico.

```
Matriz de cifrado:
[['h' 'o' 'l']
 ['a' 'm' 'u']
 ['n' 'd' 'o']]
```

El código llena la matriz de cifrado con el mensaje, fila por fila. Se utiliza un bucle anidado para recorrer cada fila y columna de la matriz. Si hay más caracteres en el mensaje de los que caben en la matriz, los caracteres adicionales se ignoran.

Se imprime la matriz de cifrado, el mensaje original y el mensaje cifrado utilizando la función print(). El mensaje cifrado se obtiene al concatenar todos los elementos de la matriz de cifrado, utilizando un orden específico (en este caso, se utiliza el orden 'F', que representa el orden de la columna).

```
Mensaje original: holamundo
Mensaje cifrado: hanomdluo
```

3. Algoritmo que realice el cifrado de un mensaje por **permutación de columnas, teniendo como clave n columnas**. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que n x n. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

Similar al código anterior, se verifica si el número de caracteres del mensaje es menor o igual a n x n.

```
Ingrese el número de columnas: 3
Ingrese el mensaje: hola mundo
```

La diferencia principal radica en cómo se llena la matriz de cifrado. En este caso, la matriz se llena con el mensaje, columna por columna. Se utiliza un bucle anidado para recorrer cada columna y fila de la matriz. Si hay más caracteres en el mensaje de los que caben en la matriz, los caracteres adicionales se ignoran.

```
Matriz de cifrado:
[['h' 'a' 'n']
 ['o' 'm' 'd']
 ['l' 'u' 'o']]
```

La matriz de cifrado, el mensaje original y el mensaje cifrado se imprimen de manera similar al código anterior. Sin embargo, es importante tener en cuenta que la disposición de los caracteres en la matriz de cifrado y cómo se genera el mensaje cifrado difieren debido al enfoque de permutación de columnas en lugar de permutación de filas.

```
Mensaje original: holamundo
Mensaje cifrado: hanomdluo
```

4. Algoritmo que realice el cifrado de una cadena de caracteres mediante un **método de sustitución Monoalfabético de desplazamiento n caracteres a la derecha**. Tanto la palabra como el valor de n se ingresan al iniciar el algoritmo. El algoritmo debe mostrar el alfabeto original, el alfabeto cifrado, la cadena de caracteres ingresada y su resultado.

El algoritmo comienza creando el alfabeto original utilizando la biblioteca string, específicamente string.ascii_lowercase, que devuelve el alfabeto en minúsculas. Este alfabeto se utiliza como base para el cifrado monoalfabético. Luego, se crea el alfabeto cifrado desplazando el alfabeto original “n” caracteres a la derecha. Esto se logra concatenando la última “n” letras del alfabeto original con las primeras letras, generando así un nuevo alfabeto para el cifrado.

```
# Crear alfabeto original
alfabeto_original = string.ascii_lowercase

# Crear alfabeto cifrado desplazado n caracteres a la derecha
alfabeto_cifrado = alfabeto_original[n:] + alfabeto_original[:n]
```

El algoritmo procede a cifrar la cadena de caracteres ingresada por el usuario utilizando el método de sustitución mono alfabética. Recorre cada carácter de la cadena y, si el carácter está presente en el alfabeto original, busca su índice en ese alfabeto y lo reemplaza con el carácter correspondiente en el alfabeto cifrado. Se tiene en cuenta si el carácter original estaba en minúscula o mayúscula para evitar un mal cifrado

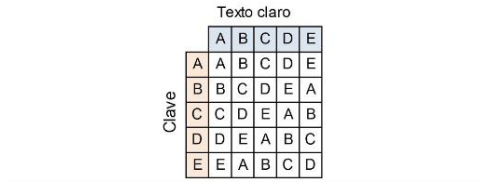
Finalmente, el algoritmo muestra los resultados al usuario, incluyendo el alfabeto original, el alfabeto cifrado, la cadena de caracteres ingresada y el resultado del cifrado.

```
Ingresa la cadena de caracteres a cifrar: hola
Ingresa el valor de desplazamiento (n): 6
Alfabeto original: abcdefghijklmnopqrstuvwxyz
Alfabeto cifrado: ghijklmnopqrstuvwxyzabcdef
Cadena de caracteres ingresada: hola
Resultado cifrado: nurg
```

5. Algoritmo que realice el cifrado de una cadena de caracteres mediante un **método de sustitución Poli alfabético de Vigenère**. La cadena se ingresa al iniciar el algoritmo. El algoritmo debe mostrar la cadena de caracteres ingresada, la clave de cifrado y la cadena de caracteres cifrada.

El algoritmo comienza importando la biblioteca string y define el alfabeto como las letras mayúsculas del inglés utilizando string.ascii_uppercase. Este alfabeto se utilizará como base para el cifrado de Vigenère.

Cifrado VIGENÈRE



Luego, se genera una clave repetida para cifrar el mensaje. La clave se repite tantas veces como sea necesario para que coincida con la longitud del mensaje. Esto se logra mediante la multiplicación de la clave por la parte entera de la división de la longitud del mensaje entre la longitud de la clave, concatenada con los primeros caracteres de la clave para completar la longitud total.

El algoritmo procede a cifrar el mensaje utilizando el método de sustitución poli alfabética de Vigenère. Recorre cada carácter del mensaje y, si el carácter está presente en el alfabeto, calcula el desplazamiento sumando los índices correspondientes del mensaje y la clave repetida, tomando el módulo 26 para asegurar que el resultado esté dentro del rango del alfabeto. Luego, se utiliza este desplazamiento para encontrar el carácter cifrado correspondiente en el alfabeto y se agrega al mensaje cifrado. Si el carácter no está en el alfabeto, se deja sin cifrar.

```python
def cifrado_vigenere(mensaje, clave):
    alfabeto = string.ascii_uppercase
    mensaje_cifrado = ''
    clave_repetida = clave * (len(mensaje) // len(clave)) + clave[:len(mensaje) % len(clave)]

    for i in range(len(mensaje)):
        if mensaje[i].upper() in alfabeto:
            mensaje_cifrado += alfabeto[(alfabeto.index(mensaje[i].upper()) + alfabeto.index(clave_repetida[i].upper())) % 26]
        else:
            mensaje_cifrado += mensaje[i]

    return mensaje_cifrado
```

Por último el algoritmo muestra los resultados al usuario, incluyendo la cadena de caracteres ingresada, la clave de cifrado y la cadena de caracteres cifrada.

```
Ingresa la cadena de caracteres a cifrar: hola
Ingresa la clave de cifrado: j
Cadena de caracteres ingresada: hola
Clave de cifrado: j
Cadena de caracteres cifrada: QXUJ
```

6. Algoritmo que realice el cifrado de una cadena de caracteres utilizando la siguiente **tabla de cifrado**:

*	A	S	D	F	G
Q	a	b	c	d	e
W	f	g	h	i	j
E	k	l	m	n	o
R	p	q	r	s	t
T	u	v	x	y	z

Lo primero es comenzar a ir definiendo una función llamada generar_matriz_cifrado(), que crea y devuelve una matriz de cifrado predefinida. Esta matriz tiene un diseño específico donde los caracteres de la primera fila y la primera columna actúan como referencias para el cifrado.

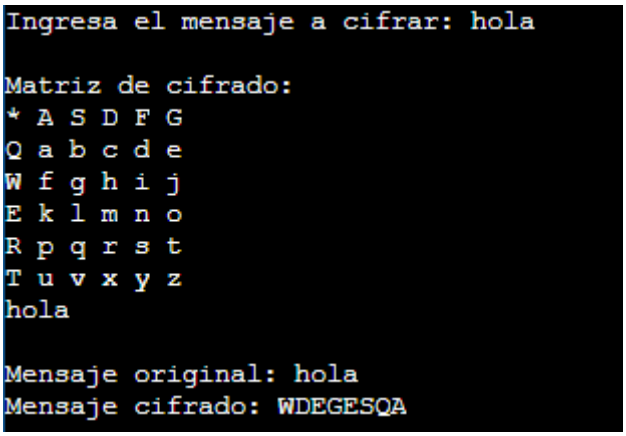
```
def generar_matriz_cifrado():
    matriz = [
        ['*', 'A', 'S', 'D', 'F', 'G'],
        ['Q', 'a', 'b', 'c', 'd', 'e'],
        ['W', 'f', 'g', 'h', 'i', 'j'],
        ['E', 'k', 'l', 'm', 'n', 'o'],
        ['R', 'p', 'q', 'r', 's', 't'],
        ['T', 'u', 'v', 'x', 'y', 'z']
    ]
    return matriz
```

Luego, hay una función llamada cifrar_mensaje(matriz, mensaje) que recibe la matriz de cifrado y el mensaje a cifrar como entrada. Itera sobre cada letra del mensaje y busca su equivalente cifrado en la matriz. Si encuentra la letra en la matriz, concatena los caracteres correspondientes de la primera fila y la primera columna para formar la letra cifrada. Si no encuentra la letra en la matriz, la sustituye por **.

```
def cifrar_mensaje(matriz, mensaje):
    mensaje_cifrado = ""
    print(mensaje)
    for letra in mensaje:
        cifrada = False
        for i, fila in enumerate(matriz):
            for j, caract in enumerate(fila):

                if caract == letra:
                    mensaje_cifrado += f"{fila[0]}{matriz[0][j]}"
                    cifrada = True
                    break
            if cifrada:
                break
        if not cifrada:
            mensaje_cifrado += "**"
    return mensaje_cifrado
```

El algoritmo también define una función llamada imprimir_matriz(matriz) que imprime la matriz de cifrado en la consola. La función principal main() llama a las funciones anteriores para generar la matriz de cifrado, recibir el mensaje del usuario, imprimir la matriz de cifrado, cifrar el mensaje utilizando la matriz y luego imprimir tanto el mensaje original como el mensaje cifrado.



Bibliografía

[1] J. D. Melo Rivero, E. Carrizosa Navarro, and L. Pineda, "Permutaciones," 2020.

[2] H. J. Herrera Mejía and C. A. Rojas Hincapié, "Permutaciones y combinaciones," 2021.

[3] A. A. Soofi, I. Riaz, and U. Rasheed, "An enhanced Vigenere cipher for data security," Int. J. Sci. Technol. Res, vol. 5, no. 3, pp. 141-145, 2016.

[4] R. R. de Araujo, "Reticulados algébricos e aplicações a códigos e criptografia," Doctoral dissertation, [sn], 2018.