Universidad Central del Ecuador

Criptografía y Seguridad de la Información

Octavo Semestre

- Arteaga Jhon
- Cacuango Mauricio
- Rivera Cristina
- Salazar Augusto

Resultados de los Algoritmos Criptográficos

2024-2025

Contenido

RESULTADOS DE LOS ALGORITMOS CRIPTOGRÁFICOS

Algoritmo 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.

CÓDIGO:

main.py +

```python
import itertools

def permutaciones_palabra(palabra):
    # Generar todas las permutaciones
    permutaciones = sorted(set(itertools.permutations(palabra)))

    # Convertir cada permutación en una cadena de caracteres
    permutaciones = [''.join(p) for p in permutaciones]

    # Mostrar el número total de permutaciones
    total_permutaciones = len(permutaciones)
    print(f"Total de permutaciones: {total_permutaciones}")

    # Mostrar las primeras 10 permutaciones ordenadas alfabéticamente
    print("Primeras 10 permutaciones:")
    for perm in permutaciones[:10]:
        print(perm)

# Solicitar la palabra al usuario
palabra = input("Ingrese una palabra: ")
permutaciones_palabra(palabra)

```

Ln: 22, Col: 1

RESULTADOS:

```
Ingrese una palabra:
movil
Total de permutaciones: 120
Primeras 10 permutaciones:
ilmov
ilmvo
ilomv
ilovm
ilvmo
ilvom
imlov
imlvo
imolv
imovl


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Algoritmo 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 "*".

CÓDIGO:

```python
import time

# Ingreso de datos
mensaje = input("Ingresa el mensaje a cifrar: ")
n = int(input("Ingresa el número de filas: "))

# Eliminar espacios
mensaje_sin_espacios = mensaje.replace(" ", "")

# Comprobar si el mensaje cabe en la matriz
if len(mensaje_sin_espacios) > n * n:
    print(f"Error: el número de caracteres del mensaje es mayor a {n * n} caracteres (n x n).")
else:
    # Completar el mensaje con '*' si es necesario
    mensaje_sin_espacios += "*" * (n * n - len(mensaje_sin_espacios))

    # Crear la matriz de cifrado llenando verticalmente
    matriz_cifrado = [[''] * n for _ in range(n)]
    index = 0
    for col in range(n):
        for row in range(n):
            matriz_cifrado[row][col] = mensaje_sin_espacios[index]
            index += 1

    # Generar el mensaje cifrado tomando las filas de la matriz
    mensaje_c = ''.join(''.join(fila) for fila in matriz_cifrado)

    # Resultados
    print("Matriz de cifrado:")
    for fila in matriz_cifrado:
        print(fila)

    print("Mensaje original:", mensaje)
    print("Mensaje cifrado:", mensaje_c)

time.sleep(10)
```

RESULTADOS:

```
PS C:\Users\Asus> & C:/Users/Asus/AppData/Local/Microsoft/Windows
Ingresa el mensaje a cifrar: Universidad Central
Ingresa el número de filas: 5
Matriz de cifrado:
['U', 'r', 'd', 'r', '*']
['n', 's', 'C', 'a', '*']
['i', 'i', 'e', 'l', '*']
['v', 'd', 'n', '*', '*']
['e', 'a', 't', '*', '*']
Mensaje original: Universidad Central
Mensaje cifrado: Urdr*nsCa*iiel*vdn**eat**
```

Algoritmo 3.

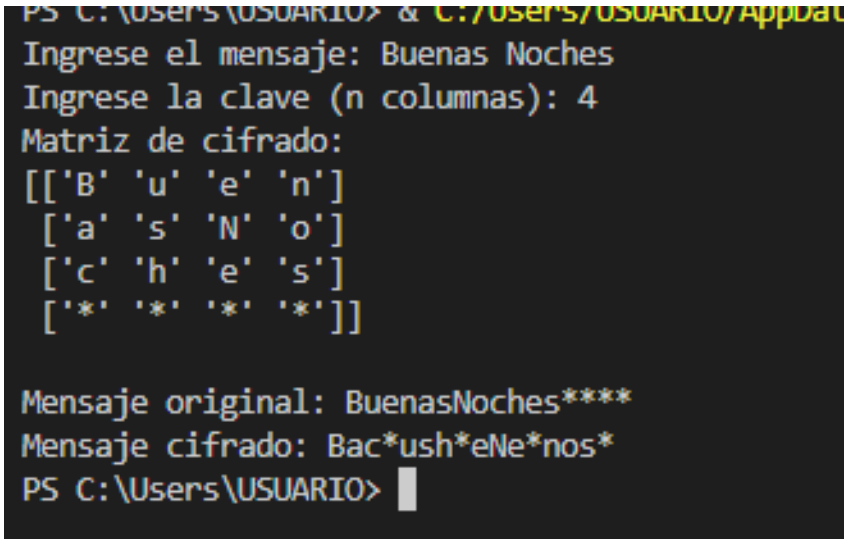
Algoritmo que realice el cifrado de un mensaje por permutación de columnas, teniendo como clave n columnas. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que n x n. Imprima la 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 "*".

CÓDIGO:

```python
import numpy as np

def permutacion_columnas_numpy(mensaje, n):
    # Eliminamos los espacios del mensaje
    mensaje = mensaje.replace(" ", "")

    # Comprobamos que el número de caracteres del mensaje no supere n x n
    if len(mensaje) > n * n:
        print("El mensaje es demasiado largo para la clave proporcionada.")
        return

    # Rellenamos el mensaje con '*' si es necesario
    while len(mensaje) < n * n:
        mensaje += '*'

    # Convertimos el mensaje a una matriz de n x n usando NumPy
    matriz = np.array(list(mensaje)).reshape((n, n))

    # Imprimimos la matriz de cifrado
    print("Matriz de cifrado:")
    print(matriz)

    # Ciframos el mensaje leyendo por columnas
    mensaje_cifrado = ''
    for col in range(n):
        for fila in range(n):
            mensaje_cifrado += matriz[fila, col]  # Extraer cada elemento

    # Imprimimos el mensaje original y cifrado
    print("\nMensaje original:", mensaje)
    print("Mensaje cifrado:", mensaje_cifrado)
```

```python
# Entrada de datos
mensaje = input("Ingrese el mensaje: ")
n = int(input("Ingrese la clave (n columnas): "))

# Ejecutamos el algoritmo con NumPy
permutacion_columnas_numpy(mensaje, n)
```

RESULTADO:


PS C:\Users\USUARIO> & C:/Users/USUARIO/AppDat
Ingrese el mensaje: Buenas Noches
Ingrese la clave (n columnas): 4
Matriz de cifrado:
[['B' 'u' 'e' 'n']
['a' 's' 'N' 'o']
['c' 'h' 'e' 's']
['*' '*' '*' '*']]
Mensaje original: BuenasNoches****
Mensaje cifrado: Bac*ush*eNe*nos*
PS C:\Users\USUARIO>

Algoritmo 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.

CÓDIGO:

```
from string import ascii_uppercase
import re

# Definición del alfabeto español con la Ñ
alfabeto_base = ascii_uppercase[:14] + 'Ñ' + ascii_uppercase[14:]

# Solicitud de palabra y limpieza de caracteres especiales
texto_usuario = re.sub(r'[^a-zA-Z\s]', '', input("Introduce un texto a cifrar: "))

# Solicitud de valor de desplazamiento (shift)
try:
    desplazamiento = int(input("Introduce el valor del desplazamiento (n): "))
except ValueError:
    print('Error: Introduce un valor numérico válido')
    desplazamiento = 0

# Generación del alfabeto desplazado
alfabeto_desplazado = alfabeto_base[desplazamiento:] + alfabeto_base[:desplazamiento]

# Cifrado del texto ingresado
texto_cifrado = ''
for letra in texto_usuario:
    posicion_original = alfabeto_base.lower().find(letra.lower())
    # Verifica si el caracter está en el alfabeto
    if posicion_original != -1:
      texto_cifrado += alfabeto_desplazado[posicion_original]
    # Agrega el caracter tal cual si no es letra
    else:
      texto_cifrado += letra



# Resultados
print("\nAlfabeto original:")
print(" ".join(alfabeto_base))
print("\nAlfabeto cifrado:")
print(" ".join(alfabeto_desplazado))
print("\nTexto ingresado:", texto_usuario)
print("Texto cifrado:", texto_cifrado)
```

RESULTADO:

```
PS C:\Users\usuario> & C:/Python311/python.exe c:/Users/us
Introduce un texto a cifrar: lomas
Introduce el valor del desplazamiento (n): 2

Alfabeto original:
A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z

Alfabeto cifrado:
C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z A B

Texto ingresado: lomas
Texto cifrado: NQÑCU
PS C:\Users\usuario>
```

Algoritmo 5.

Algoritmo que realice el cifrado de una cadena de caracteres mediante un método de sustitución Polialfabético 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.

CÓDIGO:

```python
mensaje = input("Escribe el mensaje a encriptar: ")
llave = input("Escribe la clave de encriptación: ")

#Variable para almacenar el texto encriptado
mensaje_cifrado = ""

#Índice para recorrer la clave de encriptación
indice_clave = 0

#Recorre cada caracter en el mensaje
for pos in range(len(mensaje)):
    if mensaje[pos].isalpha():  # Verifica si es una letra
        # Realiza el cifrado de Vigenère
        letra_cifrada = chr((ord(mensaje[pos]) + ord(llave[indice_clave].lower()) - 2 * ord('a')) % 26 + ord('a'))
        mensaje_cifrado += letra_cifrada
        # Actualiza el índice de la clave
        indice_clave = (indice_clave + 1) % len(llave)
    else:
        # Agrega el carácter sin cambiarlo si no es letra
        mensaje_cifrado += mensaje[pos]

# Muestra los resultados
print("Mensaje original:", mensaje)
print("Clave utilizada:", llave)
print("Mensaje encriptado:", mensaje_cifrado)
```

RESULTADO:

```
PS C:\Users\usuario> & C:/Python311/python.exe c:/Users
Escribe el mensaje a encriptar: Utilizamos python
Escribe la clave de encriptación: 2010
Mensaje original: Utilizamos python
Clave utilizada: 2010
Mensaje encriptado: twmonceptv tbyksq
PS C:\Users\usuario>
```

Algoritmo 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

La cadena de caracteres se ingresa al iniciar el programa. Si algún caracter del texto no existe en la matriz, coloque "**". Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado.

Utilice el lenguaje de programación de su preferencia. Cada algoritmo debe resolverse en un archivo diferente de tal forma que exista independencia en la ejecución.

CÓDIGO:

```
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']]

def print_matriz(matriz):
    for fila in matriz:
        print(' '.join(fila))
    print()

def obtener_coordenadas(caracter, matriz):
    # Devuelve las coordenadas de un carácter en la matriz o None si no existe
    for i, fila in enumerate(matriz):
        if caracter in fila:
            return (i, fila.index(caracter))
    return None
```

```python
def cifrar_cadena(cadena, matriz):
    #Cifra una cadena de caracteres utilizando la matriz de cifrado
    resultado = ""
    for char in cadena:
        if char.isalpha():
            coordenadas = obtener_coordenadas(char, matriz)
            if coordenadas:
                i, j = coordenadas
                resultado += matriz[i][0] + matriz[0][j]  # Agregar la fila y columna
            else:
                resultado += '**'
        else:
            resultado += ' '  # Mantener los espacios en blanco
    return resultado

# Entrada del usuario
cadena = input("Ingresa la cadena para cifrar: ").lower()

# Imprimir resultados
print("Matriz de Cifrado:")
print_matriz(matriz)
print("El mensaje es:", cadena)
print("El mensaje cifrado es:", cifrar_cadena(cadena, matriz))
```

RESULTADO:

```
Ingresa la cadena para cifrar: No tengo tarea el dia de hoy
Matriz 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

El mensaje es: no tengo tarea el dia de hoy
El mensaje cifrado es: EFEG RGQGEFWSEG RGQARDQGQA QGES QFWFQA QFQG WDEGTF
PS C:\Users\USUARIO>
```