Universidad Nacional de Loja

Carrera de Computación

Complejidad Computacional

Proyecto: Desarrollo de 3 algoritmos de etnocifrado de texto

Autor: David Alberto Soto Ramón

Introducción

El hombre siempre ha tenido la necesidad de comunicar algo ya sea hablado, verbal, escrito, etc, para transmitir un mensaje. Se evidencia en culturas de la historia [3], con sus lenguajes entendibles entre los individuos de su cultura; sin embargo, para otras sociedades o culturas esta comunicación era imposible por las diferencias entre lenguajes, formas de expresarse, formas de entender el mundo, etc. En este contexto, los individuos de una sociedad no entienden la forma de comunicación de otra sociedad, esto se podría considerar "cifrado". Y más especificamente, si en una sociedad se quisiera comunicar algo pero escondiendo el mensaje entre ciertos individuos de la misma sociedad.

La criptografía está presente en la historia de la humanidad, desde comunicar secretos de estado, conferencias secretas, el propio modus operandi de alguna cultura, enviar información crucial para una guerra. El objetivo de esta práctica nace para evitar que los mensajes no puedan ser leídos por otras partes que no sean las receptoras.[1] La palabra criptografía proviene de términos griegos, cripto que significa 'secreto' y grafía escritura. Entonces, criptografía es el estudio de lo escondido.[2]

Etno significa pueblo, raza. Cifrado, por otro lado, que está escrito con letras, símbolos o números que solo pueden comprenderse si se dispone de la clave necesaria para descifrarlos. En criptografía, el cifrado es el proceso de convertir la representación original de la información, conocida como texto plano, en una forma alternativa conocida como texto cifrado.[2]

Etnocifrado = Es el estudio de los métodos o procesos de cifrado utilizados para ocultar el significado de un mensaje, que utilizan los individuos, sociedades, instituciones, culturas para codificar información(pasar de una representación original de información a un texto cifrado, que para descifrarlo se necesita una clave de descifrado).

Diseño de los algoritmos Se parte de la idea de la fig.1 -

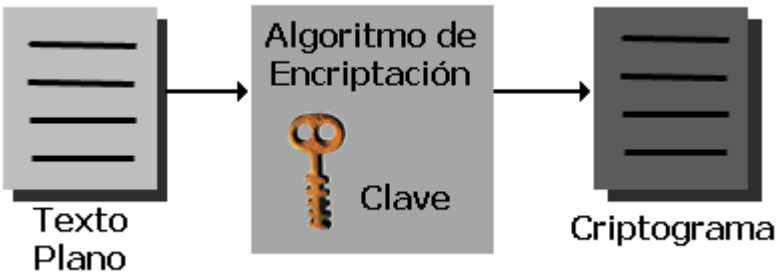


fig1.

Para el desarrollo de los 3 algoritmos se toma como base

El primer algoritmo es de tipo: sustitución o perturbación que consiste en cambiar parcialmente o totalmente cada simbolo o letra número del texto original por otro símbolo, letra o número.

Podemos destacar el cifrado César que consite en sustitituir cada letra por la 4ta letra hacia adelante en el abecedario.

Partiendo de este cifrado y haciendo unas pequeñas modificaciones:

- Las letras se sustituirán al azar por letras determinadas.

Quedando de esta manera:

-

Letras iniciales	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	0	1	2	3	4	5	6	7	8
Sustitución	B	5	3	W	U	C	H	5	0	L	7	V	*	M	6	T	P	F	4	2	O	Q	X	Y	8	K	A	N	Z	&	;	9	D	G	J	E

ESPACIO	MHH0
TILDE	VOCAL + [
ENTRADA	TENGO 10 MIL DÓLARES
SALIDA	OUMHTMHH0ZNMHH0*0VMHH0W[VB4U2

El segundo algoritmo está basado en el tablero de polibio, con la variación de:

- La encriptación es con 0s y 1s

	0	1	2	3	4	5
0		000	001	010	011	100
1	000	A	B	C	D	E
2	001	F	G	H	I	J
3	010	K	L	M	N	Ñ
4	011	O	P	Q	R	S
5	100	T	U	V	W	X
6	101	Y	Z	0	1	2
7	110	3	4	5	6	7
8	111	8	9	ESPACIO		

ENTRADA	TENGO 10 MIL DÓLARES
SALIDA	100000000100010011001001011000111010101011101010111010010010001011010001 1110100000110100010000

-

Con base en la encriptación de tableros de Vigenerá, se hizo variación:

Con base en la encriptación de tableros de Vigenere, se hizo variación:

- Acepta espacio y letras y numeros solamente
- Solo se toma en cuenta 1 hilera de alfabeto
- la clave es un número donde cada valor nos da la pauta de los espacios y los saltos, los cuales se sumarán al número de orrden del arreglo

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	0	1	2	3	4	5	6	7	8	9	Á	É	Í	Ó	Ú

ENTRADA TENGO 10 MIL DÓLARES
CLAVE 5237
SALIDA YJRLT32OLÑKFRHYLZ

1 Definnir arreglo
2 Recibir frase
3 guardar longitud frase
4 determinar tamaño arreglo de clave con base en espacios de frase
5 coonfigurar el arreglo de la clave
6 llenar arregloclave con cantidad de palabras separadas por espacio
7 encriptar cada letra con el criterio de la letra + el número de la clave que corresponde antes de encontrar un e
7 si encuentra un espacio, lo ignora
7 si la suma es mayor al tamañno del arreglo, se resta la cantidad del arreglo
8 presenta resultados

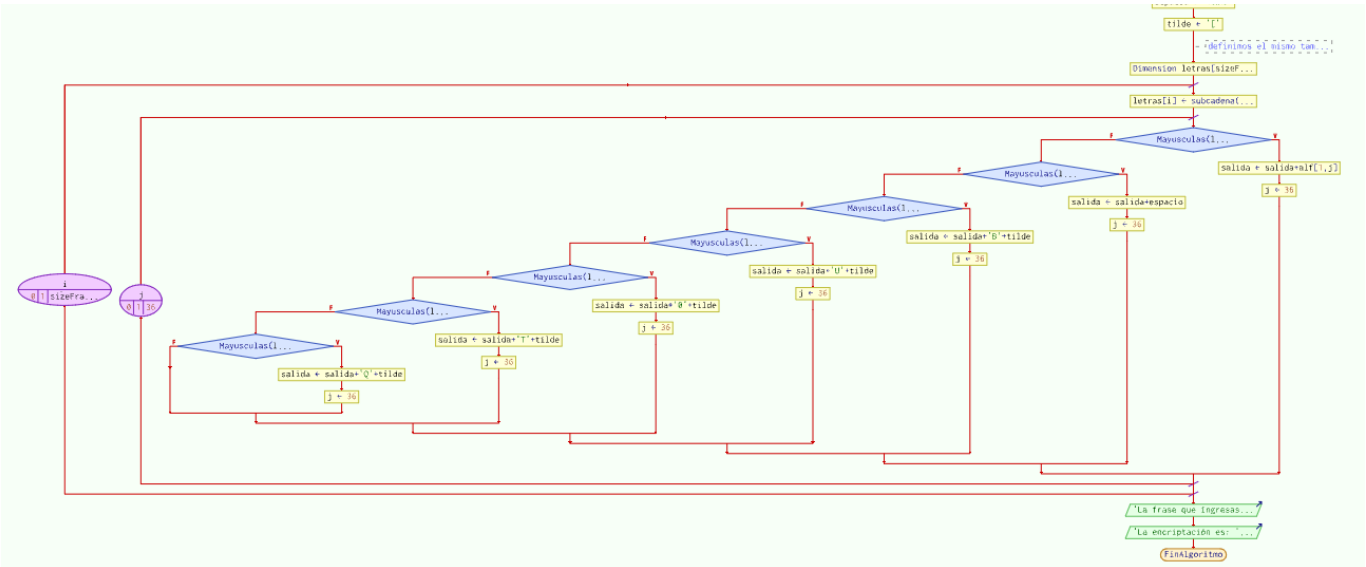
A continuación se mostrará tanto el encriptado como desencriptado en:

- Flujograma
- Pseudocódigo
- Codificación de python

La complejidad algorítimica en big O, el orden de complejidad y gráficos

Algoritmo 1

Flujograma Algoritmo 1



Pseudocódigo Algoritmo 1

```
# Algoritmo alg1

    alf, salida, espacio es caracteres

    Dimension alf[2,37];
    //definimos y declaramos el arreglo con el abecedario
    alf[0,0] = "A"; alf[0,1] = "B"; alf[0,2] = "C"; alf[0,3] = "D";alf[0,4] = "E"; alf[0,5] = "F"

    escribir "escriba frase"
    leer fraseE
    sizeFraseE = longitud(fraseE)

    salida = ""
    espacio = "MHH0"
    tilde = "["

    //definimos el mismo tamaño de la frase para el arreglo
    dimension letras[sizeFraseE]

    Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
        letras[i] = subcadena(fraseE,i,i)

        Para j <- 0 hasta 36 con paso 1 Hacer
            si Mayusculas(letras[i]) == alf[0,j] Entonces
                salida = salida + alf[1,j]
```

```
                j = 36
            Sino
                    si Mayusculas(letras[i]) == " " entonces
                        salida = salida + espacio
                        j =36
                    Sino
                        si Mayusculas(letras[i]) == "Á" Entonces
                            salida = salida + "B" +tilde
                            j =36
                        Sino
                            si Mayusculas(letras[i]) == "É" Entonces
                                salida = salida + "U" + tilde
                                j =36
                            SiNo
                                si Mayusculas(letras[i]) == "Í" Entonces
                                    salida = salida + "0" + tilde
                                    j =36
                                SiNo
                                    si Mayusculas(letras[i]) == "Ó" Entonces
                                        salida = salida + "T" + tilde
                                        j =36
                                    SiNo
                                        si Mayusculas(letras[i]) == "Ú" Entonces
                                            salida = salida + "Q" + tilde
                                            j =36
                                        FinSi
                                    FinSi
                                FinSi
                            FinSi
                        FinSi
                    FinSi
            FinSi
        FinPara
    FinPara

    escribir "La frase que ingresaste es: " fraseE
    escribir "La encriptación es: " salida

FinAlgoritmo
```

Codificación en Python Algoritmo 1

```
if __name__ == '__main__':
    alf = str()
    salida = str()
    espacio = str()
    alf = [[str() for ind0 in range(37)] for ind1 in range(2)]
    # definimos y declaramos el arreglo con el abecedario
    alf[0][0] = "A"
    alf[0][1] = "B"
    alf[0][2] = "C"
    alf[0][3] = "D"
    alf[0][4] = "E"
    alf[0][5] = "F"
    alf[0][6] = "G"
    alf[0][7] = "H"
    alf[0][8] = "I"
    alf[0][9] = "J"
    alf[0][10] = "K"
    alf[0][11] = "L"
    alf[0][12] = "M"
    alf[0][13] = "N"
    alf[0][14] = "Ñ"
    alf[0][15] = "O"
    alf[0][16] = "P"
    alf[0][17] = "Q"
    alf[0][18] = "R"
    alf[0][19] = "S"
    alf[0][20] = "T"
    alf[0][21] = "U"
    alf[0][22] = "V"
    alf[0][23] = "W"
    alf[0][24] = "X"
    alf[0][25] = "Y"
    alf[0][26] = "Z"
    alf[0][27] = "0"
    alf[0][28] = "1"
    alf[0][29] = "2"
    alf[0][30] = "3"
    alf[0][31] = "4"
    alf[0][32] = "5"
    alf[0][33] = "6"
    alf[0][34] = "7"
    alf[0][35] = "8"
    alf[0][36] = "9"
    alf[1][0] = "B"
    alf[1][1] = "5"
    alf[1][2] = "3"
    alf[1][3] = "W"
    alf[1][4] = "U"
```

```
alf[1][5] = "C"
alf[1][6] = "H"
alf[1][7] = "#"
alf[1][8] = "0"
alf[1][9] = "L"
alf[1][10] = "7"
alf[1][11] = "V"
alf[1][12] = "*"
alf[1][13] = "M"
alf[1][14] = "6"
alf[1][15] = "T"
alf[1][16] = "P"
alf[1][17] = "F"
alf[1][18] = "4"
alf[1][19] = "2"
alf[1][20] = "O"
alf[1][21] = "Q"
alf[1][22] = "X"
alf[1][23] = "Y"
alf[1][24] = "8"
alf[1][25] = "K"
alf[1][26] = "A"
alf[1][27] = "N"
alf[1][28] = "Z"
alf[1][29] = "&"
alf[1][30] = ";"
alf[1][31] = "9"
alf[1][32] = "D"
alf[1][33] = "G"
alf[1][34] = "J"
alf[1][35] = "E"
alf[1][36] = "1"
print("escriba frase")
frasee = input()
sizefrasee = len(frasee)
salida = ""
espacio = "MHH0"
tilde = "["
# definimos el mismo tamaño de la frase para el arreglo
letras = [str() for ind0 in range(sizefrasee)]
for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]

    for j in range(37):
        if str.upper(letras[i])==alf[0][j]:
            salida = salida + alf[1][j]
            break
        else:
            if str.upper(letras[i])==" ":
                salida = salida+espacio
```

```
                break
            else:
                if str.upper(letras[i])=="Á":
                    salida = salida+"B"+tilde
                    break
                else:
                    if str.upper(letras[i])=="É":
                        salida = salida+"U"+tilde
                        break
                    else:
                        if str.upper(letras[i])=="Í":
                            salida = salida+"0"+tilde
                            break
                        else:
                            if str.upper(letras[i])=="Ó":
                                salida = salida+"T"+tilde
                                break
                            else:
                                if str.upper(letras[i])=="Ú":
                                    salida = salida+"Q"+tilde
                                    break
print("La frase que ingresaste es: ",frasee)
print("La encriptación es: ",salida)
```

```
escriba frase
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in
_input_request(self, prompt, ident, parent, password)
    728             try:
--> 729                 ident, reply = self.session.recv(self.stdin_socket, 0)
    730             except Exception:

                    5 frames
zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in
_input_request(self, prompt, ident, parent, password)
    732             except KeyboardInterrupt:
    733                 # re-raise KeyboardInterrupt, to truncate traceback
```

```
--> 734                 raise KeyboardInterrupt
    735             else:
    736                 break

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

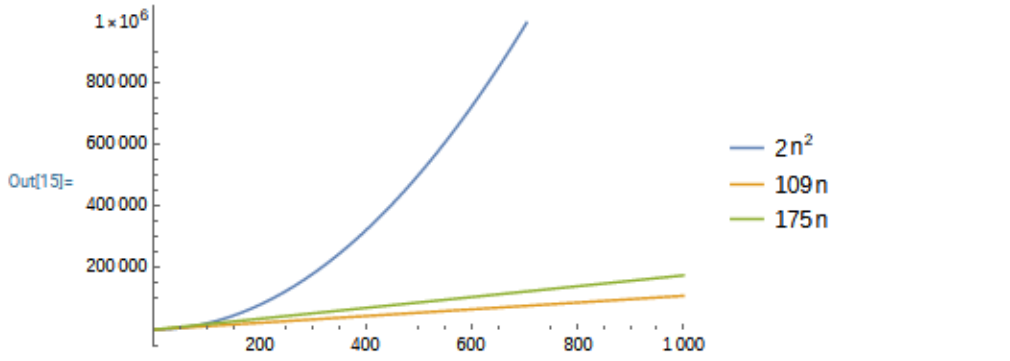
Complejidad Algoritmica en términos de Big Omicron Algoritmo 1

En este caso este algoritmo O(2n2+n+14) por tanto O(n2)

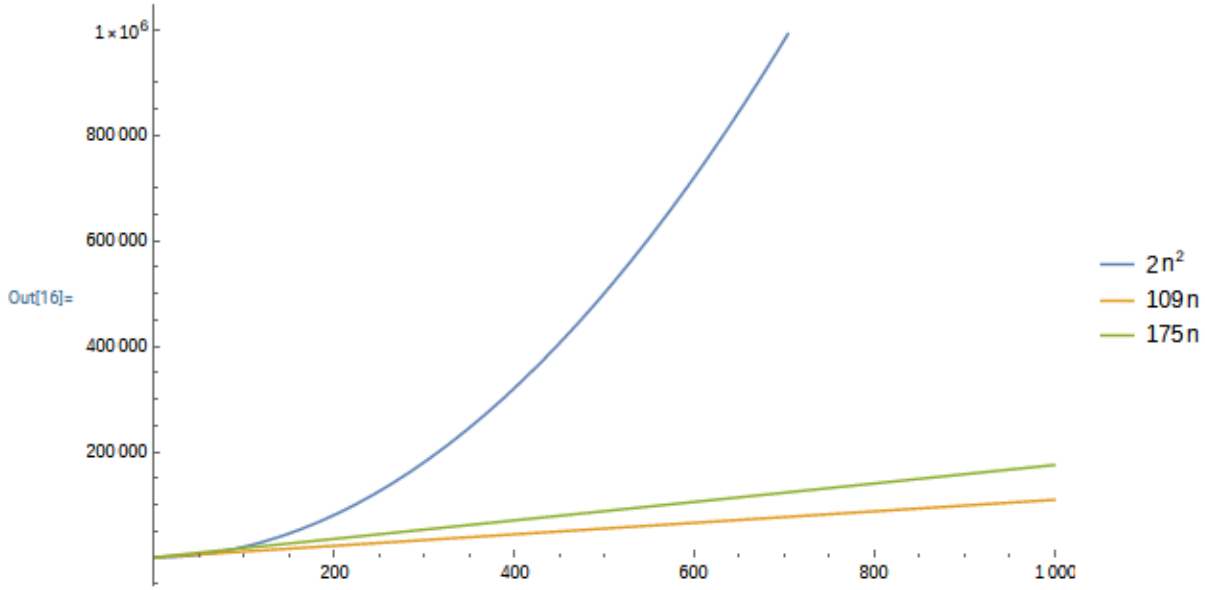
Orden de Complejidad Algoritmo 1 Es de orden cuadrático

Gráficos de complejidad algoritmica en Wolfram Algoritmo 1

```
In[15]:= Plot[{2 n ^2, 109 n, 175 n}, {n, 1, 1000}, PlotLegends → "Expressions"]
```

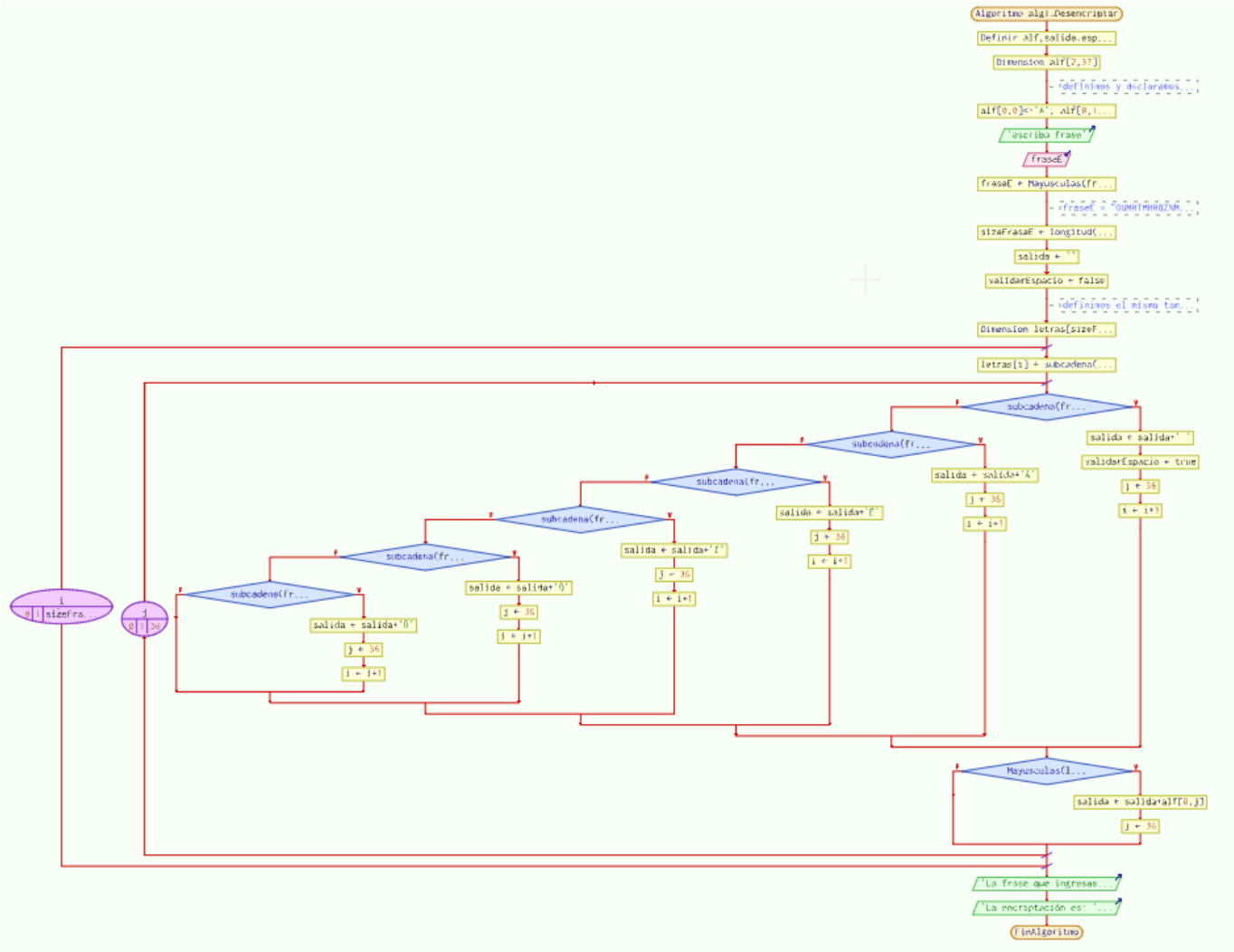



```
In[16]:= Show[%15, ImageSize → Large]
```



Algormitmo 1 Desencriptación

Flujograma Algoritmo 1 desencriptación



Pseudocódigo Algoritmo 1 desencriptación

```
# Algoritmo alg1_Desencriptar

    alf, salida, espacio es caracteres

    Dimension alf[2,37];
    //definimos y declaramos el arreglo con el abecedario
    alf[0,0] = "A"; alf[0,1] = "B"; alf[0,2] = "C"; alf[0,3] = "D";alf[0,4] = "E"; alf[0,5] = "F"
    alf[0,10] = "K"; alf[0,11] = "L"; alf[0,12] = "M"; alf[0,13] = "N"; alf[0,14] = "Ñ"; alf[0,15
    alf[0,19] = "S"; alf[0,20] = "T"; alf[0,21] = "U"; alf[0,22] = "V"; alf[0,23] = "W"; alf[0,24
```

```
alf[0,28] = "1"; alf[0,29] = "2"; alf[0,30] = "3"; alf[0,31] = "4"; alf[0,32] = "5"; alf[0,33
alf[1,0] = "B"; alf[1,1] = "5"; alf[1,2] = "3"; alf[1,3] = "W";alf[1,4] = "U"; alf[1,5] = "C"
alf[1,10] = "7"; alf[1,11] = "V"; alf[1,12] = "*"; alf[1,13] = "M"; alf[1,14] = "6"; alf[1,15
alf[1,19] = "2"; alf[1,20] = "O"; alf[1,21] = "Q"; alf[1,22] = "X"; alf[1,23] = "Y"; alf[1,24
alf[1,28] = "Z"; alf[1,29] = "&"; alf[1,30] = ";"; alf[1,31] = "9"; alf[1,32] = "D"; alf[1,33

escribir "escriba frase"
leer fraseE
fraseE = Mayusculas(fraseE)
//fraseE = "OUMHTMHH0ZNMHH0*0VMHH0WT[VB4U2"
sizeFraseE = longitud(fraseE)

salida = ""
validarEspacio = false

//definimos el mismo tamaño de la frase para el arreglo
dimension letras[sizeFraseE]

Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
    letras[i] = subcadena(fraseE,i,i)

    Para j <- 0 hasta 36 con paso 1 Hacer
        Si subcadena(fraseE,i,i+3) == "MHH0" Entonces
            salida = salida + " "
            validarEspacio = true
            j = 36
            i = i+3
        SiNo
            si subcadena(fraseE,i,i+1) == "B[" entonces
                salida = salida + "Á"
                j =36
                i = i+1
            SiNo
                si subcadena(fraseE,i,i+1) == "U[" entonces
                    salida = salida + "É"
                    j =36
                    i = i+1
                Sino
                    si subcadena(fraseE,i,i+1) == "0[" entonces
                        salida = salida + "Í"
                        j =36
                        i = i+1
                    SiNo
```

```
                        SiNo
                            si subcadena(fraseE,i,i+1) == "T[" entonces
                                salida = salida + "Ó"
                                j =36
                                i = i+1
                            SiNo
                                si subcadena(fraseE,i,i+1) == "Q[" entonces
                                    salida = salida + "Ú"
                                    j =36
                                    i = i+1
                                FinSi
                            FinSi
                        FinSi
                    FinSi
                FinSi
            FinSi

            si Mayusculas(letras[i]) == alf[1,j] y validarEspacio == falso y j < 36 Entonces
                salida = salida + alf[0,j]
                j = 36
            FinSi
        FinPara
    FinPara



    escribir "La frase que ingresaste es: " fraseE
    escribir "La encriptación es: " salida

FinAlgoritmo
```

Codificación en Python Algoritmo 1 desencriptación

```
from pickle import TRUE
from tkinter.tix import INTEGER
import trace


if __name__ == '__main__':
    alf = str()
    salida = str()
    espacio = str()
```

```
alf = [[str() for ind0 in range(37)] for ind1 in range(2)]
# definimos y declaramos el arreglo con el abecedario
alf[0][0] = "A"
alf[0][1] = "B"
alf[0][2] = "C"
alf[0][3] = "D"
alf[0][4] = "E"
alf[0][5] = "F"
alf[0][6] = "G"
alf[0][7] = "H"
alf[0][8] = "I"
alf[0][9] = "J"
alf[0][10] = "K"
alf[0][11] = "L"
alf[0][12] = "M"
alf[0][13] = "N"
alf[0][14] = "Ñ"
alf[0][15] = "O"
alf[0][16] = "P"
alf[0][17] = "Q"
alf[0][18] = "R"
alf[0][19] = "S"
alf[0][20] = "T"
alf[0][21] = "U"
alf[0][22] = "V"
alf[0][23] = "W"
alf[0][24] = "X"
alf[0][25] = "Y"
alf[0][26] = "Z"
alf[0][27] = "0"
alf[0][28] = "1"
alf[0][29] = "2"
alf[0][30] = "3"
alf[0][31] = "4"
alf[0][32] = "5"
alf[0][33] = "6"
alf[0][34] = "7"
alf[0][35] = "8"
alf[0][36] = "9"
alf[1][0] = "B"
alf[1][1] = "5"
alf[1][2] = "3"
alf[1][3] = "W"
alf[1][4] = "U"
alf[1][5] = "C"
alf[1][6] = "H"
alf[1][7] = "#"
alf[1][8] = "0"
alf[1][9] = "L"
alf[1][10] = "7"
alf[1][11] = "V"
```

```
alf[1][12] = "*"
alf[1][13] = "M"
alf[1][14] = "6"
alf[1][15] = "T"
alf[1][16] = "P"
alf[1][17] = "F"
alf[1][18] = "4"
alf[1][19] = "2"
alf[1][20] = "O"
alf[1][21] = "Q"
alf[1][22] = "X"
alf[1][23] = "Y"
alf[1][24] = "8"
alf[1][25] = "K"
alf[1][26] = "A"
alf[1][27] = "N"
alf[1][28] = "Z"
alf[1][29] = "&"
alf[1][30] = ";"
alf[1][31] = "9"
alf[1][32] = "D"
alf[1][33] = "G"
alf[1][34] = "J"
alf[1][35] = "E"
alf[1][36] = "1"
print("escriba frase")
frasee = input()
frasee = str.upper(frasee)
# fraseE = "OUMHTMHH0ZNMHH0*0VMHH0WT[VB4U2"
sizefrasee = len(frasee)
salida = ""
validarespacio = False
# definimos el mismo tamaño de la frase para el arreglo
letras = [str() for ind0 in range(sizefrasee)]

contQuitaEspacio = 0

for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]
    for j in range(37):
        if frasee[i:i+4]=="MHH0":
            salida = salida + " "
            #validarespacio = True
            contQuitaEspacio = 3
            break
        else:
            if frasee[i:i+2]=="B[":
                salida = salida+"Á"
                i = i+1
                break
```

```
            else:
                if frasee[i:i+2]=="U[":
                    salida = salida+"É"
                    i = i+1
                    break

                else:
                    if frasee[i:i+2]=="0[":
                        salida = salida+"Í"
                        i = i+1
                        break

                    else:
                        if frasee[i:i+2]=="T[":
                            salida = salida+"Ó"
                            i = i+1
                            break

                        else:
                            if frasee[i:i+2]=="Q[":
                                salida = salida+"Ú"
                                i = i+1
                                break

        if str.upper(letras[i])==alf[1][j]  and j<36:

            if contQuitaEspacio <= 0:
                salida = salida + alf[0][j]
                break
            contQuitaEspacio = contQuitaEspacio - 1

print("La frase que ingresaste es: ",frasee)
print("El mensaje es : ",salida)
```

Complejidad Algoritmica en términos de Big Omicron Algoritmo 1 desencriptación

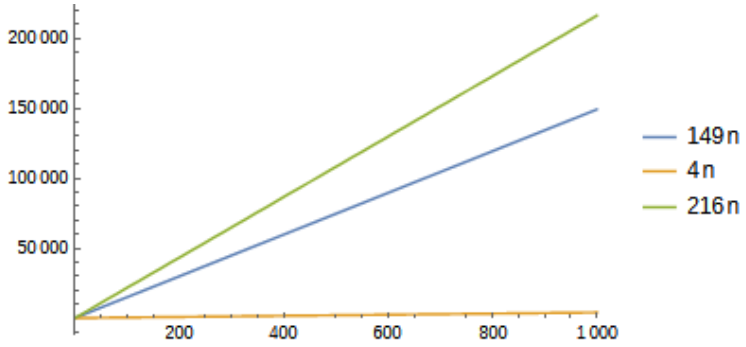
ES O(13+n+37x4xn) => O(n)

Orden de Complejidad Algoritmo 1 desencriptación orden de complejidad lineal

Gráficos de complejidad algoritmica en Wolfram Algoritmo 1 desencriptación

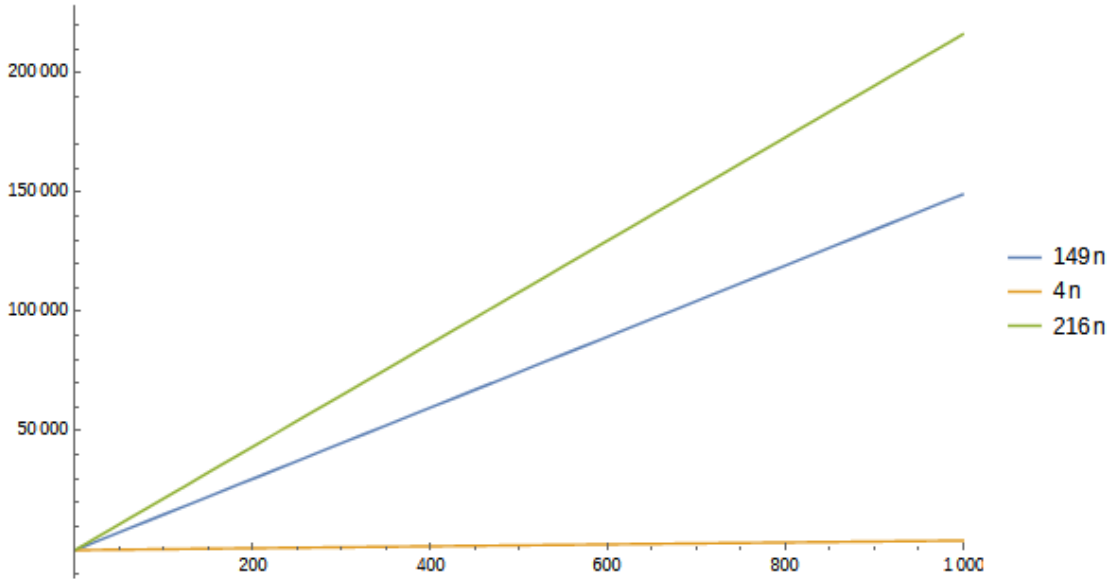
```
In[17]:= Plot[{149 n, 4 n, 216 n}, {n, 1, 1000}, PlotLegends → "Expressions"]
```

Out[17]=



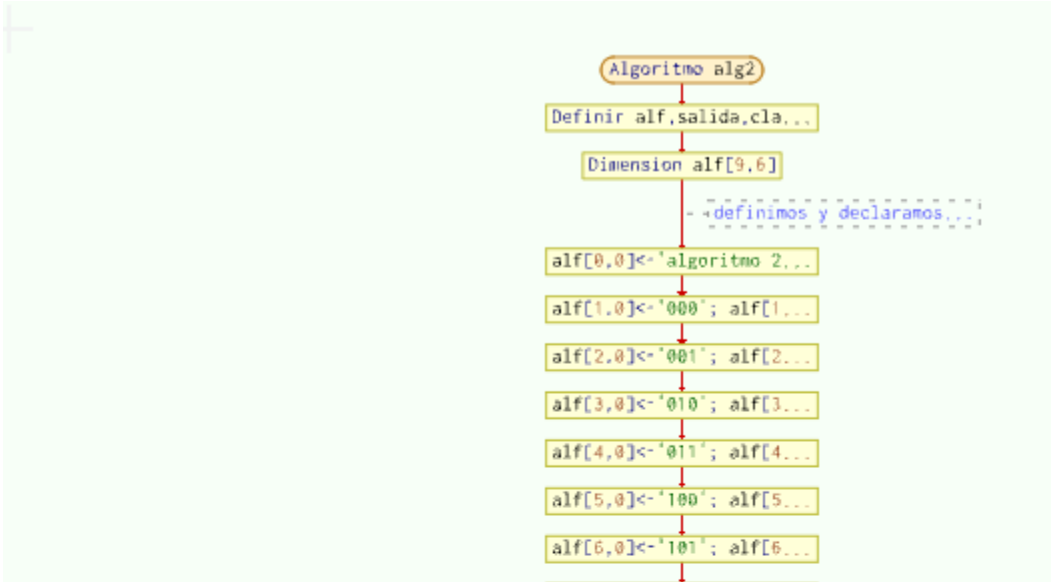
In[18]:= `Show[%17, ImageSize → Large]`

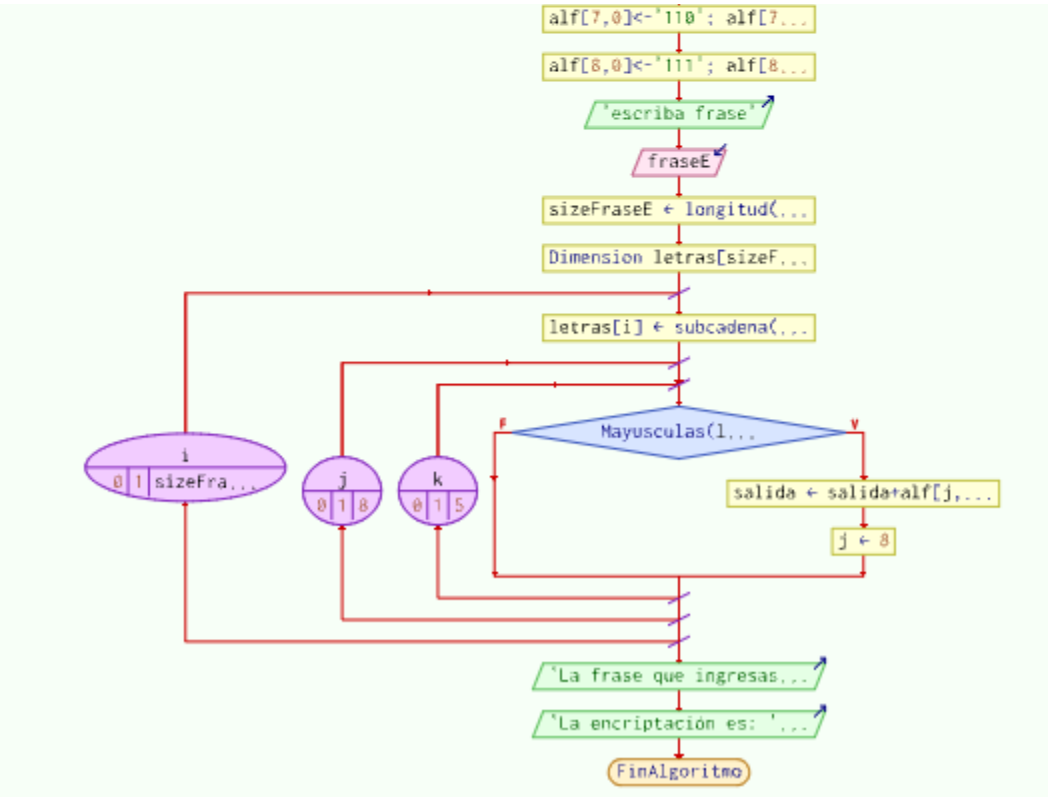
Out[18]=



Algoritmo 2

Flujograma Algoritmo 2





Pseudocódigo Algoritmo 2

```
# Algoritmo alg2

    alf, salida, clave es caracteres

    Dimension alf[9,6];
    //definimos y declaramos el arreglo con el abecedario
    alf[0,0] = "algoritmo 2"; alf[0,1] = "000"; alf[0,2] = "001"; alf[0,3] = "010";alf[0,4] = "01
    alf[1,0] = "000"; alf[1,1] = "A"; alf[1,2] = "B"; alf[1,3] = "C";alf[1,4] = "D"; alf[1,5] = "
    alf[2,0] = "001"; alf[2,1] = "F"; alf[2,2] = "G"; alf[2,3] = "H";alf[2,4] = "I"; alf[2,5] = "
    alf[3,0] = "010"; alf[3,1] = "K"; alf[3,2] = "L"; alf[3,3] = "M";alf[3,4] = "N"; alf[3,5] = "
    alf[4,0] = "011"; alf[4,1] = "O"; alf[4,2] = "P"; alf[4,3] = "Q";alf[4,4] = "R"; alf[4,5] = "
    alf[5,0] = "100"; alf[5,1] = "T"; alf[5,2] = "U"; alf[5,3] = "V";alf[5,4] = "W"; alf[5,5] = "
    alf[6,0] = "101"; alf[6,1] = "Y"; alf[6,2] = "Z"; alf[6,3] = "0";alf[6,4] = "1"; alf[6,5] = "
    alf[7,0] = "110"; alf[7,1] = "3"; alf[7,2] = "4"; alf[7,3] = "5";alf[7,4] = "6"; alf[7,5] = "
    alf[8,0] = "111"; alf[8,1] = "8"; alf[8,2] = "9"; alf[8,3] = " ";

    escribir "escriba frase"
    leer fraseE
    sizeFraseE = longitud(fraseE)
```

```
    dimension letras[sizeFraseE]

    Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
        letras[i] = subcadena(fraseE,i,i)

        Para j <- 0 hasta 8 con paso 1 Hacer
            Para k <- 0 hasta 5 con paso 1 hacer
                si Mayusculas(letras[i]) == alf[j,k] Entonces
                    salida = salida + alf[j,0] + alf[0,k]
                    j = 8
                FinSi
            FinPara
        FinPara
    FinPara

    escribir "La frase que ingresaste es: " fraseE
    escribir "La encriptación es: " salida

 FinAlgoritmo
```

Codificación en Python Algoritmo 2

```
if __name__ == '__main__':
    alf = str()
    salida = str()
    clave = str()
    alf = [[str() for ind0 in range(6)] for ind1 in range(9)]
    # definimos y declaramos el arreglo con el abecedario
    alf[0][0] = "algoritmo 2"
    alf[0][1] = "000"
    alf[0][2] = "001"
    alf[0][3] = "010"
    alf[0][4] = "011"
    alf[0][5] = "100"
    alf[1][0] = "000"
    alf[1][1] = "A"
    alf[1][2] = "B"
    alf[1][3] = "C"
    alf[1][4] = "D"
    alf[1][5] = "E"
    alf[2][0] = "001"
    alf[2][1] = "F"
    alf[2][2] = "G"
    alf[2][3] = "H"
    alf[2][4] = "I"
```

```
alf[2][5] = "J"
alf[3][0] = "010"
alf[3][1] = "K"
alf[3][2] = "L"
alf[3][3] = "M"
alf[3][4] = "N"
alf[3][5] = "Ñ"
alf[4][0] = "011"
alf[4][1] = "O"
alf[4][2] = "P"
alf[4][3] = "Q"
alf[4][4] = "R"
alf[4][5] = "S"
alf[5][0] = "100"
alf[5][1] = "T"
alf[5][2] = "U"
alf[5][3] = "V"
alf[5][4] = "W"
alf[5][5] = "X"
alf[6][0] = "101"
alf[6][1] = "Y"
alf[6][2] = "Z"
alf[6][3] = "0"
alf[6][4] = "1"
alf[6][5] = "2"
alf[7][0] = "110"
alf[7][1] = "3"
alf[7][2] = "4"
alf[7][3] = "5"
alf[7][4] = "6"
alf[7][5] = "7"
alf[8][0] = "111"
alf[8][1] = "8"
alf[8][2] = "9"
alf[8][3] = " "
print("escriba frase")
frasee = input()
sizefrasee = len(frasee)
letras = [str() for ind0 in range(sizefrasee)]
for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]
    for j in range(9):
        for k in range(6):
            if str.upper(letras[i])==alf[j][k]:
                salida = salida+alf[j][0]+alf[0][k]
                j = 8
print("La frase que ingresaste es: ",frasee)
print("La encriptación es: ",salida)
```

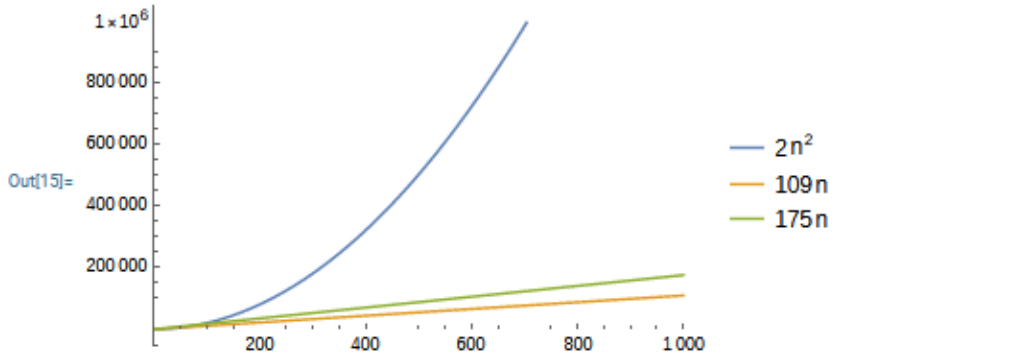
Complejidad Algoritmica en términos de Big Omicron Algoritmo 2

Es O(109n + 11) => O(n)

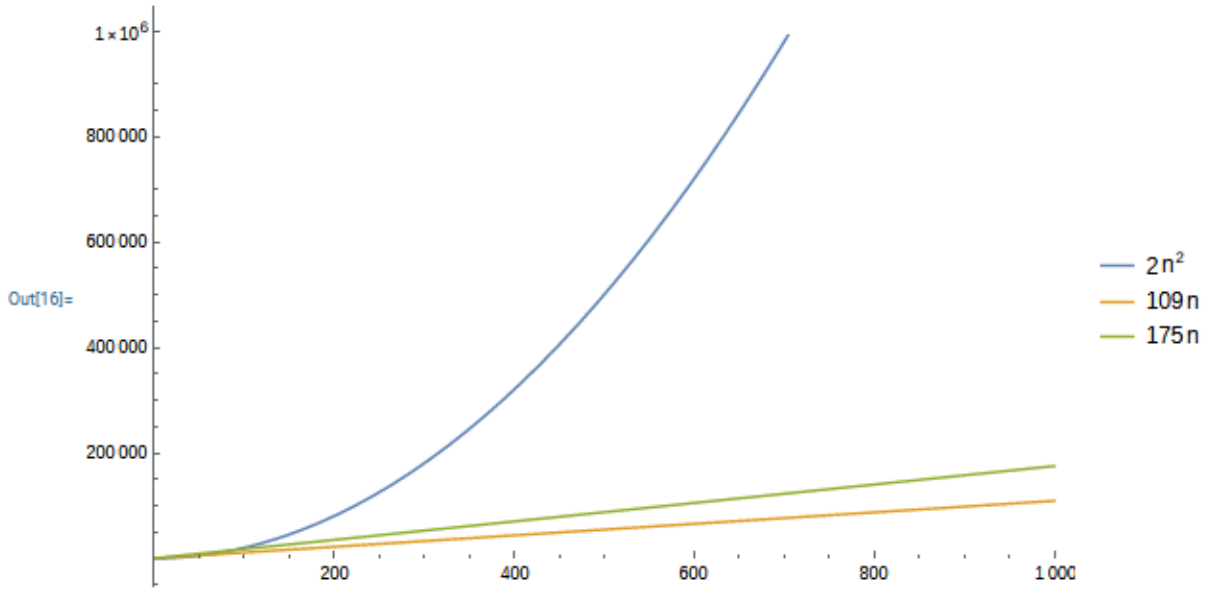
Orden de Complejidad Algoritmo 2 es de orden lineal

Gráficos de complejidad algoritmica en Wolfram Algoritmo 2

```
In[15]:= Plot[{2 n ^2, 109 n, 175 n}, {n, 1, 1000}, PlotLegends → "Expressions"]
```

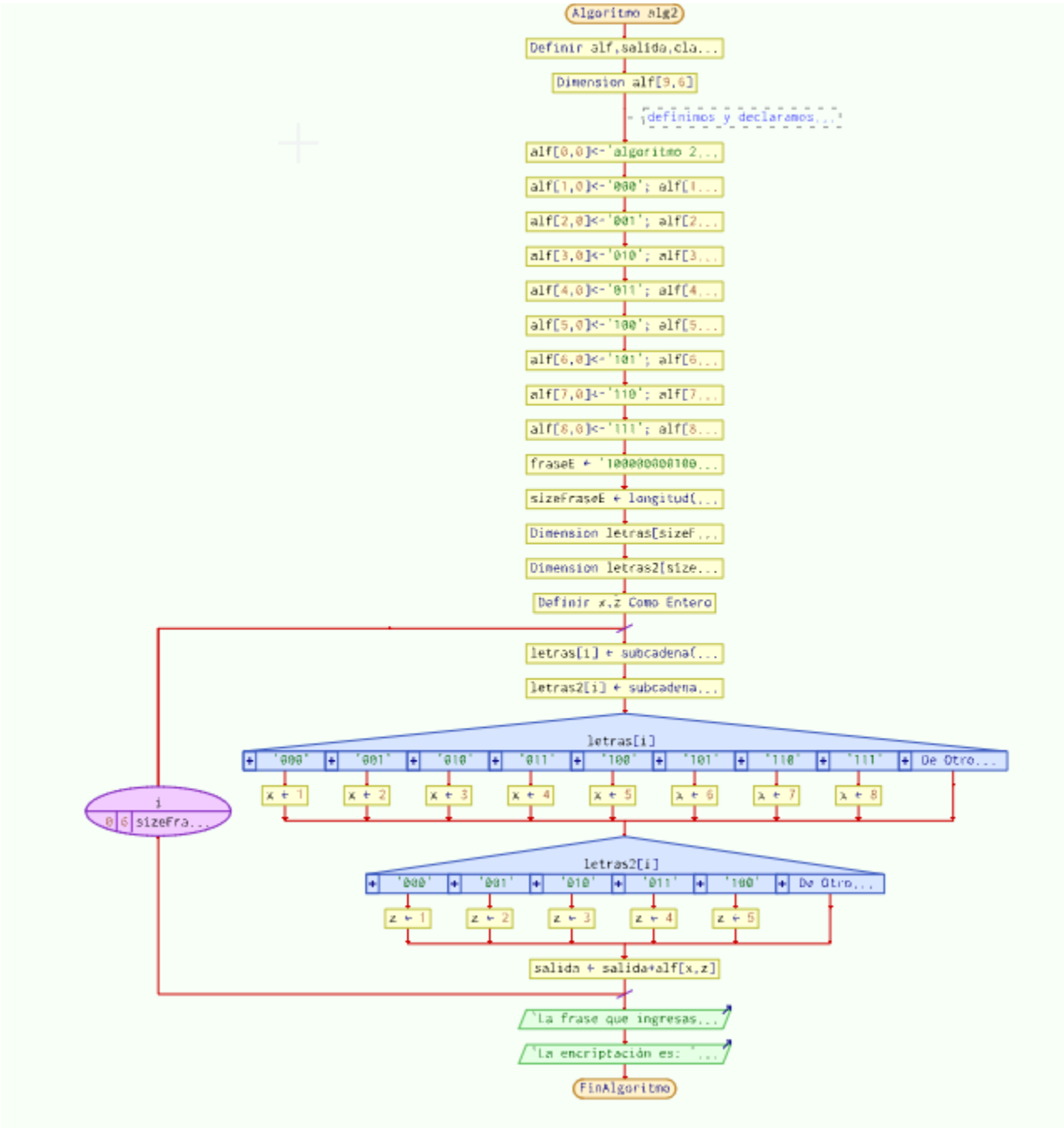



```
In[16]:= Show[%15, ImageSize → Large]
```



Algoritmo 2 desencriptación

Flujograma Algoritmo 2 Desencriptación



Pseudocódigo Algoritmo 2 Desencriptación

```
# Algoritmo alg2

    alf, salida, clave es caracteres

    Dimension alf[9,6];
    //definimos y declaramos el arreglo con el abecedario
    alf[0,0] = "algoritmo 2"; alf[0,1] = "000"; alf[0,2] = "001"; alf[0,3] = "010";alf[0,4] = "01
    alf[1,0] = "000"; alf[1,1] = "A"; alf[1,2] = "B"; alf[1,3] = "C";alf[1,4] = "D"; alf[1,5] = "
```

```
alf[2,0] = "001"; alf[2,1] = "F"; alf[2,2] = "G"; alf[2,3] = "H";alf[2,4] = "I"; alf[2,5] = "
alf[3,0] = "010"; alf[3,1] = "K"; alf[3,2] = "L"; alf[3,3] = "M";alf[3,4] = "N"; alf[3,5] = "
alf[4,0] = "011"; alf[4,1] = "O"; alf[4,2] = "P"; alf[4,3] = "Q";alf[4,4] = "R"; alf[4,5] = "
alf[5,0] = "100"; alf[5,1] = "T"; alf[5,2] = "U"; alf[5,3] = "V";alf[5,4] = "W"; alf[5,5] = "
alf[6,0] = "101"; alf[6,1] = "Y"; alf[6,2] = "Z"; alf[6,3] = "0";alf[6,4] = "1"; alf[6,5] = "
alf[7,0] = "110"; alf[7,1] = "3"; alf[7,2] = "4"; alf[7,3] = "5";alf[7,4] = "6"; alf[7,5] = "
alf[8,0] = "111"; alf[8,1] = "8"; alf[8,2] = "9"; alf[8,3] = " ";

fraseE = "1000000001000100110010010110001110101010111010101110100100100010110100011110100000
sizeFraseE = longitud(fraseE)

dimension letras[sizeFraseE]
dimension letras2[sizeFraseE]

x, z es entero
Para i <- 0 hasta sizeFraseE-6 con paso 6 Hacer
        letras[i] = subcadena(fraseE,i,i+2)
        letras2[i] = subcadena(fraseE,i+3,i+5)

        segun letras[i] Hacer
            "000":
                x = 1
            "001":
                x = 2
            "010":
                x = 3
            "011":
                x = 4
            "100":
                x = 5
            "101":
                x = 6
            "110":
                x = 7
            "111":
                x = 8
        FinSegun

        segun letras2[i] Hacer
            "000":
                z = 1
            "001":
                z = 2
```

```
                "010":
                    z = 3
                "011":
                    z = 4
                "100":
                    z = 5
            FinSegun
            salida = salida + alf[x,z]
     FinPara

     escribir "La frase que ingresaste es: " fraseE
     escribir "La encriptación es: " salida

 FinAlgoritmo
```

Codificación en Python Algoritmo 2 Desencriptación

```
if __name__ == '__main__':
    alf = str()
    salida = str()
    clave = str()
    alf = [[str() for ind0 in range(6)] for ind1 in range(9)]
    # definimos y declaramos el arreglo con el abecedario
    alf[0][0] = "algoritmo 2"
    alf[0][1] = "000"
    alf[0][2] = "001"
    alf[0][3] = "010"
    alf[0][4] = "011"
    alf[0][5] = "100"
    alf[1][0] = "000"
    alf[1][1] = "A"
    alf[1][2] = "B"
    alf[1][3] = "C"
    alf[1][4] = "D"
    alf[1][5] = "E"
    alf[2][0] = "001"
    alf[2][1] = "F"
    alf[2][2] = "G"
    alf[2][3] = "H"
    alf[2][4] = "I"
    alf[2][5] = "J"
    alf[3][0] = "010"
    alf[3][1] = "K"
    alf[3][2] = "L"
    alf[3][3] = "M"
```

```
alf[3][4] = "N"
alf[3][5] = "Ñ"
alf[4][0] = "011"
alf[4][1] = "O"
alf[4][2] = "P"
alf[4][3] = "Q"
alf[4][4] = "R"
alf[4][5] = "S"
alf[5][0] = "100"
alf[5][1] = "T"
alf[5][2] = "U"
alf[5][3] = "V"
alf[5][4] = "W"
alf[5][5] = "X"
alf[6][0] = "101"
alf[6][1] = "Y"
alf[6][2] = "Z"
alf[6][3] = "0"
alf[6][4] = "1"
alf[6][5] = "2"
alf[7][0] = "110"
alf[7][1] = "3"
alf[7][2] = "4"
alf[7][3] = "5"
alf[7][4] = "6"
alf[7][5] = "7"
alf[8][0] = "111"
alf[8][1] = "8"
alf[8][2] = "9"
alf[8][3] = " "

#frasee = "1000000001000100110010010110001111010101011101010111010010010001011010001111
print("escriba frase")
frasee = input()
frasee = str.upper(frasee)
sizefrasee = len(frasee)
letras = [str() for ind0 in range(sizefrasee)]
letras2 = [str() for ind0 in range(sizefrasee)]
x = int()
z = int()
for i in range(0,sizefrasee-5,6):
    letras[i] = frasee[i:i+3]
    letras2[i] = frasee[i+3:i+6]
    if letras[i]=="000":
        x = 1
    elif letras[i]=="001":
        x = 2
    elif letras[i]=="010":
        x = 3
    elif letras[i]=="011":
        x = 4
```

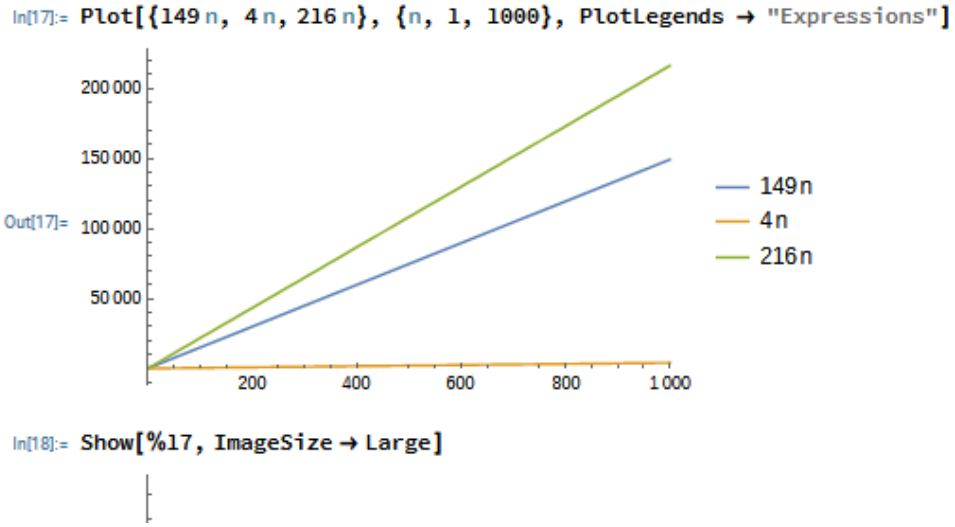
```
        x = 4
    elif letras[i]=="100":
        x = 5
    elif letras[i]=="101":
        x = 6
    elif letras[i]=="110":
        x = 7
    elif letras[i]=="111":
        x = 8
    if letras2[i]=="000":
        z = 1
    elif letras2[i]=="001":
        z = 2
    elif letras2[i]=="010":
        z = 3
    elif letras2[i]=="011":
        z = 4
    elif letras2[i]=="100":
        z = 5
    salida = salida+alf[x][z]
print("La frase que ingresaste es: ",frasee)
print("La encriptación es: ",salida)
```

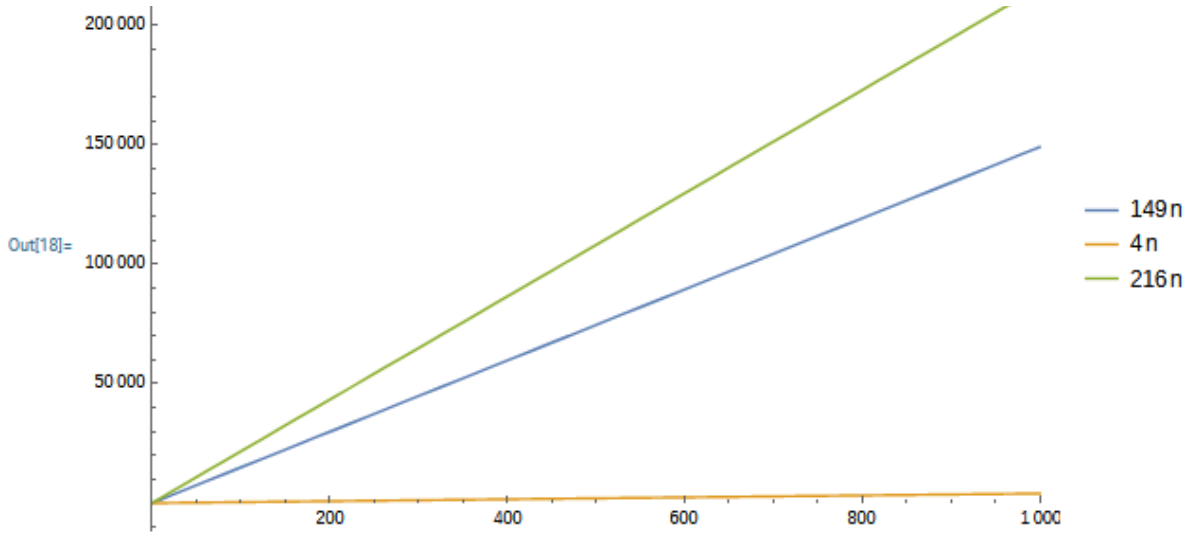
Complejidad Algoritmica en términos de Big Omicron Algoritmo 2 Desencriptación

Es O(13+2n+n+n+2) => O(n)

Orden de Complejidad Algoritmo 2 Desencriptación Orden lineal

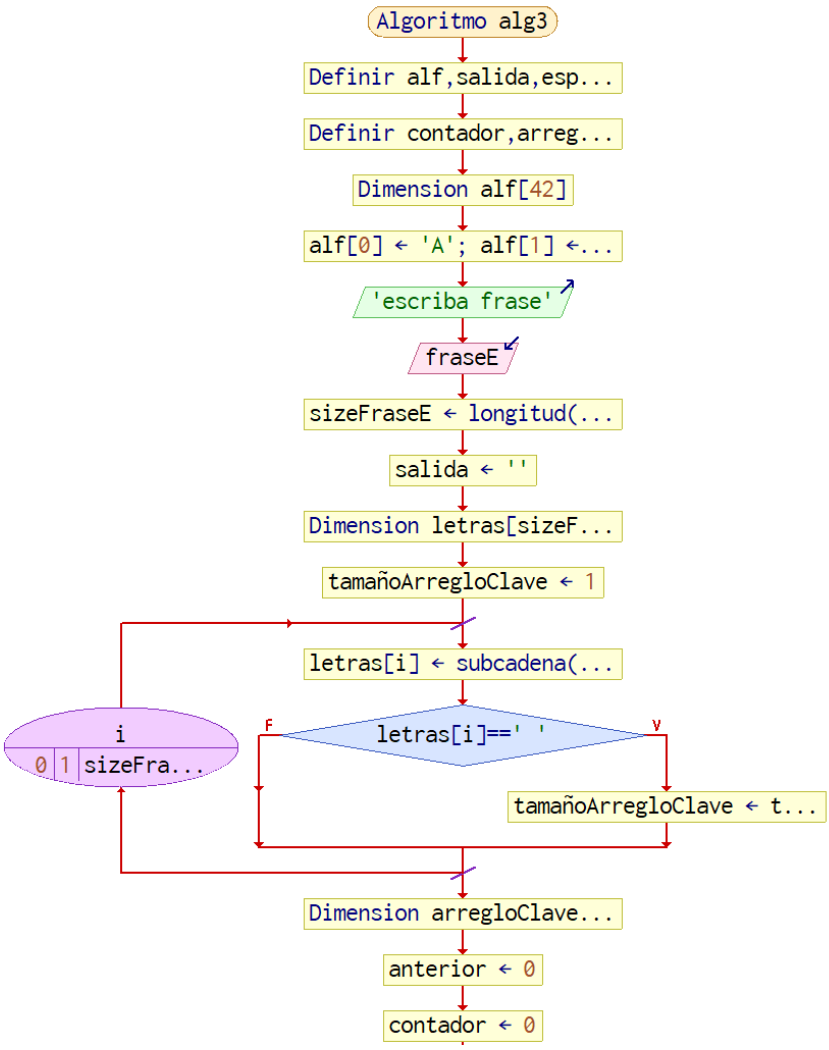
Gráficos de complejidad algoritmica en Wolfram Algoritmo 2 Desencriptación

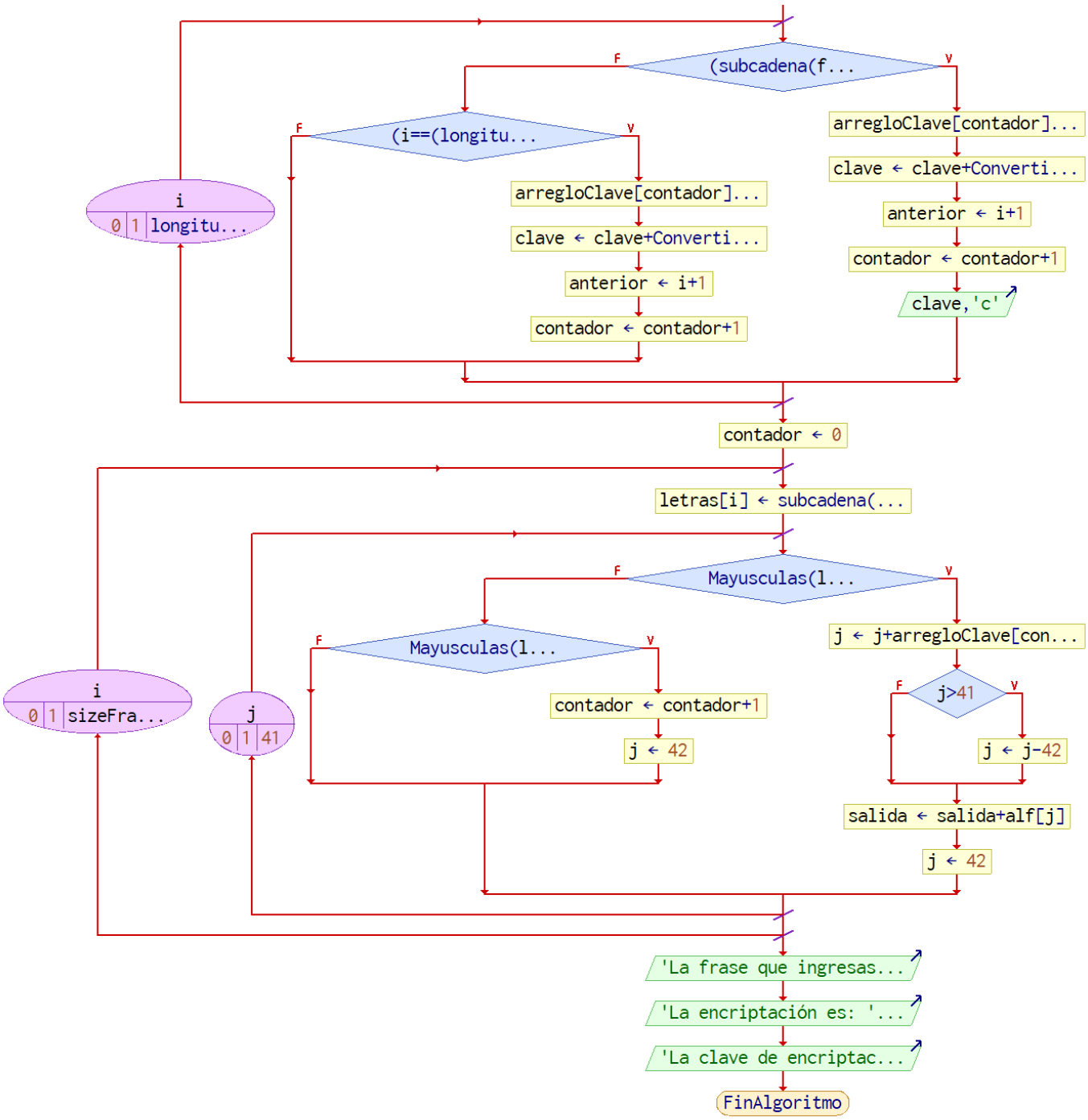




Algoritmo 3

Flujograma Algoritmo 3





Pseudocódigo Algoritmo 3

```
# Algoritmo alg3

    alf, salida, espacio, clave es caracteres
    contador, arregloClave es entero //utilizamos contador dentro de distintos for

    //definimos y declaramos el arreglo con el abecedario
```

```
Dimension alf[42];
alf[0] = "A"; alf[1] = "B"; alf[2] = "C"; alf[3] = "D";alf[4] = "E"; alf[5] = "F"; alf[6] = "

escribir "escriba frase"
leer fraseE
sizeFraseE = longitud(fraseE)

salida = ""

//definimos arreglo letras para iterar por cada letra de la fraseE
dimension letras[sizeFraseE]
tamañoArregloClave = 1

//determinar tamaño de arreglo clave
Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
    letras[i] = subcadena(fraseE,i,i)
    si letras[i] == " " entonces
        tamañoArregloClave = tamañoArregloClave + 1
    FinSi
FinPara

dimension arregloClave[tamañoArregloClave]
//Fin configuracion arreglo clave

//llenado de arregloClave
anterior = 0
contador = 0
Para i <- 0 hasta longitud(fraseE) con paso 1 Hacer
    Si (subcadena(fraseE,i,i) == " ") entonces
        arregloClave[contador] = i - anterior
        clave = clave + ConvertirATexto(arregloClave[contador])
        anterior = i + 1
        contador = contador + 1
        Escribir clave 'c'
    SiNo
        si (i == (longitud(fraseE)))Entonces
            arregloClave[contador] = i - anterior
            clave = clave + ConvertirATexto(arregloClave[contador])
            anterior = i + 1
            contador = contador + 1

        FinSi
```

```
        FinSi
    FinPara
    //fin llenado de arregloClave


    contador = 0//a cero el contador para reutilizarlo
    Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
        letras[i] = subcadena(fraseE,i,i)
        //para recorrer arregloClave alf
        Para j <- 0 hasta 41 con paso 1 Hacer
            si Mayusculas(letras[i]) == alf[j] Entonces
                j = j + arregloClave[contador]

                si j > 41 Entonces
                    j = j - 42
                FinSi

                salida = salida + alf[j]
                j = 42
            Sino
                si Mayusculas(letras[i]) == " " entonces
                    contador = contador + 1
                    j = 42
                FinSi
            FinSi
        FInPara
    FinPara



    escribir "La frase que ingresaste es: " fraseE
    escribir "La encriptación es: " salida
    escribir "La clave de encriptación es: " clave

 FinAlgoritmo
```

Codificación en Python Algoritmo 3

```
if __name__ == '__main__':
    alf = str()
    salida = str()
```

```
espacio = str()
clave = str()
# utilizamos contador dentro de distintos for
contador = int()
arregloclave = int()
# definimos y declaramos el arreglo con el abecedario
alf = [str() for ind0 in range(42)]
alf[0] = "A"
alf[1] = "B"
alf[2] = "C"
alf[3] = "D"
alf[4] = "E"
alf[5] = "F"
alf[6] = "G"
alf[7] = "H"
alf[8] = "I"
alf[9] = "J"
alf[10] = "K"
alf[11] = "L"
alf[12] = "M"
alf[13] = "N"
alf[14] = "Ñ"
alf[15] = "O"
alf[16] = "P"
alf[17] = "Q"
alf[18] = "R"
alf[19] = "S"
alf[20] = "T"
alf[21] = "U"
alf[22] = "V"
alf[23] = "W"
alf[24] = "X"
alf[25] = "Y"
alf[26] = "Z"
alf[27] = "0"
alf[28] = "1"
alf[29] = "2"
alf[30] = "3"
alf[31] = "4"
alf[32] = "5"
alf[33] = "6"
alf[34] = "7"
alf[35] = "8"
alf[36] = "9"
alf[37] = "Á"
alf[38] = "É"
alf[39] = "Í"
alf[40] = "Ó"
alf[41] = "Ú"
print("escriba frase")
```

```
frasee = input()
sizefrasee = len(frasee)
salida = ""

# definimos arreglo letras para iterar por cada letra de la fraseE
letras = [str() for ind0 in range(sizefrasee)]
tamanoarregloclave = 1
# determinar tamaño de arreglo clave
for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]
    if letras[i]==" ":
        tamanoarregloclave = tamanoarregloclave+1
arregloclave = [int() for ind0 in range(tamanoarregloclave)]
# Fin configuracion arreglo clave

# llenado de arregloClave
anterior = 0
contador = 0
for i in range(len(frasee)+1):
    if (frasee[i:i+1]==" "):
        arregloclave[contador] = i-anterior
        clave = clave+str(arregloclave[contador])
        anterior = i+1
        contador = contador+1
        #print(clave,"c")
    else:
        if (i==(len(frasee))):
            arregloclave[contador] = i-anterior
            clave = clave+str(arregloclave[contador])
            anterior = i+1
            contador = contador+1
# fin llenado de arregloClave


# a cero el contador para reutilizarlo
contador = 0
for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]
    # para recorrer arregloClave alf
    for j in range(42):
        if str.upper(letras[i])==alf[j]:
            j = j + arregloclave[contador]
            if j > 41:
                j = j-42
            salida = salida + alf[j]
            break
        else:
            if str.upper(letras[i])==" ":
                contador = contador+1
                break
print("La frase que ingresaste es: ",frasee)
```

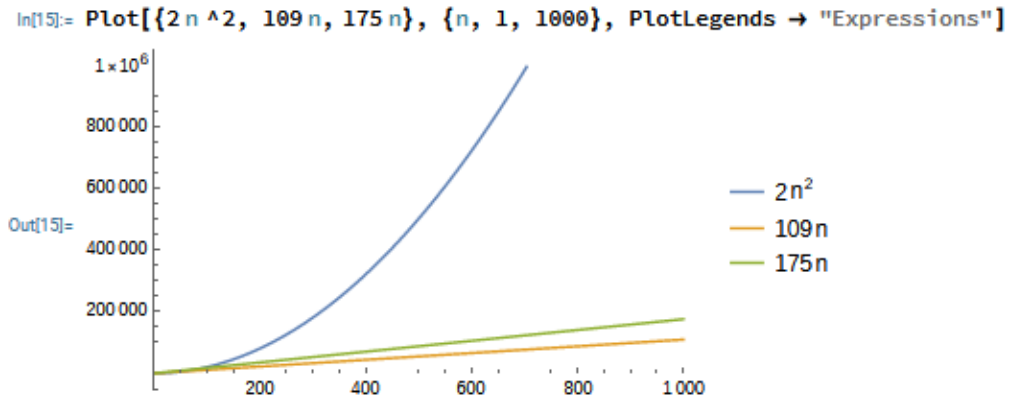
```
print("La encriptación es: ",salida)
print("La clave de encriptación es: ",clave)
```

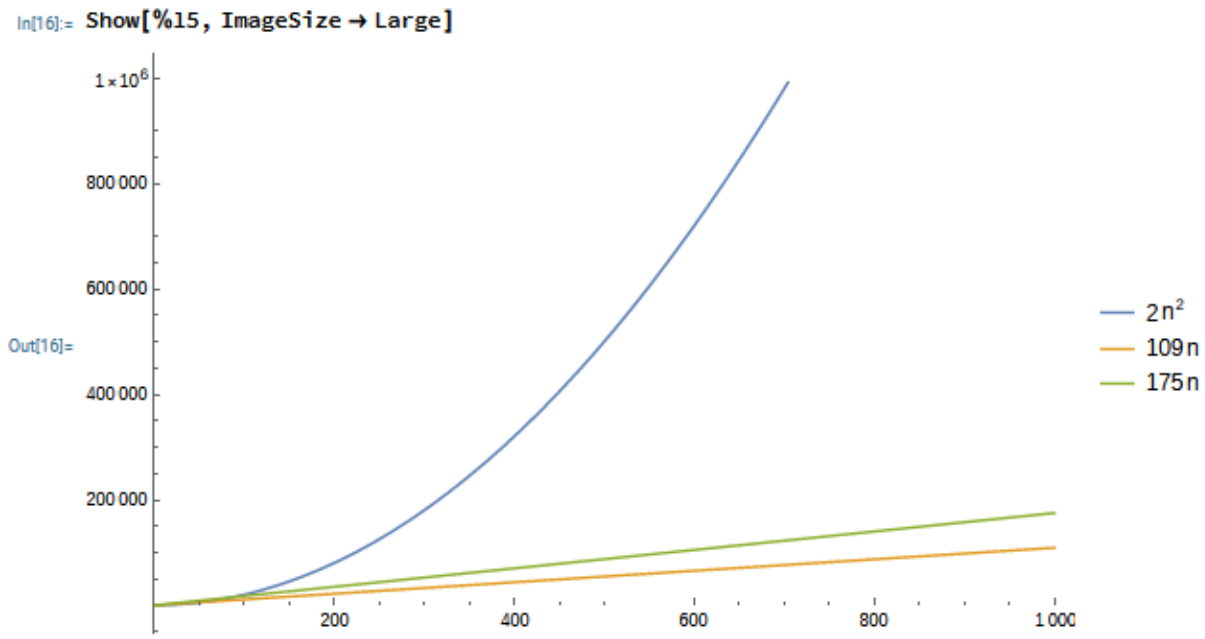
Complejidad Algoritmica en términos de Big Omicron Algoritmo 3

Es O(14+2n+1+2+4n+1+n+168n+3) => O(n)

Orden de Complejidad Algoritmo 3 orden lineal

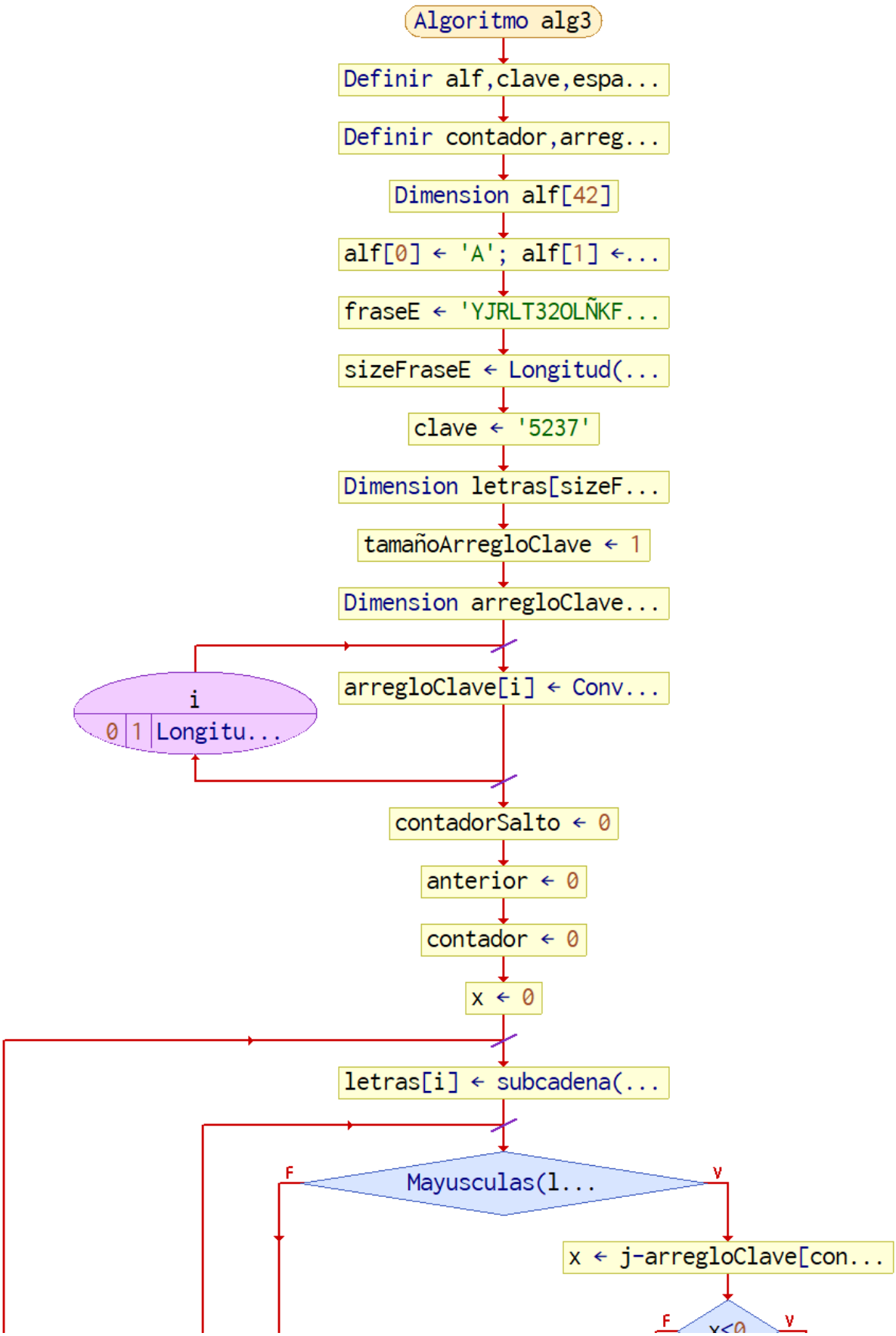
Gráficos de complejidad algoritmica en Wolfram Algoritmo 3

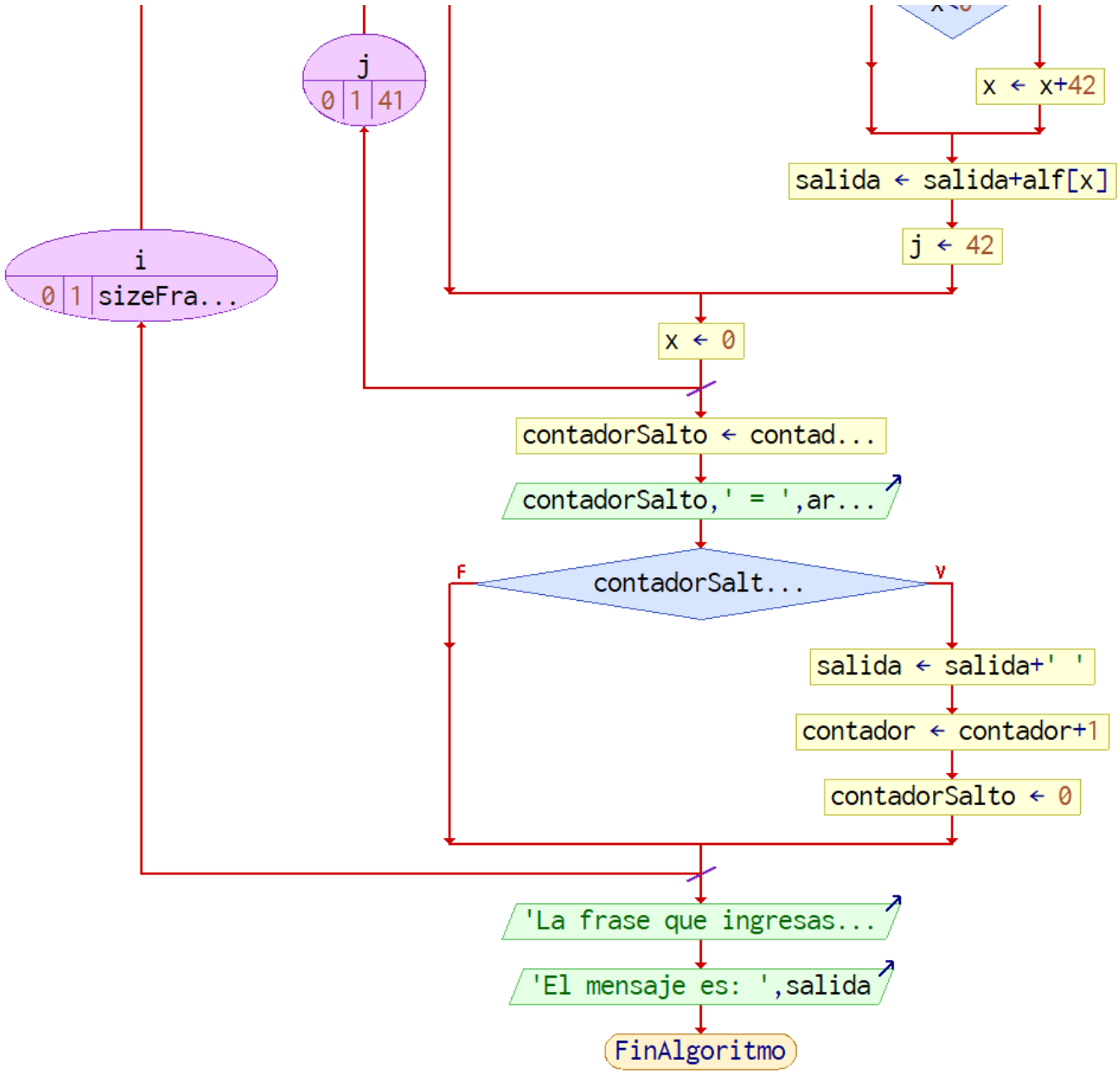




Algoritmo 3 Desencriptado

Flujograma Algoritmo 3 Desencriptación





Pseudocódigo Algoritmo 3 Desencriptación

```
# Algoritmo alg3

    alf, clave, espacio, salida es caracteres
    contador, arregloClave es entero //utilizamos contador dentro de distintos for

    //definimos y declaramos el arreglo con el abecedario
    Dimension alf[42];
    alf[0] = "A"; alf[1] = "B"; alf[2] = "C"; alf[3] = "D";alf[4] = "E"; alf[5] = "F"; alf[6] = "
```

```
fraseE = "YJRLT32OLÑKFRHYLZ"
sizeFraseE = longitud(fraseE)

clave = "5237"

//definimos arreglo letras para iterar por cada letra de la fraseE
dimension letras[sizeFraseE]
tamañoArregloClave = 1

//determinar tamaño de arreglo clave
dimension arregloClave[longitud(clave)]
//Fin configuracion arreglo clave

//llenado de arregloClave
Para i <- 0 hasta Longitud(clave) - 1 con paso 1 Hacer
    arregloClave[i] = ConvertirANumero(subcadena(clave,i,i))
FinPara
//fin llenado de arregloClave

contadorSalto = 0
anterior = 0
contador = 0//a cero el contador para reutilizarlo
x=0
Para i <- 0 hasta sizeFraseE-1 con paso 1 Hacer
    letras[i] = subcadena(fraseE,i,i)
    //para recorrer arregloClave alf
    Para j <- 0 hasta 41 con paso 1 Hacer
        si Mayusculas(letras[i]) == alf[j] Entonces
            x = j - arregloClave[contador]

            si x < 0 Entonces
                x = x + 42
            FinSi
            salida = salida + alf[x]
            j = 42
        FinSi
        x = 0
    FInPara

    contadorSalto = contadorSalto + 1
    escribir contadorSalto ' = ' arregloClave[contador]
    si contadorSalto  = arregloClave[contador] y contador <= Longitud(clave) entonces
        salida = salida + " "
```

```
            contador = contador + 1
            contadorSalto = 0
        FinSi


    FinPara



    escribir "La frase que ingresaste es: " fraseE
    escribir "El mensaje es: " salida


FinAlgoritmo
```

Codificación en Python Algoritmo 3 Desencriptación

```
if __name__ == '__main__':
    alf = str()
    clave = str()
    espacio = str()
    salida = str()
    # utilizamos contador dentro de distintos for
    contador = int()
    arregloclave = int()
    # definimos y declaramos el arreglo con el abecedario
    alf = [str() for ind0 in range(42)]
    alf[0] = "A"
    alf[1] = "B"
    alf[2] = "C"
    alf[3] = "D"
    alf[4] = "E"
    alf[5] = "F"
    alf[6] = "G"
    alf[7] = "H"
    alf[8] = "I"
    alf[9] = "J"
    alf[10] = "K"
    alf[11] = "L"
    alf[12] = "M"
    alf[13] = "N"
    alf[14] = "Ñ"
    alf[15] = "O"
    alf[16] = "P"
```

```
alf[17] = "Q"
alf[18] = "R"
alf[19] = "S"
alf[20] = "T"
alf[21] = "U"
alf[22] = "V"
alf[23] = "W"
alf[24] = "X"
alf[25] = "Y"
alf[26] = "Z"
alf[27] = "0"
alf[28] = "1"
alf[29] = "2"
alf[30] = "3"
alf[31] = "4"
alf[32] = "5"
alf[33] = "6"
alf[34] = "7"
alf[35] = "8"
alf[36] = "9"
alf[37] = "Á"
alf[38] = "É"
alf[39] = "Í"
alf[40] = "Ó"
alf[41] = "Ú"
print("escriba encriptado")
frasee = input()
#frasee = "YJRLT32OLÑKFRHYLZ"
sizefrasee = len(frasee)
print("escriba clave")
clave = input()
#clave = "5237"

# definimos arreglo letras para iterar por cada letra de la fraseE
letras = [str() for ind0 in range(sizefrasee)]
tamanoarregloclave = 1

# determinar tamaño de arreglo clave
arregloclave = [int() for ind0 in range(len(clave))]
# Fin configuracion arreglo clave

# llenado de arregloClave
for i in range(len(clave)):
    arregloclave[i] = int(clave[i:i+1])
# fin llenado de arregloClave

contadorsalto = 0
anterior = 0
# a cero el contador para reutilizarlo
contador = 0
x = 0
```

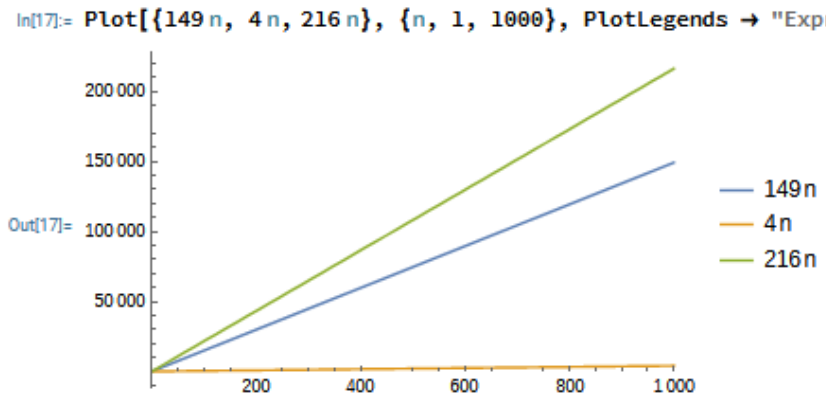
```
x = 0
for i in range(sizefrasee):
    letras[i] = frasee[i:i+1]
    # para recorrer arregloClave alf
    for j in range(42):
        if str.upper(letras[i])==alf[j]:
            x = j-arregloclave[contador]
            if x<0:
                x = x + 42
            salida = salida+alf[x]
            break
        x = 0

    contadorsalto = contadorsalto+1
    #print(contadorsalto," = ",arregloclave[contador])
    if contadorsalto==arregloclave[contador] and contador<=len(clave):
        salida = salida+" "
        contador = contador+1
        contadorsalto = 0
print("Ingresaste es: ",frasee)
print("El mensaje es: ",salida)
```

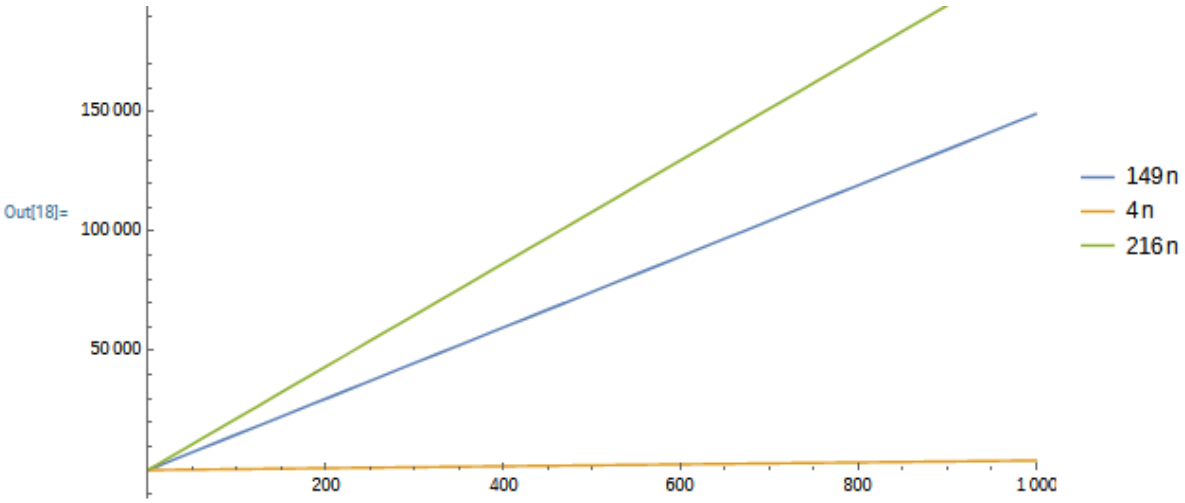
Complejidad Algoritmica en términos de Big Omicron Algoritmo 3 Desencriptación

Es O(15+n+4+n+210n+4n+2) => O(n)

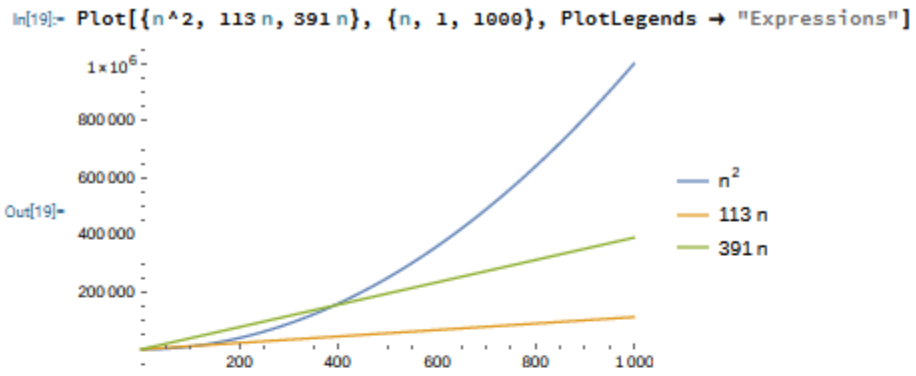
Orden de Complejidad Algoritmo 3 Desencriptación es de orden lineal

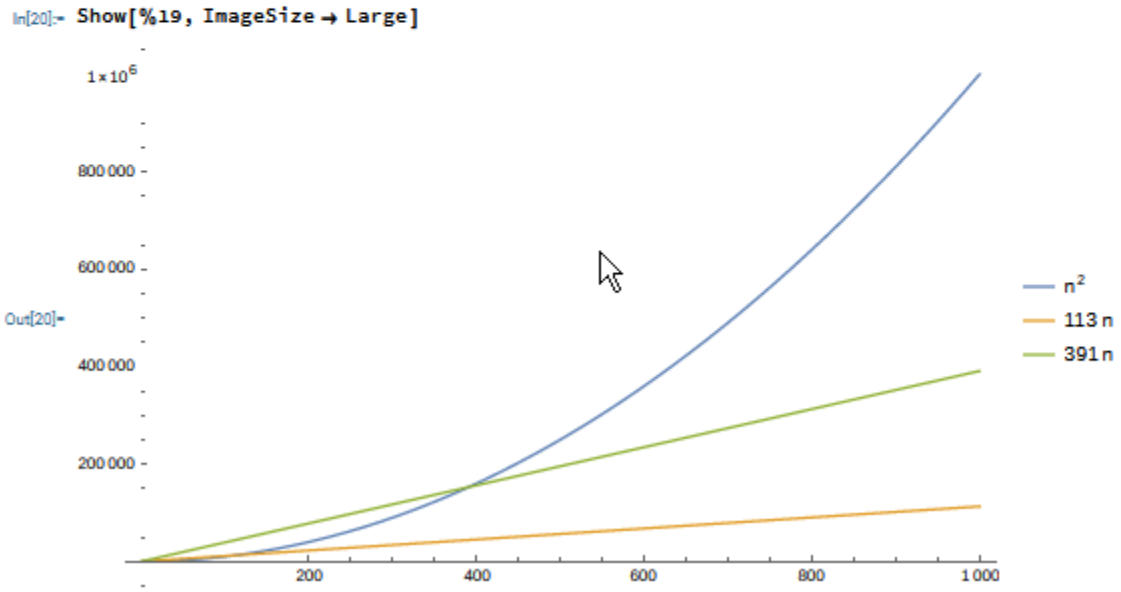
Gráficos de complejidad algoritmica en Wolfram Algoritmo 3 Desencriptación





Unión de Encriptación con desencriptación





Bibliografía

[1] M. J. Prieto and H. De, “Historia de la criptografía”.

[2] M. P. Fernández, “Criptografía Hispanoamericana Durante El Siglo Xvi,” 2013.

[3] M. I. S. K. Pozzo, “Culturas y lenguas: la impronta cultural en la interpretación linguística.,” 2011.

Anexos

Agregar enlace git: https://github.com/DavidSotoX/ProyectoFinalComplejidadComputacional.git

Enlace del proyecto en Wolfram: https://www.wolframcloud.com/env/davidsoto/ProyectoFinal.nb

Enlace público de Google Colab: https://colab.research.google.com/drive/1U13W0toEEzoO4hF9HrzKY6pZtQKtAdcF?usp=sharing