

Códigos y criptografía: Curso 2022-2023

Práctica 1: Cifrado afín.

Cifrado César (caso particular del afín)

- Mientras no se diga lo contrario, nuestro abecedario será

‘abcdefghijklmnopqrstuvwxyz’

- A cada letra le vamos a asociar un número:

a	b	c	d	...	y	z
0	1	2	3	...	25	26

1. Función *letter_number*

```
1 function num=letter_number(text)
```

Se trata de una función que convierte las letras en números de modo que podamos operar con ellos.

Entrada: texto escrito como si fuera una cadena de caracteres (puede contemplar minúsculas, mayúsculas, tildes, signos de puntuación etc.).

NOTA: El texto lo introduciremos entre comillas simples para crear un elemento de tipo *char* (array o matriz de caracteres). Trataremos así cualquier entrada de tipo texto a lo largo de las prácticas.

Salida: vector numérico asociado al texto, una vez convertidas todas las letras en minúsculas y eliminados todos los símbolos que no estén en nuestro alfabeto.

Ejemplo:

```
1 >> num=letter_number('Hola, buenos días, ¿empezamos?')
2 num = 7 15 11 0 1 21 4 13 15 19 3 8 0 19 4 12 16 4 26 0 12 ...
      15 19
```

2. Función *affine*

```
1 function code=affine(k,d,text)
```

Se trata de una función que cifra un mensaje por el método afín.

Entradas:

k: la clave multiplicativa. El programa debe comprobar que es un número de \mathbb{Z}_{27} y que $\text{mcd}(k, 27) = 1$.

d: el desplazamiento. También debe ser un número de \mathbb{Z}_{27} .

text: el texto claro que se quiere encriptar.

Salida: el criptograma.

Ejemplos:

```
1 >>code=affine(14,7,'este metodo tambien es facil')
2 code = j dq j n j q b v b q h n u l j a j d w h i l z
```

```
1 >>code=affine(15,7,'este metodo tambien es facil')
2 Error using affine (line ...)
3 The multiplicative key should satisfy gcd(k,27)=1.
```

3. Función *dec_affine*

```
1 function text=dec_affine(k,d,code)
```

Se trata de una función que descifra un mensaje que ha sido cifrado por el método afín conociendo las claves de cifrado.

Entradas:

k: la clave multiplicativa empleada en el método.

d: el desplazamiento empleado en el método.

code: el texto encriptado del que se pretende obtener el texto claro.

Salida: el mensaje claro.

Ejemplo:

```
1 >> text=dec_affine(14,7,'j dq j n j q b v b q h n u l j a j d w h i l z')
2 text = estemetodotambienesfacil
```

- Las siguientes funciones van encaminadas a realizar un análisis de frecuencias de las letras de un criptograma.
- Servirán para poder realizar el criptoanálisis y encontrar el mensaje original, suponiendo que se conozca el criptograma pero no las claves empleadas.

4. Función *crypt_ana_order*

```
1 function [freq,freq_order]=crypt_ana_order(v)
```

Se trata de una función que obtiene las frecuencias de cada letra de un criptograma.

Entrada: el criptograma. Se introducirá como un array de caracteres, es decir, entre comillas simples.

Salidas:

freq: una matriz 27×2 , cuya primera columna consiste en las frecuencias con las que aparece cada elemento de nuestro alfabeto en el criptograma y en la segunda se tienen los números de las letras correspondientes a esas frecuencias, ordenados según el orden usual de los caracteres en el alfabeto.

freq_order: la matriz 27×2 obtenida al ordenar de mayor a menor la primera columna de la matriz *frecuencia*.

Ejemplo:

```
1 v='ltbtrbnhyklrshstjñhkljbfvtvsiyltvxbolyvhjvykhyslfhklshztvcvfhlyzjyoioythkhshzwv
2 yxbllzavlyzbtaleavvhyhclyzombtjovthtrvzwyvnyhshzxbllzavfñhjoltkv'
3 >> [freq,freq_order]=crypt_ana_order(v)
4 freq =
5 0.028369 0
6 0.056738 1
7 0.014184 2
8 0 3
9 0.0070922 4
10 0.028369 5
11 0 6
12 0.12766 7
13 0.014184 8
14 0.042553 9
15 0.042553 10
16 0.11348 11
17 0.0070922 12
18 0.014184 13
19 0.014184 14
20 0.042553 15
21 0 16
22 0 17
23 0.021277 18
24 0.042553 19
25 0.085106 20
26 0 21
27 0.11348 22
28 0.014184 23
29 0.021277 24
30 0.085106 25
31 0.06383 26
32 freq_order =
33 0.12766 7
34 0.11348 11
35 0.11348 22
36 0.085106 20
37 0.085106 25
38 0.06383 26
39 0.056738 1
40 0.042553 9
```

41	0.042553	10
42	0.042553	15
43	0.042553	19
44	0.028369	0
45	0.028369	5
46	0.021277	18
47	0.021277	24
48	0.014184	2
49	0.014184	8
50	0.014184	13
51	0.014184	14
52	0.014184	23
53	0.0070922	4
54	0.0070922	12
55	0	3
56	0	6
57	0	16
58	0	17
59	0	21

5. Función *bars*

```
1 function compare=bars(v)
```

Se trata de una función que compara las frecuencias del criptograma con las frecuencias de las letras en castellano, mostrando la comparación de dos formas: mediante una matriz de datos y mediante un diagrama de barras.

Entrada: el criptograma.

Salida: una matriz 27×4 donde las columnas 1 y 3 son las frecuencias ordenadas de mayor a menor en castellano y en nuestro criptograma:

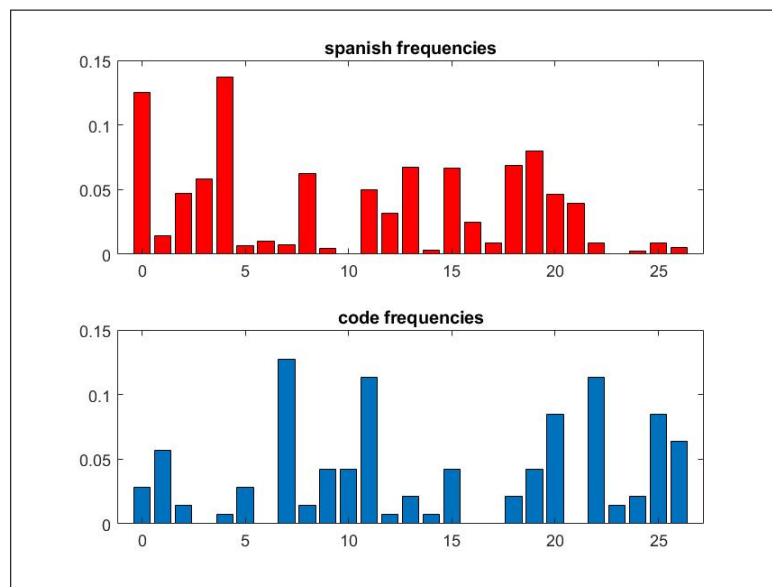
‘sp. freq.’ ‘letter num.’ ‘code freq.’ ‘letter num.’

Adicionalmente deben mostrarse dos diagramas de barras: el primero indicando las frecuencias de las letras en castellano y el segundo las frecuencias de cada letra en el criptograma.

Ejemplo:

```
1 v='ltbtrbnhyklrshstjñhkljbfvtvsiyltvxbolyvhjvykhyslfhklshztvcvfhlfzjyoioythkhshzwv
2 yxbllzavlzbtaleavvyhclyzombtjovthtrvzwyvnyhshzxbllzavfñhjoltkv'
3 >>'spanish' 'letter' 'code' 'letter'
4 >>compare=bars(v)
5 compare =
6     0.1368         4     0.12766         7
7     0.1253         0     0.11348        11
8     0.0868        15     0.11348        22
9     0.0798        19     0.085106       20
10    0.0687        18     0.085106       25
11    0.0671        13     0.06383        26
12    0.0625         8     0.056738         1
13    0.0586         3     0.042553         9
14    0.0497        11     0.042553        10
15    0.0468         2     0.042553        15
16    0.0463        20     0.042553        19
17    0.0393        21     0.028369         0
```

18	0.0315	12	0.028369	5
19	0.0251	16	0.021277	18
20	0.0142	1	0.021277	24
21	0.0101	6	0.014184	2
22	0.009	22	0.014184	8
23	0.009	25	0.014184	13
24	0.0088	17	0.014184	14
25	0.007	7	0.014184	23
26	0.0069	5	0.0070922	4
27	0.0052	26	0.0070922	12
28	0.0044	9	0	3
29	0.0031	14	0	6
30	0.0022	24	0	16
31	0.0002	10	0	17
32	0.0001	23	0	21



6. Función *inv_module*

```
1 function inver=inv_module(A,m)
```

Se trata de una función que calcula la inversa modular de una matriz con coeficientes enteros.

Entradas:

A: la matriz de la queremos calcular su inversa.

m: el módulo de trabajo.

Salida: la inversa modular de la matriz dada.

Ejemplos:

```
1 >> inver=inv_module([3 4.4;5 6],27)
2 Error using inv_module (line ...)
3 All the elements should be integer numbers.
```

```

1     >> inver=inv_module([3 0;5 6],27)
2     Error using inv_modulo (line ...)
3     The matrix is not inversible with the given module.

```

```

1     >> inver=inv_module([3 2;5 6],27)
2     inver =
3         21     20
4         23     24

```

7. Función *affine_cryptanalysis*

```

1     function affine_cryptanalysis(v)

```

Se trata de una función que realiza el criptoanálisis de un mensaje cifrado por el método afín, desconociendo las claves empleadas.

Entrada: el criptograma.

Salida: se espera una salida interactiva.

La función compara las máximas frecuencias, hace un intento de descifrar el mensaje, nos lo muestra y nos pregunta si queremos probar con otras claves.

Si la respuesta es afirmativa pasa a comparar otras dos frecuencias y nos muestra otro posible texto claro etc. En el momento en que la respuesta sea negativa debe mostrar el texto claro y el valor correcto de las claves.

Ejemplo:

```

1     v='eymcklcdmgdcscmeligvcqwseiwyccklevqgqwcldgrlcldwveuiemwqwcrgvbmccipgevseiwyccklev
2     qgqwcscdelucgvnelyciwqzucl'
3
4     >>affine_cryptanalysis(v)
5
6     k = 13
7     d = 4
8
9     text =
10    'amleoneclwcexelanswrebpxaspmeonarbwbecwznencpratsalpbpezwrglesdwarxaspmeonarbw
11    bpexecantewrjanmespbkten'
12
13    if you want to prove with different keys write 1, otherwise write 0: 1
14
15    ....
16
17    k =14
18    d = 2
19
20    text =
21
22    'estapRACTICahaterminadohemosaprendidoacifrarconelmetodoafinytambienhemosaprendid
23    oahacerlainversamodular'
24
25    if you want to prove with different keys write 1, otherwise write 0: 0

```