SPD - Exercici 1: Mètodes criptogràfics senzills

Autor: Francesc Xavier Bullich Parra

Implementació dels mètodes criptogràfics

Xifrat Cesar

Veure fitxer cesar.py

El mètode per obtenir el xifrat és molt simple.

Donat que es treballa amb caracters de l'alfabet anglès es pot aprofitar la codificació dels caracters en ASCII.

- Primer s'obté el número del caràcter en qüestió i se li resta el número que representa el caràcter 'a'.
- Un cop s'ha passat el caràcter a un número entre el 0 i el 25 s'aplica el desplaçament de forma modular (si es passa de 25 torna a començar a 0).
- Finalment s'obté el caràcter sumant el valor obtingut a al número que representa el caràcter 'a'.

```
    posZero = ord('a')
    return chr((ord(lowerChar)-posZero + despl) % L + posZero)
```

Per desxifrar un caràcter es fa la mateixa operació amb 1 canvi. En comptes de sumar el desplaçament un cop obtingut el valor del caràcter entre 0 i 25, el que es fa és restar-lo. De forma que s'obté el caràcter original.

```
    posZero = ord('a')
    return chr((ord(lowerChar)-posZero - despl) % L + posZero)
```

Proves

Es prova amb un text molt simple i desplaçament 1 per veure el correcte comportament modular.

```
python cesar.py
entreu un nombre natural corresponent al desplaçament: 1
DESPLAÇAMENT:  1
entra el text que vols xifrar: a z
TEXT XIFRAT:  b a
TEXT ORIGINAL:  a z
```

Com es pot comprovar amb un desplaçament = 1 la 'a' passa a ser 'b' i la 'z' es transforma en 'a' de forma correcta.

Altres proves més complexes

```
python cesar.py
entreu un nombre natural corresponent al desplaçament: 8
DESPLAÇAMENT:  8
entra el text que vols xifrar: 8 Aquesta es una prova. Hi ha caracters que no son a-z (aquests caracters apareixen tal qual)
TEXT XIFRAT:  8 iycmabi ma cvi xzwdi. pq pi kizikbmza ycm vw awv i-h (iycmaba kizikbmza ixizmqfmv bit ycit)
TEXT ORIGINAL:  8 aquesta es una prova. hi ha caracters que no son a-z (aquests caracters apareixen tal qual)
```

Xifrat Polybios

Veure fitxer polybios.py

Aquest tipus de xifrat es basa en una taula on es codifiquen els diferents caràcters de l'alfabet. En aquest cas l'alfabet anglès de 26 caràcters.

Es proposa una solució que permet codificar la taula en una mida de mínima de 5x5 (col·lisionant la i i la j com la que s'ha vist a classe). Per mides superiors per ex: 5x6 es codifiquen els caràcters sense col·lisions (s'haurà de tenir en compte que hi ha més posicions que caràcters).

Una possible solució seria actuar sobre una taula de 2 dimensions codificant a cada posició un dels possibles caràcters.
S'haurà de tenir en compte alhora de crear-la de les possibles col·lisions.
En el fitxer polybios.py hi ha un exemple de creació de taula tinguent en compte les col·lisions del 5x5.

Aquesta solució permet:

- Buscar la codificació d'un text en o(n^2) segons l'entrada (per cada caràcter s'haurà de buscar quin és la seva codificació)
- Desxifrar un text en o(n) segons l'entrada (per a cada codificació es pot accedir al caràcer amb accés directe)

Es proposa una solució que utilitza la taula de codificació de forma teórica.
En comptes de generar una taula i posteriorment buscar la codificació/descodificació d'un caràcter el que es fa és calcular la posició mitjançant el nombre de files i columnes.

```
if lowerChar >= 'a' and caracter <='z':
    posicio = ord(lowerChar) - ord('a')
    primer = posicio // columnes
    segon = posicio % columnes
```

Aquest seria l'esquema simple si no hi haguessin col·lisions. Posteriorment només s'hauria de transformar els valors primer i segon a la codificació desitjada. En aquest cas es transformen en lletres majúscules.

```
posZeroRes = ord('A')
return chr(posZeroRes + primer) + chr(posZeroRes + segon)
```

En aquesta proposta només es permet 1 col·lisio. Aquesta es controla movent els elements necessaris a posicions anteriors d'aquesta forma.

```
if (files * columnes) < L and caracter >= 'j':
    # si hi ha colisio (maxim 1) fem que la i i la j vagin juntes
    # hem de fer corre enrera les altres lletres 1 posicio
    segon = ((segon - 1) % columnes)
    if segon >=  (ord('j') - ord('a')) % columnes:
        # Com es colisiona la j, si la columna resultant es superior a la posicio de j:
        # hem de fer que les files tirin 1 valor enrere. per ex en un 5x5 la K seria la posicio CA,  pero ha de ser BE
        primer =((primer -1) % columnes)
```

Per la descodificació també s'ha de tenir en compte que pot existir la col·lisió ij.

```
pos1 = (ord(primer) - ord('A'))*columnes
    pos2 = ord(segon) - ord('A')
    if files * columnes < L and pos2 + pos1 >=  (ord('j') - ord('a')):
        ## si hi ha colisió, a partir de la j hem de sumar 1 a tots els caracters
        pos1 +=1
    return chr(ord('a') + pos2 + pos1)
```

Aquesta solució permet:

- Codificar el text en o(n) segons l'entrada
- Descodificar el text en o(n) segons l'entrada

Proves

Proves d'execució amb tauler de 5x5 per comprovar com funcionen les col·lisions:

```
python polybios.py
Entra un nombre de files i columnes que compleixin num_files x num_columnes >=25
Entra el nombre de files: 5
Entra el nombra de columnes: 5
entra el text que vols xifrar: a b c i j k z
TEXT XIFRAT:  AA AB AC BD BD BE EE
TEXT ORIGINAL:  a b c i i k z
```

Es pot comprovar que la 'i' i la 'j' col·lisionen en la posició BD.
També es pot veure que la 'k' es posiciona correctament a la BE i la z a la última posicio EE

El problema de la col·lisió és que no permet saber quin dels 2 caràcters era l'original. Com es pot veure en el text desxifrat apareixen 2 'i' en comptes de 'i' segit de 'j'

```
python polybios.py
Entra un nombre de files i columnes que compleixin num_files x num_columnes >=25
Entra el nombre de files: 5
Entra el nombra de columnes: 5
entra el text que vols xifrar: prova de polybios complexa. ja no es pot saber quin caracter de colisio es l'original
TEXT XIFRAT:  CEDBCDEAAA ADAE CECDCAEDABBDCDDC ACCDCBCECAAEECAA. BDAA CCCD AEDC CECDDD DCAAABAEDB DADEBDCC ACAADBAAACDDAEDB
```

```
ADAE ACCDCABDDCBDCD AEDC CA'CDDBBDBBBDCCAACA
TEXT ORIGINAL:  prova de polybios complexa. ia no es pot saber quin caracter de colisio es l'original
```

Com diu el text de prova, es perd la 'j' substituint-se per la 'i'. Tot i que un humà pot reconeixer quin caràcter hauria de ser l'original.

Proves de 5x6

```
python polybios.py
Entra un nombre de files i columnes que compleixin num_files x num_columnes >=25
Entra el nombre de files: 5
Entra el nombra de columnes: 6
entra el text que vols xifrar: a b c i j k l m n z
TEXT XIFRAT:  AA AB AC BC BD BE BF CA CB EB
TEXT ORIGINAL:  a b c i j k l m n z
```

Prova simple per comprovar com ara ja no hi ha col·lisio. També es pot veure que com hi ha més columnes disponibles ara la 'l' està a la mateixa fila que la 'j' i la 'k'.

```
python polybios.py
Entra un nombre de files i columnes que compleixin num_files x num_columnes >=25
Entra el nombre de files: 5
Entra el nombra de columnes: 6
entra el text que vols xifrar: com ja no hi ha colisions, aquest text es pot descodificar perfectament.
TEXT XIFRAT:  ACCCCA BDAA CBCC BBBC BBAA ACCCBFBCDABCCCCBDA, AACEDCAEDADB DBAEDFDB AEDA CDCCDB ADAEDAACCCADBCAFBCACAACF CDAECFAFAEACDBAACAAECBDB.
TEXT ORIGINAL:  com ja no hi ha colisions, aquest text es pot descodificar perfectament.
```

Xifrat Rail Fence

Veure fitxer railFence.py

De la mateixa forma que el xifrat Polybios, en comptes d'utilitzar una taula de N rails per codificar i descodificar el missatge, s'utilitza un mètode de càlcul de posicions per simular aquesta taula.

S'utilitza la seguent frase (vista a classe) sense espais per explicar el funcionament del mètode:
seraelpropermati

Creació de la taula de rail = 4

s						**p**						**m**			
	e				l		r				r		a		
		r		e				o		e				t	
			a						p						i

Si s'observa la taula es pot trobar un patrò que permet calculàr quina seria la següent posició.
En la primera fila podem trobar quin es el desplaçament des del primer caràcter fins el segon. Aquest es pot trobar a partir del nombre de rails. En aquest cas es de 6 posicions.
Es defineix un desplaçament de 2xnombre_rails -2
Aquest desplaçament també es pot veure en l'última fila.

Els rails intermitjos però no segueixen aquesta regla. Com es pot observar hi ha 2 mides de desplaçament entre els caràcters d'un rail.
S'observa el segon rail. De la 'e' a la 'l' hi ha un desplaçament de 4, mentre de de la 'l' a la 'r' hi ha un desplaçament de 2 posicions. El tercer rail també segueix aquest patró amb desplaçaments diferents.

Podem trobar el primer desplaçament amb la mateixa formula anterior [2xnombre_rails-2] - num_rail_actualx2
El segon desplaçament s'obte amb num_rail_actualx2

Nota: el numero de rail actual esta entre 0 (pel primer rail) i 3 (per l'últim)

Comprovació:

- Desplaçament e -l (segon rail):

```
(num segon rail = 1)
primer desplaçament = 2*4-2 - 2*1 = 4

segon desplaçament = 2*1 = 2
```

Si es van alternant aquests desplaçaments es poden anar generan les posicions corresponents a cada rail.

Veiem un exemple de la mateixa frase amb num_rails =7

s												**m**			
	e										r		a		
		r								e				t	
			a						p						i
				e				o							
					l		r								
						p									

S'observa el 3er rail:

```
(num tercer rail = 2)
primer desplaçament = 2*7-2 - 2*2 = 8

segon desplaçament = 2*2= 4
```

Es pot observar que des de la 'r' a la 'e' del 3er rail hi ha 8 posicions. També de la 'e' a la 't' hi ha 4 posicions.

Com es pot comprovar els desplaçaments calculats serveixen per a qualsevol nombre de rails.

S'utilitza la següent funció per obtenir la seguent posició segons aquests desplaçametns.

desp sempre és 2xnum_rails-2
diff val el desplaçament anterior. Es va alternant entre primer i segon desplaçament
mida es la llargada de la cadena

```
def obtenirNovaPosicio(posicio, rail, desp, diff, mida):
    # calcula la seguent posicio en base al nombre de num_rails
    if diff == 0:
        diff = desp
    # si diff = 0 es que som al primer o ultim rail, per tant hem de moure desp cada cop
    posicio = posicio + diff
    diff = desp-diff
    # excepte el primer i ultim rail, tots els rails tenen 2 desplaçaments:
    # el primer a desp - rail*2
    # el segon a rail*2
    # utilitza la variable diff per intercanviar aquest 2 desplaçaments
    if posicio >= mida:
        rail +=1;
        posicio = rail
        diff  = desp -2*rail

    return posicio, rail, diff
```

Amb aquesta funció es pot simular la taula de forma que tant serveix per codificar el missatge com per descodificar. Només s'ha de tenir en compte les assignacions que es fan.

Per codificar:
A mida que es construeix el missatge codificar calcula la posició d'on sha de llegir el caracter al missatge original

```
for k in range(len(missatge)):
        xifrat[k]= missatge[posicio]
        posicio, rail, diff = obtenirNovaPosicio(posicio,rail, desp, diff, len(missatge))
```

Per descodificar:
A mida que es llegeixen els caracters del missatge xifrat es calcula la posició on s'ha de posar al desxifrat

```
for k in range(len(missatge)):
        desxifrat[posicio] = missatge[k]
        posicio, rail, diff = obtenirNovaPosicio(posicio,rail, desp, diff, len(missatge))
```

Tal i com esta pensada la formula només serveix per num_rails > 1. Tot i que no te massa sentit tenir 1 rail (el missatge codificat = original) es corregeix el desplaçament a 1 si es demana num_rails = 1.

NOTA: Al tractarse d'un mètode de xifrat per transposició de càracters, el mètode propossat no discrimina entre lletres o simbols i ho transposa tot. Les codificacions resultants poden vairar si es compara amb un mètode que discrimini lletres i simbols.

Proves

Prova senzilla amb rails = 2

```
python railFence.py
entreu un nombre natural corresponent al rail: 2
Rail:  2
entra el text que vols xifrar: hola
TEXT XIFRAT:  hloa
TEXT ORIGINAL:  hola
```

Prova amb el text vist a classe sense espais

```
python railFence.py
entreu un nombre natural corresponent al rail: 4
Rail:  4
entra el text que vols xifrar: seraelpropermati
TEXT XIFRAT:  spmelrrareoetapi
TEXT ORIGINAL:  seraelpropermati
```

Prova del mateix text amb 7 rails

```
python railFence.py
entreu un nombre natural corresponent al rail: 7
Rail:  7
entra el text que vols xifrar: seraelpropermati
TEXT XIFRAT:  smeraretapieolrp
TEXT ORIGINAL:  seraelpropermati
```

Xifrat i desxifrat d'una cadena concreta

Es demana que es xifri i es desxifri un missatge en anglés amb uns paràmetres determinats per comprovar que els mètodes proposats funcionen correctament

Frase a provar:

it's the honky tonk w omen that gimme, gimme, gimme the honky tonk blues (honky tonk w omen, by the rolling stones)

Prova amb xifrat Cesar

Desplaçament = 17

```
python cesar.py
entreu un nombre natural corresponent al desplaçament: 17
DESPLAÇAMENT:  17
entra el text que vols xifrar: it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
TEXT XIFRAT:  zk'j kyv yfebp kfeb nfdve kyrk xzddv, xzddv, xzddv kyv yfebp kfeb sclvj (yfebp kfeb nfdve, sp kyv ifcczex jkfevj)
TEXT ORIGINAL:  it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
```

Prova amb xifrat Polybios

S'utilitza la taula amb col·lisions 5x5 amb la 'i' i la 'j' juntes.

```
python polybios.py
Entra un nombre de files i columnes que compleixin num_files x num_columnes >=25
Entra el nombre de files: 5
Entra el nombra de columnes: 5
entra el text que vols xifrar: it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
TEXT XIFRAT:  BDDD'DC DDBCAE BCCDCCBEED DDCDCCBE EBCDCBAECC DDBCAADD BBBDCBCBAE, BBBDCBCBAE, BBBDCBCBAE DDBCAE BCCDCCBEED DDCDCCBE ABCADEAEDC
(BCCDCCBEED DDCDCCBE EBCDCBAECC, ABED DDBCAE DBCDCACABDCCBB DCDDCDCCAEDC)
TEXT ORIGINAL:  it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
```

Prova de xifrat Rail Fence

Número de rails = 7
Recordar que el mètode proposat no discrimina entre lletres i simbols i per tant tots els caracters son codificats

```
python railFence.py
entreu un nombre natural corresponent al rail: 7
Rail:  7
entra el text que vols xifrar: it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
TEXT XIFRAT:  ikn,m (weotnye e meyt h oh tn'o mtmgi kosokmtrseshtohmigtnnenne o s  owaim hokukonylg)ten tgm,eh lyt,blnhk e b  i
TEXT ORIGINAL:  it's the honky tonk women that gimme, gimme, gimme the honky tonk blues (honky tonk women, by the rolling stones)
```

Criptoanàlisi

Veure fitxer criptoAnalisi.py

Es proposa el següent text per a ser analitzat i trencar la codificació. El text ha estat codificat en un dels 3 mètodes de xifrat anteriors.

xfimr litvxl. patm tkx px ebobgz yhk? tutgw hgxw ietvxl. b znxll px dghp max lvhkx. (lahp fnlm zh hg, ur jnxxg)

Pre-Anàlisi

Es conegut que el text ha estat codificat en un dels 3 metodes anteriors. Sabent aquesta informació, mirant el text, podem descartar el mètode polibyos.

El mètode polybios codifica cada un dels caràcters del text (només lletres) en coordenades XY. Tot i que en principi podrien estar codificades en minúscules (en comptes de majúscules com s'ha vist a classe) es pot observar que hi ha un caràcter 'b' que no te parella per tant aquest caràcter seria impossible de descodificar utilitzant el polybios. També podem veure que hi ha paraules de mida senar que tindrien el mateix problema.

Mètodes de trencament

Com ja s'ha descartat un dels mètodes de codificació, es proposen 2 métodes per mirar de trencar el codi.

Rail Fence

El primer serà provar amb rail Fence.
Aquí el més simple és inicialitzar el nombre de rails a 2 i anar augmentant de forma que es pugui anar mirant si algun dels numeros de rail és la solució.

Es proposa un mètode mostra per pantalla la solucio amb el num_rails actual i demana al usuari si vol provar amb un número de rails major.

```
python criptoAnalisi.py
entra el text que vols analitzar: xfimr litvxl. patm tkx px ebobgz yhk? tutgwhgxw ietvxl. b znxll px dghp max lvhkx. (lahp fnlm zh hg, ur jnxxg)
Amb quin tipus de desxifrat vols provar d'analitzar? [c = cesar, r = Rail Fence, altres = finalitzar] r
Desxifrat Rail Fence:
Prova dexifrar num_rails = 2
x fbi mzrn xllilt vpxxl .d gphapt mm atxk xl vphxk xe.b o(blgazh py hfkn?l mt uzthg whhgg,x wu ri ejtnvxxxlg.)
Vols provar amb mes rails? [s/n]s
Desxifrat Rail Fence:
Prova dexifrar num_rails = 3
xo(bfglzi aymhhkr?p  t ultfgiwnhtglxvwm xi eltzv.xhl .  pbh azgntx,lml  puxt rdkg hxpj  mnapxx xlxv hgkex).b
Vols provar amb mes rails? [s/n]s
Desxifrat Rail Fence:
Prova dexifrar num_rails = 4
xtbm kfxz n ipxzlxm lh erbp xo b hdglzggh iyp, htkm a?v xu txulrvtlgh kw.hxj.g x n(wp lxaiaehxpttv gfxmln)l.
Vols provar amb mes rails? [s/n]n
Vols provar un altre tipus de desxifrat? [c = cesar, r = Rail Fence, altres = finalitzar] n
```

S'ha provat fins a un num_rail de 30 pero no dona cap text que sigui reconeixible en anglès.

Mètode Cesar

L'estratègia en aquest mètode es provar amb diferents desplaçaments de forma que algun d'ells porporcioni un text reconeixible en anglès.

Per fer-ho s'utilitza la taula de freqüencies proporcionada a classe.
Els pasos son:

- Obtenir el caràcter amb més freqüencia del text codificat.
- Ordenar la primera taula segons de mes a menor freqüencia.
- Iterar per a cada un dels elements ordenats de la primera taula
- Obtenir un desplaçament entre el caràcter amb més freqüencia del text codificat i el caràcter actual dels elements ordenats
- Provar de descodificar el text amb el desplaçament obtingut

Per a cada caràcter de la taula ordenada és mostra una posible solució de descodificació i es demana si es vol continuar provant.

```
python criptoAnalisi.py
entra el text que vols analitzar: xfimr litvxl. patm tkx px ebobgz yhk? tutgwhgxw ietvxl. b znxll px dghp max lvhkx. (lahp fnlm zh hg, ur jnxxg)
Amb quin tipus de desxifrat vols provar d'analitzar? [c = cesar, r = Rail Fence, altres = finalitzar] c
e x
Desxifrat Cesar:
Provant desplaçament d =  19
Text desxifrat:
empty spaces. what are we living for? abandoned places. i guess we know the score. (show must go on, by queen)
Vols provar una altre opcio? [s/n] n
Vols provar un altre tipus de desxifrat? [c = cesar, r = Rail Fence, altres = finalitzar] n
```

Amb la primera iteració podem obtenir un text clar. tenim un desplaçament de 19 proporcionat per el caràcter amb més freqüència al text codificat 'x' comparat amb el caràcter amb més freqüència del alfabet anglès 'e'.

Resultat del CriptoAnàlisi

Amb les proves realitzades es pot obtenir aquesta solució:

Mètode de codificació: Cesar
Desplaçament: b = 19