Xifrat Àuric

Author : Marc Sànchez Pifarré

Primer pas: “Poguer Fer i Desfer”

“Sèrie de fibonacci”.

Raó Àurea :

$$(a+b)/a = a/b$$

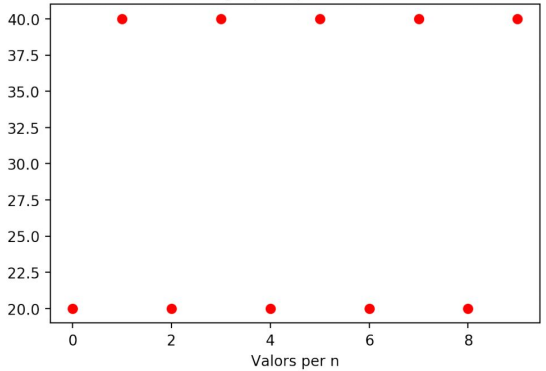
- Donada una parella de nombres extreure’n un de únic.
- Idea de clau pública i privada.

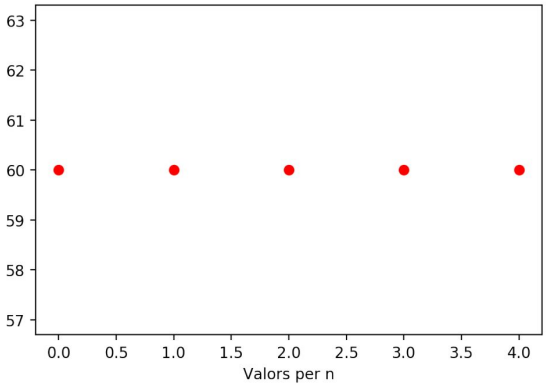
El meu cervell dóna moltes voltes...

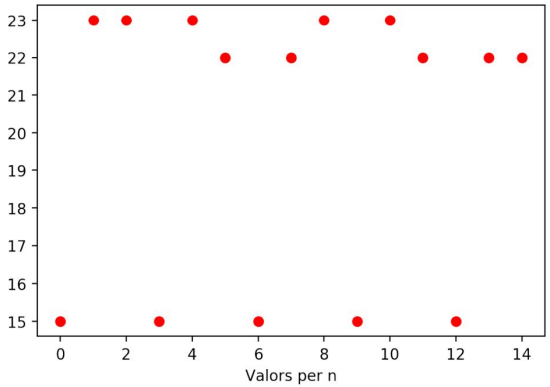
Estudi Série fibonacci

Donats els primers 300 nombres de fibonacci i f = 25

- Per tot i de 5 fins a f de amb increment de 5 (**POTRA**)
- Fer histograma de modularitats dels 300 nombres:

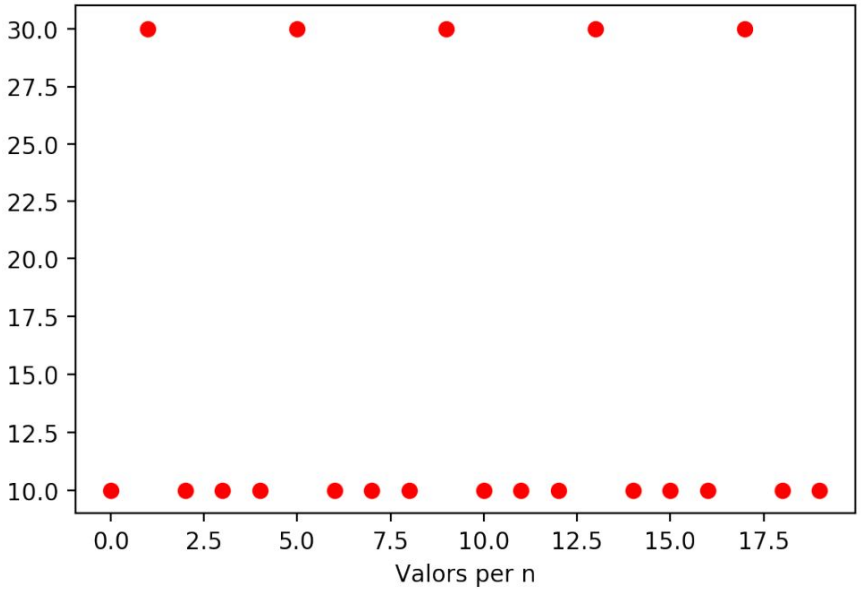


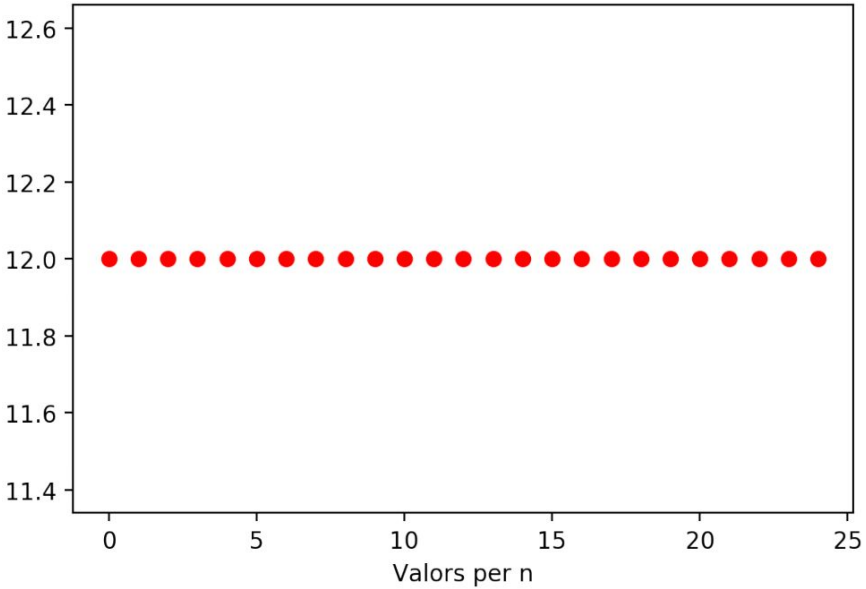




Estudi Série fibonacci

Cosa curiosa per f = 25





Recorregut cíclic

Si pogués tenir nombres que compartissin alguna propietat com la raó àurea, podria fer i desfer? Idea

- Z/5

a	b	c	d	e
0	1	2	3	4

Juguem una mica generant nombres aleatoris entre 1 i 100

```
$ python3 rotacio.py 20
Generem 20 nombres aleatoris :
{0: 6, 3: 4, 2: 4, 4: 4, 1: 2}
```

```
$ python3 rotacio.py 1000000
Generem 1000000 nombres aleatoris i els classifiquem :
{0: 199763, 2: 199614, 4: 200854, 1: 199492, 3: 200277}
```

Taula àurea : 25 columnes.

	a	b	c	d	e	f	g	h	i	j	k	l
	0	1	2	3	4	5	6	7	8	9	10	11
0	75025	1	2	3	514229	5	4181	28657	8	34	610	17711
1	12586269025	1	377	3524578	591286729879	55	4052739537881	1,3049695E+15	233	2584	1,9039249E+14	317811
2	2,1114851E+15	4807526976	14930352	1836311903	3,0806152E+14	102334155	3,7889062E+16	3,4164546E+15	701408733	2178309	2,3416728E+16	63245986
3	3,5422485E+20	2,18923E+20	8,3621143E+19	4,494557E+13	7,5401138E+18	3,1940435E+19	1,1000878E+18	5,5279397E+15	6,7989164E+17	4,6600466E+18	2,596955E+17	2504730781961
4	5,9425115E+25	5,7314784E+20	1,5005205E+21	2,4278932E+21	4,073058E+26	3,9284138E+21	3,3116481E+24	2,2698374E+25	6,356307E+21	2,6925749E+22	4,8316295E+23	1,4028367E+25
5	9,9692167E+30	9,2737269E+20	2,9861113E+23	2,7917155E+27	4,6834098E+32	4,3566776E+22	3,2100568E+33	1,0336283E+36	1,8455183E+23	2,0467111E+24	1,5080434E+35	2,5172883E+26
6	1,6724458E+36	3,8079019E+30	1,1825896E+28	1,4544891E+30	2,4400655E+35	8,10559E+28	3,0010821E+37	2,7060741E+36	5,555654E+29	1,725375E+27	1,8547708E+37	5,0095301E+28
7	2,8057117E+41	1,7340252E+41	6,6233869E+40	3,5600076E+34	5,9723043E+39	2,5299087E+40	8,7134745E+38	4,3785198E+36	5,3852234E+38	3,691087E+39	2,0569723E+38	1,9839242E+33
8	4,7068901E+46	4,5397369E+41	1,1885186E+42	1,9230634E+42	3,2261504E+47	3,111582E+42	2,6230599E+45	1,797872E+46	5,0346454E+42	2,13271E+43	3,8269929E+44	1,111146E+46
9	7,8963258E+51	7,3454487E+41	2,3652117E+44	2,2112364E+48	3,7095923E+53	3,4507973E+43	2,5425924E+54	8,1870685E+56	1,4617812E+44	1,6211402E+45	1,1944772E+56	1,9938706E+47
10	1,3246955E+57	3,0161281E+51	9,3669477E+48	1,1520584E+51	1,9327047E+56	6,4202015E+49	2,3770697E+58	2,1434024E+57	4,4004716E+50	1,3666193E+48	1,4691098E+58	3,9679027E+49
11	2,2223224E+62	1,3734708E+62	5,2461917E+61	2,8197782E+55	4,7304881E+60	2,0038669E+61	6,9016891E+59	3,4680979E+57	4,2654784E+59	2,9236024E+60	1,6292678E+59	1,5714085E+54

m	n	o	p	q	r	s	t	u	v	w	x	y
12	13	14	15	16	17	18	19	20	21	22	23	24
987	13	89	6765	165580141	24157817	46368	144	1134903170	21	1597	2971215073	7778742049
5702887	2,777789E+13	4,9845401E+14	832040	956722026041	225851433717	121393	1346269	139583862445	10946	86267571272	53316291173	20365011074
433494437	1,6050064E+17	8,9443943E+15	9227465	7,272346E+13	6557470319842	196418	39088169	1548008755920	267914296	9,9194853E+16	1,061021E+13	32951280099
365435296162	1,220016E+19	1,7799794E+18	1,716768E+13	6,1305791E+16	1,9740274E+19	8,0651553E+14	1,1766903E+14	2,8800672E+18	1,4472334E+16	5,1680709E+19	4,2019614E+17	1,3530185E+20
7,8177408E+23	1,0284721E+22	7,0492525E+22	5,3583593E+24	1,311512E+29	1,9134702E+28	3,6726741E+25	1,140593E+23	8,9892371E+29	1,6641028E+22	1,264937E+24	2,3534128E+30	6,1613147E+30
4,5170905E+27	2,2002057E+34	3,9481089E+35	6,5903462E+26	7,5779162E+32	1,7889033E+32	9,6151855E+25	1,0663404E+27	1,1056031E+32	8,6700074E+24	6,8330028E+31	4,223028E+31	1,6130531E+31
3,433583E+29	1,2712788E+38	7,0845939E+36	7,308806E+27	5,7602132E+34	5,193981E+33	1,5557697E+26	3,0960599E+28	1,2261326E+33	2,122071E+29	7,8569351E+37	8,4040378E+33	2,6099748E+31
2,8945064E+32	9,6633913E+39	1,4098698E+39	1,3598019E+34	4,8558529E+37	1,5635696E+40	6,3881744E+35	9,3202208E+34	2,2812172E+39	1,1463114E+37	4,0934782E+40	3,3282511E+38	1,0716865E+41
6,1922045E+44	8,1462274E+42	5,5835073E+43	4,2442001E+45	1,0388104E+50	1,515604E+49	2,909018E+46	9,0343046E+43	7,1201126E+50	1,3180873E+43	1,0019197E+45	1,8640697E+51	4,8801977E+51
3,5778557E+48	1,7427188E+55	3,1271819E+56	5,2200211E+47	6,0022464E+53	1,4169382E+53	7,6159081E+46	8,4461715E+47	8,7571595E+52	6,86726E+45	5,4122222E+52	3,3449373E+52	1,2776524E+52
2,719641E+50	1,0069429E+59	5,6115003E+57	5,7890921E+48	4,5624969E+55	4,1140009E+54	1,2322798E+47	2,4522988E+49	9,7118387E+53	1,6808306E+50	6,2232492E+58	6,6565933E+54	2,0672849E+52
2,2926541E+53	7,6540905E+60	1,1167167E+60	1,0770594E+55	3,8461795E+58	1,2384579E+61	5,0598866E+56	7,3822751E+55	1,8068857E+60	9,0795981E+57	3,2423248E+61	2,6362106E+59	8,4885164E+61

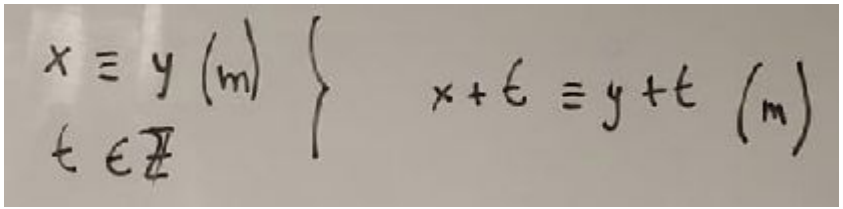
Tornem al : “Poguer fer i Desfer”

- S’abandona la idea de la propietat àurea.
- S’introdueix la propietat modular.

Supòsit : L = {a,b,c,d,e}, Z / 5

On : a = 13, b = 14, c = 15, d = 16, e = 17.

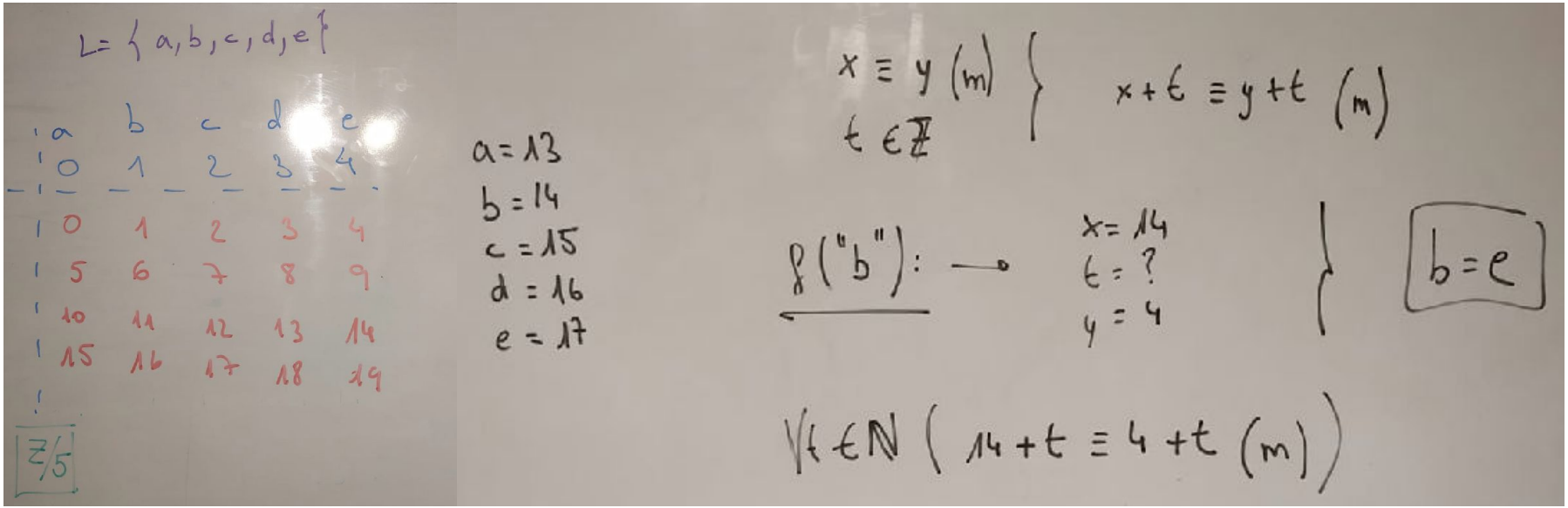
En aritmètica modular :



a	b	c	d	e
0	1	2	3	4
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Cíclic + modular

En el supòsit anterior codifiquem ‘b’ :



Descodificat Cíclic + modular

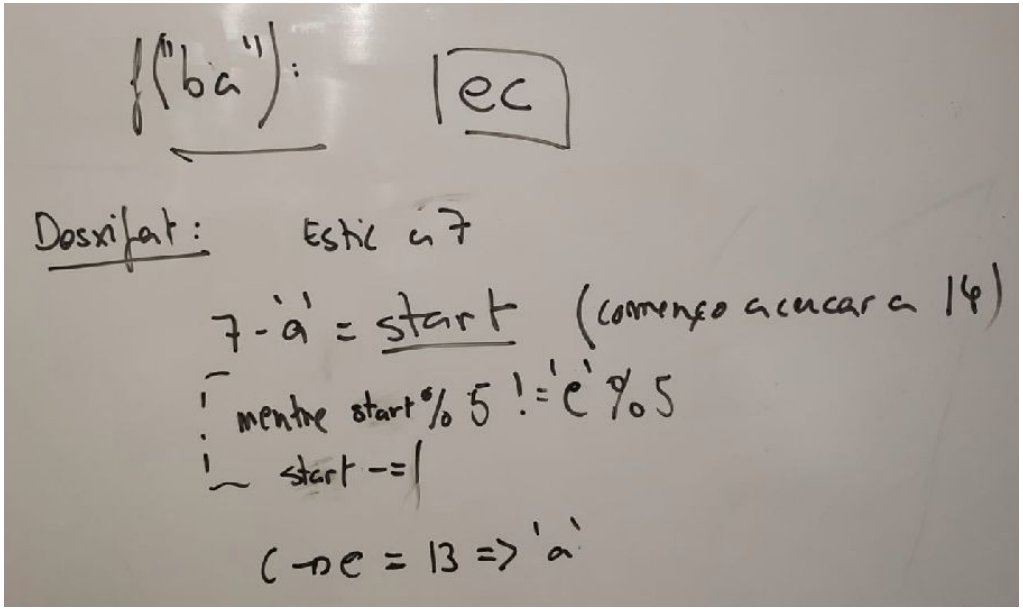
Codifiquem ‘ba’ : f(“ba”) = “ec”

- Partim de start = 0

Descodifiquem d(“ec”) = “ba”

- Si i només si coneixem start.

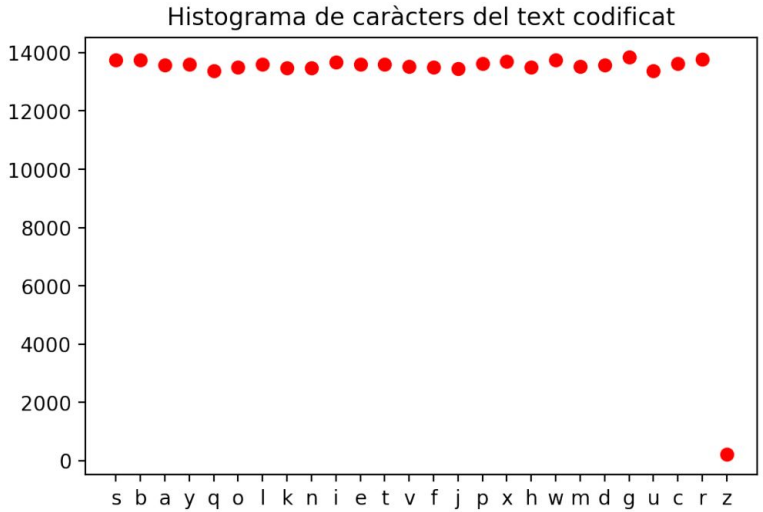
La distpància entre l’inici i el proper congruent és el caràcter descodificat.



Codifiquem Frankenstain

Resultat de codificar el llibre de frankenstain :

- a = 0 … y = 24 on L = {a,b,c...y} |L| = 25.



LA PELI FUNCIONA!

Utilitzem la última posició de la taula com a clau per a la descodificació.

Problemes...

- La força de l'algoritme recau en el desconeixement de la clau i **el desconeixement de l'algoritme!**
- Només es poden codificar 25 caràcters

En definitiva… Un nyap...

Millores

- Ampliació del rang de caràcters.
- Desacoblament de la taula àurea i generació de taula vers una clau privada.
- Millorar la fortalesa (ha de raure en el desconeixament de la clau)
- La clau no pot tenir a veure amb la posició on s'ha deixat l'últim caràcter de la taula.
- L'algoritme pot ser públic.
- S'incorpora marca de fi, text | codificat | = | text pla | + 1.
- Descodificat en el mateix ordre que el codificat i no invers.
- Rail Fence amb n rails = | clau | **(m'hi vai trencar el coco...)**

Ampliació del rang de caràcters.

Per obrir camp s'utilitzen els caràcters del 32 al 126 de la taula ascii.

- **OBLIDEM LA TAULA ÀUREA!**
- Es fa una taula quadrada per simplicitat.
- Sempre tindrà tantes columnes com caràcters es puguin codificar.
- Té 126 -32 files i 126 - 32 columnes.

Desacoblament de la clau i nova taula

- Es demana un text com a clau privada.
- S’utilitza la clau de llavor per la generació de nombres aleatòris.
- El primer nombre de la taula és generat aleatòriament.
- S’emplena la taula amb els següents (126-32)*(126-32) nombres.

Exemples

Codifiquem :
“Project Gutenberg's Frankenstein, by Mary Wollstonecraft (Godwin) 17 Shelle.”

```
ENCODED TEXT By SUBSTITUTION :
-----------------------------------------------
[L_oz!%:ZBXms#&,?Gn#C*=?NZ`o$9?IX%EHb#pr&@`Xhu#7L\kqu)+2Gg0w(-EO^(H<
   EKXeky

ENCODED TEXT By SUBSTITUTION + TRANSFORMATION :
-----------------------------------------------
[sNbL(kL#Z#\-y_&`pkEo,orqOz?$&u^!G9@)(%n?`+H:#IX2<ZCXhGEB*%ugKX=E#0Xm?
   H7we

DECODED TEXT By SUBSTITUTION + TRANSFORMATION :
-----------------------------------------------
[L_oz!%:ZBXms#&,?Gn#C*=?NZ`o$9?IX%EHb#pr&@`Xhu#7L\kqu)+2Gg0w(-EO^(H<
   EKXeky

DECODED TEXT By SUBSTITUTION + TRANSFORMATION :
-----------------------------------------------
Project Gutenberg's Frankenstein, by Mary Wollstonecraft (Godwin)
Shelle.
```

Exemples

```
[OX^~02H\bg4Tlrx*4CKw8:INn$~3Bbfv&;ETjpuBb%n/DM]s{%:Zr|2;[dt(;K^+KY
s(@FLa"/16P|=R[]r3M]s4HQaw%*JMS`jp(.Nhx/BbMcw~7FLx9QZjx9Scy~MP]cw,2
79O^s4=?Ddx)IR\dmz5U^n}.DW]b/OQ`e&>GWe&@Pf'4D[af3Skm"BDdh{"$9Obh)-/@BER
Xho02RVis"(H`isw!AP`ou69Od%:CIint,6Cclv%9?LSsw(>KPpy{39Yjp$5;Pcez!&
T9?ATtlv$1;=Kl-28:MSg|=@MSg{"'GKT^kp2Rr\|1AQ`!5>@MZz/5;[u&<\^fhr"BL
[djl$*9e&>GM`f'?Eey#%2?_an{<?Eenp"3M)IKZ_ 5>@Uuy*9M]jp%ESY&FN^hw @BVv
`bp1FVv+AHOUh)3;JZhr"<\^mr38>@U^~

OqajTWBXl",;Ab#C*:MU_v|=KQqx)<\eg~)8@`gw+K[jp1?O]cr(HMWk!4J^sy~?Yi
NFOi*/9>^x)?_cs#*0DX8Xx[q'GL\|,<Qq 0FYh5UZ`bu6>H[h7Ww\l-<La")/1Dr3S
=U_ly:K^nr ",:f'p1IS`m.?Rby @Zj!4T^m|-17FJP~?_Ii",9Fftz(=]r{"BVk{+
ENTVi~3Scj+EUk~?ETZhrx-MPj+9Ss)/1DXxz*/O`su06I],Llfv-Majly'GVf{<AKQ
rl|3_ .Hhy')CJPjjz3_ .Hhl|+<>MWgvCcq,L`j~4:My:KQdn#,L\k,AJPp%)+29IV[
|k{=]}l|>^~h)8>U[n/3CYfk,@Vi!+BHh|-MVfy~7:GMmo0>H\cs'<Ragi

ENCODED TEXT By SUBSTITUTION + TRANSFORMATION :
------------------------------------------------
[TNTs(LrJBZwD^eDhBsotsjTKM2@^$E?_j vhUXUejk
?[0H"l' 7Fkcr16lrCl,d%=Ui-sOlnj{;a3Mbj,dn&[{E"u,wp9lSRMf*eE
p@+r^l_gp!N_qF[)yp@Ff{jxDIylJ|Ln)][!7'Xr$p%K"MSMx2x}>a"R(66($?-grZh9ye5
    %BA"~"v~14Fc'Yh/:1ZJt++~X]'|P+`#+}n+:<^x~u:*/]`c9
).Gf$
H9C>5A2{\zre#n>EVH<
,|)?JOsGh71KIjPz
EMx,G3]<j,2l/BGR~*3BZ+1sjwS7IDW39X`OcK;T8"|/"&%p@SvO\
;=8O^i#L5WD^S!~(EUPzLV_j>~L9|3HMa04BbrK64p~c9RWeSOhidlPPt:'15B>2"UY
U^OAK@]s**\Uwrn`4?=Nkj*lf
zM4\I>Chmg2Cb%|YPH(7yO\]&kbos%vpclMGA;LG?3u&`hmqbQ`cy/0|Z\3rmT_]T~+/f
    {.3W:kV^Y|oiHKfn2
|Q.F~^db@mh0w:%yevSKQ[
M_MyFb)ra#qgr~9D,`lS
.^IrV?9Ov<H_gM,[~f-0\wv/;s=aNLMsm/P")2!C9{z$gT`u[`a)*Np33jCxw(?>X<b~
"?mi{iES`~Ah vyA
hkM>b8&D[(RwhxP4zOfB~
RAI?3!1|^!&dfnI9^1;87*)+HY^8Qu<=,R|""~TssMKy.C:J|),VHg:;Md@[%x9]=5Q'D/
    VPiL9&;=k5<j'{KMhFJ>W:<KMixXq6LU:b~,B3Z)uaQ'HcKPk8@f\4IE]tF]*/Qc?U`4
    d@i`nSY
=@p>\l?<Z]wVZ@BM\[W )x >a_fy19VSh/0j
)hqQp{>Vyc

DECODED TEXT By SUBSTITUTION + TRANSFORMATION :
------------------------------------------------
[OX^~02H\bg4Tlrx*4CKw8:INn$~3Bbfv&;ETjpuBb%n/DM]s{%:Zr|2;[dt(;K^+KY
```

```
s(@FLa"/16P|=R[]r3M]s4HQaw%*JMS`jp(.Nhx/BbMcw~7FLx9QZjx9Scy~MP]cw,2
79O^s4=?Ddx)IR\dmz5U^n}.DW]b/OQ`e&>GWe&@Pf'4D[af3Skm"BDdh{"$9Obh)-/@BER
Xho02RVis"(H`isw!AP`ou69Od%:CIint,6Cclv%9?LSsw(>KPpy{39Yjp$5;Pcez!&
T9?ATtlv$1;=Kl-28:MSg|=@MSg{"'GKT^kp2Rr\|1AQ`!5>@MZz/5;[u&<\^fhr"BL
[djl$*9e&>GM`f'?Eey#%2?_an{<?Eenp"3M)IKZ_ 5>@Uuy*9M]jp%ESY&FN^hw @BVv
`bp1FVv+AHOUh)3;JZhr"<\^mr38>@U^~

Oqaj7WBXl",;Ab#C*:MU_v|=KQqx)<\eg~)8@`gw+K[jp1?O]cr(HMWk!4J^sy~?Yi
NFOi*/9>^x)?_cs#*0DX8Xx[q'GL\|,<Qq 0FYh5UZ`bu6>H[h7Ww\l-<La")/1Dr3S
=U_ly:K^nr ",:f'p1IS`m.?Rby @Zj!4T^m|-17FJP~?_Ii",9Fftz(=]r{"BVk{+
ENTVi~3Scj+EUk~?ETZhrx-MPj+9Ss)/1DXxz*/O`su06I],Llfv-Majly'GVf{<AKQ
rl|3_ .Hhy')CJPjjz3_ .Hhl|+<>MWgvCcq,L`j~4:My:KQdn#,L\k,AJPp%)+29IV[
|k{=]}l|>^~h)8>U[n/3CYfk,@Vi!+BHh|-MVfy~7:GMmo0>H\cs'<Ragi

DECODED TEXT By SUBSTITUTION + TRANSFORMATION :
------------------------------------------------
She paused, weeping, and then continued, "I thought with horror, my
sweet lady, that you should believe your Justine, whom your blessed
aunt had so highly honoured, and whom you loved, was a creature
capable
of a crime which none but the devil himself could have perpetrated.
Dear William! dearest blessed child!  I soon shall see you again in
heaven, where we shall all be happy; and that consoles me, going as I
am to suffer ignominy and death."

"Oh, Justine!  Forgive me for having for one moment distrusted you.
Why did you confess?  But do not mourn, dear girl.  Do not fear.  I
will proclaim, I will prove your innocence.  I will melt the stony
hearts of your enemies by my tears and prayers.  You shall not die!
You, my playfellow, my companion, my sister, perish on the scaffold!
No!  No!  I never could survive so horrible a misfortune.
```

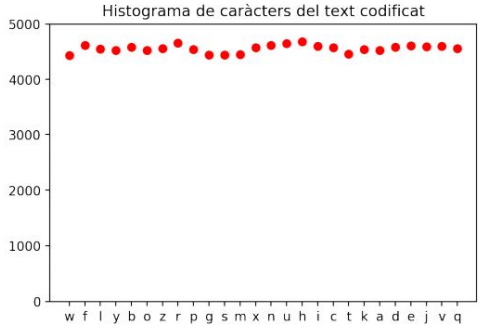
En cas de voler executar la codificació i descodificació es poden realitzar les següents instruccions.

```
python3 encode.py txt/frankenstein.txt text-codificat.txt
    lamevaparauladepas
```

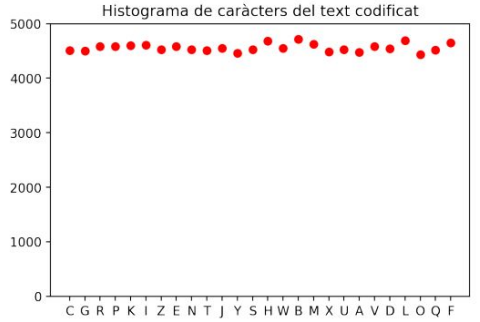
```
python3 decode.py text-codificat.txt text-descodificat.txt
    lamevaparauladepas
```

Histograma del text codificat

```
histograma = dict()
book = ""
with open("txt/codificat-modular.txt", 'r', encoding='utf-8') as
    fileobj:
    for line in fileobj:
        for ch in line:
            if ch >= 'a' and ch <= 'z':
                if ch in histograma:
                    histograma[ch] += 1
                else:
                    histograma[ch] = 1

plt.plot(histograma.keys(), histograma.values(), 'ro')
plt.title("Histograma de caràcters del text codificat")
axes = plt.gca()
axes.set_ylim([0,5000])
plt.xlabel("")
plt.ylabel("")
plt.show()
```




```

histograma = dict()
book = ""
with open("txt/codificat-modular.txt", 'r', encoding='utf-8') as
    fileobj:
    for line in fileobj:
        for ch in line:
            if ch >= 'A' and ch <= 'Z':
                if ch in histograma:
                    histograma[ch] += 1
                else:
                    histograma[ch] = 1

plt.plot(histograma.keys(), histograma.values(), 'ro')
plt.title("Histograma de caràcters del text codificat")
axes = plt.gca()
axes.set_ylim([0,5000])
plt.xlabel("")
plt.ylabel("")
plt.show()
```



Càlcul de l'Índex de Coincidència

```
def ic_calculation(file, first_char, last_char):
# Donat un fitxer file, el valor del primer caràcter, el valor de l'ultim
caràcter a la taula ascii
# Retorna l'IC que s'ha calculat d'aquell fitxer.
    histograma = dict()
    text_length = 0
    with open(file, 'r', encoding='utf-8') as fileobj:
        for line in fileobj:
            for ch in line:
                text_length += 1
                if ch >= chr(first_char) and ch <= chr(last_char):
                    if ch in histograma:
                        histograma[ch] += 1
                    else:
                        histograma[ch] = 1

    # Nombre de caràcters de llenguatge
    L=len(histograma.keys())
    print("L=" + str(L))
    suma = 0
    for value in histograma.values():
        suma += value * (value - 1)
    ic = L * suma / (text_length * (text_length - 1))
    return ic
```

$$IC = \frac{\sum_{i=1}^{L} f(x_i)(f(x_i)-1)}{N(N-1)}$$

En el nostre cas el calculem de la següent manera :

```
1  import Utils
2  ic = Utils.ic_calculation("txt/codificat-modular.txt", 32, 126)
3  print("IC = " + str(ic))
```

```
1  L=95
2  IC = 0.9648614759341749
```

Entropia per Llenguatge Anglés - Absoluta

Fitxer /usr/share/dict/words ⇒ L = 53

Per tant la seva ràtio absoluta és :

```
print("H(X)=" + str(Utils.getAbsoluteRatio(L)))
```

```
H(X)=5.7279204545632
```

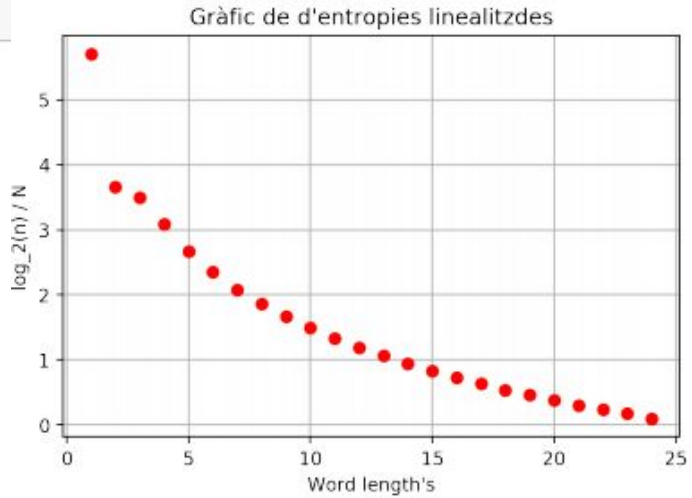
```
histograma_mots, max = Utils.getWordsLengthDict("txt/words.txt")
print(histograma_mots)
```

```
{1: 52, 2: 160, 3: 1420, 5: 10230, 4: 5272, 8: 29989, 7: 23869, 9:
32403, 6: 17706, 11: 26013, 10: 30878, 12: 20462, 14: 9765, 16: 3377,
15: 5925, 20: 198, 19: 428, 17: 1813, 13: 14939, 18: 842, 21: 82, 22:
41, 23: 17, 24: 5}
```

Entropia per el llenguatge Anglès - Verdadera

Vegem les aparicions en forma de gràfic un cop calculats els seus valors de ratio verdadera per tots els nombres d'aparicions, $log_2(n)/N$.

```
1 rs = Utils.evalDictionary(histograma_mots, max)
2 Utils.plotGraphic(histograma_mots.keys(), rs, "Gràfic de d'entropies
    linealitzdes", "Word length's", "log_2(n) / N")
```



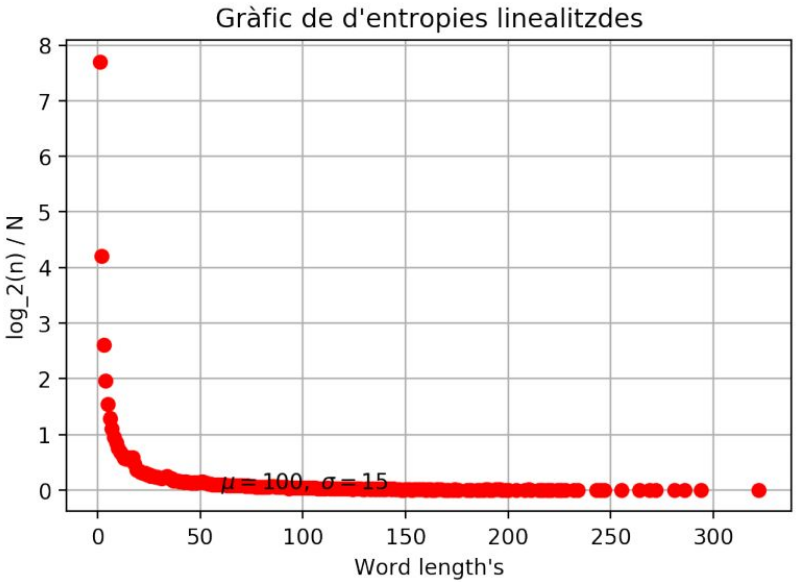
Presenta linealitat com hem vist a la pràctica 4...

Entropia sobre un text codificat

Mirem si podriem deduïr que està escrit en anglés…

```
1 histograma_mots_fran, max_fran = Utils.getWordsLengthDictSpaceSeparato
    (”txt/codificat-modular.txt”)
2 rsf = Utils.evalDictionary(histograma_mots_fran, max_fran)
3 print(rsf)
4 Utils.plotGraphic(histograma_mots_fran.keys(), rsf, ”Gràfic de d'
    entropies linealitzdes”, ”Word length's”, ”log_2(n) / N”)
```

No et sabria dir… xD



Altres propietats

Redundància

Intuitivament ja es veu que no és redundant, presenta l'aleatorietat que et pot aportar el nombre de possibles caràcters següents que poden aparèixer.

Propagació de la confusió

Presenta confusió, ja que un mateix caràcter no serà codificat sempre igual, també es propaga per què la combinació dels possibles caràcters que poden venir és la que determina quin serà el següent.

Complexitat

Complexitat de la codificació:

- O(n) on n es correspon a la llargada del text.

Complexitat de la descodificació:

- O(n) on n es correspon a la llargada del text.

Tot molt bonic, però…….L'algoritme és un cagarro!

- Pot ser desxifrat en $O(n^4)$ per força bruta… Generant totes les possibles taules + començant per totes les posicions de la taula $O(n^2) * O(n^2)$.

Millora en la fortalesa i les gràcies!

No està implementada!

- Un cop generada la taula es podria desordenar per files i per columnes en funció de la clau (pendent).
- Veure com augmentar la complexitat del desxifrat en funció del nombre de bits de la clau (pendent).

Gràcies per la vostre atenció!

Podeu trobar tot el codi aquí : https://github.com/raikkon88/seguretat-p3