LA MACHINE ENIGMA

SOMMAIRE

I)Introduction

1.Cryptographie

Source: http://laurent.flaum.free.fr/pgpintrofr.htm

I)Introduction

1.Cryptographie

CODAGE

Message	B	O	N	J	O	U	R
	2	15	14	10	15	21	18
Clé	D	E	C	O	D	E	C
	4	5	3	15	4	5	3
Addition	6	20	17	25	19	26	21
Message codé	F	T	Q	Y	S	Z	U

DECODAGE

Message codé	F	T	Q	Y	S	Z	U
	6	20	17	25	19	26	21
Clé	D	E	C	O	D	E	C
	4	5	3	15	4	5	3
Soustraction	2	15	14	10	15	21	18
Message décodé	B	O	N	J	O	U	R

Source : https://sciencetonnante.wordpress.com/2010/12/03/protegez-vos-petits-secrets-grace-aux-nombres-premiers/

Source : http://www.nymphomath.ch/crypto/cesar/index.html

I)Introduction

2.Enigma

Source :
http://www.frenchweb.fr/petite-histoire-de-la-cryptographie-de-la-machine-enigma-a-lordinateur/264879

Source :
https://www.bletchleypark.org.uk/our-story

Source :
https://www.archives.gov/files/publications/prologue/1997/fall/turing.pdf

I)Modélisation

1.Fonctions utiles

Transformation lettre/chiffre


```
def lettrechiffre(H,D): #H liste, D dictionnaire
    B=[]
    n=len(H)
    for i in range(n):
        B.append(0)
        B[i]=D.get(H[i])
    return B
```

Source : http://python.lycee.free.fr/

Comparaison mot

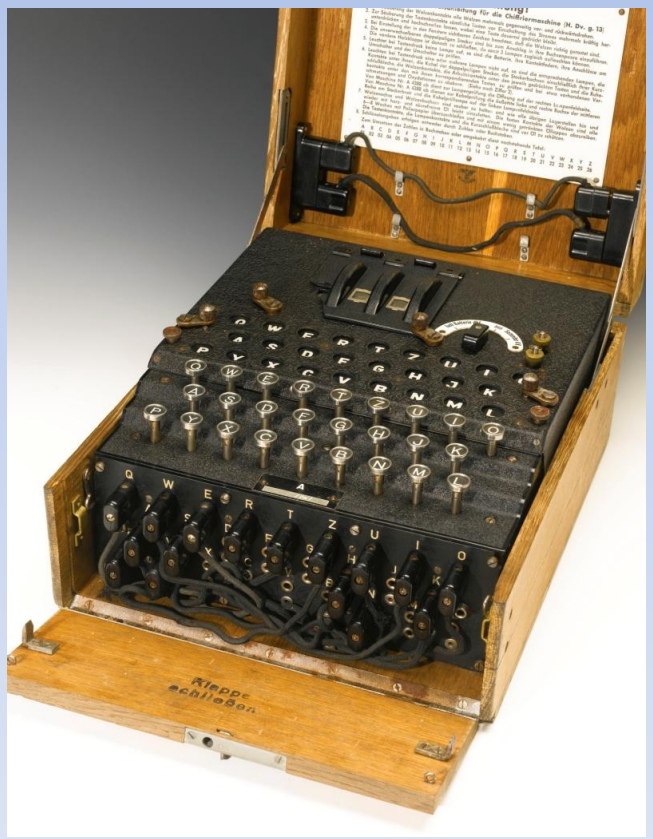

```
def mot(L,µ): #L mot (dé)crypté par
enigma,   µ première lettre du mot
   i=0
   if len(L)==len(µ):
      while i<len(L):
            if L[i]==µ[i]:
               i=i+1
            else:
               return('false')
      return('true')
    return('false')
```

Réciproque dictionnaire


```
def inverse(M): #M=dictionnaire
    T=[]
    U=[]
    MI=[]
    T = M.keys()
    U = M.values()
    MI=dict(zip(U,T))    #MI= M inverse
    return MI
```

II) Modélisation

2.Cryptage

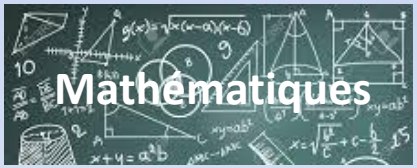


Source : https://www.actualitte.com/article/patrimoine-education/enigma-la-machine-a-ecrire-des-messages-codes-durant-la-seconde-guerre-mondiale/59506

Source : https://commons.wikimedia.org/wiki/File:Enigma-logo.svg

Clavier

Source : https://www.geocaching.com/



I=[0,25]

A=['u', 't', 'w', 'v', 'q', 'p', 's', 'r', 'y', 'x', 'z', 'e', 'd', 'g', 'f', 'a', 'c', 'b', 'm', 'l', 'o', 'n', 'i', 'h', 'k', 'j']

lettrechiffre(A,D)

Cablage

Source : http://cryptomuseum.com/crypto/enigma/i/index.htm

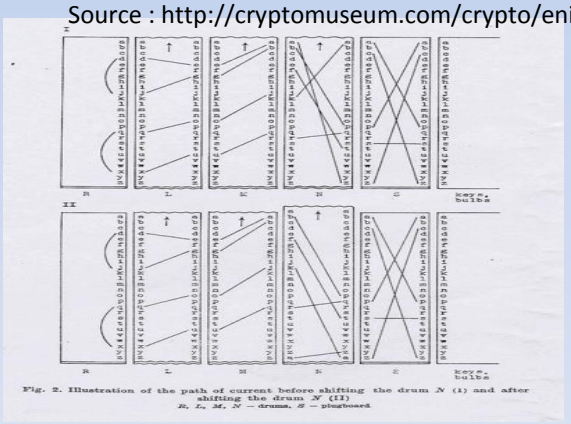
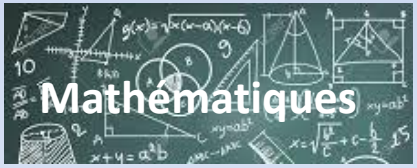


Fig. 2. Illustration of the path of current before shifting the drum N (I) and after shifting the drum N (II)
R, L, M, N — drums, S — plugboard



I=[0,25]
R : I -> I
R(8)=25

```
#H=message en entrée
#C=cablage
def cablage(H,C,i):
    H1=H
    H1[i]=C.get(H[i])
    return H1
```

C={1:2, 2:1, 3:0, 0:3, 4:5, 7:6, 5:4, 6:7, 9:10, 11:12, 13:14, 15:16, 17:18, 19:20, 21:22, 23:24, 25:8, 10:9, 12:11, 14:13, 16:15, 18:17, 20:19, 22:21, 24:23, 8:25}
H=[1,5,18,7,22]

Ex: cablage(H,C,18)->17

Cablage

- Complexité : get=O(1) => cablage=O(1)

- $n \in [0,26]$ n pair

- Choix de n lettres = $\binom{26}{n}$
- Choix des n/2 couples = A(n,n/2)
- (E,G)⇔(G,E) => $/2^{n/2}$

- Combinaisons cablage = $\binom{26}{n} * A(n,n/2)/2^{n/2}$

Rotors

Source:
http://www.cryptomuseum.com/crypto/enigma/working.htm

```
def rotor1(H,G,i):
    H1=H
    H1[i]=G.get(H[i])
    return H1
```

H=[1,5,18,7,22] i=0
R(0): G={0:20, 1:9, 2:23, 3:25, 4:22, 5:21, 6:14, 7:19, 8:24, 9:17, 10:16, 11:12, 12:10, 13:18, 14:13, 15:15, 16:8, 17:11, 18:2, 19:3, 20:4, 21:1, 22:5, 23:6, 24:7, 25:0}

Ex: rotor1(H,G,0) => 9

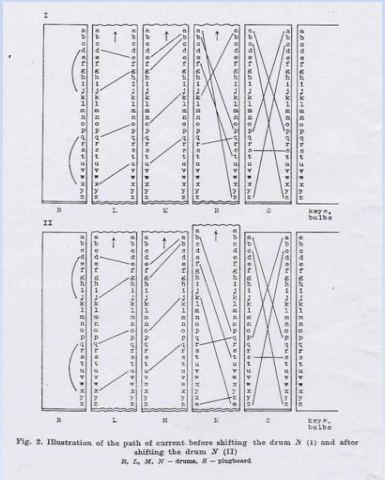
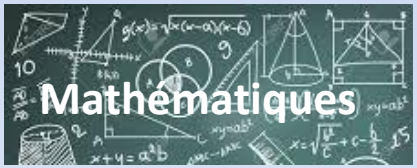


Fig. 2. Illustration of the path of current before shifting the drum N (1) and after shifting the drum N (II)
R, L, M, N – drums, S – plugboard



I=[0,25]
R(i) : I -> I
Ex: R(0)(2)=23

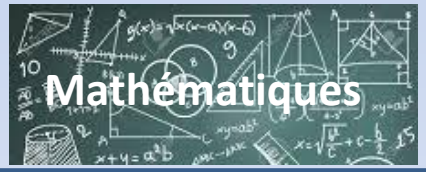
Rotors

- Complexité : rotor1 : get=O(1) => rotor1=O(1)
- Choix rotors : 5!/3!
- Ordre rotors : 3!

Rotation rotor

```
def tournerotor(V,V1,i) :
    n=len(V)
    for j in range((i)%len(V)):
        a=V1[0]
        for k in range (n-1):
            V1[k]=V1[k+1]
        V1[n-1]=a
    M1=dict(zip(V,V1))
    return M1
```

Source :https://fr.wikipedia.org/wiki/Enigma_(machine)



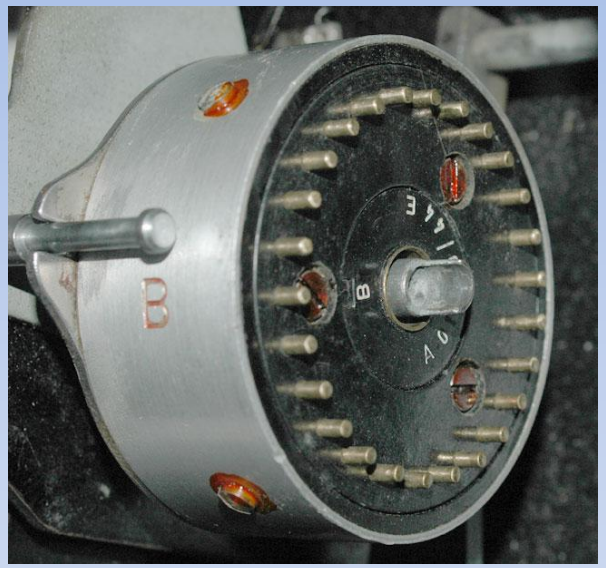
J=[R(0)..R(25)]
T : J->J
i€I T(R(i))=T(R((i+1)%26))

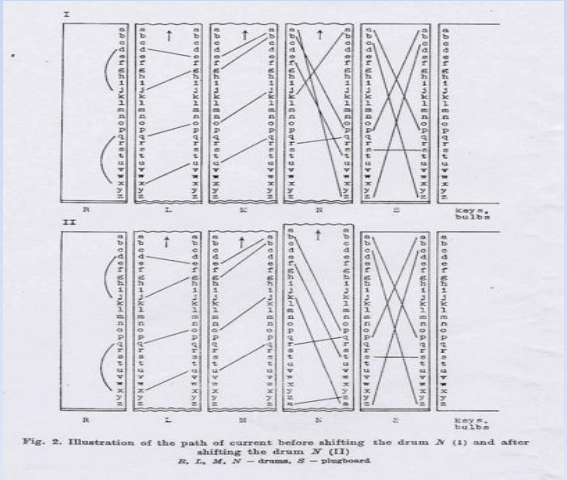
T=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
T1=[20,9,23,25,22,21,14,19,24,17,16,12,10,18,13,15,8,11,2,3,4,1,5,6,7,0]
tournerotor(T,T1,3)=> {0:25, 1:22, 2:21, 3:14, 4:19, 5:24, 6:17, 7:16, 8:12, 9:10, 10:18, 11:13, 12:15, 13:8, 14:11, 15:2, 16:3, 17:4, 18:1, 19:5, 20:6, 21:7, 22:0, 23:20, 24:9, 25:23}

Rotation rotor

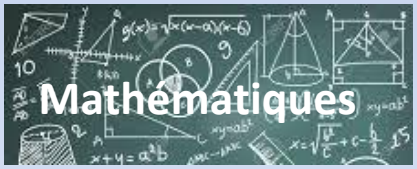
- Complexité : for=O(n) dict=O(n), zip=O(1) =>tournerotor=$O(n^2)$
- Combinaisons initiales : 26^3

Miroir






```
def miroir(H,S,i):
    H1=H
    H1[i]=S.get(H[i])
    return H1
```



I=[0,25]
M: I -> I
M(i)≠i

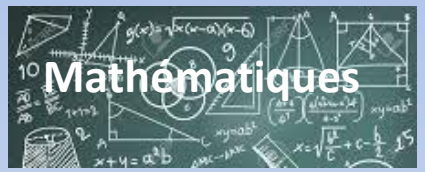
Source : http://www.minelinks.com/war/enigma.html

Miroir

- Complexité : get=O(1) => miroir=O(1)

Conséquence

- Une lettre ne peut être codée par elle même



E : I -> I
R: I -> I
a∈I p: I->I

E(a)=a
$(p \circ R \circ p^{-1})(a)=a$
$(R \circ p^{-1})(a)= p^{-1}(a)$

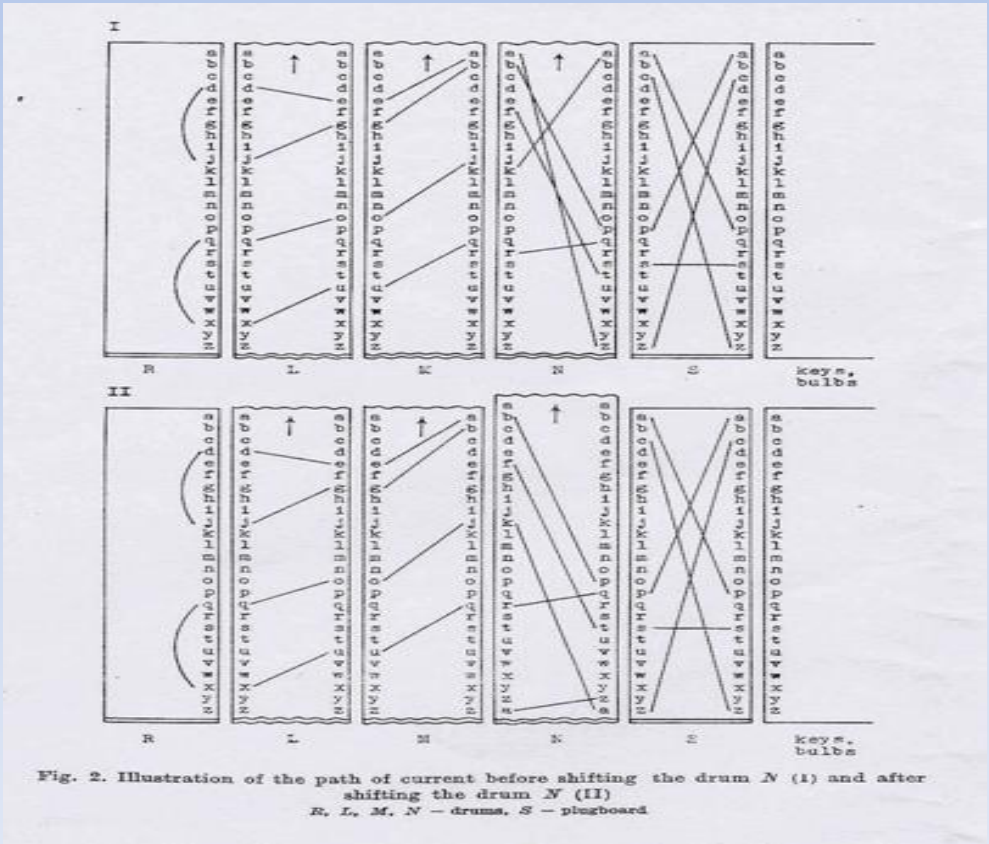
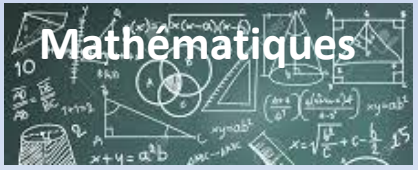


Fig. 2. Illustration of the path of current before shifting the drum N (I) and after shifting the drum N (II)
R, L, M, N — drums, S — plugboard

Tableau d'affichage

P=inverse(D)
lettrechiffre(A',P)



I=[0,25]

Source :
http://www.wikiwand.com/fr/Enigma_(machine)

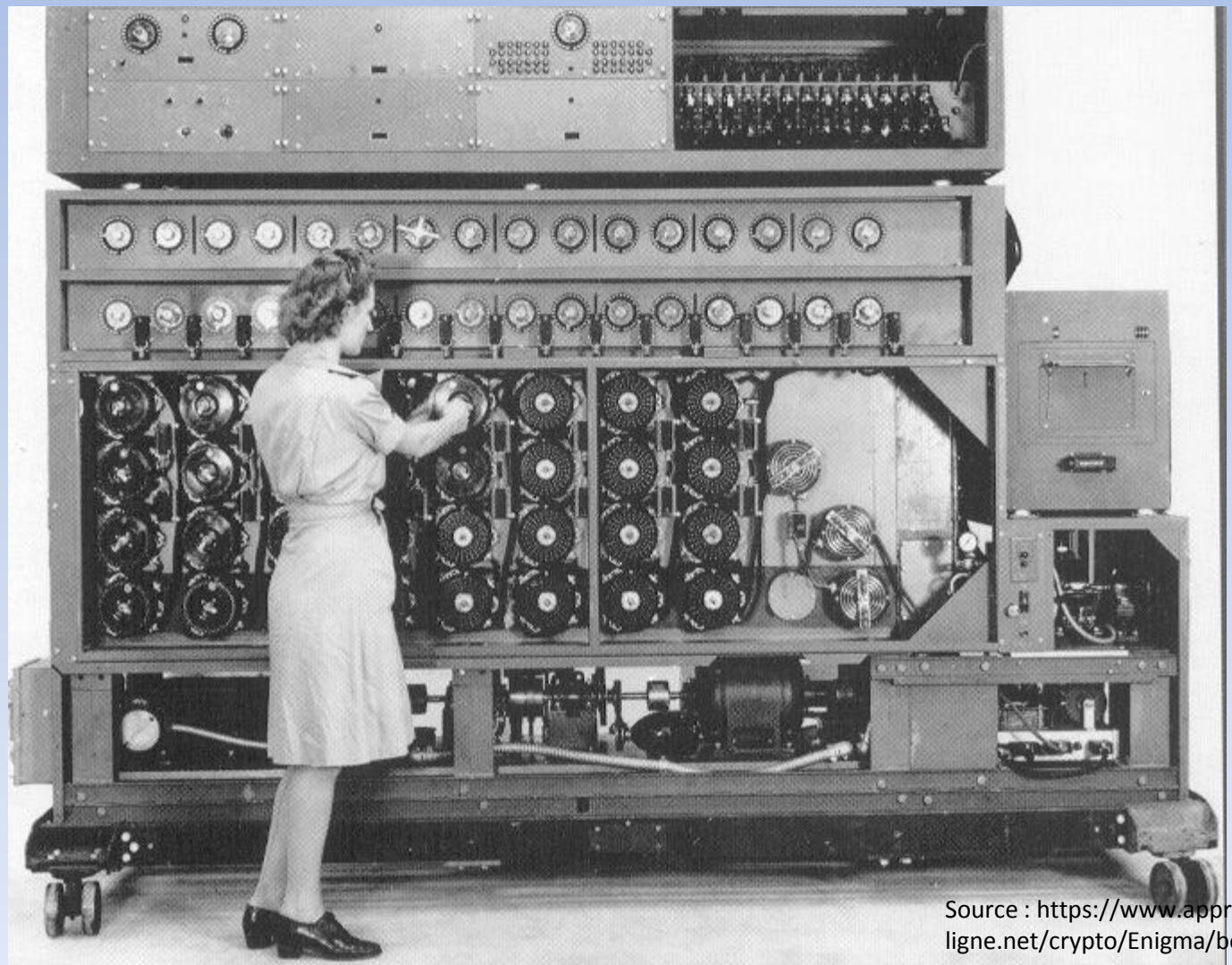
Programme principal


```
def enigma(A,T,Z,Z1,Z2,C,S,D):
    n=len(A)
    A=lettrechiffre(A,D)
    G1=dict(zip(T,Z))
    G2=dict(zip(T,Z1))
    G3=dict(zip(T,Z2))
    for i in range (n):
        A=cablage(A,C,i)
        A=rotor1(A,G1,i)
        A=rotor1(A,G2,i)
        A=rotor1(A,G3,i)
        A=miroir(A,S,i)
        K=inverse(G1)
        K1=inverse(G2)
        K2=inverse(G3)
        L=inverse(C)
        A=rotor1(A,K2,i)
        A=rotor1(A,K1,i)
        A=rotor1(A,K,i)
        A=cablage(A,L,i)
        G1=tournerotor(T,Z,i)
        G2=tournerotor(T,Z1,i//26)
        G3=tournerotor(T,Z2,i//(26*26))
    P=inverse(D)
    A=lettrechiffre(A,P)
    return (A)
```

- Complexité Enigma : Fonctions utiles=O(n)
- Fonctions précédentes=$O(n^2)$, for=O(n) => <u>Enigma=$O(n^3)$</u>
- Combinaisons initiales : 10^{16}

II)Modélisation

3.Décryptage



Source : https://www.apprendre-en-ligne.net/crypto/Enigma/bombe.html

Force brute


```
def decodage(S,C,D,T,Z1,Z2,A):
    Z=[20,9,23,25,22,21,14,19,24,17,16,12,10,18,13,15,8,11
    ,2,3,4,1,5,6,7,0]
       m=len(Z)**3
       for i in range (m):
            print(Z)
            print(Z1)
            print(Z2)
            Z=tournerotor([20,9,23,25,22,21,14,19,24,17,16,12,
    10,18,13,15,8,11,2,3,4,1,5,6,7,0],i)
            Z1=tournerotor(Z1,i//26)
            Z2=tournerotor(Z2,i//676)
             L=enigma(A,T,Z,Z1,Z2,C,S,D)
             print(L)
```

Complexité : for=O(n), fonctionsprécédentes=O(n^3)

=> <u>decodage=O(n^4)</u>


```
def decodage1(S,C,D,T,Z,Z1,Z2,A):
        Z=[20,9,23,25,22,21,14,19,24,17,16,12,10,18,13,15,8,11,2,3,4,1,5,6,7,0]
        m=len(Z)**3
        for i in range (m):
            Z=tourneZ([20,9,23,25,22,21,14,19,24,17,16,12,10,18,13,15,8,11,2,3,4,1,5,6,7,0],i+1)
            Z1=tourneZ(Z1,(i+1)//26)
            Z2=tourneZ(Z2,(i+1)//676)
            L=enigma(A,T,Z,Z1,Z2,C,S,D)
            if mot(L,μ)==‘true’
                return(L,Z,Z1,Z2)
```

- Complexité : mot=$O(n^3)$ => <u>decodage1=$O(n^3)$</u>

III)Optimisation

Source : http://yveslaurent.blogvie.com/2011/07/05/loptimisation-dune-campagne-multicanal/

III)Optimisation

1.Choix

Fonctions récurrentes ou explicites ?

$$(\forall n \in N)\ u_{n+1} = f(u_n)$$

$$(\forall n \in N)\ u_n = g(u_0)$$

Fonction récurrentes	Fonction explicte
Compatible avec le programme python Enigma	Compatible avec le programme python Enigma
O(n)	$O(n^2)$
Codage de la i-ème lettre impossible*	Codage de la i-ème lettre possible

Dictionnaires ou Listes ?

Dictionnaires	Listes
O(1)	O(n)

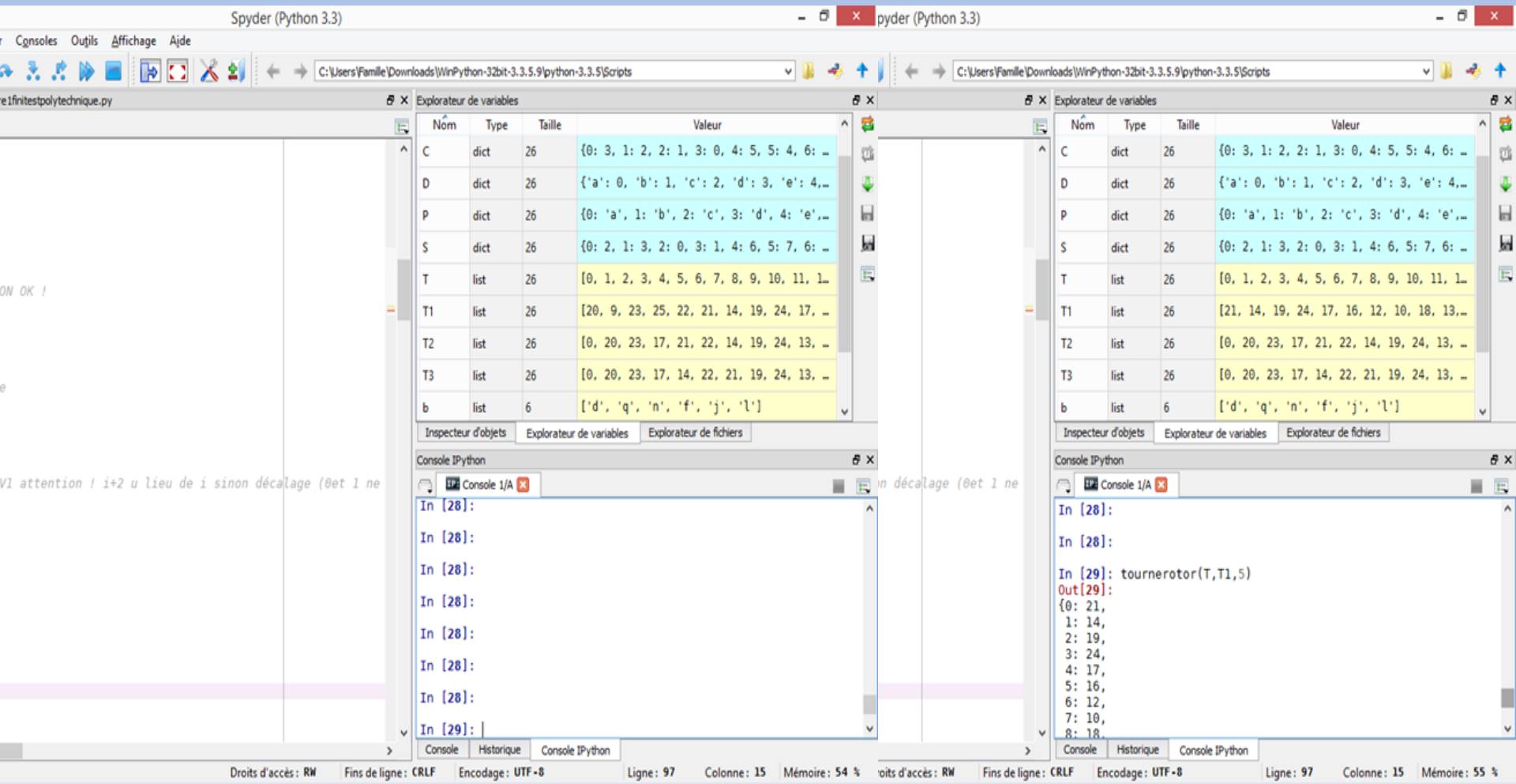
III)Optimisation

2.Contraintes

- Force Brute => méthode non optimale

III)Optimisation

2.Contraintes



III)Optimisation

3. Cycles de Marian Rejewski

Combinaisons originales	Combinaisons après travail de Marian Rejewski
10^{16}	7020