Banque « Agro- Veto »
AT - 0316

Algorithmique et Informatique
Durée : 45 mn

Si au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.
L'usage d'une calculatrice est interdit pour cette épreuve.

- On rappelle que l'opération de concaténation de deux chaines de caractères `ch1` et `ch2` s'écrit `ch1 + ch2`.

  Ainsi `"MESSAGE" + "SECRET"` *est la chaine de caractères* `"MESSAGESECRET"`.

- On rappelle que la syntaxe `a % b` (lue `a` modulo `b`) désigne le reste de la division euclidienne de l'entier `a` par l'entier (non nul) `b`.

  Ainsi l'entier `(12 + 18) % 26` *a pour valeur* `4`.

- Dans tout le sujet, on pourra utiliser une fonction `car_to_int` renvoyant un entier égal au rang (à partir de zéro) d'un caractère majuscule (écrit sans accent) dans l'alphabet passée en argument.

  La fonction `car_to_int` *est entièrement définie par l'ensemble de ses valeurs :* `car_to_int("A")= 0`, `car_to_int("B")= 1`, ..., `car_to_int("Y")= 24`, `car_to_int("Z")= 25`.

- On pourra utiliser sa réciproque notée `int_to_car`.

  La fonction `int_to_car` *est entièrement définie par l'ensemble de ses valeurs :* `int_to_car(0)= "A"`, `int_to_car(1)= "B"`, ..., `int_to_car(24)= "24"`, `int_to_car(25)= "Z"`.

1 Questions préliminaires

a. Quelle est le type de `int_to_car(21)` ? Quelle est sa valeur ?

b. Quelle est le type de `car_to_int("F")` ? Quelle est sa valeur ?

2 Chiffrement de Vigenère

Le chiffrement de Vigenère est une méthode cryptographique qui consiste à coder une chaine de caractères (appelée *texte clair*) en une autre (appelée *texte chiffré*) à partir d'une clé, connue seulement de l'expéditeur et du destinataire du message.
Pour cela, on choisit une chaine de caractères (de longueur m inférieure à celle du texte à chiffrer) qui fera office de clé, qu'on "applique" successivement à des blocs de longueur m du message à chiffrer.
Dans toute la suite, on supposera que la longueur du texte à chiffrer est un multiple de la longueur de la clé.
On illustre par l'exemple ci-dessous les étapes du chiffrement de Vigenère.
On choisit de chiffrer la chaine de caractère `"MESSAGECODEPOURBOB"` *- appelée* ***texte clair*** *- avec la clé* `"SECRET"`.

- *On construit une chaine de caractères de la longueur du texte claire en concaténant autant de fois que nécessaire la clé* `"SECRET"`.

  Puisque dans cet exemple le texte chiffré est de longueur 18 et que la clé est de longueur 6, on répète trois fois la clé, de manière à construire la chaine de caractères `"SECRETSECRETSECRET"`.

- *on convertit les chaines de caractères - le texte à chiffrer - ainsi que la clé dupliquée en deux listes d'entiers via la correspondance bijective entre caractères et entiers décrite au début du sujet.*

  On obtient alors la liste `[12, 4, 18, 18, 0, 6, 4, 2, 14, 3, 4, 15, 14, 20, 17, 1, 14, 1]` *et la liste* `[18, 4, 2, 17, 4, 19, 18, 4, 2, 17, 4, 19, 18, 4, 2, 17, 4, 19]`.

- *Les deux listes ayant la même longueur, on somme terme-à-terme* ***modulo 26*** *les entiers de ces listes. On obtient alors une troisième liste d'entiers (compris entre 0 et 25).*

- *On convertit cette liste en une chaine de caractères via la correspondance entre entiers et caractères. La chaine ainsi construite est appelée* ***texte chiffré*** *; c'est cette chaine qui constitue le message qui sera envoyé.*

Les opérations présentées dans l'exemple sont illustrées dans l'exemple ci-après.

texte clair	*"M*	*E*	*S*	*S*	*A*	*G*	*E*	*C*	*O*	*D*	*E*	*P*	*O*	*U*	*R*	*B*	*O*	*B"*
	[12,	*4,*	*18,*	*18,*	*0,*	*6,*	*4,*	*2,*	*14,*	*3,*	*4,*	*15,*	*14,*	*20,*	*17,*	*1,*	*14,*	*1]*
clé dupliquée	*"S*	*E*	*C*	*R*	*E*	*T*	*S*	*E*	*C*	*R*	*E*	*T*	*S*	*E*	*C*	*R*	*E*	*T"*
	[18,	*4,*	*2,*	*17,*	*4,*	*19*	*18,*	*4,*	*2,*	*17,*	*4,*	*19*	*18,*	*4,*	*2,*	*17,*	*4,*	*19]*
	[4,	*8,*	*20,*	*9,*	*4,*	*25,*	*22,*	*6,*	*16,*	*20,*	*8,*	*8,*	*6,*	*24,*	*19,*	*18,*	*18,*	*20]*
texte chiffré	*"E*	*I*	*U*	*J*	*E*	*Z*	*W*	*G*	*Q*	*U*	*I*	*I*	*G*	*Y*	*T*	*S*	*S*	*U"*

a. Écrire une fonction `duplique(ch, n)` qui construit une chaine de caractères comme la concaténation de `n` chaines de caractères identiques, la chaine `ch`.

   Ainsi, `duplique("SECRET", 3)` *est la chaine de caractères* `"SECRETSECRETSECRET"`.

b. Écrire une fonction `car_to_listofint` qui renvoie la liste des entiers correspondant (via la correspondance entre caractères et entiers présentée au début du sujet) aux caractères (majuscules sans accent) de la chaine de caractères passée en argument.

   Exemple : `car_to_listofint("POURBOB")` *est la liste* `[15, 14, 20, 17, 1, 14, 1]`.

c. Écrire une fonction `somme` qui renvoie une liste d'entiers construite comme la somme terme-à-terme modulo 26 de deux listes de même longueurs.

   Ainsi, `somme([12, 4, 18, 18], [18, 4, 2, 17])` *est la liste* `[4, 8, 20, 9]`.

d. Écrire une fonction `listofint_to_car`, réciproque de la fonction `car_to_listofint`, renvoyant la chaine de caractères correspondant aux élément d'une liste d'entiers (compris entre 0 et 25).

   Exemple : `listofint_to_car([4, 8, 20, 9, 4, 25])` *est la chaine de caractères* `"EIUJEZ"`.

e. Écrire une fonction `chiffrementVigenere` qui, à partir de deux chaines de caractères `text_clair` et `cle` passées en argument, renvoie le texte chiffré (une chaine de caractères) obtenu par le chiffrement de Vigenère.

3 Déchiffrement

Pour déchiffrer un texte chiffré - ce qui se fait toujours en connaissant la clé, sinon on parle de cryptanalyse - on reprend les étapes précédentes dans l'ordre inverse.

1. Écrire une fonction `difference` qui renvoie une liste d'entiers construite comme la différence terme-à-terme modulo 26 de deux listes de même longueurs. On "soustraira" la seconde liste à la première.

   Ainsi, `somme([4, 8, 20, 9], [18, 4, 2, 17])` *est la liste* `[12, 4, 18, 18]`.

2. En déduire une fonction `dechiffrementVigenere` qui, à partir de deux chaines de caractères `texte_chiffre` et `cle`, renvoie la chaine de caractères correspondant au texte clair.

4 Cryptanalyse par analyse fréquentielle

- On rappelle que la syntaxe `ch[k:]` désigne la chaine de caractères `ch` tronquée de ses `k` premiers caractères.

  Ainsi, si `ch` *désigne la chaine* `"SECRET"`, `ch[2:]` *est la chaine* `"CRET"`.

- On rappelle que si `ch` désigne une chaine de caractères, `ch[i::d]` désigne la chaine de caractères construite en extrayant les caractères de `ch` d'indice `i, i+d, i+2d`, etc.

  Ainsi, si `ch` *désigne la chaine de caractères* `"EPREUVEINFORMATIQUE"`, `ch[1::2]` *est la chaine de caractères* `"PEVIFRAIU"`.

Lorsqu'on cherche à lire un texte chiffré sans connaitre la clé, on parle de cryptanalyse.
La cryptanalyse du chiffrement de Vigenère a lieu en deux étapes : la détermination de la longueur de la clé puis de la clé elle-même, à l'aide d'outils statistiques (inhérent à la langue dans laquelle est écrit le texte).
Dans toute la suite, on désigne par `ch` le texte chiffré à décrypter.

a. On appelle **indice de coïncidence** de deux chaines de caractères la fréquence des indices des chaines où les caractères sont identiques.

Par exemple, l'indice de coïncidence des chaines "ABRACADABRA" et "SHAZAAAM" est $\frac{1}{8}$*. En effet, ces chaines ne partagent qu'un seul caractère commun situé au même indice (*`"A"` *à l'indice 5). Les trois derniers caractères (*`"BRA"`*) de la chaine* `"ABRACADABRA"` *sont ignorés.*

Parmi les quatre fonctions ci-dessous, indiquer (sans justification) l'unique fonction renvoyant l'indice de coïncidence de deux chaines de caractères.

```
def indice_coincidence1(ch1, ch2):
    m = len(ch1)
    compteur = 0
    for k in range(m+1):
        if ch1[k] in ch2:
            compteur += 1
    return compteur/m
```

```
def indice_coincidence2(ch1, ch2):
    m = min(len(ch1), len(ch2))
    compteur = 1
    for k in range(m):
        if ch1[k] == ch2[k]:
            compteur += 1
    return compteur
```

```
def indice_coincidence3(ch1, ch2):
    compteur = 0
    n1 = len(ch1)
    n2 = len(ch2)
    i = 0
    while i < n1 or i < n2:
        if ch1[i] == ch2[i]:
            compteur = compteur + 1
    return compteur
```

```
def indice_coincidence4(ch1, ch2):
    compteur = 0
    n1 = len(ch1)
    n2 = len(ch2)
    i = 0
    while i < n1 and i < n2:
        if ch1[i] == ch2[i]:
            compteur = compteur + 1
    return compteur/i
```

b. Pour deux chaines de caractères aléatoires, on peut prouver que l'indice de coïncidence est égal à $\frac{1}{26}$. Pour des chaines de caractères écrites dans une langue dite *naturelle*, l'indice de coïncidence est plus élevé (la distribution des lettres n'est pas uniforme). En français par exemple, il est supérieur à 0,07. Ainsi, si `ch` désigne une chaine de caractères écrite en français, l'indice de coïncidence de `ch` et `ch[n:]` est (statistiquement) supérieur à 0,07.

Pour déterminer la longueur de la clé, on utilise le **test de Friedman** : si `ch` désigne une chaine codée par le chiffrement de Vigenère, on calcule successivement les indices de coïncidence entre `ch` et les chaines `ch[k:]` où `k` varie dans l'ensemble des indices non nuls de la chaine `ch`.

La propriété sur l'indice de coïncidence de `ch` et `ch[d:]` nous assure que cet indice dépasse 0,07 lorsque `d` est égal à la longueur de la clé (ou un multiple), et reste proche de $\frac{1}{26}$ dans le cas contraire.

Écrire une fonction `test_Friedman` qui prend en argument une chaine de caractères `ch` codée selon le chiffrement de Vigenère et qui renvoie le plus petit entier `d` ≥ 1 tel que l'indice de coïncidence de `ch` et `ch[d:]` dépasse 0.07, i.e. la longueur (probable) de la clé de chiffrement.

c. Écrire une fonction `lettre_plus_freq` qui renvoie le caractère le plus fréquent dans une chaine de caractères entrée en argument (en cas d'égalité, on renverra le premier caractère rencontré, i.e. celui de plus petit indice).

On pourra utiliser la méthode `count` *sur le type* `str` *de la manière suivante :* `ch.count(car)` *est le nombre de fois où le caractère* `car` *apparait dans la chaine* `ch`*.*

Ainsi, si `ch` *désigne la chaine* `"SECRET"`*, l'entier* `ch.count("E")` *a pour valeur 2.*

d. Une fois la longueur de la clé connue, une simple analyse statistique suffit : en français, la lettre la plus courante est le "E". Si la longueur de la clé est `d`, le caractère le plus fréquent de chaque chaine extraite `ch[0::d], ch[1::d], ch[2::d], ..., ch[d-1::d]` correspond (statistiquement) à un "E" dans le texte clair (en supposant que le texte chiffré est suffisamment long). Chaque caractère de la clé se déduit par différence.

On souhaite écrire une fonction `determine_cle` qui prend en argument un texte chiffré selon la méthode de Vigenère et qui renvoie la clé de chiffrement.

Compléter le code fourni en annexe (qu'on rendra avec la copie).

e. En déduire une fonction `cryptanalyse_Vigenere` qui décrypte un texte chiffré selon la méthode de Vigenère.

⋆⋆⋆ FIN ⋆⋆⋆

Annexe : code à compléter (question 4.d)

```
def determine_cle(txt_chiffre):
    """renvoie la clé du texte chiffré par la méthode de Vigenère"""
    # calcul de la longueur de la clé
    d = ........................
    # détermination de chaque caractère de la clé
    cle = ""
    for k in range(d):
        car = lettre_plus_freq(........................)
        cle = cle + ........................
    return ........................
```