Épreuve d'informatique de l'X - 2008 - MP/PC

Codage de César

Question 1 : le codage de « `maitrecorbeau` » avec un décalage de 5 donne le message codé : « `hvdomzxjmwzvp` ».

Question 2 : ligne 4, on remplit le tableau représentant le texte codé avec les « lettres » du texte initial (représenté par un tableau d'entiers) décalés de d vers la gauche, ce qui se fait modulo le nombre de lettres dans l'alphabet soit 26.

```
1  def codageCesar(t, n, d):
2      tt = []
3      for i in range(n) :
4          tt.append((t[i] + 26 - d)%26)
5      return tt
```

Question 3 : même principe que dans la fonction précédente sauf que l'on effectue le décalage dans l'autre sens.

```
1  def decodageCesar(t,n,d):
2      tt = []
3      for i in range(n):
4          tt.append((t[i] + d)%26)
5      return tt
```

On aurait aussi pu utiliser la fonction précédente en effectuant un décalage dans l'autre sens de la manière suivante :

```
1  def decodageCesar(t,n,d):
2      return(codageCesar(t,n,(26-d)))
```

(PSI*)

Question 4 : On commence par créer et initialiser le tableau des fréquences (lignes 2-4) puis on parcourt le tableau représentant le texte. À la lecture de la i-ème lettre du texte, on incrémente la case correspondante dans le tableau des fréquences (lignes 5-7).

```
1  def frequences(tt,n):
2      freq = []
3      for i in range(26):
4          freq.append(0)
5      for i in range(n) :
6          lettre = tt[i]
7          freq[lettre] += 1
8      return freq
```

Question 5 : dans `decodageAuto`, on commence par récupérer le tableau des fréquences des apparitions des lettres dans le texte codé (ligne 2). On parcourt le tableau des fréquences pour repérer la fréquence d'apparition maximale ainsi que l'indice du tableau correspondant (ligne 7-10). On calcule la clé (ligne 11) en prenant en compte le fait que 'e' est la cinquième lettre de l'alphabet et enfin on décode le texte (ligne 12).

```
1  def decodageAuto(tt,n):
2      freq = frequences(tt,n)
3      code_e = 4
4      cle = 0
5      maxi = freq[0]
6      imaxi = 0
7      for i in range(26):
8          if maxi <= freq[i]:
9              imaxi = i
10             maxi = freq[i]
11     cle = (30-imaxi)%26
12     return decodageCesar(tt,n,cle)
```

Question 6 : Le codage du texte "`becunfromage`" en utilisant la clé de codage "`jean`" est "`kichwjrbvegr`".

Question 7 : Même principe que pour le codage de César sauf que le décalage dépend de la position de la lettre du texte à coder modulo k et s'obtient à partir des lettres de la clé. Le décalage pour la i-ème lettre du texte à coder est égal à $c[i \bmod k]$ (le décalage se fait cette fois-ci vers la droite).

```
1  def codageVigenere(t,n,c,k):
2      res = []
3      for i in range(n):
4          res.append((t[i]+c[i%k])%26)
5      return res
```

Question 8 : On programme un classique calcul de pgcd en utilisant les propriétés suivantes (les nombres sont supposés positifs) :

$pgcd(0,b) = b$, $pgcd(a,0) = a$, et $pgcd(a,b) = \begin{cases} pgcd(a,b-a) \text{ si } a \leqslant b \\ pgcd(a-b,b) \text{ si } a > b \end{cases}$.

```
1  def pgcd(a,b):
2      while a != 0 and b != 0:
3          if a <= b:
4              b = b-a
5          else:
6              a = a-b
7      return max(a,b)
```

Question 9 : On mémorise la répétition des trois lettres $\langle tt[i], tt[i+1], tt[i+2]\rangle$ du texte codé dans les variables $t0$, $t1$, $t2$ (ligne 4-6), puis on fait une boucle pour regarder si la séquence $\langle tt[j], tt[j+1], tt[j+2]\rangle$ coïncide avec la séquence de départ pour j variant de $i+3$ à $n-4$. Si c'est le cas, on modifie la variable `PGCD` qui contiendra le résultat final. On y met `pgcd`$(PGCD,\ j-i)$ (ligne 12).

On utilise la propriété `pgcd`$(a_1,\ldots,a_p)$ = `pgcd`$(\texttt{pgcd}(a_1,\ldots,a_{p-1}),a_p)$ et la propriété `pgcd`$(0,a) = a$.

```
1  def pgcdDesDistancesEntreRepetitions(tt,n,i):
2      a = 0
3      t0 = tt[i]
4      t1 = tt[i+1]
5      t2 = tt[i+2]
6      for j in xrange(i+3,n-4):
7          if tt[j]==t0 and tt[j+1]==t1 and tt[j+2]==t2:
8              ecart = j-i
9              if a == 0:
10                 a = ecart
11             else:
12                 a = pgcd(a,ecart)
13     return a
```

Question 10 : Dans cette fonction, on essaye de trouver la longueur de la clé. Pour cela, on calcule les distances de répétition de $\langle tt[i], tt[i+1], tt[i+2]\rangle$ pour i variant de 0 à $n-7$. Comme on a supposé que toute répétition d'une séquence de 3 lettres dans le texte codé t' provient exclusivement d'une séquence de 3 lettres répétée du texte original t, ces distances seront toutes des multiples de la longueur de la clé. On retourne donc le *pgcd* de toutes les distances non nulles pour obtenir le plus petit multiple possible de k (on espère trouver ainsi la longueur exacte de la clé avec ce procédé).

```
1  def longueurDeLaCle(tt,n):
2      res = 0
3      i = 0
4      while res == 0 and i < n-7:
5          res = pgcdDesDistancesEntreRepetitions(tt,n,i)
6          i += 1
7      return res
```

Question 11 : Dans la fonction `longueurDeLaCle`, on a au plus $n-6$ appels à la fonction `pgcd`. Mais il ne faut pas oublier que la fonction `pgcdDesDistancesEntreRepetitions` fait aussi appel à la fonction `pgcd` et que le nombre de ces appels est majoré par $n-i$.
On peut donc majorer le nombre total d'appels à la fonction `pgcd` par $n+\sum_{i=1}^{n} n-i = n+\frac{n(n+1)}{2}$ ce qui nous donne une complexité quadratique (en $O(n^2)$).

Question 12 : Une fois que l'on a k la longueur de la clé, on peut retrouver chacune des lettres de la clé en appliquant la méthode d'analyse des fréquences aux mots construits à partir du mot codé en prenant une lettre sur k. Pour le mot codé $\langle t'_0, \ldots, t'_{n-1}\rangle$ et $0 \leqslant j < k$, on fait une analyse en fréquence sur le mot $\langle t'_j, t'_{j+k}, \ldots, t'_{j+m.k}\rangle$, où $m = \lfloor \frac{((n-1)-j)}{k} \rfloor$. Cela nous permet de trouver la j-ème lettre de la clé de la même façon que dans la fonction `decodageAuto` de la question 5, à ceci près que les décalages se font vers la droite pour le codage de Vigenère et vers la gauche pour le codage de César.

Question 13 : En fait, on va créer la clé de décodage plutôt que la clé de codage pour pouvoir réutiliser la fonction `codageVigenere`. On commence par récupérer la longueur k de la clé que l'on obtient grâce à la fonction `longueurDeLaCle` (ligne 2). Pour trouver la j-ème lettre de la clé, on fait une analyse en fréquence du sous mot obtenu à partir du texte codé en partant de la j-ème lettre et en ne gardant qu'une lettre sur k (ligne 7-9). Une fois que l'on a le tableau des fréquences du sous mot, on le parcourt pour trouver la fréquence maximale qui correspondra normalement au codage de la lettre 'e' (lignes 12-15). On peut ainsi reconstituer la j-ème lettre de la clé de décodage (ligne 16). Une fois que l'on dispose de la clé de décodage, il ne reste plus qu'à décoder le texte (ligne 17).

```
1  def decodageVigenereAuto(tt,n):
2      k = longueurDeLaCle(tt,n)
3      cle = []
4      for j in range(k):
5          m = (n-1-j)/k
6          sousMot = []
7          for l in range(m+1):
8              sousMot.append(tt[j+l*k])
9          freq = frequences(sousMot,(m+1))
10         maxi = 0
11         imaxi = 0
12         for i in range(26):
13             if maxi <= freq[i]:
14                 maxi = freq[i]
15                 imaxi = i
16         cle.append((26 - (imaxi-4))%26)
17     return codageVigenere(tt,n,cle,k)
```