Épreuve d'informatique de l'X - 2009 - MP/PC

Chiffrement par blocs

Question 1. Initialement, on a `k` $= \sum_{j=0}^{N-1} a_j N^j$. À la i-ème étape de la boucle, la variable `k` est égale à $\sum_{j=i}^{N-1} a_j N^{j-i}$ et donc le reste de la division par N est a_i que l'on place bien en i-ème position dans le tableau résultat `res` à la ligne 4. À la ligne 5 modifie `k` en $\sum_{j=i+1}^{N-1} a_j N^{j-i-1}$.

```
1 def DecomposerBase(N,k) :
2     res = []
3     for i in range(N) :
4         res.append(k % N)
5         k = k / N
6     return res
```

Question 2. À la i-ème étape de la boucle, la variable `k` est égale à $\sum_{j=i}^{N-1} a_j \frac{j!}{i!}$ et donc $h - a_i$ est divisible par $(i+1)$ et a_i est le reste de la division euclidienne par $(i+1)$. On place bien a_i en i-ème position dans le tableau résultat `res` à la ligne 4. À la ligne 5 modifie `k` en $\sum_{j=i+1}^{N-1} a_j \frac{j!}{(i+1)!}$.

```
1 def DecomposerFact(N, k ) :
2     res = []
3     for i in range(N) :
4         res.append(k % (i+1))
5         k = k / (i+1)
6     return res
```

Question 3 : On supprime le j-ème élément en faisant attention au fait que les tableaux sont indexés à partir de 0.

```
1 def Retirer(L, l, j) :
2     res = []
3     for i in range(l-1) :
4         if i < j-1 :
5             res.append(L[i])
6         else :
7             res.append(L[i+1])
8     return res
```

Question 4 : On détermine le tableau représentant la permutation σ en suivant l'algorithme donné par l'énoncé. On commence par créer le tableau représentant la liste initiale $[0, 1, \ldots, N-1]$ (ligne 3 à 5). On récupère la décomposition factorielle de k (ligne 6). On applique l'algorithme dans la boucle des lignes 7 à 10.

```
1 def EcrirePermutation(N, k) :
2     sigma = []
3     L = []
4     for i in range(N) :
5         L.append(i)
6     decomp = DecomposerFact (N, k)
7     for i in xrange(N-1,-1,-1) :
8         j = decomp[i]
9         sigma.append(L[j])
10        L = Retirer(L, i+1, j+1)
11    return sigma
```

Question 5 : Pour le déchiffrage d'un bloc b, on recherche son antécédent par σ, ce qui se fait dans la boucle `while` des lignes 8 à 10.

```
1  def Chiffrer(N, k, b) :
2      sigma = EcrirePermutation(N,k)
3      return sigma[b]
4
5  def Dechiffrer(N, k, b) :
6      sigma = EcrirePermutation(N, k)
7      i = 0
8      while sigma[i] != b :
9          i += 1
10     return i
```

II. Réseau de Feistel

Question 6 : On crée une variable globale `deuxpuiss32` égale à 2^{32}.

```
1  deuxpuiss32 = 4294967296
2
3  Xor = lambda a, b : a^b
4
5  def FeistelTour(k, b) :
6      r = b % deuxpuiss32
7      q = b / deuxpuiss32
8      return deuxpuiss32 * r+ Xor(q, F(k, r))
```

Question 7 : Pour décoder, on utilise la propriété $\text{xor}(\text{xor}(b,x),x) = b$ et l'égalité $r_i = q_{i+1}$. Ceci qui permet de calculer q_i à partir de r_{i+1} et q_{i+1}.

```
1  def FeistelInverseTour(k, b) :
2      tmp = b % deuxpuiss32
3      r = b / deuxpuiss32
4      q = Xor(tmp,F(k,r))
5      return  deuxpuiss32*q+r
```

Question 8 : Le tableau K étant indexé à partir de 0, on exécute itérativement les différents tours pour les clefs contenues dans K (boucle des lignes 3 à 4).

```
1  def Feistel(K, l, b) :
2      res = b
3      for i in range(l) :
4          res = FeistelTour(K[i], res)
5      return res
```

Question 9 : Ici on décode à l'aide de la fonction `FeistelInverseTour` en utilisant les différentes clefs en partant de la fin (boucles des lignes 3 à 4).

```
1  def FeistelInverse(K, l , b) :
2      res = b
3      for i in xrange(l-1,-1,-1) :
4          res = FeistelInverseTour (K[ i ] , res )
5      return( res )
```

III. Vérification de propriétés statistiques

Question 10 : On construit le tableau S_n. Pour i variant de 0 à $\frac{n}{64}-1$ (boucle des lignes 5 à 11), on décompose $\sigma(i)$ en binaire en rangeant les bits dans le tableau de résultat `res` (boucle des lignes 10 à 14). On faisant attention de les ranger dans le bon ordre (bit de poids fort à gauche).

```
1  def alloue(n) :
2      t = []
3      for i in range(n) :
4          t.append(0)
5      return t
6
7  def sequence(n) :
8      res = alloue(n)
9      p = n /64
10     for i in range(p) :
11         s = Sigma(i)
12         for k in xrange(63,-1,-1) :
13             res[64*i+k] = s % 2;
14             s = s / 2
15     return res
```

Question 11 : On comptabilise les 0 et les 1 de la séquence S_n (boucle des lignes 5 à 9) dans les variables `n0` et `n1`. On calcule enfin V_1 ligne 11.

```
1  def CalculerV1(n) :
2      Sn = sequence(n)
3      n0 = 0
4      n1 = 0
5      for i in range(n)  :
6          if Sn[i] == 1 :
7              n1 += 1
8          else :
9              n0 += 1
10     return ((n1-n0)*(n1-n0)/float(n))
```

Question 12 : On comptabilise les occurrences des séquences `00`, `01`, `10` et `11` dans les variables `n00`, `n01`, `n10` et `n11` dans la boucles des lignes 13 à 25 ainsi que les occurrences de 0 et de 1 dans les variables `n0` et `n1`. Enfin, on calcule V_2 en ligne 26.

```
1  def CalculerV2(n) :
2      Sn = sequence(n)
3      if Sn[0] == 1 :
4          n0 = 0
5          n1 = 1
6      else :
7          n0 = 1
8          n1 = 0
9      n00 = 0
10     n01 = 0
11     n10 = 0
12     n11 = 0
13     for i in range(n-1) :
14         if Sn[i] == 1 and Sn[i+1] == 1 :
15             n11 += 1
16         elif Sn[i] == 1 and Sn[i+1] == 0 :
17             n10 += 1
18         elif Sn[i] == 0 and Sn[i+1] == 1 :
19             n01 += 1
20         else :
21             n00 += 1
22         if Sn[i+1] == 0 :
23             n0 += 1
24         else :
25             n1 += 1
26     return 4.*(n00**2+n01**2+n10**2+n11**2)\
27              /(n-1)-2.*(n0**2+n1**2)/(n+1)
```