X Informatique MP/PC 2009 — Corrigé

Ce corrigé est proposé par Arnaud Borde (École Polytechnique) ; il a été relu par Vincent Puyhaubert (Professeur en CPGE) et Céline Chevalier (ENS Cachan).

Le sujet de cette année propose l'étude d'une méthode de cryptographie basée sur les permutations d'un ensemble d'entiers et sur une structure appelée réseau de Feistel. Toutes les questions demandent d'écrire du code et, contrairement aux années précédentes, aucune justification de performance, aucun comptage de nombre d'opérations n'est demandé. Le sujet est beaucoup plus centré sur l'algorithmique (c'est-à-dire ce qu'il faut faire) que sur la syntaxe (comment le faire exactement) et les possibilités du langage utilisé (comment le faire avec élégance).

- La première partie est consacrée à la décomposition d'entiers sur deux bases différentes et à la construction de permutations.
- La deuxième est centrée sur la structure clef du codage, le réseau de Feistel.
- La troisième et dernière introduit des outils statistiques permettant d'évaluer la qualité d'un codage.

Ce sujet est un peu moins difficile que ceux des années précédentes ; il demande surtout de la concentration et de la méthode. Ses trois parties sont indépendantes et peuvent donc être traitées dans un ordre quelconque.

Indications

I. Approche naïve

1 Faire attention à l'ordre des éléments. Remarquer que k peut s'écrire

$$a_0 + \mathrm{N} \times (a_1 + a_2\mathrm{N} + \cdots + a_{\mathrm{N}-1}\mathrm{N}^{\mathrm{N}-2})$$

et effectuer des divisions euclidiennes.

2 Employer le même cheminement qu'à la question précédente en constatant cette fois que

$$k = a_0 + 1 \times (a_1 + 2 \times (...(a_{\mathrm{N}-2} + (\mathrm{N}-1) \times a_{\mathrm{N}-1}))...))$$

3 Utiliser deux boucles pour recopier les valeurs en sautant l'élément à supprimer.

4 Appliquer les fonctions `DecompositionFact` et `Retirer` écrites dans les questions 2 et 3.

5 Pour `Chiffrer`, retourner uniquement l'élément demandé de la liste calculée à la question 4. Pour `Dechiffrer`, parcourir cette liste jusqu'à trouver b.

II. Réseau de Feistel

6 Utiliser les fonctions données et appliquer directement les formules de l'énoncé pour calculer q_{i+1} et r_{i+1}.

7 Utiliser la propriété $\texttt{xor}(\texttt{xor}(x, y), y) = x$ pour exprimer q_i en fonction de r_i et r_{i+1}.

8 Appliquer ℓ fois la fonction `FeistelTour` à l'aide d'une boucle `for`.

9 Employer une boucle `for` pour appeler la fonction `FeistelInverseTour` plusieurs fois.

III. Vérification de propriétés statistiques

10 Commencer par écrire une fonction `Base2`(n) qui renvoie la décomposition en base 2 de l'entier n sur 64 bits. Faire attention à l'ordre des bits lors de la construction de S_n.

11 Utiliser deux variables pour compter les 1 et les 0 au fur et à mesure du parcours de la liste.

12 Regarder chaque séquence de deux bits à l'aide d'une boucle `for` pour les compter.

Les fonctions de ce corrigé sont toutes rédigées dans le langage Maple. Toutefois, le quotient et le reste de la division euclidienne de deux entiers `a` et `b` s'obtiennent dans ce langage respectivement par les commandes `iquo(a,b)` et `irem(a,b)`. Ces commandes *ne sont pas utilisées* dans les programmes car l'énoncé impose les notations `quo(a,b)` et `rem(a,b)`. Pour obtenir des codes qui fonctionnent, il faut donc remplacer ces notations par les vraies fonctions.

L'énoncé demande également d'utiliser une fonction `xor`. Celle-ci existe en Maple, mais ne fais pas le travail voulu par l'énoncé. Il faudra donc utiliser le code de la procédure donné à la fin de la partie II.

Pour finir, l'énoncé laisse le candidat libre d'utiliser des listes ou des tableaux. Le choix des listes a été retenu dans ce corrigé ; toutefois, Maple limite leur taille à 100 éléments. En contrepartie, ce langage est beaucoup moins limité en ce qui concerne la taille des entiers manipulables.

I. Approche naïve

1 Pour décomposer l'entier k dans la base N, comme on connaît la taille maximale de k (à savoir $\mathrm{N}^{\mathrm{N}}-1$), effectuons des divisions euclidiennes successives par N. Dans un premier temps, on remarque que k peut s'écrire

$$a_0 + \mathrm{N} \times (a_1 + a_2\mathrm{N} + \cdots + a_{\mathrm{N}-1}\mathrm{N}^{\mathrm{N}-2})$$

donc a_0 est le reste entier de la division de k par N. L'élément suivant, a_1, est ensuite obtenu en divisant le quotient de cette division par N. On reprend ainsi de suite le quotient de la division euclidienne à chaque fois. Cela est réalisé par une boucle `for` effectuant N itérations, au sein desquelles l'algorithme calcule successivement les coefficients a_0, a_1, ... (avec la fonction `irem`) en mettant à jour le quotient de la division (avec `iquo`).

En Maple, le premier élément d'une liste ou d'un tableau est celui d'indice 1, pas 0. Par conséquent, a_i est stocké à la $(i+1)$-ième place dans la liste. Ce décalage d'indice aura lieu systématiquement dans tout le corrigé.

```
DecomposerBase:=proc(N,k)
  local decomposition, i, quotient;
  quotient:=k;
  decomposition:=[seq(0,i=0..N-1)];
  for i from 0 to N-1 do
    # le reste de la division par N donne a_i
    # que l'on met dans decomposition[i+1]
    decomposition[i+1]:=irem(quotient,N);
    # on met le quotient à jour
    quotient:=iquo(quotient,N);
  od;
  RETURN(decomposition);
end;
```

Cet algorithme illustre bien une autre écriture possible de k, à savoir $k = a_0 + \mathrm{N}(a_1 + a_2\mathrm{N} + \cdots + a_{\mathrm{N}-1}\mathrm{N}^{\mathrm{N}-2})$. Cette écriture est utilisée dans un algorithme célèbre, celui de Hörner, pour minimiser le nombre d'opérations lors du calcul de la valeur d'un polynôme en un point.

2 Le principe de cette question est presque le même que celui de la précédente, mais il faut changer le diviseur à chaque itération pour décomposer dans la bonne base.

> Le nombre de possibilités pour la décomposition en base factorielle s'élève à $1 \times 2 \times 3 \times ..... \times N = N!$ (1 pour a_0, 2 pour a_1, ..., N pour a_{N-1}). De plus, l'entier k est majoré par $\sum_{i=0}^{N-1} i \times i! = N! - 1$ (il s'agit d'une somme télescopique en vertu de l'égalité $i \times i! = (i+1)! - i!$). Cette décomposition permet donc d'obtenir une bijection explicite entre les ensemble $[\![0 ; N! - 1]\!]$ et $\{0\} \times [\![0 ; 1]\!] \times ..... \times [\![0 ; N-1]\!]$ (l'égalité des cardinaux en assure l'existence, mais ne la fournit pas pour autant).

On peut déjà remarquer que a_0 vaut forcément 0. Ensuite, a_1 est le reste de la division de k par 2. Puis a_2 est le reste de la division du quotient précédent par 3. Ce changement de diviseur se voit facilement quand on écrit

$$k = a_0 + 1 \times (a_1 + 2 \times (...(a_{N-2} + (N-1) \times a_{N-1}))...))$$

```
DecomposerFact:=proc(N,k)
  local decomposition, i, quotient;
  quotient:=k;
  decomposition:=[seq(0,i=0..N-1)];
  for i from 0 to N-1 do
    # on traite ici a_i
    decomposition[i+1]:=irem(quotient,i+1);
    quotient:=iquo(quotient,i+1);
  od;
  RETURN(decomposition);
end;
```

3 L'énoncé demande de retourner une copie de L. La méthode qui semble la plus naturelle consiste donc à recopier dans une liste L′ de cardinal $\ell - 1$ tous les éléments précédant le j-ième élément puis tous les suivants. Cela est réalisé dans le code par deux boucles `for` s'occupant chacune d'une partie de la liste d'origine.

```
Retirer:=proc(L,l,j)
    local L_prime, i;
    L_prime:=[seq(0,i=0..l-2)];

    # on copie la partie de la liste avant le j-ième élément
    for i from 0 to j-1 do
        L_prime[i+1]:=L[i+1];
    od;

    # on copie la partie de la liste après le j-ième élément
    for i from j+1 to l-1 do
        L_prime[i]:=L[i+1];
    od;

    RETURN(L_prime);
end;
```