

# Option Informatique en Spé MP et MP\*

## Devoir à rendre après les vacances de Noël

Le problème du monnayeur

### 1 Notations

- ▶  $|E|$  désigne le cardinal de l'ensemble fini  $E$ .
- ▶  $\lfloor x \rfloor$  désigne la partie entière inférieure du réel  $x$ , et  $\lceil x \rceil$  sa partie entière supérieure: ce sont les relatifs définis respectivement par  $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$  et  $\lceil x \rceil - 1 < x \leq \lceil x \rceil$ .
- ▶  $\llbracket p, q \rrbracket$  désigne l'intervalle discret délimité par les naturels  $p$  et  $q$ :  $\llbracket p, q \rrbracket = \{k \in \mathbb{N} \mid p \leq k \leq q\}$ .

### 2 Le problème du monnayeur

- ▶ Nous nous intéressons au problème suivant, rencontré lors de la conception d'un monnayeur: comment rendre la monnaie, en utilisant le plus petit nombre possible de pièces?
- ▶ Formalisons ce problème: soit  $(c_i)_{1 \leq i \leq m}$  une liste d'entiers non nuls, vérifiant

$$c_1 > c_2 > \dots > c_{m-1} > c_m = 1$$

Nous utiliserons le terme *système* pour désigner une telle liste. Les  $c_i$  sont les *valeurs faciales* des pièces en service. Par exemple, si nous prenons comme unité le centime, le système utilisé en France<sup>1</sup> est (200, 100, 50, 20, 10, 5, 2, 1).

- ▶ Pour chaque  $i \in \llbracket 1, m \rrbracket$ , nous disposons d'une quantité illimitée de pièces de valeur  $c_i$ . Soit  $x$  un entier (le montant à rendre). Une *représentation* de  $x$  dans le système  $c$  est un  $m$ -uplet  $(k_i)_{1 \leq i \leq m}$  de naturels vérifiant  $x = \sum_{1 \leq i \leq m} k_i c_i$ :  $k_i$  est donc le nombre de pièces de valeur  $c_i$  que l'on doit rendre. Pour

épargner les poches des clients, nous souhaitons minimiser le *poids* de cette représentation, c'est-à-dire le nombre  $\|k\| = \sum_{1 \leq i \leq m} k_i$  de pièces rendues.

- ▶ Nous pouvons considérer les différents  $m$ -uplets utilisés comme des vecteurs de  $\mathbb{R}^m$ , muni de sa structure euclidienne canonique. Si  $k$  est une représentation de  $x$  dans le système  $c$ , alors  $x = k \cdot c$  et  $\|k\| = k \cdot \mathbf{1}$ , où  $\mathbf{1}$  désigne le  $m$ -uplet dont toutes les composantes sont égales à 1.

- ▶ Nous utiliserons des listes d'entiers pour représenter aussi bien un système (liste de valeurs de pièces), qu'une représentation d'un montant dans ce système. Par exemple, la liste  $[4; 1; 3]$  est une représentation de 30 dans le système (6, 3, 1).

**Question 1** • Rédigez en Caml une fonction de signature

```
est_un_systeme : int list -> bool
```

spécifiée comme suit: `est_un_systeme c` indique si la liste  $c$  est bien un système. Par exemple, `est_un_systeme [5;2;1]` rendra la valeur `true`, tandis que `est_un_systeme [5;7;1]` rendra la valeur `false`, tout comme le fera `est_un_systeme [7;5;2]`.

---

<sup>1</sup>et plus généralement dans la zone euro

### 3 Représentations de poids minimal

► Soient  $c = (c_i)_{1 \leq i \leq m}$  un système et  $x \geq 0$ . Nous notons  $M_c(x)$  (ou même  $M(x)$  lorsqu'aucune ambiguïté n'est à craindre) le plus petit nombre de pièces nécessaires pour représenter  $x$  dans le système  $c$ :

$$M(x) = \min\{\|k\| \mid k \in \mathbb{N}^m \text{ et } k \cdot c = x\}$$

Nous nous intéresserons aux représentations de poids minimal de  $x$  : ce sont les représentations  $k$  telles que  $\|k\| = M(x)$ . Pour alléger la rédaction, nous parlerons simplement de *représentations minimales*.

**Question 2** • Prouvez l'encadrement  $\lceil x/c_1 \rceil \leq M(x) \leq x$ .

**Question 3** • Exhibez un système  $c$  et un nombre  $x > 0$  possédant plusieurs représentations minimales dans le système  $c$ .

**Question 4** • Soient  $c$  un système et  $x > 0$ . Montrez que  $M(x) \leq 1 + M(x - c_j)$  pour tout indice  $j$  tel que  $c_j \leq x$ .

**Question 5** • Soient  $c$  un système,  $x > 0$  et  $j$  tel que  $c_j \leq x$ . Montrez que  $M(x) = 1 + M(x - c_j)$  si et seulement si il existe une représentation minimale de  $x$ , faisant intervenir  $c_j$  (c'est-à-dire telle que  $k_j > 0$ ).

**Question 6** • Soit  $x > 1$ ; notons  $s$  le plus petit indice  $i$  tel que  $c_i \leq x$ . Nous avons donc  $c_s \leq x$ ; et, soit  $s = 1$ , soit  $s > 1$  et  $c_{s-1} > x$ . Justifiez l'égalité

$$M(x) = 1 + \min_{s \leq i \leq m} M(x - c_i)$$

**Question 7** • Montrez que l'on peut construire la liste des valeurs de  $M(y)$  pour  $y \leq x$ , pour un coût  $\mathcal{O}(mx)$ .

**Question 8** • Rédigez en Caml une fonction de signature

```
poids_minimaux : int -> int list -> int list
```

spécifiée comme suit : `poids_minimaux x c` construit la liste des valeurs de  $M_c(y)$ , pour  $1 \leq y \leq x$ . Par exemple, `poids_minimaux 5 [5;2;1]` rendra la liste `[1;1;2;2;1]`. Vous utiliserez la formule de la question 6.

### 4 L'algorithme glouton

**G** ► L'algorithme *glouton* consiste à choisir le plus grand  $c \leq x$ , puis à rendre le montant  $x - c$ . Par exemple, avec le système  $(10, 5, 2, 1)$  l'algorithme glouton décomposera le montant 27 en  $27 = 10 + 10 + 5 + 2$ . Avec le formalisme proposé,  $m = 4$ ,  $c_1 = 10$ ,  $c_2 = 5$ ,  $c_3 = 2$ ,  $c_4 = 1$ ; la solution fournie par l'algorithme glouton est le quadruplet  $(k_1 = 2, k_2 = 1, k_3 = 1, k_4 = 0)$ . Le fonctionnement de l'algorithme glouton peut être accéléré avec la remarque suivante : notant  $q = \lfloor x/c_1 \rfloor$ , cet algorithme rend  $q$  pièces de valeur  $c_1$ , puis rend le montant  $x - qc_1$  en utilisant le système  $(c_i)_{2 \leq i \leq m}$ .

**Question 9** • Rédigez en Caml une fonction de signature

```
glouton : int -> int list -> int list
```

spécifiée comme suit : `glouton x c` construit la représentation de  $x$  dans le système  $c$ , en appliquant l'algorithme glouton. Par exemple, `glouton 13 [5;2;1]` rendra la liste `[2;1;1]`.

► Soient  $c = (c_i)_{1 \leq i \leq m}$  un système et  $x \geq 0$ . Nous noterons  $\Gamma_c(x)$  la représentation gloutonne de  $x$  dans le système  $c$ , et  $G_c(x)$  (ou même  $G(x)$  lorsqu'aucune ambiguïté n'est à craindre) le nombre de pièces utilisées par l'algorithme glouton :  $G_c(x) = \|\Gamma_c(x)\|$ .

► Nous dirons qu'un système  $c = (c_i)_{1 \leq i \leq m}$  est *canonique* si l'algorithme glouton donne toujours une représentation minimale; on a donc  $M(x) = G(x)$  quel que soit  $x$ .

**Question 10** • Montrez que tout système  $c = (c_1, c_2)$  de deux valeurs est canonique.

**Question 11** • Exhibez un système  $c = (c_1, c_2, c_3)$  non canonique.

**Question 12** • Montrez qu'un système  $c = (c_i)_{1 \leq i \leq m}$  vérifiant  $c_i \geq 2c_{i+1}$  pour tout  $i \in \llbracket 1, m-1 \rrbracket$  n'est pas nécessairement canonique.

**Question 13** • Soient  $\lambda \geq 2$ , et  $c = (c_i)_{1 \leq i \leq m}$  un système vérifiant  $c_i = 2c_{i+1}$  pour tout  $i \in \llbracket 1, m-1 \rrbracket$ . Montrez qu'un tel système est canonique.

► Le résultat précédent montre que le système utilisé en France est canonique : si nous prenons comme unité la pièce de 1 centime, ce système est  $(200, 100, 50, 20, 10, 5, 2, 1)$ .

**Question 14** • Avant la réforme de 1971 (introduisant un système décimal), le Royaume-Uni utilisait le système  $(30, 24, 12, 6, 3, 1)$ . Montrez que ce système n'est pas canonique.

**Question 15** • Considérons un monnayeur basé sur un système canonique. Donnez un exemple de situation où il est souhaitable que ce monnayeur n'applique pas l'algorithme glouton.

► Dans les deux questions suivantes, nous nous intéressons au système  $c = (c_i)_{1 \leq i \leq m}$  de FIBONACCI. Ce système est défini par  $c_{m-1} = 2$ , et par la relation de récurrence :  $c_i = c_{i+1} + c_{i+2}$  pour  $1 \leq i \leq m-2$ .

**Question 16** • Soient  $x$  un entier et  $\sum_{1 \leq i \leq m} k_i c_i$  sa décomposition gloutonne. Montrez que  $k_i$  est égal à 0 ou à 1 pour  $i \in \llbracket 2, m \rrbracket$ ; et que  $k_i k_{i-1} = 0$  pour  $i \in \llbracket 3, m \rrbracket$ .

**Question 17** • Montrez que le système de FIBONACCI est canonique.

## 5 L'algorithme de Kozen et Zaks

► Nous nous proposons de décrire un algorithme efficace permettant de déterminer si un système  $c$  est canonique. Nous dirons qu'un entier  $x$  est un *contre-exemple* pour  $c$  si  $M(x) < G(x)$ . Un système canonique n'admet donc aucun contre-exemple.

**Question 18** • Soit  $c$  un système non canonique. Montrez qu'il existe un contre-exemple  $x$  vérifiant l'encadrement  $c_{m-2} + 1 < x < c_1 + c_2$ .

► Nous allons montrer que l'intervalle de recherche des contre-exemples ne peut pas être diminué.

**Question 19** • Soit  $q \geq 3$ . Montrez que le système  $(q+1, q, 1)$  n'est pas canonique. Quel est le plus petit contre-exemple pour ce système ?

**Question 20** • Soit  $q \geq 3$ . Déterminez  $\alpha(q) > q$  tel que le système  $(\alpha(q), q, 1)$  ne soit pas canonique, et admette  $\alpha(q) + 2$  comme plus petit contre-exemple.

► Le résultat de la question 18 nous donne un algorithme déterminant si un système est canonique : il suffit de les rechercher dans l'intervalle discret  $\llbracket c_{m-2} + 2, c_1 + c_2 - 1 \rrbracket$ . Ceci nécessite la construction des représentations minimales de chacun des éléments de cet intervalle.

► Nous allons étudier un algorithme plus efficace, dû à KOZEN et ZAKS ; leur méthode repose sur la notion de *témoin*. Nous dirons qu'un entier  $x$  est un témoin pour le système  $c = (c_i)_{1 \leq i \leq m}$  s'il existe un indice  $i$  tel que  $G(x - c_i) < G(x) - 1$ .

**Question 21** • Montrez que tout témoin est un contre-exemple.

**Question 22** • Montrez que le résultat de la question 21 n'admet pas de réciproque ; vous exhiberez un système  $c = (s, t, 1)$  et un entier  $x$  qui est un contre-exemple, mais pas un témoin pour  $c$ .

**Question 23** • Montrez que si le système  $c$  admet des contre-exemples, alors le plus petit d'entre eux est un témoin.

► Il résulte de cette étude que, pour savoir si un système est canonique, il suffit de vérifier l'inégalité  $G(x) \leq G(x - c_i) + 1$  pour tout  $x \in \llbracket c_{m-2} + 2, c_1 + c_2 - 1 \rrbracket$ , et tout indice  $i \in \llbracket 1, m \rrbracket$  tel que  $c_i < x$ . C'est le principe de l'algorithme de KOZEN et ZAKS.

**Question 24** • Rédigez en Caml une fonction de signature

```
kozen_zaks : int list -> bool
```

spécifiée comme suit : `kozen_zaks c` indique si la liste  $c$  est canonique.

Par exemple, `kozen_zaks [5;2;1]` rendra la valeur `true`, tandis que `kozen_zaks [6;5;1]` rendra la valeur `false`. Vous utiliserez l'algorithme de KOZEN et ZAKS.

**Question 25** • Montrez que le coût de l'algorithme de KOZEN et ZAKS peut être exponentiel, par rapport au nombre  $m$  de pièces du système.

**Question 26** • Montrez qu'un système  $(s, t, 1)$  est non canonique si et seulement si, notant  $q$  le quotient et  $r$  le reste dans la division euclidienne de  $s$  par  $t$ , on a  $0 < r < t - q$ .

## 6 L'algorithme de Pearson

► Dans cette partie, nous allons étudier un algorithme, dû à PEARSON, dont le coût est polynomial par rapport au nombre  $m$  de pièces du système.

► Soient  $u = (u_i)_{1 \leq i \leq m}$  et  $v = (v_i)_{1 \leq i \leq m}$  deux  $m$ -uplets de réels. Nous noterons  $u \leq v$  si  $u_i \leq v_i$  pour tout indice  $i \in \llbracket 1, m \rrbracket$ ;  $\mathbb{R}^m$  est ainsi muni d'un ordre partiel. Nous noterons  $u \prec v$  s'il existe un indice  $j \in \llbracket 1, m \rrbracket$  tel que  $u_i = v_i$  pour  $1 \leq i < j$ , et  $u_j < v_j$ . Nous noterons  $u \preceq v$  si  $u = v$  ou  $u \prec v$  : nous avons ainsi défini sur  $\mathbb{R}^m$  l'ordre *lexicographique* ; cet ordre est total.

**Question 27** • Soient  $c = (c_i)_{1 \leq i \leq m}$  un système, et  $x > 0$ . Montrez que  $\Gamma_c(x)$  est, pour l'ordre lexicographique, le plus grand élément de l'ensemble des représentations de  $x$ .

**Question 28** • Soient  $x$  et  $y$  deux réels. Montrez que  $x < y$  implique  $\Gamma_c(x) \prec \Gamma_c(y)$ .

**Question 29** • Soient  $u$  et  $v$  deux  $m$ -uplets de naturels tels que  $u \leq v$  et  $v = \Gamma_c(v \cdot c)$ . Montrez que  $u = \Gamma_c(u \cdot c)$ .

► Soient  $c = (c_i)_{1 \leq i \leq m}$  un système, et  $x > 0$ . Parmi les représentations minimales de  $x$  dans le système  $c$ , nous noterons  $\Omega_c(x)$  celle qui est la plus grande pour l'ordre lexicographique ; nous dirons que  $\Omega_c(x)$  est la *représentation optimale* de  $x$ .

**Question 30** • Soient  $u$  et  $v$  deux  $m$ -uplets de naturels tels que  $u \leq v$  et  $v = \Omega_c(v \cdot c)$ . Montrez que  $u = \Omega_c(u \cdot c)$ .

► Dans toute la suite,  $c$  désigne un système non canonique, et  $\xi$  le plus petit contre-exemple pour ce système. Notons  $g = \Gamma_c(\xi)$  et  $\omega = \Omega_c(\xi)$ .

**Question 31** • Montrez que  $g_i \omega_i = 0$  pour tout  $i \in \llbracket 1, m \rrbracket$ .

► Notons  $i$  le plus petit indice tel que  $\omega_i \neq 0$ , et  $j$  le plus grand indice tel que  $\omega_j \neq 0$ .

**Question 32** • Prouvez que  $i$  est au moins égal à 2.

**Question 33** • Soit  $\gamma = \Gamma_c(c_{i-1} - 1)$ . Montrez que  $\omega_k = \gamma_k$  pour  $1 \leq k < j$ , et  $\omega_j = \gamma_j + 1$ .

**Question 34** • Déduisez de ce dernier résultat un algorithme déterminant si un système  $c$  donné est canonique, pour un coût  $\mathcal{O}(m^3)$ .

**Question 35** • Rédigez en Caml une fonction de signature

```
pearson : int list -> bool
```

spécifiée comme suit : `pearson c` indique si la liste  $c$  est canonique.

Par exemple, `pearson [5;2;1]` rendra la valeur `true`, tandis que `pearson [6;5;1]` rendra la valeur `false`. Vous utiliserez l'algorithme de PEARSON.

FIN