Rendu de monnaie

Durée : 3 heures

Le sujet traite du problème du monnayeur : comment rendre la monnaie en utilisant le plus petit nombre de pièces ? La première partie met en place le formalisme et les outils qui serviront par la suite. On étudiera dans la deuxième partie l'*algorithme glouton*. Enfin, la dernière partie présente un algorithme permettant de décider si l'algorithme glouton est optimal pour un système de pièces donné.

Formalisation du problème

On appelle *système* un m-uplet d'entiers $c = (c_i)_{1 \leqslant i \leqslant m}$ vérifiant : $c_1 > c_2 > \cdots > c_m = 1$. Les c_i sont les valeurs faciales des pièces (ou billets) en service. Par exemple, le système utilisé en zone Euro est : $(500, 200, 100, 50, 20, 10, 5, 2, 1)$.
Pour tout $i \in [\![1, m]\!]$, 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 $k = (k_1, \ldots, k_m)$ vérifiant :

$$x = \sum_{i=1}^{m} k_i c_i.$$

k_i est donc le nombre de pièces c_i qui seront rendues.
Pour épargner les poches des clients, nous souhaitons minimiser le poids de cette représentation, c'est à dire la quantité :

$$w(k) = \sum_{i=1}^{n} k_i.$$

Partie I. Représentations de poids minimal

Nous utiliserons des listes d'entiers pour représenter aussi bien un système 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. Rédiger 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.

Soient $c = (c_1, \ldots, c_m)$ un système, et $x \in \mathbb{N}^*$. Nous notons $\mathrm{M}(x)$ le plus petit nombre de pièces nécessaires pour représenter x dans le système c :

$$\mathrm{M}(x) = \min\left\{ w(k) \;\middle|\; k \in \mathbb{N}^m \text{ et } x = \sum_{i=1}^{m} k_i c_i \right\}.$$

Nous nous intéresserons aux représentations de poids minimal de x : ce sont les représentations k telles que $w(k) = \mathrm{M}(x)$.

Question 2. Prouver l'encadrement : $\left\lceil \dfrac{x}{c_1} \right\rceil \leqslant \mathrm{M}(x) \leqslant x$.

Question 3.

a) Montrer que pour tout indice j tel que $c_j \leqslant x$ on a : $\mathrm{M}(x) \leqslant 1 + \mathrm{M}(x - c_j)$.

b) Montrer qu'on a : $\mathrm{M}(x) = 1 + \mathrm{M}(x - c_j)$ si et seulement s'il existe une représentation minimale k de x faisant intervenir c_j, c'est à dire telle que $k_j > 0$.

c) Soit s le plus petit indice i vérifiant : $c_i \leqslant x$. Justifier l'égalité :

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

Question 4. Rédiger en Caml une fonction de signature :

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

spécifiée comme suit : **poids_minimaux x c** construit le tableau des valeurs de $M(y)$ pour $0 \leqslant y \leqslant x$. Par exemple, **poids_minimaux 5 [5; 2; 1]** rendra le vecteur **[|0; 1; 1; 2; 2; 1|]**. Cet exemple fournit d'ailleurs l'ordre dans lequel on souhaite que les $M(y)$ apparaissent dans le tableau résultat.
On rappelle que **make_vect n a** retourne un vecteur de longueur n dont tous les emplacements contiennent la valeur a.

Partie II. L'algorithme glouton

Avertissement : dans cette partie, on travaillera obligatoirement sur des listes sans passer par des vecteurs.

L'algorithme glouton pour rendre une somme $x > 0$ consiste à choisir le plus grand $c_i \leqslant x$, puis à rendre récursivement $x - c_i$. Par exemple, avec le système $c = (10, 5, 2, 1)$, l'algorithme décomposera 27 en : $10 + 10 + 5 + 2$. Avec le formalisme proposé, la solution fournie par l'algorithme glouton est donc $k = (2, 1, 1, 0)$. Le fonctionnement de l'algorithme glouton peut être accéléré par la remarque suivante :

> notant $q = \left\lfloor \frac{x}{c_1} \right\rfloor$, cet algorithme rend q pièces de valeur c_1, puis rend le montant $x - qc_1$ en utilisant le système $(c_2, \dots, c_m)$.

Question 5. Rédiger 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 utilisant l'algorithme glouton. Par exemple, **glouton 13 [5; 2; 1]** retournera la liste **[2; 1; 1]**.

Nous noterons $\Gamma(x)$ la représentation gloutonne de x dans le système c, et $G(x)$ le nombre de pièces utilisées par l'algorithme glouton : $G(x) = w(\Gamma(x))$.
Nous dirons que le système c est *canonique* lorsque l'algorithme glouton nous donne toujours une représentation minimale ; on a alors $M(x) = G(x)$ pour tout $x \in \mathbb{N}$.

Question 6.

a) Montrer que tout système (c_1, c_2) est canonique.

b) Exhiber un système (c_1, c_2, c_3) non canonique (en justifiant).

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

Partie III. L'algorithme de Kozen et Zaks

Nous allons voir ici un algorithme efficace permettant de déterminer si un système c est canonique. Nous dirons qu'un entier x est un *contre-exemple* pour c lorsque $M(x) < G(x)$. Un système canonique n'admet donc pas de contre-exemple.
Dans la suite du problème, on supposera $m \geqslant 3$.

Question 7. Soit c un système non canonique, x un contre-exemple, et k une représentation minimale de x.

a) Si $x \geqslant c_1$, montrer la relation : $G(x) = 1 + G(x - c_1)$.
En déduire que si k_1 est non nul, alors $x - c_1$ est aussi un contre-exemple.

b) On suppose maintenant $x \geqslant c_1 + c_2$ et que $x - c_1$ n'est pas un contre-exemple. Soit $i < m$ un indice tel que k_i soit non nul. Montrer que $G(x) \leqslant 2 + M(x - c_1 - c_i)$, et en déduire que $x - c_i$ est un contre-exemple.

c) Montrer alors que le plus petit des contre-exemples vérifie :

$$c_{m-2} + 1 < x < c_1 + c_2.$$

Question 8.

a) Soit $q \geqslant 3$. Montrer que le système $(q+1, q, 1)$ n'est pas canonique. Quel est le plus petit contre-exemple pour ce système ? Justifier.

b) Soit $q \geqslant 3$. Déterminer $\alpha(q) > q$ tel que le système $(\alpha(q), q, 1)$ ne soit pas canonique, et admette $\alpha(q)+2$ comme plus petit contre-exemple.

c) Que vient-on de faire dans ces deux questions ?

Le résultat de la question 7 nous donne un algorithme déterminant si un système est canonique : il suffit de rechercher un contre-exemple dans l'intervalle $[\![c_{m-2}+2, c_1+c_2-1]\!]$. Ceci nécessiterait la construction (coûteuse) 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 s'il existe un indice i tel que $c_i < x$ et $\mathrm{G}(x-c_i) < \mathrm{G}(x)-1$.

Question 9.

a) Montrer que tout témoin est un contre-exemple.

b) En considérant le système $(5,4,1)$, montrer que le résultat précédent n'admet pas de réciproque.

c) Montrer que si le système c admet des contre-exemples, 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 $\mathrm{G}(x) \leqslant \mathrm{G}(x-c_i)+1$ pour tout $x \in [\![c_{m-2}+2, c_1+c_2-1]\!]$ et tout indice $i \in [\![1, m]\!]$ tel que $c_i < x$. C'est le principe de l'algorithme de Kozen et Zaks.

Question 10. Rédiger en Caml une fonction de signature :

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

spécifiée comme suit : **`kozen_zaks c`** indique si le système c est canonique, en utilisant l'algorithme de Kozen et Zaks.

Remarque. Ne pas hésiter à appliquer une approche modulaire en utilisant des fonctions annexes (en précisant les spécifications de celles-ci). Comme le dit René Descartes, il convient *de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait, et qu'il serait requis pour mieux les résoudre*.

Questions hors barème

Question 11. Soient q et n deux entiers supérieurs ou égaux à 2. Montrer que le système $(q^n, q^{n-1}, \ldots, q, 1)$ est canonique.

Question 12. En déduire que le coût de l'algorithme de Kozen et Zaks peut être exponentiel par rapport au nombre m de pièces du système. On exhibera deux systèmes (l'un canonique, l'autre pas) pour lesquels le coût de l'algorithme est exponentiel en m.

♦ ♦
♦