Option Informatique en Sup MPSI
Le problème de partition : corrigé

Question 1 • Notons $\mathcal{P}(X,k)$ le problème consistant à exhiber une partie Y de X vérifiant $\|Y\| = k$. Avec cette notation, le problème de partition est $\mathcal{P}\big(X, \|X\|/2\big)$. Fixons donc $k = \|X\|/2$ et présentons à l'oracle le couple (X,k) : si sa réponse est négative, inutile de continuer. Sinon, choisissons $x \in X$, et présentons à l'oracle le couple (Y,k) où $Y = X \setminus \{x\}$; deux cas peuvent se produire :

- si sa réponse est positive, nous éliminons x, et il nous reste à résoudre le problème $\mathcal{P}(Y,k)$;
- sinon, nous gardons x et il nous reste à résoudre le problème $\mathcal{P}(Y,k-x)$.

Notons k_i la valeur de k et G_i l'ensemble des éléments gardés après i étapes ; on vérifie sans peine que $k_i + \|G_i\|$ est constante. Au bout de n étapes, chacun des éléments de X aura été gardé ou éliminé ; et $k_n = 0$. Donc $\|G_n\| = k$. Nous aurons consulté $n+1$ fois l'oracle. Il est clair que l'on peut arrêter les frais aussitôt que $k_i = 0$.

Question 2 • `toto` est une fonction à deux arguments. `s = 0` indique que le premier est de type `int` ; le deuxième est de type `int list` au vu du filtrage et de `s = t` ; enfin, le résultat est de type `bool` au vu de `s = 0`. On vérifie sans peine que les différents motifs sont cohérents. Concluons : `toto` est de type `int -> int list -> bool`.

Question 3 • Nous dirons que s est réalisable avec la liste x lorsque l'on peut extraire de x une sous-liste dont la somme est s. Nous allons prouver, par induction structurelle sur la liste x, que `toto s x` rend la valeur `true` ssi s est réalisable avec la liste x.

- si x est la liste vide, le résultat est clair : la seule somme réalisable est 0.
- sinon, soit t la tête de la liste x et q sa queue. Distinguons deux cas de figure :
  - si $s < t$, alors t ne peut servir à réaliser s ; donc s est réalisable avec la liste x ssi il peut l'être avec la liste q ;
  - si $s \geqslant t$, alors s est réalisable en faisant éventuellement intervenir t (c'est le cas ssi $s-t$ est réalisable avec q) ou sans faire intervenir t (c'est le cas ssi s est réalisable avec q).

Question 4 • L'essentiel du travail est fait par la fonction `réaliser_total` : appliquée au couple `(c,x)`, elle extrait de la liste x une sous-liste dont la somme est égale à c, en appliquant les règles suivantes :

- si $c = 0$, la liste vide répond à la question ;
- sinon : on ne peut réaliser $c \neq 0$ avec la liste vide ;
- sinon : $c \neq 0$ et $x = t :: q$; si c est réalisable sans avoir recours à t, faisons-le !
- sinon, on réalise $c - t$ avec la liste q, et on adjoint t au résultat.

Pas de remarques particulières concernant la programmation :

```
exception Somme_impaire ;;
exception Partition_impossible ;;

let rec réaliser_total = function
 | (0,_) -> []
 | (_,[]) -> raise Partition_impossible
 | (c,t::q) when toto c q -> réaliser_total (c,q)
 | (c,t::q) -> t::(réaliser_total (c-t,q)) ;;

let partition l =
 let s = sum l in
 if (s mod 2) = 1
  then raise Somme_impaire
  else réaliser_total (s/2,l) ;;
```

Question 5 • Il n'est pas nécessaire de tester les parties qui contiennent x_n : en effet, dans chaque partition de X en deux parts, l'une des deux ne contient pas x_n. Ceci économise très précisément 50 % des calculs !

Question 6 • $m(0, j) =$ `vrai` s'il existe une partie Y de $\{x_0\}$ vérifiant $\|Y\| = j$; or $\{x_0\}$ n'a que deux parties, l'ensemble vide (de somme nulle) et lui-même (de somme x_0). Donc tous les coefficients de cette ligne ont la valeur `faux`, sauf ceux des colonnes 0 et x_0.

Question 7 • Si $m(i, j) =$ `vrai`, alors $m(i + 1, j) =$ `vrai` à plus forte raison. Sinon, $m(i + 1, j) =$ `vrai` ssi $j \geqslant x_{i+1}$ et $j - x_{i+1}$ est réalisable avec une partie de $\{x_0, \ldots, x_i\}$: donc $m(i + 1, j) = m(i, j - x_{i+1})$.

Question 8 • Le problème de décision admet une réponse positive ssi $m(n - 1, \|X\|/2) =$ `vrai` ; ceci sous réserve que $\|X\|$ soit pair, bien entendu !

Question 9 • Observons la colonne d'indice $\|X\|/2$: le coefficient du bas vaut `vrai`. Déterminons le premier indice i tel que $m(i, \|X\|/2) =$ `vrai` : x_i appartient à la solution. Nous recommençons, en observant cette fois la colonne d'indice $\|X\|/2 - x_i$, et en remontant à partir du coefficient de la ligne i ; de proche en proche, nous reconstituons un sous-ensemble Y de X vérifiant $\|Y\| = \|X\|/2$.

Question 10 • $n = 2$ et $\|X\| = 10$. Pour faciliter la lecture, seuls sont marqués (par un **t**) les coefficients de la matrice égaux à `vrai`.

$i \downarrow \quad j \rightarrow$	0	1	2	3	4	5	6	7	8	9	10
0	**t**					**t**					
1	**t**		**t**			**t**		**t**			
2	**t**		**t**	**t**		**t**		**t**	**t**		**t**

Question 11 • La fonction suivante applique le schéma que nous venons d'exposer aux questions 6 et 7.

```
let partition_dynamique x =
 let (n,v,s) = (list_length x , vect_of_list x , sum x) in
 let m = make_matrix n (s+1) false in
  m.(0).(0) <- true ; m.(0).(v.(0)) <- true ;
  for i = 1 to n-1 do
   for j = 0 to s do
    m.(i).(j) <- m.(i-1).(j) or (j >= a.(i) & m.(i-1).(j-v.(i)))
   done;
  done ;
 m ;;
```

Question 12 • L'argument de la question 5 s'applique tel quel : nous pouvons éviter de calculer la dernière ligne ; mais cette économie est négligeable. Plus intéressant : nous pouvons nous arrêter dès que nous trouvons un indice i tel que $m(i, \|X\|/2) =$ `vrai`. En fait, la véritable économie repose sur la remarque suivante : comme le résultat se trouve dans la case $m(n - 1, \|X\|/2)$ et ne dépend que des cases situées au-dessus et/ou à gauche de celle-ci, il est inutile de remplir les colonnes d'indice supérieur à $\|X\|/2$. Enfin, notons que la première colonne ne sert à rien ...

FIN