Mpsi 1 et 2 - option informatique **Devoir surveillé 1**

Les algorithmes seront écrits en CAML. Ils doivent être suffisamment commentés pour faciliter la tâche du correcteur. Toute affirmation de votre part doit être justifiée.

Problème 1

On considère la fonction suivante dans laquelle n est un entier naturel :

```
let rec f n =
  if n mod 7 = 0 then 1
  else if n mod 8 = 1 then 2
  else f (n/2) + f(n/3 + 1);;
```

Montrez que cette fonction termine.

Problème 2 - Rendu de monnaie

Certaines machines automatiques doivent rendre la monnaie : étant donnée une famille d'entiers naturels non nuls (les valeurs des pièces disponibles) $a_1, \ldots, a_p$, on cherche à écrire un entier naturel quelconque n sous la forme $n = a_1u_1 + \cdots + a_pu_p$ où $u_1, \ldots, u_p$ sont des entiers naturels : dans ce cas, n est dit $(a_1, \ldots, a_p)$-décomposable.

Partie 1 - Avec deux entiers

On se donne deux entiers naturels non nuls a, b tel que $a < b$. n désigne un entier naturel.

Question 1)

a) Montrez que si b ne divise pas n, alors n est (a, b)-décomposable si et seulement si $n \geqslant a$ et $n - a$ est (a, b)-décomposable.

b) Et si b divise n ? n est-il (a, b)-décomposable ?

Question 2)

a) Donnez un algorithme récursif dec2 de paramètres a, b, n qui retourne true si n est (a, b)-décomposable et false sinon, et justifiez-le.

b) Justifiez la terminaison de votre algorithme : vous mettrez en évidence les hypothèses essentielles du théorème du cours.

c) On note $C(n)$ la complexité de l'algorithme. Montrez qu'en dehors des cas de bases, $C(n) = 4 + C(n - a)$. Déduisez-en que $C(n) \leqslant 3 + 4\dfrac{n}{a}$.

Question 3)

a) Écrivez un algorithme récursif sol2 de paramètres a, b, n qui retourne un couple $(u, v) \in \mathbb{N}^2$ tel que $n = au + bv$ si n est (a, b)-décomposable et le couple $(-1, -1)$ sinon.

b) Justifiez sa terminaison, sa correction et donnez sa complexité en fonction de n.

Partie 2 - Avec trois entiers

On se donne trois entiers naturels a, b, c tel que $a < b < c$.

À l'aide de l'algorithme sol2, écrivez un algorithme sol3 de paramètres a, b, c, n qui donne une solution $(u, v, w) \in \mathbb{N}^3$ telle que $n = au + bv + cw$ si n est (a, b, c)-décomposable et $(-1, -1, -1)$ sinon.

Justifiez la terminaison de votre algorithme, sa correction et donnez sa complexité en fonction de n.

Problème 3 - Décomposition en facteurs premiers

Question 1) On considère la fonction décrite ci-dessous

```
let rec f d n =
  if n mod d = 0 then d
  else f (d+1) n;;
```

n et d sont deux entiers naturels tels que $2 \leqslant d \leqslant n$: montrez que `p = f d n` est le plus petit diviseur de `n` au moins égal à `d`.

Montrez que `f 2 n` est le plus petit diviseur premier de `n`.

Question 2) Écrivez un algorithme récursif `decpr n` de paramètre un entier naturel $n \geqslant 1$ qui calcule la liste des diviseurs premiers de n, rangés par ordre croissant, comme dans les exemples qui suivent :

`decpr 1 = []`, `decpr 11 = [11]` et `decpr 84 = [2; 2; 3; 7]`.

Question 3) La décomposition en facteurs premiers peut contenir plusieurs fois consécutives le même entier. On souhaite une réponse différente.

Écrivez un algorithme récursif `decpr2 n` de paramètre un entier naturel $n \geqslant 1$ qui calcule la liste des couples (p, a) où p est un diviseur premier de n et a le nombre de fois où il divise n (*i.e.* son exposant dans la décomposition en facteurs premiers de n), rangés dans l'ordre croissant des diviseurs premiers, comme dans les exemples suivants :

`decpr2 1 = []`, `decpr2 7 = [(7,1)]`, `decpr2 1176 = [(2,3); (3,1); (7,2)]`

Vous pouvez vous servir, si vous le souhaitez, des fonctions `fst` et `snd` qui associent à un couple (a, b) le premier élément a et le second élément b respectivement, ou alors utiliser un filtrage (les couples peuvent être des motifs de filtrage).

Problème 4 - Tri par sélection

On calcule les complexités en fonction de la longueur n de la liste paramètre.

Question 1) Écrivez une fonction récursive `retire x li` de paramètres un objet `x` et une liste `li`, qui calcule la liste obtenue à partir de `li` en supprimant la première apparition de `x` dans la liste si elle existe, ou la liste `li` elle-même dans le cas contraire.

Exemples : `retire 2 [3; 5; 3; 6] = [3; 5; 3; 6]`, `retire 3 [1; 3; 5; 3; 6] = [1; 5; 3; 6]`

Quelle est sa complexité ?

Question 2) Écrivez une fonction récursive `minimum li` qui calcule le plus petit élément de la liste `li`. Quelle est sa complexité ?

Question 3) Écrivez une fonction récursive `tri li` qui calcule la liste obtenue à partir de `li` en ordonnant ses éléments par ordre croissant, en vous servant des deux fonctions précédentes.

Justifiez que votre fonction est correcte par récurrence sur n, qu'elle termine et donnez sa complexité en fonction de n.

Devoir surveillé 1 - Corrigé

Problème 1

Soit n un entier en dehors des cas de base, c'est-à-dire tel que n n'est pas divisible par 7 et non congru à 1 modulo 8.

Alors en particulier, $n \geqslant 2$ (car $n = 0$ est divisible par 7 et $n = 1$ est congru à 1 modulo 8) :
- si $n = 2$, alors $n/2 = 1 < n$ et $n/3 + 1 = 1 < n$;
- si $n \geqslant 3$, alors $n = 3q + r$, avec $q \geqslant 1$ et $r \in \{0, 1, 2\}$, donc $n/3 + 1 = q + 1$, donc $n - (n/3 + 1) = 3q + r - (q + 1) = 2q - 1 + r \geqslant 1 + r > 0$, donc $(n/3 + 1) < n$; de plus, comme $n > 0$, alors $n/2 < n$.

Dans tous les cas, les deux appels récursifs se font sur des entiers naturels strictement inférieurs à celui de l'appel initial, donc on est dans les conditions du théorème du cours. Les entiers paramètres dans les appels récursifs décroissent strictement dans l'ensemble bien fondé $\mathbb{N}$, donc l'algorithme termine.

Problème 2

Partie 1

Question 1)

a) Si $n \geqslant a$ et $n - a$ est (a, b)-décomposable, alors il existe u, v entiers naturels tels que $n - a = au + bv$, donc $n = a(u + 1) + bv$, donc n est (a, b)-décomposable.
Réciproquement, si n est (a, b)-décomposable, alors il existe u, v entiers naturels tels que $n = au + bv$; comme b ne divise pas n, on a forcément $u \geqslant 1$ donc $n \geqslant a$ et $n - a = a(u - 1) + bv$ est (a, b)-décomposable.

b) Réponse évidente : b divise n donc $n = bq$ où $q \in \mathbb{N}$, donc $n = a \times 0 + bq$, donc n est (a, b)-décomposable.

Question 2)

a) Première solution à coups de "if then else" :

```
let rec dec2 a b n =
  if (n mod b = 0) then true
  else if n < a then false
  else dec2 a b (n-a);;
```

Autre solution utilisant l'évaluation paresseuse des booléens :

```
let rec dec2 a b n =
  (n mod b = 0) || ( (n >= a) && dec2 a b (n-a) );;
```

Dans les deux cas, si b divise n, alors $n = a \times 0 + b \times v$ donc n est (a, b)-décomposable ; sinon on utilise l'équivalence précédente, ce qui justifie la correction de l'algorithme.

b) Soit $B = \{n \in \mathbb{N} \ / \ b|n \text{ ou } n < a\}$ (cas de base). En dehors des cas de base, on a $n \geqslant a$ donc l'appel récursif a pour argument un entier naturel strictement inférieur à celui de l'appel initial, donc comme $\mathbb{N}$ est un ensemble bien fondé, l'algorithme termine.

c) Soit $C(n)$ le coût du calcul de `dec2 a b n` : si $n \in B$, alors $C(n) \leqslant 3$, sinon $C(n) = 4 + C(n - a)$, le terme 4 provenant du calcul du reste, des deux tests et de la soustraction $n - a$. On déduit facilement par induction que $C(n) \leqslant 3 + 4 \times (n/a) \leqslant 3 + \dfrac{4n}{a}$, donc la complexité de l'algorithme est linéaire $O(n/a)$:
soit $\mathscr{P}(n)$ la proposition « $C(n) \leqslant 3 + \dfrac{4n}{a}$ » ;
pour tous les cas de base, $\mathscr{P}(n)$ est vraie ;
si $\mathscr{P}(n - a)$ est vraie pour n en dehors des cas de base, alors comme $C(n) \leqslant 4 + C(n - a)$, on a $C(n) \leqslant 4 + 3 + \dfrac{4(n - a)}{a} = 3 + \dfrac{4n}{a}$ donc $\mathscr{P}(n)$ est vraie ;
d'après le principe d'induction, pour tout $n \in \mathbb{N}$, $\mathscr{P}(n)$ est vraie.

Question 3)

a) Il suffit d'adapter l'algorithme précédent :

```
let rec sol2 a b n =
  if (n mod b = 0) then (0, n/b)
  else if n < a then (-1,-1)
  else
    let (u,v) = sol2 a b (n-a) in
    if u = -1 then (u,v) else (u+1,v);;
```

b) La terminaison se justifie de la même façon que précédemment, la correction est conséquence de la réponse à la question 1 a, qui donne un moyen explicite de fabriquer une solution pour n à partir d'une solution pour $n-a$. La complexité est à peine plus élevée : $C(n) \leqslant 4 + \dfrac{7n}{a}$, mais l'ordre de grandeur n'a pas changé, cela reste linéaire.

Partie 2

On essaye d'abord d'écrire n sous la forme $bv+cw$: si on peut, alors $n = a\times 0+bv+cw$ est (a,b,c)-décomposable. Sinon on a l'équivalence : n est (a,b,c)-décomposable si et seulement si $n \geqslant a$ et $n-a$ est (a,b,c)-décomposable (même démonstration).

```
let rec sol3 a b c n =
  let (v,w) = sol2 b c n in
  if v > -1 then (0,v,w)
  else
  if n < a then (-1,-1,-1)
  else let (u,v,w) = sol3 a b c (n-a) in
  if u = -1 then (u,v,w) else (u+1,v,w);;
```

La terminaison est justifiée par le fait que sol2 termine et qu'en dehors des cas de base, l'appel récursif a pour argument un entier naturel strictement inférieur à celui de l'appel initial, donc comme $\mathbb{N}$ est un ensemble bien fondé, l'algorithme termine.

On note $C_2(n)$ le coût de l'appel à sol2 b c n, qui est en $O(n/b)$ et $C_3(n)$ le coût du calcul de sol3 a b c n.

Dans les cas de base, on a $C_3(n) \leqslant 2+C_2(n) \leqslant 7+\dfrac{7n}{b}$ et en dehors des cas de base, $C_3(n) = C_2(n)+5+C_3(n-a) \leqslant 9 + \dfrac{7n}{b} + C_3(n-a)$, donc par induction, on montre que $C_3(n) \leqslant 7 + \dfrac{7n}{b} + \dfrac{9n}{a} + \dfrac{7n^2}{ab} = O(n^2)$.

Soit $\mathscr{P}(n)$ la proposition « $C_3(n) \leqslant 7 + \dfrac{7n}{b} + \dfrac{9n}{a} + \dfrac{7n^2}{ab}$ »

$\mathscr{P}(n)$ est vraie pour tous les cas de base

Si $\mathscr{P}(n-a)$ est vraie en dehors des cas de base, alors

$$C_3(n) = C_2(n) + 5 + C_3(n-a) \leqslant 9 + \frac{7n}{b} + C_3(n-a) \leqslant 9 + \frac{7n}{b} + 7 + \frac{7(n-a)}{b} + \frac{9(n-a)}{a} + \frac{7(n-a)^2}{ab}$$

donc $C_3(n) \leqslant 9 + \dfrac{7n}{b} + 7 + \dfrac{7n}{b} - \dfrac{7a}{b} + \dfrac{9n}{a} - 9 + \dfrac{7n^2}{ab} - \dfrac{14n}{b} + \dfrac{7a}{b} \leqslant 7 + \dfrac{9n}{a} + \dfrac{7n^2}{ab} \leqslant 7 + \dfrac{7n}{b} + \dfrac{9n}{a} + \dfrac{7n^2}{ab}$

donc $\mathscr{P}(n)$ est vraie.

D'après le principe d'induction, pour tout $n \in \mathbb{N}$, $\mathscr{P}(n)$ est vraie.

Problème 3

Question 1) On procède par récurrence descendante sur d :

si $d = n$, alors d divise n donc la fonction retourne la valeur n et n est bien le plus petit diviseur de n au moins égal à n ;

si q = f (d+1) n est le plus petit diviseur de n au moins égal à $d+1$, alors l'appel p = f d n se déroule comme suit :

— si d divise n, alors $p = d$, donc p est bien le plus petit diviseur de n au moins égal à d ;
— sinon d ne divise pas n, donc le plus petit diviseur de n au moins égal à d est au moins égal à $d+1$, donc c'est q ; or c'est justement ce que retourne l'appel de fonction `p = f d n` grâce à l'appel récursif.

Donc on a montré par récurrence que pour tout d compris entre 2 et n, `p = f d n` est le plus petit diviseur de n au moins égal à d.

Posons maintenant `p = f 2 n` : p est donc le plus petit diviseur de n au moins égal à 2.

Si p n'était pas premier, alors il aurait un diviseur propre (*i.e.* au moins égal à 2) strictement inférieur e : e diviserait alors n en étant plus petit que p, ce qui contredit la définition de p.

Question 2)

```
let rec decpr n =
  if n = 1 then []
  else
    let d = f 2 n in
    d :: decpr (n/d);;
```

Question 3)

```
let rec decpr2 n =
  if n = 1 then []
  else
    let d = f 2 n in
    let li = decpr2 (n/d) in
    match li with
    | [] -> [(d,1)]
    | (p,a) :: q -> if p = d then (p,a+1) :: q
                    else (d,1) :: li;;
```

Problème 4

Question 1)

```
let rec retire x l =
  match li with
  | [] -> []
  | a :: q -> if a = x then q else a :: retire x q;;
```

En notant $R(n)$ la complexité de cette fonction, on a $R(0) = 1$ et si $n \geqslant 1$, $R(n) \leqslant 4 + R(n-1)$, donc $R(n) = O(n)$: la complexité est linéaire.

Question 2)

```
let rec minimum li =
  match li with
  | [a] -> a
  | a :: q -> let m = minimum q in
              if m < a then m else a;;
```

De même, en notant $M(n)$ la complexité de cette fonction, $M(1) = 1$ et si $n \geqslant 2$, $M(n) = 3 + M(n-1)$ donc $M(n) = O(n)$: encore une complexité linéaire.

Question 3)

```
let rec tri li =
  match li with
  | [] -> []
  | [a] -> [a]
```

```
  | _ -> let m = minimum li in
         let li' = retire m li in
         let li'' = tri li' in
         m :: li'';;
```

On note $T(n)$ la complexité de la fonction tri. Alors on constate que $T(0) = 1$, $T(1) = 2$ et si $n \geqslant 2$, $T(n) = M(n) + R(n) + T(n-1) + 1 = T(n-1) + O(n)$ donc $T(n) = O(n^2)$: la complexité est quadratique.

La fonction tri termine, car les deux appels aux deux autres fonctions terminent et l'appel récursif se fait sur une liste de longueur strictement inférieure à celle de l'appel initial, donc comme $\mathbb{N}$ est un ensemble bien fondé, la fonction termine.

De plus, elle est correcte, car

- elle trie bien par ordre croissant les listes de longueur 0 ou 1 ;
- si elle trie les listes de longueur $n-1$, alors sur une liste de longueur n, elle commence par retrouver son minimum, qu'elle retire, on obtient une liste de longueur $n-1$ qui est triée par hypothèse de récurrence, puis on remet en tête le plus petit élément de la liste, donc la liste obtenue contient les mêmes éléments que la liste initiale et elle est donc rangée par ordre croissant.