

Corrigé de la composition Hors Classement

Jean-Jacques Lévy

16 décembre 1998

Ensembles et relations en Java

Question 1 On représente une partie par la liste de ses éléments.

```
class Partie {  
  
    int    elt;  
    Partie suivant;  
  
    Partie (int n, Partie x) {elt = n; suivant = x; }  
  
    static boolean estEnOrdre (Partie x) {  
        return x == null || x.suivant == null ||  
            (x.elt <= x.suivant.elt &&  
             estEnOrdre (x.suivant));  
    }  
}
```

Question 2 Les deux fonctions suivantes calculent en temps $O(1 + \text{Card } X + \text{Card } Y)$.

```
static Partie union (Partie x, Partie y) {  
    if (x == null)  
        return y;  
    if (y == null)  
        return x;  
    if (x.elt < y.elt)  
        return new Partie (x.elt, union (x.suivant, y));  
    if (x.elt > y.elt)  
        return new Partie (y.elt, union (x, y.suivant));  
    else  
        return new Partie (x.elt, union (x.suivant, y.suivant));  
}  
  
static Partie inter (Partie x, Partie y) {  
    if (x == null || y == null)  
        return null;  
    if (x.elt == y.elt)  
        return new Partie (x.elt, inter (x.suivant, y.suivant));  
    else  
        if (x.elt < y.elt)  
            return inter (x.suivant, y);  
        else  
            return inter (x, y.suivant);  
}
```

Question 3 On doit trouver les listes $R_j^{-1} = \{i \mid (i,j) \in R\}$ pour $0 \leq j < n$. Il faut commencer par les i grands pour avoir les listes R_j^{-1} en ordre croissant. On parcourt donc tout le tableau s et tous les éléments de R . Donc le temps est en $O(1 + n + \text{Card } R)$, qui est dans le pire cas en $O(n^2)$.

```
final static int N = 100; // taille max de E

static Partie[] reverse (Partie[] r) {
    Partie[] s = new Partie [N];
    for (int i = s.length - 1; i >= 0; --i)
        for (Partie p = r[i]; p != null; p = p.suivant)
            s[p.elt] = new Partie (i, s[p.elt]);
    return s;
}
```

Question 4 Soit $T = RS$. On doit trouver les listes $T_i = \{k \mid (i,j) \in R, (j,k) \in S\}$ pour $0 \leq i < n$. Donc $T_i = \cup\{S_j \mid (i,j) \in R\}$ où $S_j = \{k \mid (j,k) \in S\}$. On parcourt tout le tableau R et pour chaque paire dans R , on fait une union entre deux classes de longueur n dans le pire cas. Le tout est donc en $O(n^3)$.

```
static Partie[] comp (Partie[] r, Partie[] s) {
    Partie[] t = new Partie [r.length];
    for (int i = 0; i < r.length; ++i) {
        t[i] = null;
        for (Partie p = r[i]; p != null; p = p.suivant)
            t[i] = union (t[i], s[p.elt]);
    }
    return t;
}
```

Ensembles et relations en Ocaml

```
let rec est_en_ordre x = match x with
    [] -> true
  | [v] -> true
  | v1 :: (v2 :: x2 as x1) -> (v1 <= v2) && est_en_ordre x1;;
```

```
let rec union x y = match x, y with
    [], y -> y
  | x, [] -> x
  | v::x1, w::y1 ->
    if v < w then
        v:: union x1 y
    else if v > w then
        w :: union x y1
    else
        v :: union x1 y1 ;;
```

```
let rec inter x y = match x, y with
    [], y -> []
  | x, [] -> []
  | v::x1, w::y1 ->
    if v < w then
```

```

        inter x1 y
    else if v > w then
        inter x y1
    else
        v :: inter x1 y1 ;;

let reverse r =
    let s = Array.make n [] in
    for i = Array.length r - 1 downto 0 do
        List.iter
            (function j -> s.(j) <- i :: s.(j) )
            r.(i);
        done;
    s;;

let comp r s =
    let t = Array.make (Array.length r) [] in
    for i = 0 to Array.length r - 1 do
        List.iter
            (function k -> t.(i) <- union t.(i) s.(k) )
            r.(i);
        done;
    t;;

```

Fonctions récursives primitives

Question 5 $x \times y$ est en $O(xy)$, $x!$ en $O(x!)$, $x - 1$ en $O(x)$, $x - y$ en $O(x)$, $\lfloor x/2 \rfloor$ en $O(x)$.

```

// multiplication x y
z = 0;
for x do
    for y do
        z = z+1;

// predecesseur x
z = 0;
for x do {
    t = z;
    t = t + 1;
    if (t < x)
        z = z+1;
    else
        z = z;
}

// moins x y
z = 0;
for x do
    if (x > y) {
        z = z + 1;
        y = y + 1;
    } else
        z = z;

// div2 x

```

```

z = 0;
y = 1;
for x do {
  if (y < x)
    z = z+1;
  else
    z = z;
  y = y+1; y = y+1;
}

// fact x
z = 0;
n = 0;
z = z+1;
n = n+1;
for x do {
  // z = mult z n
  z' = 0;
  for z do
    for n do
      z' = z' + 1;
  z = z';
  n = n+1;
}

```

Question 6 L'instruction `for` termine toujours. Donc il en est de même pour tout programme Stupa. Mais en Borobudur, le programme suivant boucle.

```

x = 0;
y = x + 1;
while (x < y)
  x = x;

```

Question 7 Par exemple, $f(0) = 0$, $f(x+1) = f(x+1)$. Un autre exemple est la définition récursive classique de fibonacci, ou d'Ackermann.

Question 8 Les cas 1-4 sont triviaux. Pour le cas 5, supposons que par récurrence, on ait trouvé les programmes Stupa représentant h et g . Alors le programme pour f est:

```

y1 = x1; y2 = x2; ... yn = xn;
h;
x1 = 0;
for y1 do {
  y1 = x1;
  x2 = y2; ... xn = yn;
  xn+1 = x0;
  g;
  x1 = y1 + 1;
}

```