

## Corrigé du DS type CCINP

**Q20.**

**Q21.**

```
type arbre = Vide | Noeud of fraction * arbre * arbre;;
```

**Q22.** Si  $n/d$  est un noeud du niveau 0, alors  $n = d = 1$  donc  $n \wedge d = 1$ . Supposons que tous les noeuds  $n/d$  de l'étage  $k$  sont tels que  $n \wedge d = 1$ . Un noeud de l'étage  $k+1$  est du type  $n/(n+d)$  ou  $(n+d)/n$  avec  $n/d$  un noeud du niveau  $k$ . Tout diviseur de  $n$  et de  $n+d$  est un diviseur de  $n$  et de  $n+d-n = d$  donc est un diviseur de  $n$  et de  $d$  donc vaut 1 ; c'est donc que  $n \wedge (n+d) = 1$  donc que les noeuds du niveau  $k+1$  du type  $n'/d'$  vérifient  $n' \wedge d' = 1$ . Par récurrence, pour tout noeud  $n/d$  de l'arbre, on a  $n \wedge d = 1$ .

**Q23.**

```
let rec pgcd a b = if a=b then a else if a<b then pgcd a (b-a) else pgcd (a-b) b;;
```

**Q24.**

```
let fraction n d = if (n<0 && d>0) || (n>0 && d<0) then failwith "erreur de signe";
  if n>0 && d>0 then let g=pgcd n d in {n=n/g; d=d/g}
  else let g=pgcd (-n) (-d) in {n=-n/g; d=-d/g};;
```

**Q25.** Supposons que ces noeuds ont le même fils gauche. Alors  $k/(k+l) = n/(n+d)$  donc  $k(n+d) = n(k+l)$ . Comme  $n \wedge (n+d) = 1$  et  $n+d$  divise  $n(k+l)$ ,  $n+d$  divise  $k+l$ . De même,  $n$  divise  $k(n+d)$  et  $n \wedge n+d = 1$  donc  $n$  divise  $k$ . Par symétrie des rôles de  $(n, d)$  et  $(k, l)$ ,  $k$  divise  $n$  et  $k+l$  divise  $n+d$ . On a donc  $k = n$  et  $k+l = n+d$  donc  $k = n$  et  $l = d$ . Supposons que ces noeuds ont le même fils droit. Alors  $(n+d)/d = (k+l)/l$  donc  $(k+l)d = (n+d)l$  donc  $d$  divise  $(n+d)l$  donc, étant premier avec  $n+d$ ,  $d$  divise  $l$  et  $(n+d)$  divise  $(k+l)d$  donc, étant premier avec  $d$ ,  $n+d$  divise  $k+l$ ; par symétrie,  $k+l$  divise  $n+d$  et  $l$  divise  $d$  donc  $d = l$  puis  $n = k$ .

**Q26.** Supposons que  $N = n/d$  avec  $n \wedge d = 1$ . Le fils gauche de  $N$  est  $n/(n+d) = \frac{n/d}{n/d+1} = \frac{N}{N+1}$ . Soit  $k \in \mathbb{N}$  tel que le noeud provenant d'une suite de  $k$  fils gauches de  $N$  est  $\frac{N}{1+kN} = \frac{n}{d+kn}$  et  $n \wedge (d+kn) = n \wedge d = 1$ . Le fils gauche de ce noeud est donc  $\frac{n}{d+kn+n} = \frac{n/d}{1+(k+1)n/d} = \frac{N}{1+(k+1)N}$  donc le noeud issu de  $k+1$  fils gauches issus de  $N$  est  $N/(1+(k+1)N)$ . Par récurrence le noeud issu de  $k$  fils gauches de  $N$  est  $N/(1+kN)$ .

Le noeud issu d'une suite de  $k$  fils droits est  $N+k$ .

**Q27.**  $v_0 = 0, v_1 = 1, v_2 = 1/2, v_3 = 2, v_4 = 1/3, v_5 = 3/2, v_6 = 2/3, v_7 = 3$  (et  $v_8 = 1/4$  si l'on demande les huit premiers termes à partir du rang 1).

**Q28.** Supposons que  $n+d = 2$ . Alors  $n = d = 1$  et  $n/d = 1$  est la racine de l'arbre. Soit  $N \geq 2$ . Supposons que, pour tout  $n, d$  premiers entre eux tels que  $n+d \leq N$ ,  $n/d$  apparaît. Supposons que  $n+d = N+1$ . Supposons que  $n > d$ . Posons  $k = n-d$  et  $l = d$ . Alors  $k+l = n < n+d = N+1$ . Par hypothèse de récurrence,  $k/l$  apparaît dans l'arbre donc a pour fils droit  $(k+l)/l = n/d$  donc  $n/d$  apparaît dans l'arbre. Supposons que  $n < d$ . Posons  $l = d-n$  et  $k = n$ . Alors  $k+l = d < n+d = N+1$  donc  $k/l$  apparaît dans l'arbre et a pour fils gauche  $k/(k+l) = n/d$  donc  $n/d$  apparaît dans l'arbre. Si  $n = d$  alors, comme  $n \wedge d = 1$ ,  $n = d = 1$  donc  $n/d$  est la racine de l'arbre.

Par récurrence, pour tout  $n, d$  premiers entre eux,  $n/d$  apparaît dans l'arbre.

**Q29.** Soit  $n, d$  premiers entre eux. Si  $n = d$ ,  $n = d = 1$  donc  $n/d$  est la racine de l'arbre donc n'apparaît qu'une fois. Soit  $N \geq 2$ . Supposons que, si  $n+d \leq N$ ,  $n/d$  n'apparaît qu'une fois. Supposons que  $n/d = N+1$ .  $n/d \neq 1$  donc  $n/d$  n'est pas la racine donc a un père  $k/l$ . On a donc  $n = k$  et  $d = k+l$  ou  $n = k+l$  et  $d = l$ . De plus,  $n \neq d$  (sinon  $n = d = 1$ ). Supposons que  $n < d$ . Alors  $k = n$  et  $d = k+l$ . Comme  $k+l \leq N$ ,  $k/l$  n'apparaît qu'une fois donc  $n/d$  n'apparaît qu'une fois (comme fils gauche de  $k/l$ ). Supposons que  $n > d$ . Alors  $n = k+l$  et  $d = l$ . Comme  $k/l$  n'apparaît qu'une fois (puisque  $k+l \leq N$ ),  $n/d$  n'apparaît qu'une fois (comme fils droit de  $k/l$ ).

Par récurrence, pour tout  $n, d$  premiers entre eux,  $n/d$  n'apparaît qu'une fois dans l'arbre.

**Q30.** Soit  $q \in \mathbb{Q}_+$ . Si  $q = 0$ ,  $q = v_0$ . Si  $q > 0$ , il existe un unique  $(n, d) \in (\mathbb{N}^*)^2$  tels que  $n \wedge d = 1$  et  $q = n/d$ . D'après les questions 28 et 29,  $n/d$  n'apparaît qu'une seule fois dans l'arbre donc il existe un unique  $i \in \mathbb{N}^*$  tel que  $q = v_i$ . Ceci prouve que  $(v_i)_{i \in \mathbb{N}}$  est une bijection de  $\mathbb{N}$  dans  $\mathbb{Q}_+$ .

**Q31.** Montrons, par récurrence sur  $k \in \mathbb{N}$ , que les noeuds de profondeur  $k$  vont de  $2^k$  à  $2^{k+1} - 1$ . Le noeud de profondeur 0 est indexé par  $1 = 2^0 = 2^{0+1} - 1$ . Supposons que les noeuds de profondeurs  $k \in \mathbb{N}$  vont de  $2^k$  à  $2^{k+1} - 1$ . Les noeuds de profondeur  $k+1$  sont en nombre double de celui de profondeur  $k$ . Il y en a donc  $2 \times (2^{k+1} - 1 - 2^k + 1) = 2 \times 2^k = 2^{k+1}$ . Comme ils sont indexés à partir de  $2^{k+1}$ , ils sont indexés de  $2^{k+1}$  à  $2^{k+1} + 2^{k+1} - 1 = 2^{k+2} - 1$ . Ceci prouve, par récurrence, que les noeuds de profondeur  $k$  sont indexés de  $2^k$  à  $2^{k+1} - 1$ .

Soit  $k$  une profondeur fixé. Les fils du premier noeud de profondeur  $k$  sont les deux premiers noeuds de profondeur  $k+1$  : les fils du noeud d'indice  $2^k$  sont en indice  $2^{k+1}$  et  $2^{k+1} + 1$ , c'est-à-dire  $2 \times 2^k$  et  $2 \times 2^k + 1$ . Supposons qu'un noeud de profondeur  $k$  d'indice  $i < 2^{k+1} - 1$  a ses fils d'indice  $2i$  et  $2i+1$ . Les fils de son voisin  $i+1$  sont les deux voisins qui suivent  $2i+1$  ; les fils du noeud d'indice  $i+1$  sont donc  $2i+2$  et  $2i+3$  c'est-à-dire  $2(i+1)$  et  $2(i+1)+1$ . Par récurrence, pour toute profondeur  $k$ , pour tout noeud  $i$  de profondeur  $k$ , les fils de ce noeud ont pour indice  $2i$  et  $2i+1$ .

**Q32.**  $s_0 = 0, s_1 = 1, s_2 = s_1 = 1, s_3 = s_1 + s_2 = 2, s_4 = s_2 = 1, s_5 = s_2 + s_3 = 3, s_6 = s_3 = 2, s_7 = s_3 + s_4 = 3, s_8 = s_4 = 1, s_9 = s_4 + s_5 = 4$ .

**Q33.**

```
let rec stern i = if i<=1 then i else if i mod 2=0 then stern (i/2) else let k=i/2 in stern k + stern (k+1);;
```

**Q34.**  $v_0 = 0 = \frac{s_0}{s_1}$  et  $v_1 = 1 = \frac{s_1}{s_2}$  et  $s_0 \wedge s_1 = s_1 \wedge s_2 = 1$ . Soit  $n \in \mathbb{N}$ . Supposons que, pour tout  $i \leq n$ ,  $v_i = \frac{s_i}{s_{i+1}}$  et  $s_i \wedge s_{i+1} = 1$ . Supposons que  $n+1$  est pair :  $n+1 = 2k$ . D'après la question 31,  $v_{n+1} = v_{2k}$  est la valeur du noeud qui est le fils gauche du noeud qui a la valeur  $v_k$ ; par hypothèse de récurrence  $v_k = \frac{s_k}{s_{k+1}}$  et  $s_k \wedge s_{k+1} = 1$ . On en déduit que  $v_{2k} = \frac{s_k}{s_k + s_{k+1}} = \frac{s_{2k}}{s_{2k+1}}$ ; en outre,  $s_{2k} \wedge s_{2k+1} = s_k \wedge (s_k + s_{k+1}) = s_k \wedge s_{k+1} = 1$ . Supposons que  $n+1$  est impair :  $n+1 = 2k+1$ . D'après la question 31,  $v_{n+1} = v_{2k+1}$  est la valeur du noeud qui est le fils droit du noeud qui a la valeur  $v_k = \frac{s_k}{s_{k+1}}$  donc  $v_{2k+1} = \frac{s_k + s_{k+1}}{s_{k+1}} = \frac{s_{2k+1}}{s_{2k+2}}$ ; en outre  $s_{2k+1} \wedge s_{2k+2} = (s_k + s_{k+1}) \wedge s_{k+1} = s_k \wedge s_{k+1} = 1$ . Ceci prouve, par récurrence, que, pour tout  $i \in \mathbb{N}^*$ ,  $v_i = \frac{s_i}{s_{i+1}}$  (et  $s_i \wedge s_{i+1} = 1$ ).

**Q35.** Dans ce cas,  $i = 2k$ ,  $v_i = \frac{n}{d}$  est le fils de  $k/l$  avec  $k+l = d$ ,  $k = n$  donc  $l = d - n$  et  $v_{i+1} = \frac{k+l}{l} = \frac{n+d-n}{d-n} = \frac{d}{d-n}$  donc  $\frac{1}{v_{i+1}} = 1 - \frac{n}{d} = 1 - v_i$ . En outre,  $0 < n < d$  donc  $0 < v_i < 1$  donc  $\lfloor v_i \rfloor = 0$ . On a donc  $\frac{1}{v_{i+1}} = 1 + 2\lfloor v_i \rfloor - v_i$  donc  $v_{i+1} = f(v_i)$ .

**Q36.** Dans ce cas,  $v_i$  est une succession de fils droits et est à profondeur  $k$ ; d'après la question 26,  $v_i = 1 + k$ . Son successeur est la succession de fils gauche et est à profondeur  $k+1$ . D'après la question 26,  $v_{i+1} = \frac{1}{1+k+1} = \frac{1}{k+2}$ . Remarquons que  $f(v_i) = f(1+k) = \frac{1}{1+2\lfloor k+1 \rfloor - k - 1} = \frac{1}{1+k+1} = v_{i+1}$ .

**Q37.**

```
let rec chemin f = if n.f=d.f then [f] else
  if n.f<d.f then rev (G::(chemin {n=n.f;d=d.f-n.f}))
  else rev (D::(chemin {n=n.f-d.f;d=d.f}));;
```

**Q38.**

```
let noeud l = let rec aux f l = match l with
  | [] -> f;;
  | G::q -> aux {n=n.f; d=n.f+d.f} q
  | D::q -> aux {n=n.f+d.f; d=d.f} q in aux {n=1;d=1} l;;
```

**Q39.**

```
let ancetre f1 f2 =
  let rec aux l1 l2 = match l1,l2 with
    | [],_ -> []
    | _,[] -> []
    | t1::q1,t2::q2 -> if t1=t2 then t1::(aux q1 q2) else []
  in let c1=chemin f1 and c2=chemin f2 in let c=aux c1 c2 in
  noeud c;;
```

**Q40.** Le chemin de  $v_p$  à  $v_i$  est  $[G; D; \dots; D]$ . Celui de  $v_p$  à  $v_{i+1}$  est  $[D; G; \dots; G]$ .

**Q41.** Supposons que  $v_p = n/d$ . Alors le fils gauche de  $v_p$  est  $n/(n+d)$ . En vertu de la question 26, le successeur de  $k' - 1$  fils droits est  $\frac{n}{n+d} + k' - 1$ . Le fils droit de  $v_p$  est  $\frac{n+d}{d}$ . En vertu de la question 26, le successeur de  $k' - 1$  fils gauches est  $\frac{\frac{n+d}{d}}{1+(k'-1)\frac{n+d}{d}}$ . On a donc  $v_i = \frac{n}{n+d} + k' - 1$  et  $v_{i+1} = \frac{1}{\frac{d}{n+d} + k' - 1}$ . Comme  $\lfloor v_i \rfloor = k' - 1$ ,  $1 + 2\lfloor v_i \rfloor - v_i = 1 + 2k' - 2 - \frac{n}{n+d} - k' + 1 = k' - \frac{n}{n+d} = k' - 1 + \frac{d}{n+d} = \frac{1}{v_{i+1}}$  donc  $v_{i+1} = \frac{1}{1+2\lfloor v_i \rfloor - v_i} = f(v_i)$ .

## Partie 2 (sujet 2014)

### Question III. 1

L'appel "tri exemple" va créer l'appel "selection exemple" qui va appeler "aux 3 [1;4;2]". Le programme va ensuite calculer "(m,r)=aux 1 [4;2]" et renvoyer "(m,3 : r)". Or "aux 1 [4;2]" calcul "(m0,r0)=aux 1 [2]" et renvoie "(m0,4 : r0)". Enfin, "aux 1 [2]" calcule "aux 1 []" qui est (1,[]) et renvoie (1,[2]). Par conséquent aux 1 [4;2] renvoie (1,[4;2]), puis aux 3 [1;4;2] renvoie (1, [3;4;2]). Ainsi, "selection exemple" renvoie (1,[3;4;2]). La fonction "tri" va ensuite calculer "selection [3;4;2]", résultat auquel elle ajoutera 1 en tête de liste. Or "selection [3;4;2]" va appeler "aux 3 [4;2]" puis "aux 3 [2]" puis "aux 2 []" qui renvoie (2,[]), ce qui crée le renvoi, pour "aux 3 [2]" de (2,[3]), qui crée ensuite le renvoi pour "aux 3 [4;2]" de (2,[4;3]).

La fonction "tri [4;3]" va alors appeler "selection [4;3]" qui va appeler "aux 4 [3]" puis "aux 3 []" qui renvoie (3,[]) puis (3,[4]).

La fonction "tri [4]" va alors appeler "selection [4]" qui appelle "aux 4 []" qui renvoie (4,[]).

Ainsi, le dernier appel de "tri" renvoie [4], le précédent renvoie [3;4], celui d'avant [2;3;4] et l'appel initial renvoie [1;2;3;4].

### Question III. 2

Montrons, par récurrence sur la taille de la liste  $a = \langle a_1, \dots, a_p \rangle$ , que "aux c a" renvoie un couple (m,r) tel que, en notant  $r = \langle r_1, \dots, r_m \rangle$ ,

(i)  $\forall 1 \leq i \leq p$ ,  $\delta(a_i, m) + \text{card}(a_i, r) = \delta(a_i, c) + \text{card}(a_i, a)$  et  $\delta(c, m) + \text{card}(c, r) = 1 + \text{card}(c, a)$ .

(ii)  $m = p$ .

(iii)  $\forall 1 \leq i \leq m$ ,  $m \leq r_i$  et  $m \leq c$ .

Si  $a$  est la liste vide,  $r$  est la liste vide, donc  $m = p = 0$ . Comme  $p = 0$ , les débuts des propriétés (i) et (iii) sont des tautologies pour  $1 \leq i \leq p$ ; de plus, on a bien  $m \leq c = m$  et  $\delta(c, m) + \text{card}(c, r) = 1 + 0 = 1 + \text{card}(c, a)$ .

Supposons le résultat pour toute liste de taille  $p$ . Soit  $a = \langle a_1, \dots, a_{p+1} \rangle$ . On étudie la fonction "aux c t : q"

Notons  $q = \langle a_2, \dots, a_{p+1} \rangle$  et  $t = a_1$ .

Supposons que  $c < t = a_1$ . Alors on pose (m0,r0)=aux c q. Par hypothèse de récurrence, la taille de r0 est celle de q, à savoir  $p$ . On note  $r0 = \langle r_2, \dots, r_{p+1} \rangle$ .

On a alors  $m = m0$  et  $r = t :: r0 = \langle a_1, r_2, \dots, r_{p+1} \rangle$ ; on pose  $r_1 = a_1$  et on a  $r = \langle r_1, \dots, r_{p+1} \rangle$ .

La taille de  $r$  est donc  $m = p + 1$  qui est la taille de  $a$ , d'où la propriété (ii).

Par hypothèse de récurrence,  $\forall 2 \leq i \leq p+1$ ,  $m \leq r_i$ . En outre,  $m < t = a_1 = r_1$ . On a donc (iii).

Par hypothèse de récurrence,  $\forall 2 \leq i \leq p+1$ ,  $\delta(a_i, m) + \text{card}(a_i, r0) = \delta(a_i, c) + \text{card}(a_i, q)$  et  $\delta(c, m) + \text{card}(c, r) = 1 + \text{card}(c, a)$ .

Or  $\text{card}(a_i, r) = \delta(a_i, a_1) + \text{card}(a_i, r0)$  et  $\text{card}(a_i, a) = \delta(a_i, a_1) + \text{card}(a_i, q)$  donc  $\delta(a_i, m) + \text{card}(a_i, r) = \delta(a_i, m) + \delta(a_i, a_1) +$

$card(a_i, r0) = \delta(a_i, m) + card(a_i, r0) + \delta(a_i, a_1) = \delta(a_i, c) + card(a_i, q) + \delta(a_i, a_1) = \delta(a_i, c) + card(a_i, a)$ . Enfin, par hypothèse de récurrence,  $\delta(c, m) + card(c, r0) = 1 + card(c, q)$  donc  $\delta(c, m) + card(c, r) = \delta(c, m) + \delta(c, a_1) + card(c, r0) = 1 + \delta(c, a_1) + card(c, q) = 1 + card(c, a)$ . On a donc (i).

Ceci prouve (i), (ii) et (iii) pour le cas  $c < t = a_1$ .

Supposons désormais que  $c \geq a_1$ . On pose alors  $(m, r0) = \text{aux } t \text{ } q \text{ et } r = c : r0$ . Par hypothèse de récurrence, la taille de  $r0$  est celle de  $q$  donc  $p$ . Notons  $r0 = \langle r_2, \dots, r_{p+1} \rangle$ . La taille de  $r$  est donc  $1 + p$  qui est la taille de  $a$ ; on a donc (ii).

Par hypothèse de récurrence,  $\forall 2 \leq i \leq p+1, m \leq a_i$  et  $m \leq t = a_1$  donc  $\forall 1 \leq i \leq p+1, m \leq a_i$ . En outre,  $m \leq a_1 \leq c$ . On a donc (iii). Enfin, par hypothèse de récurrence,  $\forall 2 \leq i \leq p+1, \delta(a_i, m) + card(a_i, r0) = \delta(a_i, t) + card(a_i, q)$ ; or  $card(a_i, r) = card(a_i, r0) + \delta(a_i, c)$  et  $card(a_i, a) = card(a_i, q) + \delta(a_i, t)$ ; donc  $\delta(a_i, m) + card(a, r) = \delta(a_i, m) + card(a_i, r0) + \delta(a_i, c) = \delta(a_i, t) + card(a_i, q) + \delta(a_i, c) = card(a_i, a) + \delta(a_i, c)$ . Par hypothèse de récurrence, on a aussi  $\delta(t, m) + card(t, r0) = 1 + card(t, q)$ . On a donc  $\delta(a_1, m) + card(a_1, r0) = 1 + card(t, q)$  donc  $\delta(a_1, m) + card(a_1, r0) + \delta(a_1, c) = 1 + card(a_1, q) + \delta(a_1, c)$  donc  $\delta(a_1, m) + card(a_1, r) = card(a_1, a) + \delta(a_1, c)$ . Pour finir, on a  $\delta(c, m) + card(c, r) = \delta(c, m) + card(c, r0) + \delta(c, c) = \delta(c, m) + card(c, r0) + 1$ ; par hypothèse de récurrence  $card(c, r0) + \delta(c, m) = card(c, q) + \delta(c, t)$ ; on a donc  $\delta(c, m) + card(c, r) = card(c, q) + \delta(c, t) + 1 = 1 + card(c, a)$ . Ceci prouve (iii).

Ceci prouve (i), (ii) et (iii) dans le cas  $c \geq t = a_1$ .

On a donc bien prouvé les propriétés (i), (ii) et (iii) pour la fonction aux.

Passons à la fonction "selection". Si  $s = t : q$ , cette fonction applique aux  $t \text{ } q$ . Notons  $a = \langle s_1, \dots, s_n \rangle$ . On a donc  $q = \langle s_2, \dots, s_n \rangle$ .

Notons  $(m, r) = \text{aux } t \text{ } q = \text{selection } s$ . On note  $r = \langle r_1, \dots, r_p \rangle$ . D'après ce qui précède, (ii)  $p = n - 1$  donc  $n = p + 1$  ce qui est (b).

D'après ce qui précède,  $\forall 2 \leq i \leq n, m \leq a_i$  et  $m \leq t = a_1$ . donc (c).

D'après ce qui précède,  $\forall 2 \leq i \leq n, \delta(s_i, m) + card(s_i, r) = \delta(s_i, t) + card(s_i, q)$  donc  $\delta(s_i, m) + card(s_i, r) = card(s_i, s)$ . En outre, toujours d'après ce qui précède,  $\delta(t, m) + card(t, r) = 1 + card(t, q)$  donc  $\delta(a_1, m) + card(a_1, r) = 1 + card(a_1, q) = card(a_1, s)$ . On a donc (a).

Ceci prouve (a), (b) et (c) pour toute liste non vide  $s$ .

### Question III. 3

Si  $s$  est la liste vide, la liste  $r$  est vide. Les trois propriétés sont dans ce cas triviales.

Supposons que ces trois propriétés soient vraies pour toute liste de taille au plus  $n$ . Soit une liste  $s$  de taille  $n + 1$ . On note  $(m0, sp) = \text{selection } s$ .

D'après la question précédente,  $s$  étant de taille  $n + 1$ ,  $sp$  est de taille  $n + 1 - 1 = n$ ; par hypothèse de récurrence,  $rp$  est de taille  $n$  donc la liste  $r = m0 : tp$  est de taille  $1 + n$ . On a donc (a).

De plus, par hypothèse de récurrence, pour tout  $1 \leq i \leq m$ ,  $card(s_i, sp) = card(s_i, rp)$ ; d'après la question précédente,  $\delta(s_i, m0) + card(s_i, sp) = card(s_i, s)$ . En outre, comme  $r = m0 : rp$ ,  $card(s_i, r) = \delta(s_i, m0) + card(s_i, rp) = \delta(s_i, m0) + card(s_i, sp) = card(s_i, s)$ . Ceci prouve (b).

Enfin,  $r = m0 : rp = \langle m, r_2, \dots, r_{n+1} \rangle$ . Par hypothèse de récurrence, pour tout  $2 \leq i < n + 1, r_i \leq r_{i+1}$ . D'après le point (c) de la question précédente,  $m0 \leq r_i$  pour tout  $2 \leq i \leq n$ . On a donc, en notant  $r_1 = m0$ , pour tout  $1 \leq i < n + 1, r_i \leq r_{i+1}$ .

Ceci prouve, par récurrence sur la longueur d'une liste, que les trois propriétés sont vérifiées.

### Question III. 4

La fonction "aux" se termine pour une liste vide et pour une liste non vide effectue un appel récursif sur une liste de taille un de moins.

Par conséquent, cette fonction se termine.

La fonction "selection" s'arrête donc sur une liste non vide.

La fonction "tri" s'arrête pour une liste vide; pour une liste non vide, elle appelle la fonction "selection" qui se termine bien pour une liste non vide, et effectue un appel récursif sur une liste de taille un de moins. Par conséquent, la fonction "tri" va bien se terminer sur une liste quelconque.

### Question III. 5

La fonction "aux" effectue sur une liste non vide un appel récursif sur une liste de taille un de moins. Par récurrence, la fonction "aux" effectue  $n$  appels récursifs où  $n$  est la taille de la liste.

La fonction "selection" effectue donc un nombre d'appels à la fonction "aux" égal à la taille de la liste.

La fonction "tri" effectue, sur une liste non vide, un appel récursif sur une liste de taille un de moins que la liste initiale. Ainsi, la fonction "tri" effectue un nombre d'appels récursifs égal à la taille de la liste.

La fonction "tri" fait appel  $n$  fois à la fonction "selection" qui fait appel  $n$  fois à "aux", puis  $n - 1$  fois à "aux"... Au total, on a donc

$$\sum_{k=1}^n k \text{ appels récursifs, à savoir } \frac{n(n+1)}{2} = O(n^2).$$

### Question III. 6

$\forall n \in \mathcal{N}(a), \mathcal{X}(n) > 0, \mathcal{Y}(n) > 0, \mathcal{T}(n)$ .

$\forall d \in \mathcal{D}(a)$ , pour tout fils  $f$  de  $d$ ,  $\mathcal{T}(f) = \frac{\mathcal{T}(d)}{2}$ .

$\forall d \in \mathcal{D}(a), \mathcal{X}(f_{SO}) = \mathcal{X}(d), \mathcal{Y}(f_{SO}) = \mathcal{Y}(d), \mathcal{X}(f_{SE}) = \mathcal{X}(d) + \frac{\mathcal{T}(d)}{2}, \mathcal{Y}(f_{SE}) = \mathcal{Y}(d), \mathcal{X}(f_{NO}) = \mathcal{X}(d), \mathcal{Y}(f_{SE}) = \mathcal{Y}(d) + \frac{\mathcal{T}(d)}{2},$

$\mathcal{X}(f_{NE}) = \mathcal{X}(d) + \frac{\mathcal{T}(d)}{2}, \mathcal{Y}(f_{NE}) = \mathcal{Y}(d) + \frac{\mathcal{T}(d)}{2}$ .

$\forall d \in \mathcal{D}(a), \exists f_1, f_2$  deux fils de  $d$  tels que  $\exists n_1 \in \mathcal{B}(f_1), n_2 \in \mathcal{B}(f_2)$ , tels que  $\mathcal{C}(n_1) \neq \mathcal{C}(n_2)$ .

### Question III. 7

```
let scinder a = match a with
| Division(_,_,_,_,_,_) -> a
| Bloc(x,y,t,c) -> Division(x,y,t, Bloc(x,y,t/2,c), Bloc(x+t/2,y,t/2,c), Bloc(x,y+t/2,t/2,c), Bloc(x+t/2,y+t/2,t/2,c));;
```

### Question III. 8

```
let fusionner so se no ne = match so,se,no,ne with
| (Bloc(x,y,t,c1), Bloc(_,_,_,c2), Bloc(_,_,_,c3), Bloc(_,_,_,c4)) when c1=c2 && c2=c3 && c3=c4 -> Bloc(x,y,2*t,c1)
| (Bloc(x,y,t,_) ,_,_,_) -> Division(x,y,2*t,so,se,no,ne)
| (Division(x,y,t,_,_,_,_),_,_,_) -> Division(x,y,2*t,so,se,no,ne);;
```

### Question III. 9

```
let rec profondeur a = match a with
| Bloc(_,_,_,_) -> 0
```

```
| Division(_,_,_,so,se,no,ne) -> 1+ max (max (profondeur so) (profondeur se)) (max (profondeur no) (profondeur ne));;
```

### Question III. 10

```
let rec consulter x y a = match a with
| Bloc(_,_,_,c) -> c
| Division(x0,y0,t,so,_,_,_) when x<x0+t/2 && y<y0+t/2 -> consulter x y so
| Division(x0,y0,t,_,se,_,_) when x>=x0+t/2 && y<y0+t/2 -> consulter (x-t/2) y se
| Division (x0,y0,t,_,_,no,_) when x<x0+t/2 && y>=y0+t/2 -> consulter x (y-t/2) no
| Division(x0,y0,t,_,_,_,ne) -> consulter (x-t/2) (y-t/2) ne;;
```

### Question III. 11

```
let rec peindre x y c a = match a with
| Bloc(_,_,t,_) when t=1 -> Bloc(x,y,t,c)
| Bloc(_,_,_,c0) when c=c0 -> a
| Bloc(x0,y0,t,_) -> peindre x y c scinder a
| Division(x0,y0,t,so,se,no,ne) when x<x0+t/2 && y<y0+t/2 -> fusionner (peindre x y c so) se no ne
| Division(x0,y0,t,so,se,no,ne) when x>=x0+t/2 && y<y0+t/2 -> fusionner so (peindre x y c se) no ne
| Division (x0,y0,t,so,se,no,ne) when x<x0+t/2 && y>=y0+t/2 -> fusionner so se (peindre x y c no) ne
| Division(x0,y0,t,so,se,no,ne) -> fusionner so se ,no (peindre x y c ne);;
```

### Question III. 12

```
let rec valider a =
let rec abs a = match a with
| Bloc(x,_,_,_) -> x
| Division(x,_,_,_,_,_,_) -> x
and ord a = match a with
| Bloc(_,y,_,_) -> y
| Division(_,y,_,_,_,_,_) -> y
and taille a = match a with
| Bloc(_,_,t,_) -> t
| Division(_,_,t,_,_,_,_) -> t
in match a with
| Bloc(x,y,t,c) -> (x>0)&&(y>0)&&(t>0)
| Division(_,_,_, Bloc(_,_,_,c1), Bloc(_,_,_,c2), Bloc(_,_,_,c3), Bloc(_,_,_,c4)) when c1=c2 && c2=c3 && c3=c4 -> false
| Division(x,y,t,so,se,no,ne) -> (x>0)&&(y>0)&&(t>0)&&(x=abs so)&&(y=abs se)&&(t/2=taille so)&&(x+t/2=abs se)&&(t/2=taille se)&&
```

### Question III. 13

```
let sauvegarder a =
let rec aux a i = match a with
| Bloc(x,y,t,c) -> ([i,c],1)
| Division(x,y,t,so,se,no,ne) ->
let (l1,t1)=aux so i in let (l2,t2)=aux se (i+t1) in let (l3,t3)=aux no (i+t1+t2) in let (l4,t4)=aux ne (i+t1+t2+t3)
in( l1 @ l2 @ l3 @ l4, t1+t2+t3+t4)
in fst(aux a 1);;
```

### Question III. 14

```
let restaurer s =
let rec aux l x y t = match (l,t) with
| (i,c)::q when t=1 -> (Bloc(x,y,t,c),q)
| (l,t) -> let (so,q1)=aux l x y (t/2) in let (se,q2)=aux q1 (x+t/2) y (t/2)
in let (no,q3)=aux q2 x (y+t/2) t/2 in let (ne,q4)=aux q3 (x+t/2) (y+t/2) (t/2)
in (fusionner so se no ne , q4)
and racine n = match n with
| 1 -> 1
| _ -> 1+racine(n/4)
in fst(aux s 1 1 (racine (List.length s)));;
```