Algorithmique III (I41) - Licence d'Informatique

Correction du contrôle terminal (Session 1 / Mai 2021)

Soit $n \in \mathbb{N}$ avec $n \geqslant 3$. On note $\mathscr{L}_n$ l'ensemble (infini) des listes d'entiers naturels de longueur n dont toutes les valeurs sont identiques sauf une, l'*intrus*. Par exemple $L := [2,2,3,2,2,2] \in \mathscr{L}_6$ où l'intrus est 3. Ces listes définissent des applications de $[\![1,n]\!]$ dans $\mathbb{N}$, on notera donc $L(i)$ l'image de i quand nous ferons référence à l'application.

On rappelle que si $f : X \to Y$ est une application, $A \subseteq X$ et $B \subseteq Y$, alors $f(A) = \{f(x) \mid x \in A\}$ et $f^{-1}(B) = \{x \in X \mid f(x) \in B\}$.

On définit une relation binaire $\bowtie$ sur les listes de $\mathscr{L}_n$ par :

$$L_1 \bowtie L_2 \;\Leftrightarrow\; \forall i \in [\![1,n]\!] \;\; L_1^{-1}(\{L_1(i)\}) = L_2^{-1}(\{L_2(i)\}). \qquad (1)$$

(1) Vérifiez que $[7,7,7,3,7] \bowtie [0,0,0,1,0]$.

Solution. Si y est un élément d'une liste L, l'image réciproque $L^{-1}(\{y\})$ fournit l'ensemble des indices i tels que $L[i] = y$. L'expression (1) ne fait qu'exprimer que deux listes sont en relation si deux valeurs égales dans l'une impose que les valeurs aux mêmes positions de l'autre sont elles aussi égales. Par conséquent deux listes de $\mathscr{L}_n$ sont en relation si et seulement si leurs intrus sont à la même position.

On note $L_1 := [7,7,7,3,7]$ et $L_2 := [0,0,0,1,0]$:

$$\begin{array}{lll}
L_1(1) = L_1(2) = L_1(3) = L_1(5) = 7 & \text{donc} & L_1^{-1}(\{7\}) = \{1,2,3,5\},\\
L_1(4) = 3 & \text{donc} & L_1^{-1}(\{3\}) = \{4\},\\
L_2(1) = L_2(2) = L_2(3) = L_2(5) = 0 & \text{donc} & L_2^{-1}(\{0\}) = \{1,2,3,5\},\\
L_2(4) = 1 & \text{donc} & L_2^{-1}(\{1\}) = \{4\}.
\end{array}$$

Et on en déduit que

$$\begin{aligned}
L_1^{-1}(\{L_1(1)\}) &= \{1,2,3,5\} = L_2^{-1}(\{L_2(1)\}),\\
L_1^{-1}(\{L_1(2)\}) &= \{1,2,3,5\} = L_2^{-1}(\{L_2(2)\}),\\
L_1^{-1}(\{L_1(3)\}) &= \{1,2,3,5\} = L_2^{-1}(\{L_2(3)\}),\\
L_1^{-1}(\{L_1(5)\}) &= \{1,2,3,5\} = L_2^{-1}(\{L_2(5)\}).\\
L_1^{-1}(\{L_1(4)\}) &= \{4\} = L_2^{-1}(\{L_2(4)\}).
\end{aligned}$$

On a bien vérifié que $[7,7,7,3,7] \bowtie [0,0,0,1,0]$.

(2) Démontrez que $\bowtie$ est une relation d'équivalence sur $\mathscr{L}_n$.

Solution. Il suffit de réaliser que la relation $\bowtie$ définie en (1) s'appuie sur une égalité, qui est à la fois réflexive, symétrique et transitive. Si cet argument ne paraît pas évident, on peut aussi le faire directement :

- On a évidemment $\forall i \in [\![1,n]\!] \;\; L_1^{-1}(\{L_1(i)\}) = L_1^{-1}(\{L_1(i)\})$, donc $L_1 \bowtie L_1$ et $\bowtie$ est réflexive.
- Si $L_1 \bowtie L_2$ alors $\forall i \in [\![1,n]\!] \;\; L_2^{-1}(\{L_2(i)\}) = L_1^{-1}(\{L_1(i)\})$ et donc $L_2 \bowtie L_1$ ce qui prouve que $\bowtie$ est symétrique.
- Si $L_1 \bowtie L_2$ et $L_2 \bowtie L_3$, alors on a

$$\begin{aligned}
\forall i \in [\![1,n]\!] \;\; L_1^{-1}(\{L_1(i)\}) &= L_2^{-1}(\{L_2(i)\}),\\
\forall i \in [\![1,n]\!] \;\; L_2^{-1}(\{L_2(i)\}) &= L_3^{-1}(\{L_3(i)\}),
\end{aligned}$$

et par transitivité de l'égalité on en déduit que

$$\forall i \in [\![1,n]\!] \;\; L_1^{-1}(\{L_1(i)\}) = L_3^{-1}(\{L_3(i)\}).$$

Ceci prouve que $L_1 \bowtie L_3$ et que $\bowtie$ est transitive.

(3) Combien y-a-t-il de classes d'équivalence pour la relation $\bowtie$?

Solution. Deux listes de $\mathscr{L}_n$ étant en relation si et seulement si leurs intrus sont à la même position, il y a n positions possibles et donc autant de classes d'équivalence.

(4) Démontrez que l'on peut trouver l'intrus en faisant au plus $\lfloor \frac{n}{2} \rfloor$ comparaisons entre termes de la liste ($\lfloor x \rfloor$ est la partie entière du réel x).

Solution. L'algorithme s'appuie sur la proposition (évidente) suivante : si l'on compare deux éléments $L[i]$ et $L[j]$ de la liste avec $i \neq j$, soit ils sont égaux et l'intrus est ailleurs, soit ils sont différents et l'intrus est l'un des deux.

(a) Si L est de longueur paire $n = 2k$, on peut partitionner l'ensemble des indices $[\![1, n]\!]$ en k paires d'indices adjacents $\{2i-1, 2i\}$, $i \in [\![1, k]\!]$, et dans le pire des cas l'intrus appartient à la dernière classe $\{L[n-1], L[n]\}$. Dans ce cas, on aura réalisé les $k-1$ comparaisons $L[2i-1] = L[2i]$ pour $i \in [\![1, k-1]\!]$ et une dernière comparaison entre $L[n-1]$ et un élément précédent est nécessaire pour décider qui de $L[n-1]$ et $L[n]$ est l'intrus, ce qui fait k comparaisons au total.

(b) Si la liste est de longueur impaire $n = 2k+1$, on peut partitionner l'ensemble des indices $[\![1, n]\!]$ en k paires d'indices adjacents $\{2i-1, 2i\}$, $i \in [\![1, k]\!]$ et le singleton $\{n\}$, et dans le pire des cas, l'intrus est le dernier élément $L[n]$ de la liste. Dans ce cas on aura réalisé les k comparaisons $L[2i-1] = L[2i]$ pour $i \in [\![1, k]\!]$.

Quelle que soit la parité de n, on a $k = \lfloor \frac{n}{2} \rfloor$ et en conclusion, dans le pire des cas, il faut $\lfloor \frac{n}{2} \rfloor$ comparaisons pour décider qui est l'intrus.

(5) Écrivez un algorithme `Intrus(L)` qui renvoie la position de l'intrus dans la liste L en faisant au plus $\lfloor \frac{n}{2} \rfloor$ comparaisons entre éléments de la liste. Commentez votre peudo-code.

Solution. On suppose que la liste contient au moins 3 éléments. Le principe consiste à avancer de deux en deux dans la liste tant que les valeurs côte à côte sont égales. La conjonction de la condition de boucle fait que l'on en sort :

(a) Parce que la première condition `(i < |L| - 1)` n'est plus satisfaite, c'est-à-dire parce que $i = n-1$ dans le cas où n est pair ou $i = n$ dans le cas ou n est impair. Dans les deux cas il suffit de comparer $L[i]$ avec $L[i-2]$ (le terme existe puisque $|L| \geqslant 3$) pour décider qui de $L[i]$ et de $L[i+1]$ est l'intrus (notons que cette comparaison est inutile pour n impair puisque l'intrus est nécessairement $L[n]$, mais évite un nouveau test).

(b) Parce que la deuxième condition `(L[i] = L[i+1])` n'est plus satisfaite, auquel cas n est nécessairement pair et il suffit de comparer $L[i]$ avec $L[i+2]$ (le terme existe puisque $i < |L| - 1$) pour décider qui de $L[i]$ et $L[i+1]$ est l'intrus.

La variable `dec` est initialisée à ± 2 selon la valeur de i.

```
ALGO Intrus(L):entier
DONNEES
   L: liste d'entiers
   i, dec: entiers
DEBUT
   i ← 1
   TQ ((i < |L| - 1) ET (L[i] = L[i + 1])) FAIRE
      i ← i + 2  // L'intrus est ailleurs
   FTQ
   dec ← (i > 1) ? -2 ; 2
   SI (L[i] = L[i + dec]) ALORS  // L'intrus est L[i + 1]
      RENVOYER i + 1
   SINON  // L'intrus est L[i]
      RENVOYER i
   FSI
FIN
```

(6) Calculez la complexité moyenne en nombre de comparaisons de votre algorithme.

Solution. Après la sortie de la boucle principale, il y a deux comparaisons, celle qui décide de la parité de la variable `dec` et celle qui permet de décider qui est l'intrus. Chaque test de boucle effectue deux comparaisons, celle qui évite de dépasser la fin de la liste et celle qui compare deux éléments adjacents de la liste (pour ne pas compliquer inutilement le calcul, on ne fait pas l'hypothèse que l'on arrête l'évaluation d'une conjonction dès qu'une des propositions n'est pas satisfaite).

Le nombre d'occurrence du test est compris entre 1 et $k := \lfloor \frac{n}{2} \rfloor$ (ne pas oublier le dernier test qui fait sortir de la boucle). Comme l'algorithme fait exactement le même nombre de passages dans la boucle pour deux listes d'une même classe d'équivalence de $\mathscr{L}_n / \bowtie$, il suffit de calculer la moyenne des coûts pour chacune des positions possible de l'intrus. Par hypothèse, on suppose que cette position est uniformément distribuée dans l'intervalle $[\![1, n]\!]$.

On peut encore simplifier le calcul en remarquant que le coût est identique pour deux listes s'il existe un entier $i \in [\![1, \lfloor \frac{n}{2} \rfloor]\!]$ tel que les positions de

leurs intrus sont $2i-1$ et $2i$. On a donc en notant $k := \lfloor \frac{n}{2} \rfloor$:

$$\begin{aligned}\overline{T} &= 2 + \frac{2}{k}\sum_{i=1}^{k} 2i \\ &= 2 + \frac{4}{k}\sum_{i=1}^{k} i \\ &= 2 + \frac{4k(k+1)}{2k} \\ &= 2(\lfloor \frac{n}{2} \rfloor + 2).\end{aligned}$$

On a donc une complexité linéaire en n en moyenne.

On appelle représentant *minimal* d'une classe d'équivalence de $\mathscr{L}_n/\bowtie$ la liste de cette classe dont la somme des valeurs est minimale.

(7) Montrez que cette liste est unique. Quels sont alors les représentants minimaux de $\mathscr{L}_n/\bowtie$?

Solution. On définit les n listes U_i de $\mathscr{L}_n$ qui ne contiennent qu'un seul 1 et des 0 partout ailleurs :

$$\forall i \in [\![1,n]\!] \quad U_i[j] := \begin{cases} 0 & \text{si } j \neq i \\ 1 & \text{si } j = i. \end{cases}$$

Dans une même classe d'équivalence de $\mathscr{L}_n/\bowtie$, on trouve toutes les listes dont l'intrus est à la même position. Notons que deux listes de $\mathscr{L}_n$ peuvent avoir la même somme sans pour autant être égales, y compris si elles appartiennent à la même classe d'équivalence, $[2,0,2,2]$ et $[1,3,1,1]$ par exemple. La liste U_i appartient à la classe d'équivalence des listes dont l'intrus est en position i, elle est trivialement minimale puisque sa somme vaut 1 et aucune autre liste de sa classe ne peut avoir pour somme 1, elle est donc unique. Les représentants minimaux de $\mathscr{L}_n/\bowtie$ sont donc les listes U_i.

On définit la somme $L_1 \oplus L_2$ de deux listes de $\mathscr{L}_n$ comme la liste qui est le ou exclusif bit-à-bit des représentants minimaux des classes de L_1 et L_2 dans $\mathscr{L}_n/\bowtie$.

(8) Calculez $[12,12,3,12] \oplus [2,2,2,1]$.

Solution. Le représentant minimal de $[12,12,3,12]$ est la liste $[0,0,1,0]$ et celui de $[2,2,2,1]$ est la liste $[0,0,0,1]$, on a donc

$$[12,12,3,12] \oplus [2,2,2,1] = [0,0,1,1].$$

(9) Montrez que l'ensemble $\mathscr{L}_n \oplus \mathscr{L}_n = \{L_1 \oplus L_2 \mid (L_1, L_2) \in \mathscr{L}_n^2\}$ est fini et déterminez son cardinal.

Solution. Pour un entier n donné, les éléments de $\mathscr{L}_n \oplus \mathscr{L}_n$ sont définis par $U_i \oplus U_j$ pour $(i,j) \in [\![1,n]\!] \times [\![1,n]\!]$ qui est fini. Le XOR de deux listes U_i et U_j est la liste qui contient la valeur 1 aux positions i et j et 0 partout ailleurs, sauf si $i = j$, auquel cas c'est la liste nulle. Il y a $\binom{n}{2}$ combinaisons possibles pour constituer une liste de poids 2, on a donc finalement

$$|\mathscr{L}_n \oplus \mathscr{L}_n| = \binom{n}{2} + 1.$$

(10) Écrivez un algorithme `XOR(L1,L2)` qui renvoie la liste $L_1 \oplus L_2$ (utilisez l'algorithme `Intrus`).

Solution. L'algorithme est simple, il suffit d'initialiser une liste nulle, de repérer la position des intrus dans chacune des listes L_1 et L_2 et de placer un 1 à ces positions si ces positions sont différentes :

```
ALGO XOR(L1,L2):liste
DONNEES
   L1,L2: listes d'entiers
   i,j: entiers
DEBUT
   Allouer(L,0,|L1|) // Crée une liste de 0 de la taille de L1
   i ← Intrus(L1)
   j ← Intrus(L2)
   SI (i != j) ALORS
      L[i] ← 1
      L[j] ← 1
   FSI
   RENVOYER L
FIN
```

(11) Calculez la complexité de l'algorithme `XOR`.

Solution. On pourraît distinguer un meilleur des cas et un pire des cas selon que les indices i et j sont égaux ou non, mais le coût de l'initialisation des deux entrées d'indices i et j étant en $\Theta(1)$, la complexité reste

fondamentalement la même. C'est le coût de l'initialisation de la liste de n valeurs à 0 qui est prépondérant avec une complexité de $\Theta(n)$ (ceux qui ont considéré que l'allocation était en $\Theta(1)$ n'ont pas été pénalisés). Donc

$$T(n) = \Theta(1) + \Theta(n) = \Theta(n).$$