TD 1 : Correction et terminaison des algorithmes

13 septembre 2018

1 Le tri comptage

On considère un algorithme de tri agissant sur un tableau A d'entiers bornés, appartenant à $\{1, 2, \ldots, k\}$, où $k \in \mathbb{N}$. Le principe est le suivant : on commence par compter le nombre d'occurences de chacun des entiers de $\{1, \ldots, k\}$ dans le tableau A. On parcourt ensuite ce tableau des occurences dans l'ordre croissant pour reconstruire les données de A, triées cette fois-ci.

Question 1 – Écrire la fonction TRI-COMPTAGE, qui prend en entrée un tableau A et implémente la stratégie décrite ci-dessus.

Question 2 – Quelle est la complexité de TRI-COMPTAGE, en fonction de k et de n, la taille du tableau d'entrée ?

Question 3 – Que dire de cette complexité, par rapport à la borne maximale vue en cours pour les algorithmes de tri par comparaison ?

2 Exponentiation rapide : preuve de correction

Dans le cas d'algorithmes exprimés de façon récursive, on remplace l'utilisation d'invariants de boucle par l'exploitation de la notion de précondition et postcondition : on vérifie qu'avant chaque appel récursif, la précondition de la fonction appelée est satisfaite ; on peut alors supposer la postcondition vraie après l'appel.

Considérons l'algorithme suivant, dit d'exponentiation rapide.

Précondition: $n >= 0$
  Fonction $\mathrm{EXP}(x, n)$
    si $n = 0$ **alors**
      renvoyer 1
    sinon
      si $n \bmod 2 = 0$ **alors**
        $a \leftarrow \mathrm{EXP}(x, n/2)$
        renvoyer $a * a$
      sinon

$b \leftarrow \text{EXP}(x, (n-1)/2)$
renvoyer $x * b * b$

QUESTION 4 – Spécifier et justifier la correction de cet algorithme.

3 Tri par tas

La première étape du tri par tas est consacrée à la construction du tas en démarrant par le bas. On rappelle le code de CONSTRUIRE-TAS ci-après :

Fonction TASSER(A, i)
  $i_{\max} \leftarrow i$
  si $2i \leq taille[A]$ **et** $A[2i] > A[i_{\max}]$ **alors**
    $i_{\max} \leftarrow 2i$
  si $2i+1 \leq taille[A]$ **et** $A[2i+1] > A[i_{\max}]$ **alors**
    $i_{\max} \leftarrow 2i+1$
  si $i_{\max} \neq i$ **alors**
    $A[i] \leftrightarrow A[i_{\max}]$
    TASSER$(A, i_{\max})$

Fonction CONSTRUIRE-TAS(A)
  pour $i \leftarrow taille[A]$ **à** 1 **décr faire**
    TASSER(A, i)

On rappelle que la complexité de TASSER(A, i) en $\mathcal{O}(h(i))$, où $h(i)$ est la hauteur du nœud en position i dans le tas A. On cherche à calculer la complexité de CONSTRUIRE-TAS.

Par convention, on définit $h(i) = 0$ si le nœud en position i est une feuille de l'arbre. On note également $h(A) = h(1)$ la hauteur de l'arbre, et n le nombre total de nœuds dans l'arbre.

QUESTION 5 – Donner la relation qui relie n et $h(A)$.

QUESTION 6 – Pour une hauteur donnée, h, déterminer le nombre de nœuds qui ont cette hauteur dans le tas. On se limitera au cas du tas complet pour simplifier.

QUESTION 7 – En déduire la complexité de CONSTRUIRE-TAS. De même, on se limitera au cas du tas complet pour simplifier.

4 Terminaison d'algorithmes

Symétriquement à la notion d'invariant utilisée pour la preuve de correction d'algorithme, une notion de variant peut être utilisée pour établir leur terminaison. On rappelle ici quelques définitions utiles.

Définition 1 (Ordre bien fondé). *Soit $(E, \preccurlyeq)$ un ensemble ordonné. On dit que l'ordre $\preccurlyeq$ est bien fondé s'il n'existe pas de suite infinie strictement décroissante d'éléments de E.*

Exemple 1. *$(\mathbb{N}, \leq)$ est un ensemble bien fondé, mais $(\mathbb{Z}, \leq)$ n'est pas un ensemble bien fondé.*

On peut alors exploiter cette propriété pour garantir qu'une boucle ou une fonction récursive ne pourra itérer qu'un nombre fini de fois.

Définition 2 (Variant). *Soit $(E, \preccurlyeq)$ un ensemble bien fondé.*

Cas itératif : *un variant de boucle est une fonction V à valeur dans E, qui dépend des variables du programme (l'état courant), et telle que si s est un état valide (satisfaisant l'invariant de boucle) en début de boucle, et s' l'état résultant après exécution d'un tour de boucle, alors on a $V(s') \prec V(s)$.*

Cas récursif : *un variant récursif est une fonction V : $\texttt{Arg} \to E$ telle que si arg est la valeur des arguments lors de l'appel de la fonction, et arg' la valeur des arguments lors d'un de ses appels récursifs, alors on a $V(arg') \prec V(arg)$.*

On admettra que l'existence d'un variant est une condition suffisante à la terminaison d'un programme [1].

4.1 Opération de fusion du tri fusion

QUESTION 8 – Établir la terminaison de la fonction auxiliaire de fusion utilisée dans le tri fusion, rappelée ci-dessous.

Fonction FUSION($l1, l2$)
  $\rho \leftarrow []$
  tant que $l1 \neq []$ ou $l2 \neq []$ **faire**
    si $l1 = []$ **alors**
      $\rho \leftarrow \rho \cdot l2$
      $l2 \leftarrow []$
    sinon si $l2 = []$ **alors**
      $\rho \leftarrow \rho \cdot l1$
      $l1 \leftarrow []$
    sinon
      si TÊTE($l1$) $<$ TÊTE($l2$) **alors**
        $\rho \leftarrow \rho +$ TÊTE($l1$)
        $l1 \leftarrow$ QUEUE($l1$)
      sinon
        $\rho \leftarrow \rho +$ TÊTE($l2$)
        $l2 \leftarrow$ QUEUE($l2$)
  renvoyer(ρ)

4.2 Ackermann

On considère la fonction suivante. $A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m, n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases}$

QUESTION 9 – Écrire les quelques premiers termes de la fonction d'Ackermann.

1. En réalité, c'est également une condition nécessaire !

Définition 3. *Ordre lexicographique*

Soit $(E, \preccurlyeq_E)$ et $(F, \preccurlyeq_F)$ deux ensembles bien fondés. On définit l'ordre lexicographique sur leur produit cartésien $(E \times F, \preccurlyeq_{lex})$ par :

— *si $e_1 \preccurlyeq_E e_2$, alors pour tout f_1, f_2, $(e_1, f_1) \preccurlyeq_{lex} (e_2, f_2)$;*

— *si $f_1 \preccurlyeq_F f_2$, alors pour tout e, $(e, f_1) \preccurlyeq_{lex} (e, f_2)$*

QUESTION 10 – Montrer que $(E \times F, \preccurlyeq_{lex})$ est un ensemble bien fondé.

QUESTION 11 – Prouver la terminaison de la fonction d'Ackermann.

5 Problème du drapeau tricolore

On aligne n boules, réparties en un nombre quelconque de boules de couleur bleue, blanche ou rouge. Ces boules sont diposées dans un ordre quelconque. La ligne de boules est représentée par un tableau B de taille n, dont chaque élément $B[i]$ appartient à l'ensemble {bleu, blanc, rouge}.

QUESTION 12 – Écrire un algorithme qui trie le tableau de telle façon que toutes les boules bleues apparaissent au début, suivies des boules blanches puis des boules rouges. La contrainte est que le tri du tableau doit être réalisé en seul parcours (on ne peut tester qu'une seule fois la couleur de chaque boule) et en place (sans utiliser de deuxième tableau).