TD 1 : Correction et terminaison d'algorithmes

14 septembre 2017

1 Union-Find avec équilibre des arbres basé sur la hauteur

Écrire une variante de UNION qui s'appuie sur la hauteur des arbres plutôt que le nombre d'éléments. On se passera de compression de chemin pour cet exercice.

2 Correction d'algorithmes

2.1 Tri à bulles

On rappelle le code du tri à bulles :

Fonction TRI-BULLE(A)
  pour $i \leftarrow 1$ **à** $|A| - 1$ **faire**
    pour $j \leftarrow |A|$ **à** $i + 1$ **décr faire**
      si $A[j] < A[j-1]$ **alors**
        $temp \leftarrow A[j-1]$
        $A[j-1] \leftarrow A[j]$
        $A[j] \leftarrow temp$

Rappel : un invariant de boucle est une propriété que l'on établit sur la tête d'une boucle en prouvant :

— *Initialisation* : la propriété est vraie en entrée de boucle ;
— *Conservation* : si la propriété est vraie avant une itération, elle est encore vraie après l'exécution du corps de boucle.

On obtient ainsi une propriété valide en sortie de boucle, en ajoutant la condition de sortie.

QUESTION 1 – Rappeler la spécification d'un algorithme de tri.

QUESTION 2 – Proposer et démontrer un invariant de boucle pour la boucle interne.

QUESTION 3 – De même pour la boucle externe ; en déduire la correction de l'algorithme.

2.2 Exponentiation rapide

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**
      return 1
    sinon
      si $n \bmod 2 = 0$ **alors**
        $a \leftarrow \mathrm{EXP}(x, n/2)$
        return $a * a$
      sinon
        $b \leftarrow \mathrm{EXP}(x, (n-1)/2)$
        return $x * b * b$

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

3 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 : $\mathtt{Arg} \to E$ 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].

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

3.1 Opération de fusion du tri fusion

QUESTION 5 – 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)

3.2 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 6 – Écrire les quelques premiers termes de la fonction d'Ackermann.

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 7 – Montrer que $(E \times F, \preccurlyeq_{lex})$ est un ensemble bien fondé.
QUESTION 8 – Prouver la terminaison de la fonction d'Ackermann.

4 Bonus : Fonction de McCarthy

Considérons l'algorithme suivant.

$$f(x) = \begin{cases} x - 10 & \text{si } x > 100 \\ f(f(x+11)) & \text{sinon} \end{cases}$$

QUESTION 9 – Donner une formulation non récursive de la fonction de McCarthy. Justifier sa terminaison et sa correction.