TD : Invariant

Olivier Raynaud

raynaud@isima.fr

Résumé

Cette fiche de Td est consacrée à l'étude des invariants d'algorithme. Cette fiche est composée de 3 exercices. Le premier propose 5 algorithmes pour lesquels il est demandé de déterminer et de prouver les invariants. Le deuxième exercice expose deux algorithmes dont il faut deviner le résultat et prouver la correction. Enfin le troisième exercice propose d'étudier les conditions d'évolution d'une configuration donnée sur un damier.

Rappel : L'une des techniques pour prouver la correction d'un algorithme est d'établir l'invariance de propriétés portant sur des variables manipulées par l'algorithme. Cette invariance est relative aux instructions de l'algorithme qui ne doivent pas changées la validité de l'ensemble des propriétés. On parlera aussi parfois d'un invariant de boucle pour faire référence à une propriété dont la validité n'est pas modifiée lors de l'exécution de la boucle.

Exercice 1. *Déterminer et prouver les invariants des algorithmes suivants :*

Algorithm 1: *multiplicationRusse*()

Données: A, B : entier ;
Résultat: z : entier ; le produit de A et B par la méthode Russe ;
Variables : x, y, z : entier ;
début
  $x \leftarrow A$; $y \leftarrow B$; $z \leftarrow 0$;
  tant que $(x \neq 0)$ **faire**
    si x *est impair* **alors**
      $z \leftarrow z + y$;
    fin
    $x \leftarrow x \ Div \ 2$; $y \leftarrow 2 * y$;
  fin
  retourner z
fin

Algorithm 2: *triSelection*()

Données: $tab[n]$: tableau d'entiers ;

Résultat: / ; le tableau est trié

Variables : $indiceMin$, $valeurMin$: entier ;

début

    pour ($i = 1$ *à* $n - 1$) **faire**

        $indiceMin \leftarrow i$; $valeurMin \leftarrow tab[i]$;

        pour ($j = i + 1$ *à* n) **faire**

            si ($tab[j] < valeurMin$) **alors**

                $indiceMin \leftarrow j$; $valeurMin \leftarrow tab[j]$;

            fin

        fin

        $tab[indiceMin] \leftarrow tab[i]$;$tab[i] \leftarrow valeurMin$;

    fin

fin

Algorithm 3: *triInsertion*()

Données: $tab[]$: tableau d'entiers ;

Résultat: / ; le tableau est trié

Variables : $indiceC$, $valeurC$: entier ;

début

    pour ($j = 2$ *à* n) **faire**

        $valeurC \leftarrow tab[j]$; $indiceC \leftarrow j - 1$;

        tant que ($indiceC > 0$ *et* $valeurC < tab[indiceC]$) **faire**

            $tab[indiceC + 1] \leftarrow tab[indiceC]$;

            $indiceC \leftarrow indiceC - 1$;

        fin

        $tab[indiceC + 1] \leftarrow valeurC$;

    fin

fin

Algorithm 4: *multiplicationRusse*()

Données: A, B : entier ;
Résultat: : entier ; le produit de A et B par la méthode Russe en mode récursif ;
Variables : x, y : entier ;
début
  $x \leftarrow A$; $y \leftarrow B$;
  si $(y = 0)$ **alors**
    retourner 0 ;
  sinon
    si *(y est impair)* **alors**
      retourner $multiplicationRusse(2x, \lfloor y/2 \rfloor + x)$;
    sinon
      retourner $multiplicationRusse(2x, \lfloor y/2 \rfloor)$;
    fin
  fin
fin

Algorithm 5: *puissanceRusse*()

Données: A, B : entier ;
Résultat: : entier ; y puissance z par la méthode Russe en mode récursif ;
Variables : x, y : entier ;
début
  $x \leftarrow A$; $y \leftarrow B$;
  si $(y = 0)$ **alors**
    retourner 1;
  sinon
    si *(y est impair)* **alors**
      retourner $puissanceRusse(x^2, \lfloor y/2 \rfloor * x)$;
    sinon
      retourner $puissanceRusse(x^2, \lfloor y/2 \rfloor)$;
    fin
  fin
fin

Algorithm 6: $minMax()$

Données: $tab[2N]$: tableau d'entiers ;
Résultat: (min, max) couple d'entiers ;

```
début
    si (tab[1] < tab[2]) alors
        min ← tab[1] ; max ← tab[2] ;
    sinon
        min ← tab[2] ; max ← tab[1] ;
    fin
    pour (i = 3 à 2N − 1 pas de 2) faire
        si (tab[i] < tab[i + 1]) alors
            si (tab[i] < min) alors
                min ← tab[i] ;
            fin
            si (tab[i + 1] > max) alors
                max ← tab[i + 1] ;
            fin
        sinon
            si (tab[i + 1] < min) alors
                min ← tab[i + 1] ;
            fin
            si (tab[i] > max) alors
                max ← tab[i] ;
            fin
        fin
    fin
    retourner (min, max) ;
fin
```

Exercice 2 (Preuve d'algorithme)**.**

Question 1. *Montrer que l'algorithme 7 vérifie le test d'arrêt.*

Question 2. *On note i_k, m_k et z_k les valeurs respectives des variables i, m et z après k passages dans la boucle "tant que". Montrer que l'algorithme 7 vérifie chacun des invariants suivants :*

- $i_k = k$;
- $z_k = 2i_k + 1$;
- $m_k = N - {i_k}^2$;
- $m_k \geq 0$.

Question 3. *Déduire des deux questions précédentes le résultat retourné par l'algorithme 7.*

Algorithm 7: $mystere()$

Données: N : entier ;

Résultat: à chercher ;

Variable : i, m, z : entier ;

début

$i \leftarrow 0$; $m \leftarrow N$; $z \leftarrow 1$;

tant que $(m \geq z)$ **faire**

$m \leftarrow m - z$; $z \leftarrow z + 2$; $i \leftarrow i + 1$;

fin

retourner i ;

fin

Algorithm 8: $mystere()$

Données: X : entier ;

Résultat: à chercher ;

Variable : a, b, c, z : entier ;

début

$a \leftarrow 1$; $b \leftarrow 0$; $c \leftarrow X$; $z \leftarrow 0$;

tant que $(c > 0)$ **faire**

$z \leftarrow z + a + b$;

$b \leftarrow b + 2a + 1$;

$a \leftarrow a + 3$;

$c \leftarrow c - 1$;

fin

retourner z ;

fin

Question 4. *Montrer que l'algorithme 8 s'arrête.*

Question 5. *On note a_k, b_k, c_k et z_k les valeurs respectives des variables a, b, c et z après k passages dans la boucle "tant que". Montrer les invariants suivants :*

- $a_k = 3 * (X - c_k) + 1$;
- $b_k = 3 * (X - c_k)^2$;
- $z_k = (X - c_k)^3$.

Question 6. *Déduire des deux questions précédentes le résultat retourné par l'algorithme 8.*

Exercice 3 (Le damier infecté)**.**

Soit une matrice de dimension $n \times n$. Chaque case de la matrice est soit saine, soit infectée. Une case devient infectée **si et seulement si** *au moins deux de ses cases adjacentes sont infectées. Comme pour le jeu de la vie, le système évolue de proche en proche.*

Question 1. *Proposer des conditions nécessaires sur la configuration initiale pour que celle-ci puisse aboutir à l'infection totale du damier.*

Question 2. *Evaluer l'évolution de paramètres de description d'une configuration lors du passage vers une autre configuration (surface, distance, périmètre, connexité, diamètre...).*

Question 3. *Déduire de la question précédente qu'une configuration initiale contenant moins de n cases infectées ne peut pas aboutir à l'infection totale du damier.*