Fiche 100 **Niveau : 1** **Jaune**

Preuve par invariant de boucle

1 Comment ça marche

Une preuve d'algorithme par invariant de boucle utilise la démarche suivante. Nous prouvons tout d'abord que l'algorithme s'arrête en montrant qu'une condition d'exécution de boucle finit par ne plus être réalisée. Nous exhibons alors un **invariant de boucle**, c'est-à-dire une propriété P qui, si elle est valide **avant l'exécution d'un tour de boucle**, est aussi valide **après l'exécution du tour de boucle**. Nous vérifions alors que les conditions initiales rendent la propriété P vraie en entrée du premier tour de boucle. Nous en concluons que cette propriété est vraie en sortie du dernier tour de boucle. Un bon choix de la propriété P prouvera qu'on a bien produit l'objet recherché. La difficulté de cette méthode réside dans la détermination de l'invariant de boucle. Quand on l'a trouvé il est en général simple de montrer que c'est bien un invariant de boucle.

2 Exemples

2.1 Algorithme de division euclidienne par soustraction

$B := b$;
$R := a$;
$Q := 0$;
Tant que $R \geq B$ **faire**
    $R := R - B$;
    $Q := Q + 1$;
fintq ;

Remarquons que les conditions initiales donnent :

$$a = B * Q + R.$$

Montrons que la propriété $a = B * Q + R$ est un invariant de boucle : notons R', B', Q' les nouvelles valeurs en sortie de B, Q, R. Alors $R' = R - B$ et $Q' = Q + 1$. Ceci prouve que $B' * Q' + R' = B * Q + R$. De plus la quantité entière $R - B$ diminue strictement à chaque tour, donc le programme se termine et après la boucle on a :

$$a = B * Q + R \quad et \quad R < B.$$

2.2 Version binaire de l'algorithme de division euclidienne

On calcule avant toute chose par duplications successives le plus petit entier $n \geq 0$ tel que $2^n b > a$.

$B := b$;
$R := a$;
$Q := 0$;
$N := n$;
$Aux := 2^N B$;
Tant que $N > 0$ **faire**
    $Aux := Aux/2$;
    $N := N - 1$;
    si $R < Aux$ **alors**
        $Q := 2 * Q$;
    sinon
        $Q := 2 * Q + 1$;
        $R := R - Aux$;
    finsi ;
fintq ;

Montrons que les conditions :

$$\begin{cases} Aux = 2^N B \\ a = Aux * Q + R \\ 0 \leq R < Aux \\ N \geq 0 \end{cases}$$

sont un invariant de boucle. Ces conditions sont bien réalisées à l'état initial.
Nous noterons Aux', Q', R', N' les valeurs en sortie de Aux, Q, R, N. Si en entrée de boucle les conditions précédentes sont remplies alors : dans la boucle $Aux' = Aux/2$ et $N' = N - 1$ donc $Aux' = 2^{N'} B$. De plus :

– 1^{e}cas : $R < Aux'$. Dans ce cas $R' = R$, Aux est divisé par 2 tandis que Q est multiplié par 2. On a donc bien les conditions indiquées en sortie.
– 2^{e}cas : Si $R \geq Aux'$ alors on sait que :

$$Aux' \leq R < Aux = 2 * Aux'.$$

On a aussi $Aux' = Aux/2$, $Q' = 2 * Q + 1$, $R' = R - Aux'$ et $R' < Aux'$. On a donc bien les conditions attendues.

De plus N décroît strictement, donc le programme se termine avec $N = 0$. Quand $N = 0$ la variable Aux contient b, Q contient q et R contient r.

2.3 Algorithme d'Euclide de calcul du pgcd

$R0 := |a|$;
$R1 := |b|$; $\quad (b \neq 0)$
Tant que $R1 > 0$ **faire**
$\quad R := Reste\_Division(R0, R1)$;
$\quad R0 := R1$;
$\quad R1 := R$;
fintq ;

En sortie $R1 = 0$, et $R0 = \text{pgcd}(a, b)$.
Les conditions :

$$\begin{cases} \text{L'ensemble des diviseurs communs de } R0 \text{ et } R1 \text{ est} \\ \quad \text{l'ensemble des diviseurs communs de } a \text{ et } b. \\ \\ R_1 \geq 0 \end{cases}$$

constituent un invariant de boucle.
Remarquons qu'initialement l'ensemble des diviseurs communs de $R0$ et $R1$ est l'ensemble des diviseurs communs de a et de b. Notons $R0', R1'$ les nouvelles valeurs de $R0, R1$ en sortie d'un tour de boucle. Nous avons alors $R0' = R1$ et $R1' = R0 - QR1$ avec $0 \leq R1' < R1$. Donc tout diviseur de $R1$ et $R0$ est diviseur de $R0'$ et $R1'$, et réciproquement.
Cet algorithme se termine car $R1$ décroît strictement à chaque tour de boucle. A la fin $R1 = 0$, donc l'ensemble des diviseurs de $R0$ et de $R1$ est l'ensemble des diviseurs de $R0$, et par conséquent $R0 = \text{pgcd}(a, b)$.

2.4 Algorithme d'Euclide étendu

Là encore nous supposerons que $a \geq 0$ et $b > 0$. Le cas général s'en déduit. Notons $d = pgcd(a, b)$.
Voici un algorithme (algorithme d'Euclide étendu, adaptation de l'algorithme précédent) qui permet de trouver explicitement un couple (u, v) qui vérifie :

$$ua + vb = d.$$

$R0 := a; \quad (a \geq 0)$
$R1 := b; \quad (b > 0)$
$U0 := 1; U1 := 0;$
$V0 := 0; V1 := 1;$
Tant que $R1 > 0$ **faire**
$Q := Quotient\_Division(R0, R1)$;
$R := Reste\_Division(R0, R1)$;
$U := U0 - Q * U1$;
$V := V0 - Q * V1$;
$R0 := R1$; $R1 := R$;
$U0 := U1$; $U1 := U$;
$V0 := V1$; $V1 := V$;
fintq ;

Remarquons qu'il s'agit d'une amélioration de l'algorithme d'Euclide donné précédemment pour le calcul du pgcd. Comme précédemment l'algorithme se termine avec $R1 = 0$ et $R0 = \text{pgcd}(a, b)$.
Montrons que les conditions :

$$\begin{cases} U0a + V0b &= R0 \\ U1a + V1b &= R1 \\ R1 \geq 0 \end{cases}$$

sont un invariant de boucle. Pour cela notons $R0', R1', U0', U1', V0', V1'$ les nouvelles valeurs de $R0, R1, U0, U1, V0, V1$ en sortie d'un tour de boucle.
On a :

$$R0 = Q * R1 + R,$$

$$U := U0 - Q * U1,$$

$$V := V0 - Q * V1,$$

puis $R0' = R1$, $R1' = R = R0 - Q * R1$, $U0' = U1$, $U1' = U = U0 - Q * U1$, $V0' = V1$, $V1' = V = V0 - Q * V1$. Si bien que :

$$U0'a + V0'b = U1a + V1b = R1 = R0'.$$

La première condition est bien réalisée en sortie. De même on a :

$$U1'a + V1'b = U0a + V0b - Q * (U1a + V1b) = R0 - Q * R1 = R'1,$$

et la deuxième condition est aussi réalisée.
Il est facile de voir qu'à l'instant initial ces deux conditions sont bien réalisées. En sortie on a $R1 = 0$ et $R0 = \text{pgcd}(a, b)$ si bien que $U0$ et $V0$ contiennent une solution du problème.

2.5 Calcul d'une puissance

Soit $n \geq 1$ un entier. On veut calculer a^n (a est par exemple dans $\mathbb{Z}/n\mathbb{Z}$ ou dans $\mathbb{R}$, $\mathbb{C}$,...). On considère l'algorithme suivant :

$A := a$;
$N := n$;
$R := 1$;
Tant que $N > 0$ **faire**
  si N *pair* **alors**
    $A := A * A$;
    $N := N/2$;
  sinon
    $R := R * A$;
    $N := N - 1$;
  finsi ;
fintq ;

Cet algorithme se termine et en sortie R contient a^n.
Preuve : La valeur de $N \geq 0$ décroît strictement à chaque tour de boucle, donc l'algorithme se termine.
Au début on a $A^N \times R = a^n$. Si en entrée de boucle $A^N \times R = a^n$ alors il est facile de voir que dans les deux cas N pair ou N impair on a la même égalité en sortie de boucle. Mais à la fin on a $N = 0$ et par conséquent $R = a^n$.