Terminaison et correction

INFORMATIQUE COMMUNE - TP n° 2.1 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ programmer les algorithmes donnés en exemples.
- ☞ prouver la terminaison d'un algorithme simple.
- ☞ prouver la correction d'un algorithme simple.

A Terminaison

A1. Prouver la terminaison de l'algorithme 1 puis le traduire en Python.

Algorithme 1 Palindrome

Fonction PALINDROME(w)
  $n \leftarrow$ la taille de la chaîne de caractères w
  $i \leftarrow 0$
  $j \leftarrow n-1$
  tant que $i < j$ **répéter**
    si $w[i] = w[j]$ **alors**
      $i \leftarrow i+1$
      $j \leftarrow j-1$
    sinon
      renvoyer Faux
  renvoyer Vrai

A2. Prouver la terminaison de l'algorithme 2 puis le traduire en Python.

Algorithme 2 Est une puissance de deux

Fonction EST_PUISSANCE_DE_DEUX(n)
  si $n = 0$ **alors**
    renvoyer Faux
  sinon
    $m \leftarrow n \mod 2$
    tant que $m = 0$ **répéter**
      $n \leftarrow n//2$
      $m \leftarrow n \mod 2$
    renvoyer n = 1

Algorithme 3 Somme des n premiers entiers

```
Fonction INT_SUM(n)
    si n=0 alors
        renvoyer 0
    sinon
        renvoyer n + INT_SUM(n-1)
```

A3. Prouver la terminaison de l'algorithme récursif 3 puis le traduire en Python.

A4. Prouver la terminaison de l'algorithme récursif 4.

Algorithme 4 Exponentiation rapide a^n

```
Fonction EXP_RAPIDE(a,n)
    si n = 0 alors                                  ▷ Condition d'arrêt
        renvoyer 1
    sinon si n est pair alors
        p ← EXP_RAPIDE(a, n//2)                     ▷ Appel récursif
        renvoyer p × p
    sinon
        p ← EXP_RAPIDE(a, (n-1)//2)                 ▷ Appel récursif
        renvoyer p × p × a
```

B Algorithme d'Euclide du PGCD

Algorithme 5 Algorithme d'Euclide (optimisé)

Fonction PGCD(a, b) ▷ On suppose pour simplifier que $a \in \mathbb{N}$, $b \in \mathbb{N}^\star$ et $b \leqslant a$.
  $r \leftarrow a \bmod b$
  tant que $r > 0$ **répéter** ▷ On connaît la réponse si r est nul.
    $a \leftarrow b$
    $b \leftarrow r$
    $r \leftarrow a \bmod b$
  renvoyer b ▷ Le pgcd est b

On cherche à prouver la terminaison et la correction de l'algorithme d'Euclide 5. Dans ce but, on rappelle quelques éléments mathématiques importants.

Théorème 1 — Division euclidienne . Soient $a \in \mathbb{Z}$ et $b \in \mathbb{N}^\star$. Alors il existe un unique couple $(q, r) \in \mathbb{Z} \times \mathbb{N}$ tel que les deux critères suivants sont vérifiés :

$$\begin{cases} a = bq + r \\ 0 \leqslant r < b \end{cases}$$

Démonstration.

1. Existence : a et b étant donné, on pose $q = \lfloor \frac{a}{b} \rfloor$. Par définition de partie entière, on a : $0 \leqslant \frac{a}{b} - \lfloor \frac{a}{b} \rfloor < 1$. En multipliant par b, on obtient : $0 \leqslant a - b \times \lfloor \frac{a}{b} \rfloor < b$. En choisissant donc $q = \lfloor \frac{a}{b} \rfloor$ et $r = a - b \times \lfloor \frac{a}{b} \rfloor$, on a bien :
$$\begin{cases} a = bq + r \\ 0 \leqslant r < b \end{cases}$$
2. Unicité : supposons que l'on ait deux couples (q, r) et (q', r') appartenant à $\mathbb{Z} \times \mathbb{N}$: $a = bq + r = bq' + r'$ avec $0 \leqslant r < b$ et $0 \leqslant r' < b$. Cela peut également s'écrire : $b(q' - q) = r - r'$. Or, on a l'encadrement $-b < r - r' < b$. On en conclut que $-b < b(q' - q) < b$ et donc que $-1 < q' - q < 1$. Mais q et q' sont des entiers d'après nos hypothèses de départ. Donc, on en déduit de $q' - q = 0$. Il s'en suit que $q = q'$ et que $r = r'$. Il s'agit donc bien du même couple.

■

Théorème 2 — Existence du PGCD. Parmi tous les diviseurs communs de deux entiers a et b non nuls, il y en a **un** qui est le plus grand. Ce dernier est nommé plus grand commun diviseur de a et de b. On le note PGCD(a, b).

Démonstration. Soit $a \in \mathbb{N}^\star$. Tous les diviseurs de a sont bornés par $|a|$. On peut tenir le même raisonnement pour ceux de b. Donc, parmi les diviseurs de a et de b, il y en a donc un plus grand. ■

Théorème 3 — Propriété du PGCD. Soit a et b deux entiers.

1. Si $b = 0$, alors PGCD$(a, b) = a$.
2. Si $b \neq 0$, alors PGCD$(a, b) =$ PGCD$(b, a \mod b)$.

Démonstration. Démonstration de l'égalité de l'ensemble $\mathcal{D}_{ab}$ des diviseurs de a et de b et de l'ensemble $\mathcal{D}_{br}$ des diviseurs de b et de r par double inclusion.

$\mathcal{D}_{ab} \subset \mathcal{D}_{br}$ **:** La division euclidienne étant unique comme nous l'avons montré au théorème 1, il existe un entier q tel que $a = qb + r$. Ce qui peut s'écrire : $a - qb = r$. Si γ est un diviseur de a et de b, alors on peut écrire : $a - bq = \gamma a' + \gamma b' q = \gamma(a' - b' q) = r$. On a donc montrer qu'un diviseur de a et de b est un diviseur de r.

$\mathcal{D}_{br} \subset \mathcal{D}_{ab}$ **:** De même, si η est un diviseur de b et de r, alors on a : $a = bq + r = \eta(b' q + r')$, ce qui signifie que η est un diviseur de a.

Donc, $\mathcal{D}_{ab} = \mathcal{D}_{br}$. Ceci est vrai, y compris pour le plus grand des diviseurs de a et de b. ■

■ **Définition 1 — Suite des restes de la division euclidienne.** Soient a et b des entiers. On définit la suite des restes de la division euclidienne comme suit :

$$r_0 = |\, a \,| \tag{2}$$
$$r_1 = |\, b \,| \tag{3}$$
$$q_k = \lfloor r_{k-1} / r_k \rfloor, 1 \leqslant k \leqslant n \tag{4}$$

Alors on a :

$$r_{k-1} = q_k r_k + r_{k+1} \tag{5}$$
$$r_{k+1} = r_{k-1} \bmod r_k \tag{6}$$

Théorème 4 — Stricte décroissance de $(r_n)_{n\in\mathbb{N}}$. La suite des restes de la division euclidienne est positive, strictement décroissante et minorée par zéro.

B1. Coder l'algorithme 5 en Python.

B2. Grâce au théorème 3, coder une version récursive de l'algorithme du PGCD.

B3. Donner une preuve du théorème 4.

B4. Montrer que r est un variant de boucle pour l'algorithme d'Euclide.

B5. Prouver la correction de l'algorithme d'Euclide.

C Correction d'algorithmes classiques

C1. Prouver la correction partielle de l'algorithme 6 puis le traduire en Python en matérialisant l'invariant utilisé par des assertions.

C2. Prouver la correction partielle de l'algorithme de tri par sélection 7

C3. Prouver la correction de l'algorithme du tri par insertion 8.

Algorithme 6 Élément maximum d'un tableau

```
Fonction MAX(t)
    si t est vide alors
        renvoyer ∅
    sinon
        n ← la taille du tableau
        m = t[0]
        pour i = 1 à n − 1 répéter
            si m < t[i] alors
                m ← t[i]
        renvoyer m
```

Algorithme 7 Tri par sélection

```
Fonction GET_MIN_INDEX(t, i)
    n ← taille(t)
    min_index ← i
    pour j de i + 1 à n − 1 répéter
        si t[j] < t[min_index ] alors
            min_index ← j
    renvoyer min_index
Fonction TRIER_SELECTION(t)
    n ← taille(t)
    min_index ← GET_MIN_INDEX(t, 0)
    échanger(t,0, min_index)                ▷ Initialisation : le plus petit en tête
    pour i de 1 à n − 1 répéter
        min_index ← GET_MIN_INDEX(t, i)     ▷ indice du prochain plus petit
        échanger(t,i,min_index)             ▷ c'est le plus grand des triés !
```

Algorithme 8 Tri par insertion

```
Fonction INSERTION(t, i)
    à_insérer ← t[i]
    j ← i
    tant que t[j-1] > à_insérer et j>0  répéter
        t[j] ← t[j-1]                       ▷ faire monter les éléments
        j ← j-1
    t[j] ← à_insérer                        ▷ insertion de l'élément
Fonction TRIER_INSERTION(t)
    n ← taille(t)
    pour i de 1 à n-1 répéter
        INSERTION(t,i)
```