

# 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

---

```
1: Fonction PALINDROME( $w$ )
2:    $n \leftarrow$  la taille de la chaîne de caractères  $w$ 
3:    $i \leftarrow 0$ 
4:    $j \leftarrow n - 1$ 
5:   tant que  $i < j$  répéter
6:     si  $w[i] = w[j]$  alors
7:        $i \leftarrow i + 1$ 
8:        $j \leftarrow j - 1$ 
9:     sinon
10:      renvoyer Faux
11:   renvoyer Vrai
```

---

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

---

### Algorithme 2 Est une puissance de deux

---

```
1: Fonction EST_PUISSANCE_DE_DEUX( $n$ )
2:   si  $n = 0$  alors
3:     renvoyer Faux
4:   sinon
5:      $m \leftarrow n \bmod 2$ 
6:     tant que  $m = 0$  répéter
7:        $n \leftarrow n // 2$ 
8:        $m \leftarrow n \bmod 2$ 
9:   renvoyer  $n = 1$ 
```

---

---

**Algorithme 3** Somme des  $n$  premiers entiers

---

```
1: Fonction INT_SUM(n)
2:   si n=0 alors
3:     renvoyer 0
4:   sinon
5:     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  $a^n$ 

---

```
1: Fonction EXP_RAPIDE(a,n)
2:   si n = 0 alors                                     ▷ Condition d'arrêt
3:     renvoyer 1
4:   sinon si n est pair alors
5:     p ← EXP_RAPIDE(a, n//2)                           ▷ Appel récursif
6:     renvoyer p × p
7:   sinon
8:     p ← EXP_RAPIDE(a, (n-1)//2)                         ▷ Appel récursif
9:     renvoyer p × p × a
```

---

## B Algorithme d'Euclide du PGCD

---

### Algorithme 5 Algorithme d'Euclide (optimisé)

---

```

1: Fonction PGCD( $a, b$ )                                ▷ On suppose pour simplifier que  $a \in \mathbb{N}$ ,  $b \in \mathbb{N}^*$  et  $b \leq a$ .
2:    $r \leftarrow a \bmod b$ 
3:   tant que  $r > 0$  répéter                               ▷ On connaît la réponse si  $r$  est nul.
4:      $a \leftarrow b$ 
5:      $b \leftarrow r$ 
6:      $r \leftarrow a \bmod b$ 
7:   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  $b \in \mathbb{N}^*$ . Alors il existe un unique couple  $(q, r) \in \mathbb{Z} \times \mathbb{N}$  tel que les deux critères suivants sont vérifiés :

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

*Démonstration.* 1. Existence :  $a$  et  $b$  étant donné, on pose  $q = \lfloor \frac{a}{b} \rfloor$ . Par définition de partie entière, on a :  $0 \leq \frac{a}{b} - \lfloor \frac{a}{b} \rfloor < 1$ . En multipliant par  $b$ , on obtient :  $0 \leq a - b \times \lfloor \frac{a}{b} \rfloor < b$ . En choisissant donc  $q = \lfloor \frac{a}{b} \rfloor$  et  $r = a - b \times \lfloor \frac{a}{b} \rfloor$ , on a bien :

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

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  $0 \leq r < b$  et  $0 \leq r' < b$ . 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  $a \in \mathbb{N}^*$ . 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( $b, a \bmod b$ ).

*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  $\gamma$  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 montré qu'un diviseur de  $a$  et de  $b$  est un diviseur de  $r$ .

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

$$r_0 = |a| \quad (2)$$

$$r_1 = |b| \quad (3)$$

$$q_k = \lfloor r_{k-1} / r_k \rfloor, 1 \leq k \leq n \quad (4)$$

Alors on a :

$$r_{k-1} = q_k r_k + r_{k+1} \quad (5)$$

$$r_{k+1} = r_{k-1} \bmod r_k \quad (6)$$

**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

---

```

1: Fonction MAX( $t$ )
2:   si  $t$  est vide alors
3:     renvoyer  $\emptyset$ 
4:   sinon
5:      $n \leftarrow$  la taille du tableau
6:      $m = t[0]$ 
7:     pour  $i = 1$  à  $n - 1$  répéter
8:       si  $m < t[i]$  alors
9:          $m \leftarrow t[i]$ 
10:    renvoyer  $m$ 

```

---



---

**Algorithme 7** Tri par sélection

---

```

1: Fonction GET_MIN_INDEX( $t, i$ )
2:    $n \leftarrow$  taille( $t$ )
3:    $\text{min\_index} \leftarrow i$ 
4:   pour  $j$  de  $i + 1$  à  $n - 1$  répéter
5:     si  $t[j] < t[\text{min\_index}]$  alors
6:        $\text{min\_index} \leftarrow j$ 
7:   renvoyer  $\text{min\_index}$ 
8: Fonction TRIER_SELECTION( $t$ )
9:    $n \leftarrow$  taille( $t$ )
10:   $\text{min\_index} \leftarrow$  GET_MIN_INDEX( $t, 0$ )
11:  échanger( $t, 0, \text{min\_index}$ )
12:  pour  $i$  de 1 à  $n - 1$  répéter
13:     $\text{min\_index} \leftarrow$  GET_MIN_INDEX( $t, i$ )
14:    échanger( $t, i, \text{min\_index}$ )

```

▷ Initialisation : le plus petit en tête

▷ indice du prochain plus petit

▷ c'est le plus grand des triés!

---



---

**Algorithme 8** Tri par insertion

---

```

1: Fonction INSERTION( $t, i$ )
2:    $\text{à\_insérer} \leftarrow t[i]$ 
3:    $j \leftarrow i$ 
4:   tant que  $t[j-1] > \text{à\_insérer}$  et  $j > 0$  répéter
5:      $t[j] \leftarrow t[j-1]$ 
6:      $j \leftarrow j-1$ 
7:    $t[j] \leftarrow \text{à\_insérer}$ 
8: Fonction TRIER_INSERTION( $t$ )
9:    $n \leftarrow$  taille( $t$ )
10:  pour  $i$  de 1 à  $n-1$  répéter
11:    INSERTION( $t, i$ )

```

▷ faire monter les éléments

▷ insertion de l'élément

---