

Travaux Pratiques – Algorithme d'Euclide (Python)

Nom et prénom : Mouhcine Elarfaoui

Filière : Génie industriel et logistique

Module : Mathématique

Objectif

L'objectif principal de ce TP est de maîtriser l'implémentation et le fonctionnement de l'algorithme d'Euclide pour calculer le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels.

1 Rappel théorique

Le PGCD de deux entiers naturels a et b est le plus grand entier qui divise à la fois a et b sans laisser de reste.

Principe de l'algorithme d'Euclide : Si $a > b$, alors $\text{PGCD}(a, b) = \text{PGCD}(b, a \bmod b)$. On répète jusqu'à ce que le reste soit nul.

2 Exemples manuels

Exemple 1 : PGCD (48, 18)

Étape	a	b	Reste ($a \bmod b$)
1	48	18	12
2	18	12	6
3	12	6	0

PGCD= 6

Exemple 2 : PGCD (81, 57)

Etape	a	b	Reste (a mod b)
1	81	57	24
2	57	24	9
3	24	9	6
4	9	6	3
5	6	3	0

PGCD= 3

Exemple 3 : PGCD (270,192)

Étape	a	b	Reste
1	270	192	78
2	192	78	36
3	78	36	6
4	36	6	0

PGCD = 6

3 Implémentation Python

```
algorithme_euclide.py X
algorithme_euclide.py > pgcd
1 def pgcd(a, b):
2     """Calcule le PGCD de deux nombres en affichant les étapes"""
3     print(f"\nCalcul du PGCD de {a} et {b} :")
4     while b != 0:
5         reste = a % b
6         print(f"{a} = {b} * ({a // b}) + {reste}")
7         a, b = b, reste
8     print(f"Le PGCD est {a}")
9     return a
10
11 # --- Programme principal ---
12 x = int(input("Entrez le premier nombre : "))
13 y = int(input("Entrez le deuxième nombre : "))
14
15 resultat = pgcd(x, y)
16 print(f"\n✓ Résultat final : PGCD({x}, {y}) = {resultat}")
17
```

4 Vérification

Exemple 1 : PGCD (48, 18)

```
ds/algorithme_euclide/algorithme_euclide.py
Entrez le premier nombre : 48
Entrez le deuxième nombre : 18

Calcul du PGCD de 48 et 18 :
48 = 18 * (2) + 12
18 = 12 * (1) + 6
12 = 6 * (2) + 0
Le PGCD est 6

✓ Résultat final : PGCD(48, 18) = 6
```

Exemple 2 : PGCD (81, 57)

```
ds/algorithme_euclide/algorithme_euclide.py
● Entrez le premier nombre : 81
Entrez le deuxième nombre : 57

Calcul du PGCD de 81 et 57 :
81 = 57 * (1) + 24
57 = 24 * (2) + 9
24 = 9 * (2) + 6
9 = 6 * (1) + 3
6 = 3 * (2) + 0
Le PGCD est 3

 Résultat final : PGCD(81, 57) = 3
```

Exemple 3 : PGCD (270, 192)

```
● ds/algorithme_euclide/algorithme_euclide.py
Entrez le premier nombre : 270
Entrez le deuxième nombre : 192

Calcul du PGCD de 270 et 192 :
270 = 192 * (1) + 78
192 = 78 * (2) + 36
78 = 36 * (2) + 6
36 = 6 * (6) + 0
Le PGCD est 6

 Résultat final : PGCD(270, 192) = 6
```

Test	a	b	Résultat attendu	Résultat obtenu
1	48	18	6	6
2	81	57	3	3
4	270	192	6	6

5 Conclusion

L'algorithme d'Euclide est simple, rapide et efficace pour calculer le PGCD. Sa complexité est logarithmique, ce qui le rend bien plus performant que de tester tous les diviseurs.

PARTIE 2 : ÉQUATIONS DIOPHANTIENNES LINÉAIRES

6. Objectif

Déterminer toutes les solutions entières $(x, y) \in \mathbb{Z}^2$ de l'équation :

$$ax + by = c$$

7. Méthode à suivre

Étape 1 — Vérifier l'existence de solutions entières

- Calculer $d = \text{pgcd}(a, b)$ à l'aide de l'algorithme d'Euclide.
- Si $d \nmid c$: il n'existe aucune solution entière.
- Si $d \mid c$: il existe au moins une solution entière.

Étape 2 — Trouver une solution particulière

- Utiliser l'algorithme d'Euclide étendu pour écrire :

$$D = ax_0 + by_0$$

- En déduire une solution particulière :

$$(x_p, y_p) = \left(x_0 \cdot \frac{c}{d}, y_0 \cdot \frac{c}{d} \right)$$

Étape 3 — Écrire la solution générale

- Toutes les solutions de l'équation $ax+by=c$ s'écrivent sous la forme :

$$(x, y) = (x_p + \alpha k, y_p + \beta k), \quad k \in \mathbb{Z}$$

où $x_p, y_p, \alpha, \beta \in \mathbb{Z}$ sont des constantes fixées.

8. Implémentation complète en Python

```

# audidddd.py > ...
7   def euclide_etendu(a, b):
13
14     print("\n*** Algorithme d'Euclide Étendu ===")
15     print(f"Objectif : Trouver PGCD({a}, {b}) et (u, v) tels que {a}u + {b}v = PGCD({a}, {b})\n")
16
17     # Étape 1 : Algorithme d'Euclide classique
18     etapes = []
19     A, B = a, b
20     while b != 0:
21         q = a // b
22         r = a % b
23         etapes.append((a, b, q, r))
24         print(f"\n{a} = {b} * {q} + {r}")
25         a, b = b, r
26
27     d = a # PGCD
28     print(f"\nPGCD({A}, {B}) = {d}\n")
29
30     # Étape 2 : Remontée de l'algorithme (calcul de u et v)
31     print("==> Étapes de la remontée (calcul de u et v) ==>\n")
32
33     u1, v1 = 1, 0
34     u2, v2 = 0, 1
35
36     for i, (a_i, b_i, q_i, r_i) in enumerate[Any](etapes):
37         u1, u2 = u2, u1 - q_i * u2
38         v1, v2 = v2, v1 - q_i * v2
39         print(f"\nÉtape {i+1}: q = {q_i} → u = {u1}, v = {v1}\n")
40
41     u, v = u1, v1
42     print(f"\nCoefficients de Bézout : u = {u}, v = {v}\n")
43     print(f"\nVérification : {A}*(u) + {B}*(v) = {A*u + B*v}\n")
44
45     # Vérification de la relation
46     if A * u + B * v == d:
47         print("✓ Relation vérifiée : a*u + b*v = PGCD(a,b)\n")
48     else:
49         print("✗ Relation non vérifiée.\n")
50
51     return d, u, v
52
53
54 def resoudre_diophantienne(a, b, c):
55     """
56     Résout l'équation diophantienne ax + by = c.
57     Utilise l'algorithme d'Euclide étendu pour trouver les solutions.
58     """
59
60     print("\n*** Résolution de l'équation diophantienne ===")
61     print(f"\nÉquation : {a}x + {b}y = {c}\n")
62
63     # Étape 1 : Calcul du PGCD et coefficients de Bézout
64     d, u, v = euclide_etendu(a, b)
65
66     # Étape 2 : Vérifier l'existence de solutions
67     if c % d != 0:
68         print(f"\n✗ Pas de solution entière car {d} ne divise pas {c}.")
69         return
70     else:
71         print(f"\n✓ Il existe des solutions entières car {d} divise {c}.\n")

```

```
71 |     print(f"✓ Il existe des solutions entières car {d} divise {c}.\n")  
72 |  
73 | # Étape 3 : Calcul d'une solution particulière  
74 | k = c // d  
75 | x_p = u * k  
76 | y_p = v * k  
77 |  
78 | print(f"→ Solution particulière : (xp, yp) = ({x_p}, {y_p})")  
79 | print(f"Vérification : {a}*(xp) + {b}*(yp) = {a*x_p + b*y_p} = {c}")  
80 |  
81 | # Étape 4 : Écriture de la solution générale  
82 | alpha = b // d  
83 | beta = -a // d  
84 | print("\n--- Solution générale ---")  
85 | print(f"(x, y) = ({x_p}) + {alpha}k, (yp) + {beta}k), k ∈ ℤ")  
86 |  
87 | print("\n✓ Résolution complète terminée.")  
88 |  
89 |  
90 | # --- Programme principal ---  
91 | print("--- TP : Équations diophantiennes linéaires ---")  
92 | a = int(input("Entrez la valeur de a : "))  
93 | b = int(input("Entrez la valeur de b : "))  
94 | c = int(input("Entrez la valeur de c : "))  
95 |  
96 | resoudre_diophantienne(a, b, c)  
97 | Ctrl+I to chat, Ctrl+K to generate
```

Exemples d'exécution

Exemple 1 :

```
c11aaaa.py
==== TP : Équations diophantiennes linéaires ====
Entrez la valeur de a : 56
Entrez la valeur de b : 98
Entrez la valeur de c : 14

==== Résolution de l'équation diophantienne ====
Équation :  $56x + 98y = 14$ 

==== Algorithme d'Euclide Étendu ====
Objectif : Trouver PGCD(56, 98) et  $(u, v)$  tels que  $56u + 98v = \text{PGCD}(56, 98)$ 

 $56 = 98 \times 0 + 56$ 
 $98 = 56 \times 1 + 42$ 
 $56 = 42 \times 1 + 14$ 
 $42 = 14 \times 3 + 0$ 

→ PGCD(56, 98) = 14

==== Étapes de la remontée (calcul de u et v) ====
Étape 1:  $q = 0 \rightarrow u = 0, v = 1$ 
Étape 2:  $q = 1 \rightarrow u = 1, v = 0$ 
Étape 3:  $q = 1 \rightarrow u = -1, v = 1$ 
Étape 4:  $q = 3 \rightarrow u = 2, v = -1$ 

→ Coefficients de Bézout :  $u = 2, v = -1$ 
→ Vérification :  $56*(2) + 98*(-1) = 14$ 
✓ Relation vérifiée :  $a*u + b*v = \text{PGCD}(a,b)$ 

✓ Il existe des solutions entières car 14 divise 14.

→ Solution particulière :  $(x_p, y_p) = (2, -1)$ 
Vérification :  $56*(2) + 98*(-1) = 14 = 14$ 
```

```
→ Solution particulière :  $(x_p, y_p) = (2, -1)$ 
Vérification :  $56*(2) + 98*(-1) = 14 = 14$ 

==== Solution générale ====
 $(x, y) = (2 + 7k, -1 + -4k), k \in \mathbb{Z}$ 

✓ Résolution complète terminée.
```

Exemple 2

```

==== IP : Équations diophantiennes linéaires ====
Entrez la valeur de a : 21
Entrez la valeur de b : 15
Entrez la valeur de c : 03

==== Résolution de l'équation diophantienne ====
Équation :  $21x + 15y = 3$ 

==== Algorithme d'Euclide Étendu ====
Objectif : Trouver PGCD(21, 15) et (u, v) tels que  $21u + 15v = \text{PGCD}(21, 15)$ 

 $21 = 15 \times 1 + 6$ 
 $15 = 6 \times 2 + 3$ 
 $6 = 3 \times 2 + 0$ 

■ PGCD(21, 15) = 3

==== Étapes de la remontée (calcul de u et v) ====
Étape 1:  $q = 1 \rightarrow u = 0, v = 1$ 
Étape 2:  $q = 2 \rightarrow u = 1, v = -1$ 
Étape 3:  $q = 2 \rightarrow u = -2, v = 3$ 

■ Coefficients de Bézout :  $u = -2, v = 3$ 
■ Vérification :  $21*(-2) + 15*(3) = 3$ 
✓ Relation vérifiée :  $a*u + b*v = \text{PGCD}(a,b)$ 

✓ Il existe des solutions entières car 3 divise 3.

■ Solution particulière :  $(x_p, y_p) = (-2, 3)$ 
Vérification :  $21*(-2) + 15*(3) = 3 = 3$ 

```

```

■ Solution particulière :  $(x_p, y_p) = (-2, 3)$ 
Vérification :  $21*(-2) + 15*(3) = 3 = 3$ 

==== Solution générale ====
 $(x, y) = (-2 + 5k, 3 + -7k), k \in \mathbb{Z}$ 

✓ Résolution complète terminée.

```

Conclusion

L'algorithme d'Euclide est efficace et rapide, avec une complexité de $O(\log(\min(a,b)))$. Son extension permet de résoudre des équations diophantiennes linéaires, démontrant l'utilité pratique de cet algorithme fondamental en arithmétique.