

Sujet d'Approfondissement 2 – Le calcul du PGCD

A / Algorithmes naïfs

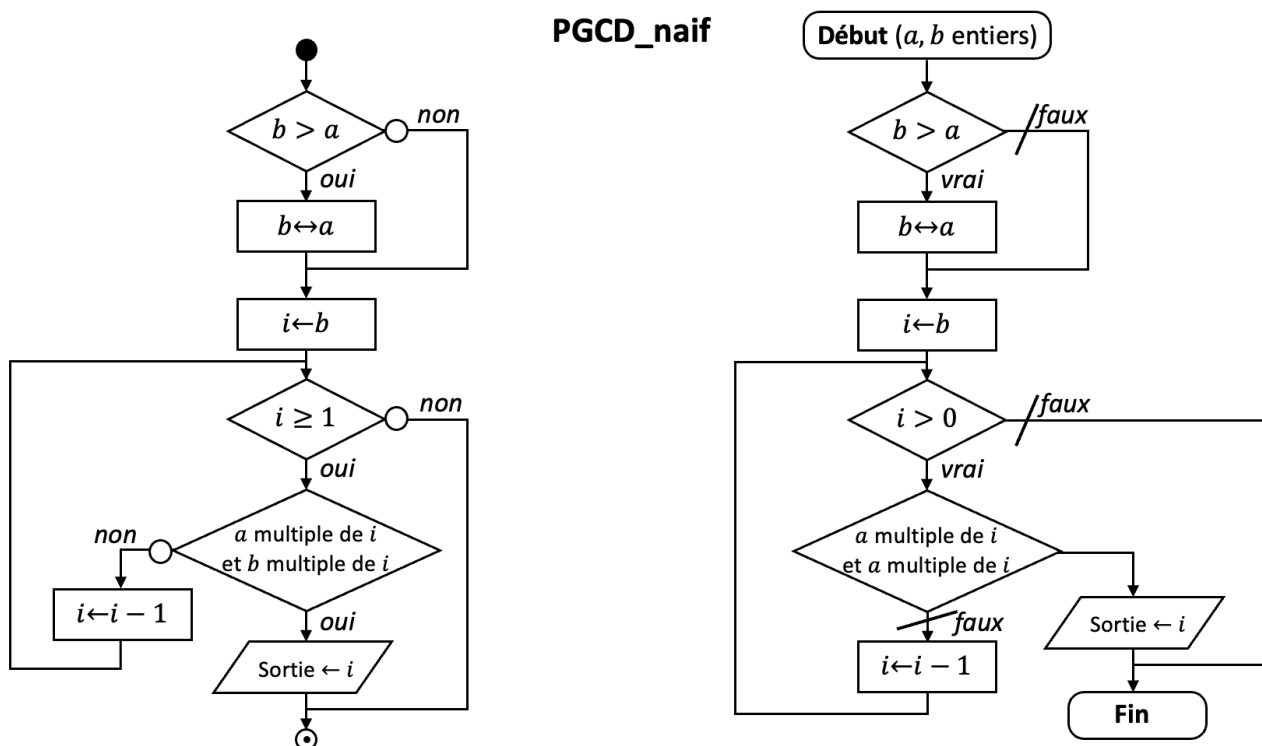
A.1 – Rappels sur le PGCD

- Notation : Le PGCD (**Plus Grand Commun Diviseur**) de deux nombres entiers a et b se note $\text{PGCD}(a, b)$.
- Propriété : on a $\text{PGCD}(a, b) = \text{PGCD}(-a, b) = \text{PGCD}(a, -b) = \text{PGCD}(-a, -b)$ donc si a ou b est négatif, le PGCD est égal à celui de leurs valeurs absolues. On ne considèrera donc que des **entiers positifs** par la suite pour simplifier nos algorithmes.
- Avec les mains : le $\text{PGCD}(a, b)$ est toujours $\leq \min(a, b)$: plus petit que le plus petit des deux nombres. Il correspond au plus grand nombre entier tel que a et b soient tous les deux divisibles par ce nombre (multiples).
- Exemple : $\text{PGCD}(10, 5) = 5$; $\text{PGCD}(10, 6) = 2$; $\text{PGCD}(10, 3) = 1$ (PGCD = 1 si pas d'autre diviseur commun).

A.2 – L'algorithme

Dans la suite de l'année, nous appellerons algorithme « naïf » un algorithme intuitif, qui permette de résoudre un problème de façon simple à coder, mais pas forcément optimisée en termes de temps d'exécution.

On donne ci-dessous l'algorithme de détermination du PGCD entre deux entiers a et b



Q1 – Écrire une fonction `pgcd_naïf(a, b)` prenant en argument deux nombres entiers a, b (supposés positifs) et renvoyant la valeur du PGCD de ces deux nombres.

Contrainte : l'algorithme s'appuiera sur une boucle `for`.

Q2 – Tester votre fonction pour déterminer :

- Le PGCD entre 36 et 27, qui devrait être 9
- Le PGCD entre 15 et 3, qui devrait être 3
- Le PGCD entre 16 et 3, qui devrait être 1

B / Algorithme d'Euclide

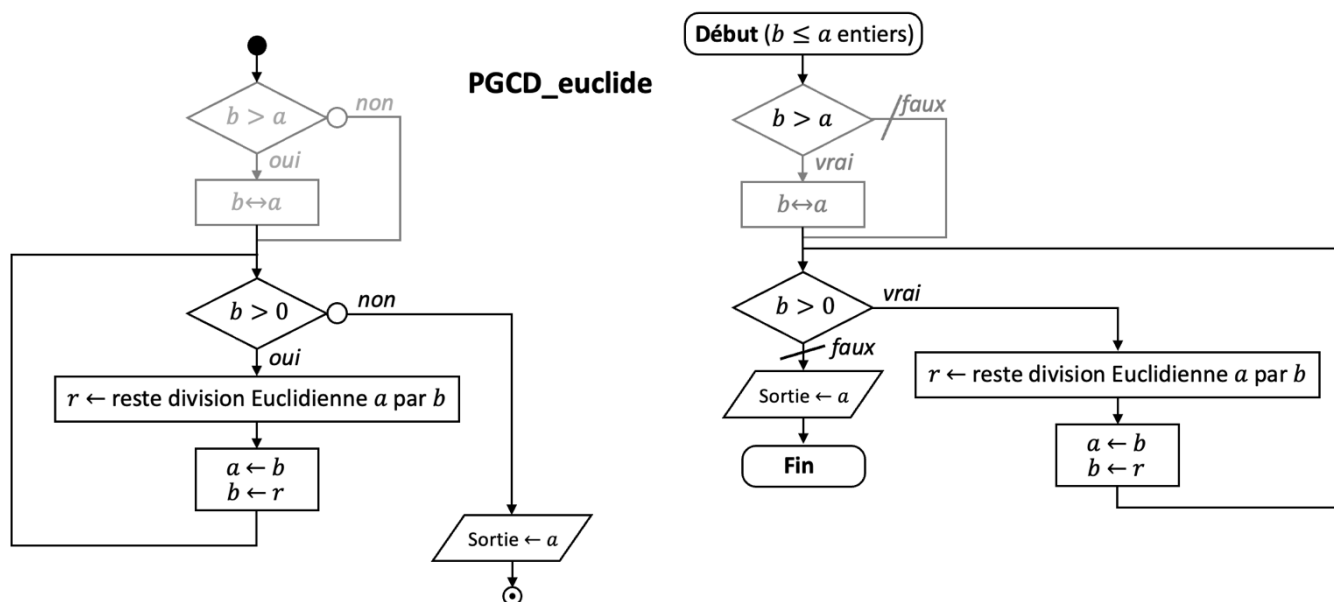
Euclide remarque que pour deux entiers strictement positifs $b \leq a$, le $\text{PGCD}(a, b) = \text{PGCD}(b, r)$ où r est le reste de la division Euclidienne de a par b (rappel : $a = bq + r$ avec q le quotient et $0 \leq r < b$ le reste.). Cela permet de baisser considérablement le nombre de calculs car si $b \ll a$ alors $r \ll a$, auquel cas on passe à chaque itération d'un calcul sur $(a \text{ et } b)$ avec a grand, à un calcul sur $(b \text{ et } r)$.

L'algorithme d'Euclide se décrit donc ainsi : soient deux entiers positifs $b \leq a$:

- Si $b = 0$ alors l'algorithme se termine et renvoie la valeur de a
- Sinon, l'algorithme calcule le reste r de la division Euclidienne de a par b . Puis on recommence avec
 - a prenant la valeur de b
 - b prenant la valeur de r

À chaque itération, la valeur de a décroît (car $b \leq a$), la valeur de b également (car le reste de la division Euclidienne de a par b est forcément $\leq \min(a, b) = b$). Par conséquent, la valeur de r décroît aussi à chaque itération.

On donne ci-dessous l'algorithme d'Euclide pour le calcul de PGCD entre deux entiers a et b :



Q3 – Écrire une fonction `pgcd_euclide(a,b)` prenant en argument deux nombres entiers a, b (supposés positifs) et renvoyant la valeur du PGCD de ces deux nombres en appliquant l'algorithme d'Euclide.

Remarque : l'algorithme s'appuie sur une boucle `while`.

Q4 – Tester votre fonction sur les mêmes cas tests qu'à la question **Q2**.

C / Comparaison de l'ordre de complexité des deux algorithmes

On donne ci-dessous un script à copier-coller à la suite de votre code.

```
import time
from random import randint

def comparaison(N): # N est le nombre de digits des nombres tirés aléatoirement
    a, b = randint(10**(N-1), 10**N), randint(10**(N-1), 10**N-1) # tirage aléatoire
    # de entiers à N digits (entre 10N-1 et 10N - 1)
    tic = time.perf_counter() # Valeur précise de l'heure (processeur) actuelle ?
    pgcd_naif(a,b) # on exécute un code dont on veut estimer le temps d'exécution
    tac = time.perf_counter() # Quelle heure est-il ?
    print("Le temps écoulé par l'algorithme naïf est", tac - tic, "secondes")
    # Le différentiel entre tic et tac est le temps d'exécution

    tic = time.perf_counter() # Idem, mais avec l'algorithme d'Euclide
    pgcd_euclide(a,b) # Pour que la comparaison ait du sens, c'est sur les ...
    tac = time.perf_counter() # ... mêmes entiers qu'on la fait
    print("Le temps écoulé par l'algorithme d'Euclide est", tac - tic, "secondes")
```

Q5 – Exécuter cette fonction pour comparer les temps d'exécution des 2 méthodes pour des nombres à 5, 6, 7 et 8 chiffres. Que constate-t-on ? Comment le temps d'exécution de la méthode naïve dépend-il de la taille du nombre ?