Étude de l'algorithme d'Euclide

TD-P#2 – Remise à niveau pour l'informatique

1 Introduction

L'algorithme d'Euclide est une méthode classique et efficace pour calculer le plus grand commun diviseur (PGCD) de deux nombres entiers. L'algorithme ne requiert pas de connaître la factorisation de ces deux nombres. Ce TP vise à étudier la complexité de cet algorithme en comparant deux méthodes itératives et la méthode récursive. Il s'agit aussi de présenter le cas pire de l'algorithme trouver sa complexité temporelle. Présenter les codes, résultats, discussions et conclusions dans un rapport LaTeX, Python note book ou autre.

2 Algorithme d'Euclide

2.1 Méthode Originale

Dans la méthode originale de l'algorithme, il n'y a pas de division ni de calcul du reste mais une simple soustraction, il s'agit d'appliquer les propriétés suivantes:

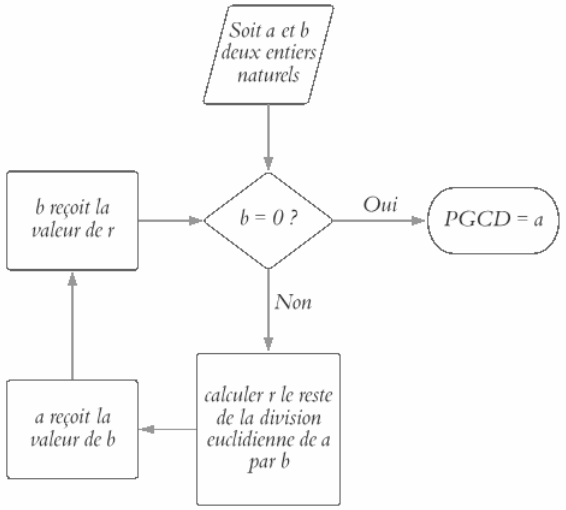
- Si $a > b$ alors $\text{pgcd}(a, b) = \text{pgcd}(a - b, b)$ sinon $\text{pgcd}(a, b) = \text{pgcd}(a, b - a)$
- $\text{pgcd}(a, a) = a$

Questions 1.

1. Écrire un algorithme itératif qui calcule le pgcd avec la méthode originale.
2. Écrire une fonction `gcd_original(a,b)`
3. Compter le nombre d'étapes

2.2 Méthode Itérative

Donald Knuth, dans "The Art of Computer Programming", propose une version itérative de l'algorithme d'Euclide.
Pour deux entiers $a > b \geq 0$, le calcul du PGCD se fait selon l'organigramme suivant :



Questions 2.

1. Écrire un algorithme itératif qui calcule le pgcd de deux entiers a, b.
2. Écrire une fonction `gcd_iterative(a,b)`
3. Compter le nombre d'étapes

2.3 Méthode Récursive

L'algorithme d'Euclide récursif utilise une approche similaire, mais il applique la division euclidienne de manière récursive.

- Si $a > b \geq 0$ alors $\text{pgcd}\,(a, b) = \text{pgcd}\,(b, a \bmod b)$
- $\text{pgcd}\,(a, 0) = a$

QUESTIONS 3.

1. Écrire un algorithme récursif qui calcule le pgcd.
2. Écrire une fonction `gcd_recursive(a,b)`
3. Compter le nombre d'étapes

3 Comparaison des Performances

Pour comparer les performances des trois versions de l'algorithme d'Euclide, on mesure le nombres d'étapes (nombres de divisions) pour différentes paires de nombres entiers aléatoires. En 1841, P.-J.-E. Finck démontre que le nombre d'étapes est borné par $2\log_2 b + 117$. Cela prouve que l'algorithme d'Euclide s'exécute en temps polynomial en la taille de l'entrée (nombre de chiffres pour écrire les nombres a et b). En 1844, Gabriel Lamé raffine les résultats de Finck : il démontre que l'algorithme effectue au plus 5 fois le nombre de chiffres en base 10 du plus petit nombre b[1].

QUESTIONS 4.

1. Écrire une fonction `measure_steps(func, a, b)` qui retourne le nombre d'étapes de l'appel de la fonction `func(a,b)`.
2. Écrire une fonction `Euclide_steps_compare` qui trace les courbes mesurant le nombres d'étapes des trois algorithmes avec `matplotlib` pour des valeurs de $1 \leq a \leq 256$ et b un entier aléatoire plus petit que a.
3. Écrire une nouvelle fonction `Euclide_moy_max_steps_compare` qui compare les courbes du nombres d'étapes $I(a, b)$ des appels fonctions itératives et récursives pour le calcul du PGCD de a et b :
   (a) Valeur maximum : pour $1 \leq a \leq 256$, $\max_{b<a} I(a, b)$
   (b) Valeur moyenne : pour $1 \leq a \leq 256$,$(1/a) \sum_{b<a} I(a, b)$

   Commenter les courbes.
4. Ajouter les courbes $log(a)$ et $1 + \log_\phi(a)$ où ϕ est le nombre d'or $\frac{1+\sqrt{5}}{2}$.
5. En déduire la complexité du nombre d'étapes pour l'algorithme d'Euclide

4 Étude du cas pire

Les résultats qui suivent sont essentiellment dûs à Lamé. Le pire des cas de l'algorithme d'Euclide est atteint pour le calcul du pgcd de deux valeurs consécutives de la suite de Fibonacci définie par

$$F_0 = 0, \quad F_1 = 1, \qquad \forall n \geq 0, \quad F_{n+2} = F_{n+1} + F_n$$

les premiers termes de la suite sont :

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, \ldots$$

Quand l'algorithme d'Euclide est appelé sur deux nombres de Fibonacci consécutifs F_{k+2} et F_{k+1} , en utilisant le pgcd original, obtient $\text{pgcd}\,(F_{k+2}, F_{k+1}) = \text{pgcd}\,(F_{k+1}, F_k)$, en itérant le procédé k fois :

$$\text{pgcd}\,(F_{k+2}, F_{k+1}) = \text{pgcd}\,(F_{k+1}, F_k) = \cdots = pgcd(F_2, F_1) = \text{pgcd}\,(F_1, F_0) = 1$$

et comme pour chaque division euclidienne invoquée par l'algorithme le quotient est 1, cette situation est la pire du point de vue de la complexité.

QUESTIONS 5.

1. Écrire une fonction itérative `Fibonacci(n)` qui retourne le n-ième terme de la suite de Fibonacci.

[1] Wikipedia

2. Écrire une fonction `FibonacciSequence(n,m)` qui retourne la liste des termes de la suite de Fibonacci de rang n à m.

3. Vérifier grahiquement la formule de Binet : F_n est équivalent à $\phi^n/\sqrt{5}$ où ϕ est le nombre d'or $\frac{1+\sqrt{5}}{2}$ en traçant les courbes.

4. Écrire une fonction `Euclide_Fibo_steps_compare(n,m)` pour des valeurs consécutives de la suite de Fibonacci de rang n à m pour la calcul du PGCD avec les algorithmes itératif et récursif.

5. Ajouter la courbe du nombre d'étapes de l'algorithme itératif avec $1 + \log(b)/\log(\phi)$ qui vérifie le résultat de Lamé sur la majoration du nombre d'étapes dans le cas pire.

6. Théorème de Lamé : l'algorithme effectue au plus 5 fois le nombre de chiffres en base 10 du plus petit nombre b

   (a) Montrer que $F_{n+5} > 10F_n$ pour $n \geq 2$. En déduire que F_{n+5} a un chiffre de plus que la représentation décimale de F_n.

   (b) Vérifier que pour $k.5 + 1 < n \leq (k + 1).5 + 1$, on a F_n s'écrit avec au mois $k + 1$ chiffres en base 10.

   (c) Soit k le nombre de chiffres de la représentation en décimal de b ($b < 10^k$). Soit $j + 1$ le nombre d'étapes dans l'application de l'algorithme d'Euclide on admet que $b \geq F_{j+2}$. Montrer que $j + 1 \leq 5k$. Conclure.

5 Rapport

Présenter les codes, résultats, discussions et conclusions dans un rapport LaTeX, Python Notebook ou autre.