Étude de la suite de Fibonacci

TD-P#4A – Remise à niveau pour l'informatique

1 Introduction

Il s'agit d'étudier la suite de Fibonacci et de comparer la complexité de deux algorithmes. Présenter les codes, résultats, discussions et conclusions dans un rapport LaTeX, Python note book ou autre.

2 La suite de Fibonacci

La suite de Fibonacci est 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 :

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

2.1 La méthode itérative

La méthode itérative consiste à calculer le n-ième terme de la suite après avoir calculé tous les précédents.

Questions 1.

1. Écrire une fonction itérative `Fibonacci_iterative(n)` qui retourne le n-ième terme de la suite de Fibonacci.
2. Compter le nombre d'étapes.
3. Donner une estimation de la complexité du nombre d'étapes de la fonction en justifiant la réponse.

2.2 La méthode récursive

Il s'agit ici d'utiliser la formule qui définit la suite.

1. Écrire une fonction récursive `Fibonacci_recursive(n)` qui retourne le n-ième terme de la suite de Fibonacci.
2. Compter le nombre d'appels récursifs.
3. Donner une estimation de la complexité du nombre d'appels récursifs de la fonction en justifiant la réponse.

2.3 La méthode matricielle

On définit la matrice $\Phi = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

Questions 2.

1. Verifier que $\forall n \geq 0$,
$$\Phi^{n+1} = \begin{pmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{pmatrix}$$
2. Écrire une fonction `mult_matrix(M1,M2)` qui retourne le produit matriciel de deux matrices 2×2. Une matrice $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ est représentée par la liste `[[a,b],[c,d]]`.
3. Écrire une fonction `square_matrix(M)` qui retourne le carré de la matrice `M`.

4. Écrire une fonction `pow_phi(n)` qui retourne Φ^n en utilisant la méthode "Square and Multiply" itérative. Indication : l'élément neutre pour la multiplication des matrices est la matrice identité, ici $Id = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

5. En déduire une fonction `Fibonacci_matrix(n)` qui retourne le n-ième terme de la suite de Fibonacci.

6. Estimer la complexité du nombres d'étapes de cette fonction.

3 Comparaison des performances

QUESTIONS 3.

1. Comparer les les performances des trois fonctions pour des valeurs raisonnables de n

2. Quelle est la plus grande valeur pour n pour l'algorithme récursif ?

3. Comparer les performances des algorithmes `Fibonacci_iterative(n)` et `Fibonacci_matrix(n)` pour des grandes valeurs de n.

4 Rapport

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