


EA4 – Éléments d’algorithmique

TP n° 2 : autour de la suite de Fibonacci


Modalités de rendu : à chaque TP, vous devrez rendre les exercices marqués par un symbole . Le rendu de l’exercice K du TP N doit être inclus dans le fichier `tpN_exK.py`, à télécharger depuis la section « Énoncés de TP ». Vous devez remplir les zones marquées par le commentaire `A REMPLIR` dans ce fichier. Ne modifiez pas les autres fonctions du fichier, sauf demande explicite de l’énoncé. Chaque fichier contient une fonction `main` qui teste les fonctions que vous devez programmer et qui vous affiche un score donné par le nombre de tests passés avec succès. Pour passer ces tests, vous devez exécuter le programme écrit.

Version de Python : attention à ne pas utiliser une version 2.x.x de Python mais exclusivement une version 3.x.x, elles ne sont pas compatibles entre elles. L’interpréteur correspondant est en général lancé par la commande `python3`, et non `python`, et nous utiliserons systématiquement le nom `python3` dans les énoncés. Pour connaître la version exacte d’un interpréteur Python, lancez-le depuis votre shell avec l’option `--version`.



Dépendances : pour ce TP vous aurez besoin de la bibliothèque graphique `matplotlib`.¹. N’oubliez pas également de télécharger le fichier `ea4lib.py`. En cas d’affichage étranges, changez les variables de configuration au début de ce fichier.

Exercice 1 : Fibonacci revisité

Dans le fichier `tp2_ex1.py`, vous trouverez le code des fonctions `fibo_1`, `fibo_2` et `fibo_3` vues en cours.

1.  Modifier les fonctions `fibo_i_adds` pour $i \in \llbracket 1, 3 \rrbracket$ pour qu’elles renvoient, comme second composant du résultat, le nombre *d’additions d’entiers* faites lors d’un appel à chacune d’entre elles.

Le fichier fourni effectue des tests de ces fonctions, puis affiche les courbes des nombres d’additions obtenus, d’abord pour les trois algorithmes, puis seulement pour `fibo_2` et `fibo_3` (en rouge pour `fibo_1`, cyan pour `fibo_2`, vert pour `fibo_3`). Ces mesures reflètent-elles les temps d’exécution constatés en cours ?

2. En mode interactif, appeler la fonction `courbes_adds` avec différentes valeurs des paramètres.
3.  Écrire la fonction `nbOfBits(i)` qui calcule le nombre de bits nécessaires pour coder la valeur de son paramètre entier.
4.  Modifier les fonctions `fibo_i_bits` pour $i \in \llbracket 1, 3 \rrbracket$ pour qu’elles renvoient, comme second composant du résultat, le nombre *d’opérations élémentaires sur les bits* faites lors d’un appel à chacune d’entre elles.

Rappel : une addition entre entiers dont le *résultat* est codé sur n bits nécessite n opérations sur les bits.







1. Pour l’obtenir avec votre gestionnaire de paquets : `apt-get install python3-matplotlib` (remplacer `apt-get` par `yum`, `brew`, ... bref votre gestionnaire de paquets si jamais ce n’est pas `apt`).

5. Compléter la fonction `courbes_ops` pour qu'elle affiche les courbes des opérations élémentaires sur les bits. Les courbes obtenues reflètent-elles les conclusions du cours ?

Exercice 2 : Fibonacci par produit de matrices

Pour la fonction `fibo_4` vue en cours, on utilise l'identité :

$$\forall n \geq 1, \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n.$$

1.  Compléter la fonction `produit_matrice_2_2` pour qu'elle calcule le produit de deux matrices de dimension 2×2 .
2.  Compléter la fonction `puissance_matrice_2_2` pour qu'elle calcule la puissance n -ième d'une matrice 2×2 *en utilisant l'algorithme d'exponentiation binaire*, à l'aide de la fonction `produit_matrice_2_2` ci-dessus.
3.  Compléter les fonctions `puissance_matrice_2_2_ops`, `produit_matrice_2_2_ops` et `fibo_4_ops` pour compter les opérations arithmétiques sur des entiers (multiplication, addition, modulo, division) effectuées à chaque appel.
4.  Compléter la fonction `courbes_ops` pour obtenir un graphique qui compare le nombre d'additions effectuées par `fibo_3_adds` avec le nombre d'opérations arithmétiques effectuées par `fibo_4_ops` (en bleu pour `fibo_4`).
5.  Compléter les fonctions `puissance_matrice_2_2_bits`, `produit_matrice_2_2_bits` et `fibo_4_bits` pour compter les opérations élémentaires sur les bits, en supposant qu'une multiplication entre deux entiers sur m et n bits nécessite $m \times n$ opérations sur des bits.
6.  Modifier la fonction `courbes_bits` pour obtenir un graphique qui compare les nombre d'opérations sur les bits de `fibo_4` et de `fibo_3` (en bleu pour `fibo_4`), et constater que cela ne correspond pas avec les courbes de temps vues en cours. Que peut-on en conclure ?