TP X1

Temps d'exécution et complexité : compléments sur le TP 3

Instructions

Les questions marquées d'astérisques (* ou **) peuvent être plus difficiles ou demander plus de temps.

1 Exponentiation rapide

Le calcul naïf de l'exponentiation (ou fonction « puissance ») repose sur des multiplications itérées. Pour calculer a^n, où n est un entier naturel, on fait

$$\underbrace{a \times \cdots \times a}_{n \text{ fois}}$$

Une implémentation Python de ce principe est

```
1 def expmult(a,n) :
2     r = 1
3     for i in range(n) :
4         r = a*r
5     return r
```

Question 1. *Tracer en fonction de* `n` *les temps d'exécution de* `expmult(2,n)` *et de* `2**n`. *On pourra utiliser des fonctions du TP 3.* [1]

Les algorithmes calculant l'exponentiation sont en fait plus efficaces. Ils reposent sur le fait que pour tout $n \in \mathbb{N}$, on a

$$\begin{array}{lcl} a^0 & = & 1 \\ a^{2n} & = & (a^2)^n \\ a^{2n+1} & = & a(a^2)^n \end{array}$$

Question 2.

(1) Écrire une fonction Python *récursive implémentant le calcul de l'exponentiation rapide ci-dessus.*

(2) Pourquoi l'exponentiation rapide porte-elle ce nom ?

Question* 3. *Écrire une fonction* Python **itérative** *implémentant l'exponentiation rapide. Comparer avec les fonctions des Questions 1 et 2.*

2 Retour sur la suite de Fibonacci

On rappelle que la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ est donnée par

$$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_n + F_{n+1}$$

1. Le TP 3 est disponible à https://perso.ens-lyon.fr/colin.riba/teaching/cpes/tp/tp03.pdf

Il est aisé de voir que les termes de la suite de Fibonacci peuvent être obtnus par produits de matrices :

$$\begin{pmatrix} F_{n+2} \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}$$

On s'intéresse dans un premier temps à des matrices Python représentées par des listes de listes. Le cas des matrices NumPy est abordé en §2.2.

2.1 Exponentiation de matrices Python

On pourra utiliser des fonctions du TP 1. [2]

Question 4. *Écrire une fonction* Python `exp_mat(a,m,n)`*, où* `a` *est une matrice* $\mathtt{m} \times \mathtt{m}$*, et qui renvoie la matrice* $\mathtt{a}^{\mathtt{n}}$. **Il est recommandé d'implémenter l'algorithme d'exponentiation rapide (§1).**

Question 5. *Écrire une fonction* Python `fibmat(n)` *qui renvoie le terme* $F_{\mathtt{n}}$ *de la suite de Fibonacci, et qui procède par exponentiation matricielle. Comparer avec les fonctions du TP 3, et commenter au vu de la complexité attendue de* `fibmat`*.*

2.2 Matrices NumPy

On utilise la bibliothèque NumPy avec la convention de nommage suivante.

```
import numpy as np
```

La bibliothèque NumPy propose (entre autres) des implémentations efficaces d'opérations matricielles. Cette efficacité repose sur l'utilisation de nombres de taille fixée.

Question 6. *Comparer les exécutions des commandes suivantes dans la console* Python*.*

- `np.array(2**64)`
- `np.array(2**64,dtype=int)`
- `np.array(2**32)`
- `np.array(2**32,dtype=int)`
- `np.array(-2**32)`
- `np.array(-2**32,dtype=int)`

Le *dtype* d'un tableau NumPy est le type des éléments du tableau (on parle parfois de « scalaires »). Les *dtypes* NumPy sont en général différents des types Python correspondants. Le *dtype* `int` correspond à des entiers « signés » de taille fixe (64 bits). C'est sur les *dtypes* de taille fixe que les primitives NumPy sont particulièrement efficaces. Le *dtype* NumPy `object` permet de représenter en NumPy les entiers Python usuels (de taille variable).

Question 7. *Implémenter le calcul de la suite de Fibonacci par produit de matrices* NumPy

(1) avec des matrices NumPy *de* dtype `int`*,*

(2) avec des matrices NumPy *de* dtype `object`*.*

On pourra utiliser les primitives `np.eye(n,dtype=X)` *(création de matrices unité de dimension* $\mathtt{n} \times \mathtt{n}$ *et de* dtype `X`*), et* `np.matmul(a,b)` *(produit de matrices* $\mathtt{a} \cdot \mathtt{b}$*) de la bibliothèque* NumPy*.*

Comparer avec les autres implémentations de la suite de Fibonacci.

2. Le TP 1 est disponible à https://perso.ens-lyon.fr/colin.riba/teaching/cpes/tp/tp01.pdf.