Faculté des Sciences
Aix*Marseille Université

Master de mathématiques, 1re année

Sixième TP

décembre 2016
B. Cadorel, P. Mercat

Un peu de programmation

1 Une fonction de tri

Le *tri à bulles* est l'une des fonctions de tri les plus simples. Pour trier une liste de longueur L, l'algorithme effectue les $L-1$ étapes suivantes :

Étape 1. On parcourt la liste en entier, en échangeant à chaque fois les éléments adjacents s'ils ne sont pas dans l'ordre. A la fin de cette étape, l'élément le plus grand est à la fin de la liste.

Étape 2. On parcourt la liste de la case 0 à la case $L-2$, en échangeant à chaque fois les éléments adjacents s'ils ne sont pas dans l'ordre. A la fin de cette étape, le deuxième élément le plus grand est à l'avant dernière case de la liste.

...

Étape $L-1$. On échange les deux premiers éléments s'ils ne sont pas dans l'ordre.

1. Programmer une fonction `tri_bulle()` qui prend en entrée une liste d'entiers et la trie en utilisant l'algorithme du tri à bulles.

2. Améliorer votre programme pour que la fonction s'arrête dès que le tableau est trié.

2 Programmation récursive

3. Que fait la fonction suivante ?

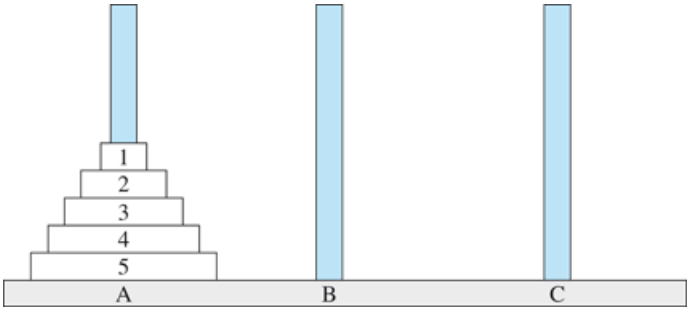
```
sage: def factorielle(n):
sage:     if n==0:
sage:         return 1
sage:     return factorielle(n-1)*n
```

4. Écrire une fonction `fibo` qui prend un entier `n` en entrée et renvoie le n-ième terme de la suite de Fibonacci (u_n) commençant par $u_0 = 1$, $u_1 = 1$.

5. Calculer quelques termes de la suite de Fibonacci. Que se passe-t-il si l'on essaie de calculer `fibo(50)` ?

6. *Le jeu des tours de Hanoï.*

Pour jouer au jeu des tours de Hanoï, il faut prendre trois tiges, que l'on appellera A, B et C, sur lesquelles peuvent coulisser des disques de bois percés en leur milieu.



On commence par placer une pile de disques de longueur décroissante sur la tige A. Le seul mouvement autorisé est de déplacer un disque situé sur le haut d'une pile pour le placer soit sur une tige vide, soit sur un disque de taille plus grande. Le but du jeu est de déplacer toute la pile sur la tige C.

Écrire une fonction `hanoi()` qui prend en entrée un entier `n`, et qui affiche la liste de mouvement à effectuer pour déplacer une pile de taille n. Les mouvements sont affichés sous la forme de paires de lettres : `A -> B` signifie par exemple qu'il faut déplacer un disque de la tige A vers la tige B.

```
sage: hanoi(2)
sage: A -> B
sage: A -> C
sage: B -> C
```

On pourra écrire d'abord une fonction `hanoi_tiges()` qui prend en entrée un entier `n` et trois chaines de caractères `D`, `A`, `I`, représentant le nom des tiges, et écrivant la séquence de mouvements à faire pour déplacer n disques de la tige de départ D vers la tige d'arrivée A, avec la tige intermédiaire I.

7. *Substitution de Thue-Morse.* Les mots de Thue-Morse s_n sont des suites de symboles 0 et 1, s'obtenant itérativement comme suit :

1. Le mot s_0 est composé uniquement du symbole 0 ;
2. Pour obtenir le mot s_n, on part du mot s_{n-1}, et on remplace chaque 0 par 01, et chaque 1 par 10.

```
0
01
0110
01101001
```

Écrire une fonction `thue_morse()` qui prend en entrée un entier `n` et qui renvoie la liste des symboles du n-ième mot de Thue-Morse.

8. *(Questions bonus) Amélioration du calcul de la suite de Fibonacci*

1. Quel est le nombre d'additions effectuées lors du calcul de u_n avec la fonction `fibo()` implémentée à la question 4 ?
2. Ecrire une fonction `fibo2()` calculant u_n avec au plus n additions. Vérifier que cela permet de calculer `fibo2(50)` et même `fibo2(10000)`.
3. *Methode encore plus efficace* Trouver une matrice $M \in M_2(\mathbb{Z})$ telle que $\forall n \geq 1$, $\begin{pmatrix} u_{n+1} \\ u_n \end{pmatrix} = M \cdot \begin{pmatrix} u_n \\ u_{n-1} \end{pmatrix}$ et en déduire l'égalité $\begin{pmatrix} u_{n+1} \\ u_n \end{pmatrix} = M^n \cdot \begin{pmatrix} u_1 \\ u_0 \end{pmatrix}, \forall n \in \mathbb{N}$. Ecrire une fonction `fibo3()` calculant u_n en utilisant l'exponentiation de matrice. Vérifier que le calcul de `fibo3(10000000)` termine rapidement contrairement à `fibo2(10000000)`. Expliquer comment l'on peut calculer une puissance en effectuant peu de produits.