TP 2 : Complexité et tableaux

20 mars 2014

1 Suite de Fibonacci

Question 1

La *suite de Fibonacci* est définie par $F_n = F_{n-1} + F_{n-2}$ et $F_1 = F_0 = 1$. Les premières valeurs prises par F sont $1, 1, 2, 3, 5, 8, 13, \ldots$ Définir une fonction récursive `fibo_rec` prenant un entier n en argument et renvoyant F_n.

Question 2

De même, définir une fonction itérative `fibo_it`. Quelle est sa complexité en nombre d'additions ?

Question 3

Afin de tester ces fonctions, afficher les valeurs de `i`, `fibo_rec(i)` et `fibo_it(i)` pour des valeurs de `i` allant de 0 à 10.

Question 4

Afficher les valeurs de `fibo_it(i)` pour `i` allant de 0 à 100. Afficher les valeurs de `fibo_rec(i)` pour `i` allant de 0 à 40. Comparer les temps d'exécution

Question 5

Un entier a est un *nombre de Fibonacci* s'il apparait dans la suite de Fibonacci, c'est à dire s'il existe i tels que $a = F_i$. Définir une fonction `nombre_fibo` prenant un nombre en argumant et renvoyant `True` si c'est un nombre de Fibonacci et `False` sinon.

Question 6

Définir une fonction `indice_fibo` prenant un nombre a en argument et renvoyant le plus petit nombre i tel que $a = F_i$ si a est un nombre de Fibonacci et `-1` sinon. Vérifier que `indice_fibo(8)`, `indice_fibo(10)`, `indice_fibo(34)` renvoient respectivemet `5`, `-1`, `8`.

2 Tables

Un tableau de taille n peut être construite en Python avec la syntaxe `[`t_0`, ..., `t_{n-1}`]`, où l'élément d'indice $i-1$, t_{i-1}, est le $i^{\text{ème}}$ élément. Par exemple, `[1, 1, 3, 5, 8]` est une tableau à 5 éléments où l'élément d'indice i représente F_i. On peut accéder à l'élément d'indice i d'un tableau `tab` avec la syntaxe `tab[`i`]`. On met à jour le tableau en associant l'indice i à une nouvelle valeur v avec `tab[`i`] = v`. On peut connaître la taille d'un tableau avec la fonction `len`. On peut également afficher un tableau directement avec la commande `print tab` Par exemple,

```
tab = [1, 1, 3, 5, 8, 13, 12, 4]
somme = tab[4] + tab[6]
print somme
print tab[2+3]
print len(tab)
tab[7] = 34
print tab
```

Question 1

Définir une fonction `affiche_tableau` affichant les éléments d'un tableau passé en argument les uns après les autres.

Question 2

Définir une fonction `min_tableau` retournant l'élément le plus petit d'un tableau passé en argument.

Question 3

Définir une fonction `membre` prenant en argument un tableau et un élément et renvoyant `True` si le tableau contient l'élément, et `False` sinon.

Question 4

Il est possible de déterminer l'appartenance d'un élément à un tableau plus efficacement si l'on suppose que les éléments du tableau sont ordonnés du plus petit au plus grand. Proposer une fonction `membre_rapide` prenant en argument un élément et un tableau dont les éléments sont ordonnés, et étant plus efficace que `membre`.

Question 5

Quelle est la complexité dans le pire des cas de `member` et de `member_rapide` ?

Question 6

Définir une fonction `indice` prenant en argument un tableau et un élément et renvoyant l'indice de la première occurence de l'élément dans le tableau s'il est présent et `-1` sinon.

Question 7

Un tableau est *sans répétition* s'il ne contient pas deux fois le même élément. Définir une fonction `sans_repet` prenant un tableau en argument et permettant de savoir s'il est sans répétition. Quelle est la complexité dans le pire cas de cette fonction ?

Question 8

En supposant que les éléments du tableau peuvent être ordonnés, proposer une idée pour détecter si un tableau est sans répétition avec une complexité $n \lg n$.

Question 9

Un mot (ou une phrase) est un *palindrome* s'il est identique lu de gauche à droite ou de droite à gauche. Un très simple parlindrome est la phrase «le sel». Étendre la notion de palindrome aux tableaux et définir une fonction `palindrome` prenant en argument un tableau et renvoyant `True` si c'est un palindrome et `False` sinon.

3 Nombres Premiers

Question 1

Un nombre $p > 1$ est *premier* s'il n'a pas d'autres diviseurs positifs que 1 et lui-même. Définir une fonction `est_premier` prenant un nombre entier en argument et renvoyant `True` s'il est premier et `False` sinon.

Question 2

On appelle P_i le $i^{\text{ème}}$ nombre premier. On a donc $P_1 = 2$, $P_2 = 3$, $P_3 = 5$, ..., $P_7 = 17$, ...Afficher les nombres P_{10i} pour i allant de 1 à 100.

Question 3

Le *crible d'Ératosthène* est un procédé qui permet de trouver les nombres premiers inférieurs à un certain nombre donné N. Il utilise le fait qu'un nombre p n'est pas premier si et seulement si il est divisible par un nombre premier $a \leq \sqrt{p}$. Au début de l'algorithme, on crée un tableau T de taille $N + 1$ contenant initialement `True` à tous les indices `[True for i in range(N+1)]`. On met T_0 et T_1 à `False`, car on sait que 0 et 1 ne sont pas des nombres premiers. Dans le corps de l'algorithme, on traverse T du plus petit indice vers le plus grand avec une variable i. Pour chaque valeur de i, si T_i est à `True`, on met à `False` tous les multiples de i strictement supérieur à i. L'algorithme renvoie un tableau T tel que l'on a T_i à `True` si et seulement si i est un nombre premier. Coder une fonction `eratos` implémentant cet algorithme.

Question 4

Utiliser la fonction `eratos` pour afficher les nombres P_{10i} pour i allant de 1 à 100.

Question 5

La *conjecture de Goldbach* stipule que tout nombre pair peut s'écrire comme la somme de deux nombres premiers. Par exemple, $20 = 7 + 13$, $98 = 31 + 67$, $100 = 29 + 71$. Vérifier la conjecture de Goldbach pour les nombres pairs jusqu'à 10 000.