Travaux Pratiques d'algorithmique n°2

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Récursivité

L'objet de cette séance est d'introduire la notion de récursivité. Nous nous intéresserons à plusieurs exemples basiques qui permettent de se familiariser avec cette technique. Vous devrez présenter clairement les algorithmes, ainsi que **le calcul de leur complexité**.

▸ **Exercice 1.** ***(Nombres de Fibonacci)***
Les nombres de Fibonacci sont définis par la relation :

$$\begin{cases} F_0 & = & 0 \\ F_1 & = & 1 \\ F_n & = & F_{n-1} + F_{n-2} \end{cases}$$

Écrire une fonction C récursive, et une autre itérative, qui calcule le n^{ieme} nombre de Fibonacci.
Comparer les temps d'exécution entre les deux fonctions pour $n = 30,\ 35, \ldots$

▸ **Exercice 2.** ***(Plus Grand Commun Diviseur)*** *On souhaite calculer le PGCD de deux entiers a et b par la méthode d'Euclide en supposant tout d'abord $a \geq b > 0$.*
Soit q et r tels que $a = b \times q + r$.

$$PGCD(a,b) = \begin{cases} b & \text{si } r = 0, \\ PGCD(b,r) & \text{sinon.} \end{cases}$$

1. *Calculer à la main le PGCD de 142 et 12 ;*
2. *Implémenter cet algorithme de façon récursive ;*
3. *Tester votre programme avec $a < b$. Que se passe-t-il ? Pourquoi ? Expliquer sur le PGCD de 12 et 142.*

▸ **Exercice 3.**

Écrire une fonction de prototype `void AffDecomp(int n, int b)` *qui affiche la décomposition de l'entier* `n` *dans la base* `b`*. Par exemple, pour* `n`*= 172 et* `b`*= 8, l'affichage sera 254. Pour cela, vous utiliserez un algorithme* ***récursif*** *dont vous estimerez la complexité.*

Écrire maintenant une fonction `Envers` *qui affiche un entier à l'envers. Par exemple,* `Envers(1995)` *donnera* `5991`*.*

▸ **Exercice 4.** ***(Somme des chiffres d'un entier)***

- *On veut calculer la somme des chiffres d'un entier. Par exemple, la somme des chiffres formant* 3281 *est* 14.

  Écrire une fonction `SommeChiffre` *prenant en argument un entier* `n` *et qui calcule récursivement la somme des chiffres qui le forment.*

- *Le critère de divisibilité suivant est connu de tous : un entier est divisible par* 3 *si la somme des chiffres qui le forment est elle-même divisible par* 3*. Autrement dit :*

$$n \equiv 0\ [3] \Leftrightarrow \sum_{i=1}^{k} c_i \equiv 0\ [3]$$

  où c_i *est le* i^{eme} *chiffre de* n *en partant de la droite.*

  On remarquera que la somme des chiffres d'un entier supérieur à 10 *est toujours strictement plus petit que cet entier. Cette simplification (passer d'un entier à la somme de ses chiffres) fait naître un algorithme récursif.*

  Un entier `n` *est-il divisible par* 3 *? Si l'entier est strictement plus petit que* 10 *: facile à déterminer ; sinon on se pose la même question sur la somme de ses chiffres.*

  Effectuer le calcul complet pour 3281 *et* 1847394.

  Écrire une fonction `MultDe3` *récursive réalisant cet algorithme. Cette fonction ne devra pas utiliser l'opérateur modulo (*`%`*).*

- *Effectuer le même travail pour la divisibilité par* 11 *:*

$$n \equiv 0\ [11] \Leftrightarrow \sum_{i=1}^{k} (-1)^i c_i \equiv 0\ [11]$$

  On utilisera une nouvelle fonction `SommeAlternée` *pour laquelle on remarquera que :*

$$\begin{aligned} SommeAlternee(17328) &= 8 - 2 + 3 - 7 + 1 \\ &= 8 - (2 - (3 - (7 - (1)))) \end{aligned}$$