Travaux Pratiques d'algorithmique n°4

Cours d'Informatique de Deuxième Année

—DEUG Sciences M3-MI—

Récursivité

Au TP précédent, nous avons introduit la notion de récursivité sur des exemples simples. Nous allons maintenant appliquer cette méthode pour résoudre des problèmes plus complexes. Vous devrez présenter clairement les algorithmes, ainsi que **le calcul de leur complexité**, que nous généraliserons la séance suivante.

▸ **Exercice 1.** **(Polynômes de Legendre)** *Ecrire une fonction récursive permettant de calculer les polynômes de Legendre définis par :*

$$\begin{cases} P_0(x) = 1 \ , \\ P_1(x) = x \ , \\ P_n(x) = \dfrac{(2\,n-1)\,x\,P_{n-1}(x) - (n-1)\,P_{n-2}(x)}{n} \ . \end{cases}$$

▸ **Exercice 2.** *Ecrire une fonction de prototype* `void Binaire (int n)` *qui affiche la décomposition binaire de l'entier* `n`*. Par exemple :*

pour `n` $= 21$*, l'affichage sera* 10101.

Pour cela, on utilisera une version **récursive** *de l'algorithme décrit lors de la deuxième séance de stage et que l'on rappelle ci-dessous :*

On effectue la division entière de `n` *par* 2*. On trouve un quotient* q_0 *et un reste* r_0*. On recommence avec* q_0*. On construit ainsi une suite de reste* $(r_0, r_1, r_2, \ldots)$*. On arrête le processus dès que l'on trouve un quotient* q_k *égal à* 0*. La décomposition de* `n` *en base* 2 *est alors* $(r_k, r_{k-1}, \ldots, r_2, r_1, r_0)$*. Par exemple pour* $n = 10$ *:*

$$\begin{array}{rclcclc} 10 & = & 5\times 2 & + & 0 & \longrightarrow & r_0 = 0 \\ 5 & = & 2\times 2 & + & 1 & \longrightarrow & r_1 = 1 \\ 2 & = & 1\times 2 & + & 0 & \longrightarrow & r_2 = 0 \\ 1 & = & 0\times 2 & + & 1 & \longrightarrow & r_3 = 1 \end{array}$$

La représentation de 10 *en base* 2 *est donc* 1010.

▸ **Exercice 3.** *On définit les nombres de Fibonacci par :*

$$\begin{cases} F(0) = 0 \ , \qquad F(1) = 1 \ , \\ F(n) = F(n-1) + F(n-2) \quad si \ n \geq 2 \ . \end{cases}$$

1. *Calculer F(6) ;*

2. *programmer de façon récursive, la fonction* `int FiboR(int n)` *qui retourne le nième nombre de Fibonacci ;*

3. *quel est le défaut de cette méthode ?*

4. *programmer une nouvelle fonction* `int FiboIt(int n)` *qui calcule le nième nombre de Fibonnaci de façon itérative. Celle-ci utilisera un tableau auxiliaire qui stockera les calculs intermédiaires ;*

5. *évaluer la complexité en temps et en espace, de ce dernier algorithme.*

6. *quel est le défaut de cette méthode ?*

7. *finalement, programmer une nouvelle fonction* `int FiboItBis(int n)` *qui sera optimale en temps et en espace.*