Algorithmique et programmation
TP 9 : Le langage Java, récursivité

Pour chacun des exercices ci-dessous, il est demandé d'écrire une solution *récursive* au problème posé.

1 Fibonacci

Écrire une fonction permettant de calculer le n^{e} terme F_n de la suite de Fibonacci. Cette suite est définie pour $n \geq 0$ de la manière suivante :

$$F_n = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ F_{n-2} + F_{n-1} & \text{si } n > 1 \end{cases}$$

Écrire ensuite un programme permettant de tester votre fonction.

2 Puissance

Écrire deux fonctions récursives de calcul x^n (bien évidemment sans utiliser la fonction `Math.pow`) :

– la première en utilisant le fait que :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ x^{n-1} \times x & \text{si } n > 0 \end{cases}$$

– la deuxième en utilisant le fait que :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ \left(x^{n/2}\right)^2 & \text{si } n > 0 \text{ et } n \text{ est pair} \\ \left(x^{(n-1)/2}\right)^2 \times x & \text{si } n > 0 \text{ et } n \text{ est impair} \end{cases}$$

Comparer ensuite les vitesses d'exécutions des deux fonctions. Utiliser pour cela l'une, puis l'autre de ces fonctions pour évaluer la valeur de l'expression

$$\sum_{i=1}^{M} \sum_{k=0}^{N} \frac{1}{2^k}.$$

Mesurer ensuite le temps d'exécution pour chaque valeur de N entre 100 et 1 000, par pas de 100.

La valeur de M sera choisie égale à 1 000. Le but est d'effectuer un grand nombre de calculs pour avoir des temps d'exécution suffisamment longs pour pouvoir être mesurés. Si les temps d'exécution sont encore trop courts, augmenter la valeur de M.

La mesure du temps d'exécution peut se faire, soit de manière externe en utilisant la commande `time(1)`, soit de manière interne au programme en utilisant la fonction standard `System.nanoTime()` [1].

1. Cf. `http://docs.oracle.com/javase/7/docs/api/java/lang/System.html#nanoTime%28%29`

3 Sierpinski

Écrire une fonction permettant de dessiner un triangle de Sierpinski. Étant donné ses trois sommets A, B et C, ce triangle est dessiné de la manière suivante :

(i) dessiner le triangle ABC ;

(ii) calculer D, E et F, respectivement les milieux des segments AB, BC et CA ;

(iii) dessiner (récursivement) les triangles de Sierpinski ADF, BED et CFE.

4 Fibonacci (suite)

Reprendre la fonction écrite pour l'exercice 1. Construire l'arbre des appels de fonctions pour $n = 5$. Que constate-t-on ?

Récrire, toujours de manière récursive, la fonction, de manière à éviter les calculs inutiles.

5 Tours de Hanoï

Écrire un programme affichant la solution au problème des tours de Hanoï (cf. exercice 4.4 de la feuille de TD).

6 Flocon de Von Koch

Écrire un programme permettant de dessiner le flocon de Von Koch (cf. exercice 4.5 de la feuille de TD).

Réfléchir ensuite à un algorithme permettant de colorier l'intérieur d'une forme arbitraire (et du flocon de Von Koch en particulier).