IFT 2125 – Introduction à l'algorithmique – TP4

21 février 2017, à remettre **au début** de la démo du 17 mars

Attention : La Question 1 ci-dessous inclut une précision ne se trouvant pas dans l'exercice 7.23 du livre et les Questions 2 et 4 correspondent à des *versions simplifiées* des exercices 7.33 et 8.30.

Question 1. Faites l'exercice 7.23 du livre :

Assuming n is a power of 2, find the exact number of scalar additions and multiplications needed by Strassen's algorithm to multiply two $n \times n$ matrices. Your answer will depend on the threshold used to stop making recursive calls. Bearing in mind what you learnt in Section 7.2, propose a threshold that minimizes the total number of scalar operations.

Vous pouvez prendre pour acquis qu'il suffit de 15 additions et 7 multiplications de matrices $n/2 \times n/2$ afin de multiplier deux matrices $n \times n$ par la méthode de Strassen, en autant que n soit pair, en dépit du fait que les équations 7.9 et 7.10 à la page 242 du livre donnent l'impression qu'il faudrait 24 additions en plus des 7 multiplications.

Question 2. Faites la *version simplifiée* suivante de l'exercice 7.33 du livre :

Consider the matrix

$$F = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

Let i and j be any two integers. What is the product of the vector (i, j) and the matrix F? What happens if i and j are two consecutive numbers from the Fibonacci sequence? Use this idea to invent a divide-and-conquer algorithm to calculate this sequence, and analyse its efficiency, counting all arithmetic operations at unit cost. (On parle ici du temps requis pour obtenir seulement le n-ème élément de la suite de Fibonacci, en fonction de n, et non pas toute la suite. De plus, ne faites *pas* la partie de l'exercice dans le livre qui demande d'analyser l'efficacité en tenant compte du temps réellement requis pour multiplier les grands entiers qui s'accumuleront à l'exécution de votre algorithme.)

Question 3. Faites l'exercice 8.28 du livre :

Consider the alphabet $\Sigma = \{a, b, c\}$. The elements of Σ have the following multiplication table, where the rows show the left-hand symbol and the columns show the right-hand symbol.

$$\begin{array}{c|ccc} & a & b & c \\ \hline a & b & b & a \\ b & c & b & a \\ c & a & c & c \end{array}$$

Thus $ab = b$, $ba = c$, and so on. Note that the multiplication defined by this table is neither commutative nor associative.

Find an efficient algorithm that examines a string $x = x_1 x_2 \cdots x_n$ of characters of Σ and decides whether or not it is possible to parenthesize x in such a way that the value of the resulting expression is a. For instance, if $x = bbbba$, your algorithm should return "yes" because $(b(bb))(ba) = a$. This expression is not unique. For example, $(b(b(b(ba)))) = a$ as well. In terms of n, the length of the string x, how much time does your algorithm take?

Question 4. Faites la *version simplifiée* suivante de l'exercice 8.30 du livre :

Let u and v be two strings of characters. We want to transform u into v with the smallest possible number of operations of the following three types: delete a character, add a character, or change a character. For instance, we can transform *abbac* into *abcbc* in three stages:

$$\begin{array}{lll} abbac & \to abac & \text{(delete } b\text{)} \\ & \to ababc & \text{(add } b\text{)} \\ & \to abcbc & \text{(change } a \text{ into } c\text{).} \end{array}$$

Show that this transformation is not optimal.

Write a dynamic programming algorithm that finds the minimum number of operations needed to transform u into v. *Contrairement à la version du problème dans le livre, je ne vous demande* pas *de trouver quelle est la suite d'opérations qui arrive à transformer u en v, mais seulement le* nombre *de ces opérations.* As a function of the lengths of u and v, how much time does your algorithm take?