IFT 2125 – Introduction à l'algorithmique – TP3

7 février 2017, à remettre **au début** de la démo du ~~17 février~~ 24 février

Question 1. Faites l'exercice 6.21 du livre :

Let $P_1, P_2, \ldots, P_n$ be n programs to be stored on a disk. Program P_i requires s_i kilobytes of storage, and the capacity of the disk is D kilobytes, where $D < \sum_{i=1}^{n} s_i$.

(a) We want to maximize the number of programs held on the disk. Prove or give a counter-example: we can use a greedy algorithm that selects programs in order of nondecreasing s_i.

(b) We want to use as much of the capacity of the disk as possible. Prove or give a counter-example: we can use a greedy algorithm that selects programs in order of nonincreasing s_i.

Question 2. Faites l'exercice 7.2 du livre :

In Section 7.1 we saw a divide-and-conquer algorithm to multiply two n–figure integers in a time in $\Theta(n^{\lg 3})$. The key idea was to reduce the required multiplication to three half-size multiplications. Show that the operands to be multiplied can be separated into three parts rather than two, so as to obtain the required product after five multiplications of integers of size approximately $n/3$ rather than the nine submultiplications that would appear necessary at first sight. Analyse the efficiency of the divide-and-conquer algorithm suggested by this idea. Is it better than our algorithm from Section 7.1?

Indice : Considérez les produits suivants... à vous de trouver les deux qui manquent et comment combiner le tout.

$$\begin{aligned} \beta &= (r+s+t)(u+v+w) \\ \gamma &= (r-s+t)(u-v+w) \\ \delta &= (r+2s+4t)(u+2v+4w) \end{aligned}$$

Question 3. Faites l'exercice 7.19 du livre :

Let $a_1, a_2, \ldots, a_k$ be positive real numbers whose sum is strictly less than 1. Consider a function $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ such that

$$f(n) \leq f(\lfloor a_1 n \rfloor) + f(\lfloor a_2 n \rfloor) + \cdots + f(\lfloor a_k n \rfloor) + cn$$

for some positive c and all sufficiently large n. Prove by constructive induction that $f(n) \in O(n)$.

Would the above work in general if the a_i's sum to exactly 1? Justify your answer with an easy argument.

Question 4. Faites l'exercice 7.34 du livre :

Represent the polynomial $p(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_d n^d$ of degree d by an array $P[0\,..\,d]$ containing its coefficients. Suppose you already have an algorithm capable of multiplying a polynomial of degree k by a polynomial of degree 1 in a time in $O(k)$, as well as another algorithm capable of multiplying two polynomials of degree k in a time in $O(k \log k)$. Let $n_1, n_2, \ldots, n_d$ be integers. Give an efficient algorithm based on divide-and-conquer to find the unique polynomial $p(n)$ of degree d whose coefficient of highest degree is 1, such that $p(n_1) = p(n_2) = \ldots = p(n_d) = 0$. Analyse the efficiency of your algorithm.